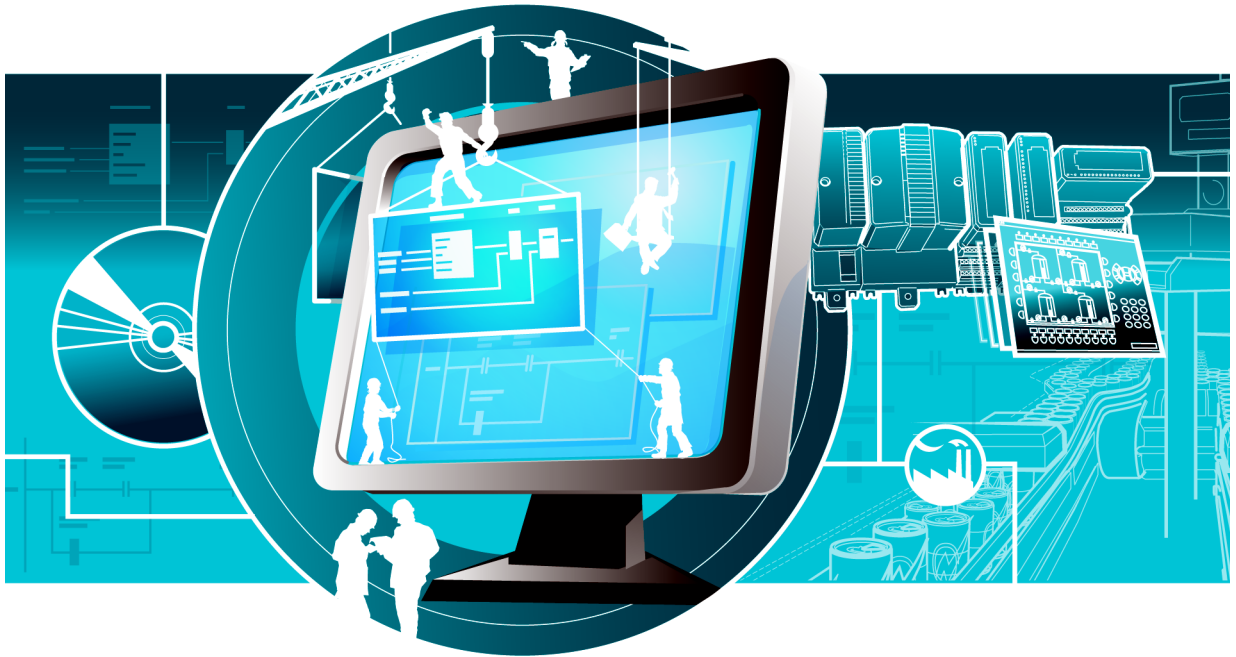


# Compact 800 Engineering Compact Control Builder AC 800M 5.1

## Getting Started



**ABB**



# **Compact 800 Engineering**

## **Compact Control Builder AC 800M 5.1**

### **Getting Started**

---

## NOTICE

The information in this document is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this document.

In no event shall ABB be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall ABB be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without written permission from ABB, and the contents thereof must not be imparted to a third party nor used for any unauthorized purpose.

The software or hardware described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

This product meets the requirements specified in EMC Directive 89/336/EEC and in Low Voltage Directive 72/23/EEC.

Copyright © 2003-2010 by ABB.  
All rights reserved.

Release: June 2010  
Document number: 3BSE041584-510

## TRADEMARKS

All rights to trademarks reside with their respective owners.

# TABLE OF CONTENTS

## About This Book

Document Conventions .....	10
Use of Warning, Caution, Information, and Tip Icons .....	11
Applicable Specifications .....	11

## Section 1 - Introduction

Documentation Strategy .....	13
Product Overview .....	13
Project Explorer .....	14
Libraries .....	15
Supported PLC and Configurations .....	16
Compact Control Builder Functions .....	17
Functions from 800xA .....	18
Multi-User Engineering .....	19
Using Online Help in Compact Control Builder .....	20
Online Manuals .....	20
Getting Help in Compact Control Builder .....	20

## Section 2 - Installing Software

Installation Prerequisites .....	24
Step-by-Step Instructions .....	25
Installing the Software .....	25
Coexistence with Previous Versions .....	27
Starting Up .....	29
Control Builder .....	29
SoftController .....	29
OPC Server .....	30
Configuration Issues .....	32

## **Section 3 - Compact Control Builder User Interface**

Introduction .....	33
About Programs and Projects .....	33
Project Templates .....	34
Project Explorer .....	35
Title Bar, Menu Bar, and Tool Bar.....	35
Project Explorer Pane .....	36
Libraries Folder.....	38
Applications Folder.....	40
Controllers Folder .....	43
Drag-and-Drop in Project Explorer .....	45
Context Menus .....	47
Message Pane.....	47
Editors .....	48
Project Documentation .....	49

## **Section 4 - MyDoors Project**

MyDoors Project .....	52
Specifications .....	52
Defined Variables.....	53
Creating MyDoors Project.....	54
Variables .....	56
Function Blocks .....	59
Code Blocks.....	62
Code Input.....	64
Testing MyDoors Project .....	71
Project Examples in Control Builder .....	75

## **Section 5 - Hardware Configuration**

Configure Hardware .....	79
Connect Variables to I/O Channels .....	83
Method 1 - Using Dot Notation .....	84
Method 2 - Using a Path Selector .....	84

Reading I/O addresses from the Application .....	86
<b>Section 6 - Connecting the PLC and Go Online</b>	
Firmware Upgrade .....	87
Setting an IP Address .....	90
Setting IP Address for PLC .....	90
Setting IP Address for PC .....	92
Downloading the Project via Ethernet .....	94
Setting the System Identity in Control Builder .....	94
Downloading the Project to a PLC .....	96
Test the Program Online .....	97
<b>Appendix A - Compact Control Builder AC 800M Settings</b>	
Product Settings for Compact Control Builder .....	100
Language .....	100
File Locations .....	101
Multi-User Configuration .....	102
Creating a Shared Project Folder .....	103
Setting Up Compact Control Builder Stations .....	106
Setting Up OPC Server .....	108
Configuration Example .....	112
Guide lines for Multi-User Engineering .....	114
Download .....	115
General Download .....	115
Download New Project to Controller .....	118
Download Project to Selected Controllers .....	118
Disable/Enable Difference Report .....	121
<b>Appendix B - Network Redundancy</b>	
Setting Up Redundant Network .....	123
Two Separate Redundant Networks .....	123
Decide IP Addresses .....	126
Setup Using the IPConfig Tool .....	127

Configure PLC Ports from Project Explorer.....	127
Configure PC Ports in Windows 7 .....	128
Download Project and Go Online .....	129
Setting Clock Synchronization using the CNCP Protocol.....	129
Design .....	131
IP Address.....	131
Separating Client/Server and Control Network .....	134
Summary of Configuration Steps.....	135

**Appendix C - Upgrade**

Introduction .....	138
Licenses .....	138
Products .....	139
Applications .....	140
Saving Application and Configuration Data in CCB 4.1/5.0.2.....	140
Remove Products .....	141
Install Products .....	142
Restore Application and Configuration Data in CCB 5.1 .....	142
Handling a download .....	144
Compatibility Issues .....	145
Removed Project Constants .....	164

**Appendix D - Communication Cables**

Connecting Control Builder PC to a PLC .....	169
--	-----

**Appendix E - Programming Languages**

General .....	171
---------------	-----

**Appendix F - Glossary**

**INDEX**



# About This Book

Welcome to Compact Control Builder AC 800M - an effective and easy to use programming tool. This manual helps the users to get start with the Compact Control Builder for the first time.

This manual contains main sections and appendixes. The appendixes contain examples of upgrading projects and details about multi-user engineering and network redundancy.

This manual is organized as follows:

[Section 1, Introduction](#), provides an overview of the Compact Control Builder.

[Section 2, Installing Software](#), describes the steps to install a single user configuration, that is, a Compact Control Builder and an OPC Server installed on the same PC.

[Section 3, Compact Control Builder User Interface](#), explains the Compact Control Builder and its core interface, the Project Explorer.

[Section 4, MyDoors Project](#), contains a small sample project that helps in familiarizing with the Control Builder environment.

[Section 5, Hardware Configuration](#), contains steps to add or remove hardware units from the tree structure in the Project Explorer.

[Section 6, Connecting the PLC and Go Online](#), contains the pre-requisites for connecting a PLC and downloading a project to go online.

## Document Conventions

The following document conventions are used in this manual:

- The names of screen elements (for example, the title in the title bar of a window, the label for a field of a dialog box, and so on) are initially capitalized.
- Capital letters are used to name the keyboard key. For example, press the ENTER key.
- Lowercase letters are used for the name of a keyboard key that is not labeled on the keyboard. For example, the **space bar**, **comma key**, and so on.
- Press CTRL+C indicates that the user must hold down the CTRL key while pressing the C key (to copy a selected object in this case).
- Press **ESC E C** indicates that the user must press and release each key in sequence (to copy a selected object in this case).
- The names of push and toggle buttons are boldfaced. For example, click **OK**.
- The names of menus and menu items are boldfaced. For example, the **File** menu.
  - The following convention is used for menu operations: **MenuName > MenuItem > CascadedMenuItem**. For example: select **File > New > Type**.
  - The **Start** menu name always refers to the **Start** menu on the Windows Task Bar.
- System prompts/messages are shown in the Courier font, and user responses/input are in the boldfaced Courier font. For example, if the user enter a value out of range, the following message is displayed:

Entered value is not valid. The value must be 0 to 30.

Variables are shown using letters in *Italic style*.

*MaxLimit*

## Use of Warning, Caution, Information, and Tip Icons

This publication includes **Warning**, **Caution**, and **Information** where appropriate to point out safety related or other important information. It also includes **Tip** to point out useful hints to the reader. The corresponding symbols should be interpreted as follows:



Electrical warning icon indicates the presence of a hazard which could result in *electrical shock*.



Warning icon indicates the presence of a hazard which could result in *personal injury*.



Caution icon indicates important information or warning related to the concept discussed in the text. It might indicate the presence of a hazard which could result in *corruption of software or damage to equipment/property*.



Information icon alerts the reader to pertinent facts and conditions.



Tip icon indicates advice on, for example, how to design the project or how to use a certain function.

Although **Warning** hazards are related to personal injury, and **Caution** hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, users are expected to comply fully with all **Warning** and **Caution** notices.

## Applicable Specifications

This product meets the requirements specified in EMC Directive 89/336/EEC and in Low Voltage Directive 72/23/EEC.



# Section 1 Introduction

The Compact Control Builder contains type solutions that are used for simple logic control, device control, loop control, alarm handling, and so on, and packaged as standard libraries. Self-defined types from other projects can also be inserted into the current project. The Control Builder supports five different programming languages, namely, Function Block Diagram, Structured Text, Instruction List, Ladder Diagram, and Sequential Function Chart. These conform to IEC 61131-3 standard.

## Documentation Strategy

This manual provides an introduction about the Compact Control Builder and provides instructions about installing and using the product, for the new users.

For information about online help in Control Builder, see [Getting Help in Compact Control Builder](#) on page 20.

## Product Overview

Compact Control Builder is a fully integrated application that runs on Windows 7 or Windows Server 2008 R1.

It provides tools for programming applications and configures hardware units from the AC 800M hardware family.

Besides the operating system requirements, the minimum software requirements are:

- Microsoft Word 2007.
- Adobe Acrobat Reader version 9.0 or later.

Microsoft Word is required for creating project documentation and Acrobat Reader is required to read the online manuals.

Project Explorer

The core user interface of the Compact Control Builder is called Project Explorer. The Project Explorer is used to create and build control projects. A project contains the entire configuration needed for an AC 800M based control solution, including control applications and hardware settings. Context menus are helpful while configuring hardware units or connecting parameters. Right-click an object to open its corresponding context menu.

Both the software (programs, functions, and so on) and the hardware (the actual hardware connected to the PLC) are modelled in a project. The relationships are shown in Figure 1.

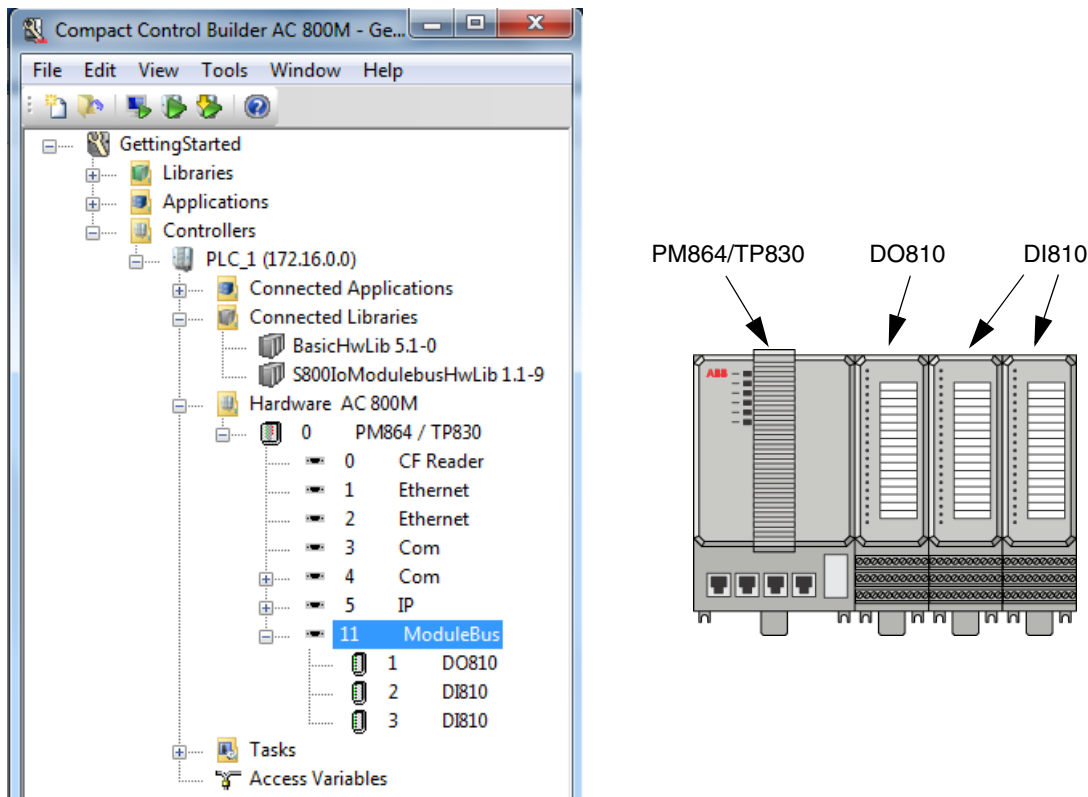


Figure 1. Project Explorer and actual hardware setup

## Libraries

Compact Control Builder provides an extensive set of predefined type solutions stored in standard libraries. These include data types, functions, function blocks and Control Modules that can be used in the projects.

All standard libraries are included during the Control Builder installation and are available in the projects.

Compact Control Builder contains the following libraries:

- The Basic library contains basic building blocks for AC 800M control software like data types, function block types, and control module types, with extended functionality, designed by ABB.

The contents inside the Basic library can be categorized as follows:

IEC 61131-3 Function Block Types, Other Function Block Types, and Control Module Types.

- The Communication Libraries include function blocks for MMS, ModBus, ModBus TCP, SattBus, COMLI, MOD5-to-MOD5 (MTM), and Siemens 3964R protocols.
- The Control Libraries include single PID control and cascade PID control function blocks, control modules, and so on.
- The Alarm and Event Library contains function blocks for alarm and event detection, and alarm printouts on a local printer.
- The Signal Libraries contain types for adding supervision, alarm handling and error handling to I/O signals, and also for the overview and forcing of boolean and real signals.

### Hardware

An extensive set of predefined hardware types, stored in standard hardware libraries, are delivered with Compact Control Builder. These hardware types are used in the projects when configuring the PLC hardware.

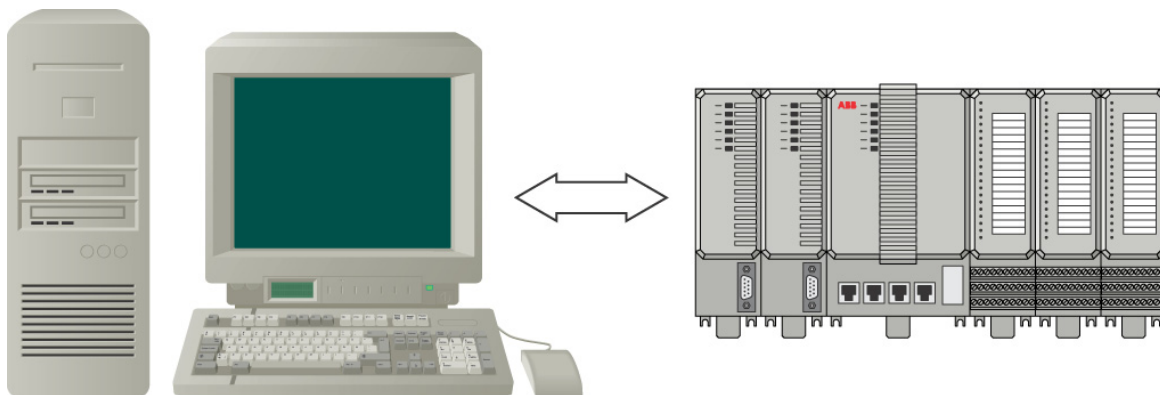
All the hardware types are included during the Compact Control Builder installation and are available in the projects.

The hardware types can be classified into the following:

- The Basic Hardware contains basic hardware types for PLC hardware, such as types for AC 800M, CPUs, Ethernet communication link, Com port, ModuleBus, and so on.
- The PROFIBUS Hardware contains hardware types for PROFIBUS communication interfaces, ABB Drives and ABB Panel 800.
- The Communication Hardware contains hardware types for the communication interfaces, MasterBus 300, ModBus TCP, IEC 61850, MOD5, AF 100, EtherNet/IP, PROFINET IO, INSUM, DriveBus and RS-232C.
- Serial Communication Protocol Hardware contains hardware types for SerialProtocol, COMLI, ModBus and Siemens 3964R.
- The I/O System Hardware contains hardware types for I/O communication interfaces, I/O adapters and I/O units; S100 (incl. S100 Rack), S200, S800 and S900.

## Supported PLC and Configurations

The AC 800M is the destination for the applications which are downloaded to the PLC from the Project Explorer. The programming code is then executed in the PLC.



*Figure 2. The Compact Control Builder station communicates with a PLC*



Do not run more than one Compact Control Builder simultaneously on a PC.



## Compact Control Builder Functions

The Compact Control Builder is used to create control solutions. These solutions are created within Control Builder projects.

Several levels of structuring are available inside one project. A project in the Control Builder can handle up to 1024 applications, and each application can handle up to 64 programs.

A maximum of 32 Control Builder PCs can be used together in multi-user environment, and a maximum of 32 PLCs can be created and handled within a project.

Using the Control Builder, self-defined libraries containing data types, function block types and control module types can be created in any project.

[Table 1](#) lists the main Compact Control Builder functions.

*Table 1. Main Control Builder Functions*

Functions
Backup/Restore
Create/change/insert libraries
Create/change/use data types, function block types and control module types
Difference report (between previous/new application)
Distribute code in an application to several PLCs
Downloading projects and go online
Multi-user engineering
Search and Navigation Tool
Testing projects offline

## Functions from 800xA

Additional functionality for building DCS type of control solutions can be used from the Control Builder available in the ABB 800xA DCS system.

The 800xA Control Builder (the Control Builder Professional) adds the following functions to the set of functions available in Compact Control Builder:

- Online Upgrade.
- Load Evaluate Go.
- Batch handling.
- Audit Trail.
- SFC Viewer.
- Diagrams.
- High Integrity Controller for SIL applications.
- CI860 for FF HSE, and CI862 for TRIO I/O.
- Information routing via HART protocol.
- Security (see [Appendix F, Glossary](#)).



The additional functions from 800xA are not included in the Compact Control Builder AC 800M.

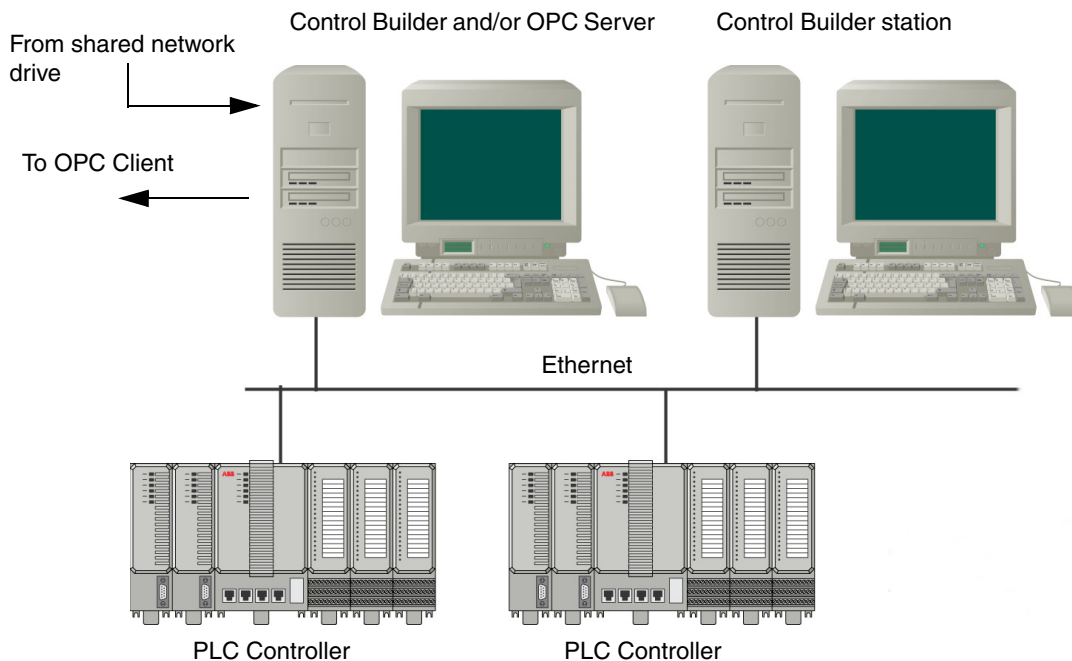


It is possible to migrate Compact Control Builder solutions to 800xA systems, and the PLC projects can be opened in the Control Builder Professional.

## Multi-User Engineering

Compact Control Builder supports multi-user engineering with a maximum of 32 separate engineering workplaces. In a multi-user configuration, all Control Builder PCs and the OPC Server must have access to the common project file(s). This means that a common Project folder must be created on a shared network server.

The network server can be placed anywhere in the network; in a Control Builder PC, in an OPC Server PC, or located as a stand-alone file server.



*Figure 3. Programmers can share the same project. Multi-user engineering stores projects on a shared network drive*

## Using Online Help in Compact Control Builder

The Compact Control Builder provides online documentation.

### Online Manuals

The Compact Control Builder provides related manuals online.



Acrobat Reader is required to open and read the online manuals provided by the Compact Control Builder.

#### Accessing Online Manuals

To access the online manuals from the Project Explorer, select **Help > Manuals**.

### Getting Help in Compact Control Builder

The Online help in the Compact Control Builder can be obtained by:

- Context-Sensitive Help (F1).
- Contents Topic.
- Index.
- Keyword Search.

#### Accessing context-sensitive Help

To access context-sensitive help for items in Project Explorer:

1. Select the element for which help is required (any item from the tree, command inside an editor, and so on).
2. Press the F1 key.

#### Accessing contents topic

Click Help in any pop-up window to view the Online help based on topics.

### Using the Online Help Index

The index offers many ways to find the information:

- Enter the action about which the information is needed (for example, “configure” or “download”).
- Enter the name of the object about which the information is needed (for example, “PM864” or “project explorer”).



It is not always possible to find information about a single object by entering its name. Enter the name of the category instead (for example, “I/O units” or “data types”). This lists the objects or units, from which the topic can be selected.

- Enter the subject about which information is needed (for example, “function block types” or “communication interfaces”).



For information about a specific library object or a specific hardware unit, select the object in Project Explorer, and press F1 key.

### Text Search

The text search runs through all topics and finds all matches. The text must be specific, else the search ends with too many search hits.



## Section 2 Installing Software

This section explains how to install and start up a single-user configuration, which means a Compact Control Builder and an OPC Server installed together on the same PC station. The software delivered on the DVD consists of three parts - the ABB Common 3rd Party Install kit, the Compact Control Builder AC 800M, and the OPC Server for AC 800M. Each of these is installed with the help of installation wizards.

- The first installation wizard installs the common 3rd party software required to work with ABB Compact Control Builder AC 800M. This is different from the software requirement mentioned in [Product Overview](#) on page 13.
- The second installation wizard installs Compact Control Builder, Base Software for SoftController, and User Documentation.
- The third installation wizard installs OPC Server for AC 800M.



Run the Compact Control Builder installation before running the OPC Server installation.



The Compact Control Builder opens the projects stored in a project folder created during the installation.

If the project folder path is changed in a recent installation, the previous projects cannot be found by the Control Builder. This problem is solved by either changing the project folder path back to previous location, or copying the previous projects from the Windows explorer into the current Project folder location.

## Installation Prerequisites

Before installing the Compact Control Builder on a PC that has Windows 7 or Windows Server 2008 R1 installed:

- Install all the other software that conforms to the minimum software requirement for Compact Control Builder. See [Product Overview](#) on page 13.
- Login to Windows 7 or Windows Server 2008 R1, with Administrator privileges.
- Turn off the User Account Control (UAC) in Windows on the PC<sup>1</sup>
- Remove previous Control Builder versions from the PC<sup>2</sup>. This also includes other products that comes with a Compact Control Builder installation (for example, OPC Server for AC 800M).



Do not install Compact Control Builder on a PC that already has Control Builder Professional installed. A Compact Control Builder and a Control Builder Professional cannot coexist in a PC.

- 
1. In Windows 7, go to **Start > Control Panel > System and Security > User Accounts** (in the left pane) > **User Accounts** (in the right pane) > **Change User Account Control Settings** (in the right pane), and drag the control to **Never Notify**.  
In Windows Server 2008, go to **Start > Control Panel > User Accounts > Turn User Account Control on or off**, and uncheck the checkbox "Use User Account Control (UAC)". Click **OK**.
  2. In Windows 7, go to **Start > Control Panel > Programs > Uninstall a program** to remove the old version.  
In Windows Server 2008, go to **Start > Control Panel > Programs and Features** to remove the old version.

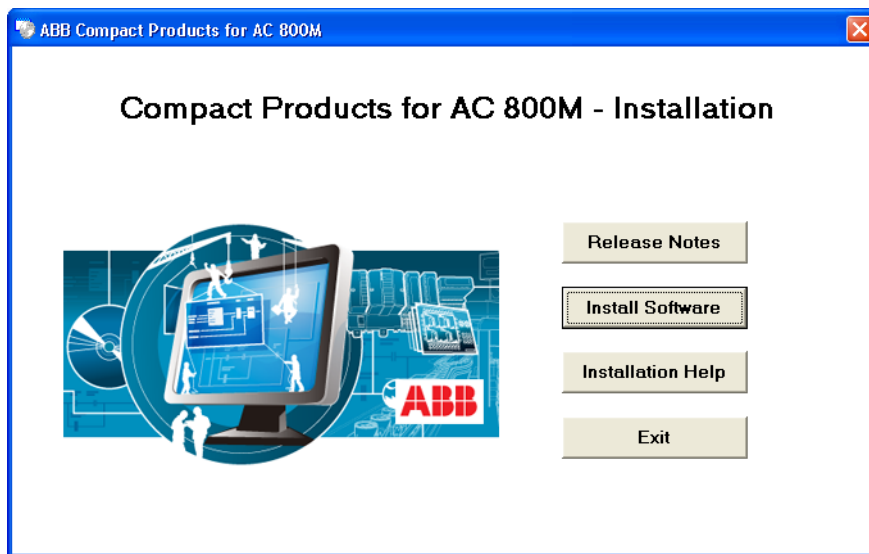


## Step-by-Step Instructions

Install the software from the DVD onto the local disk, as the software cannot be run from the DVD or a network drive.

### Installing the Software

1. Login as Administrator in Windows.
2. Insert the DVD into the drive. After a few seconds the Welcome dialog appears as shown in [Figure 4](#). If the dialog does not appear, start the file *Startme.bat*, located in the root directory of the DVD.



*Figure 4. The Welcome dialog of the installation process*

The installation dialog contains the following buttons:

- The **Release Notes** button provides the latest information.
- The **Install Software** button activates the installation procedure. The Install Software dialog opens as shown in [Figure 5](#).
- The **Installation help** button accesses information on how to install a product.
- The **Exit** button quits the installation procedure.

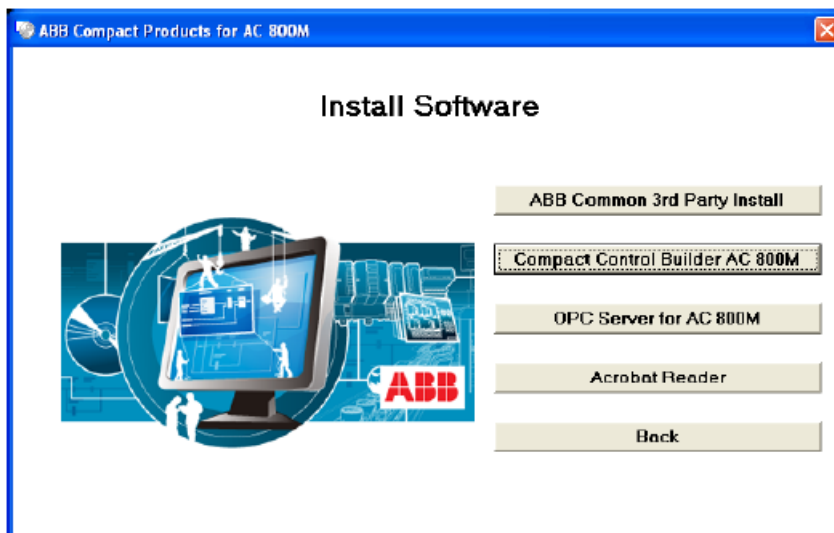


Figure 5. Install Software dialog

### Installing the ABB Common 3rd Party Software

The required common 3rd party software components (see also [Product Overview](#) on page 13) must be installed before installing the Compact Control Builder and OPC Server.

1. In the Install Software dialog, click **ABB Common 3rd Party Install** to start the Installation Wizard.
2. Follow the installation instructions that is displayed in the wizard. This installation does not include the requirements mentioned in [Product Overview](#) on page 13.

### Installing the Compact Control Builder



Always start with the Compact Control Builder installation. The OPC Server installation needs to read the Control Builder settings that are created during the Compact Control Builder installation.

1. In the Install Software dialog, click **Compact Control Builder AC800M** to start the Installation Wizard.
2. Follow the installation instructions that is displayed in the wizard.



If any project saved from the previous version need to be used through upgrade, select the installation type - **Custom** (see [Coexistence with Previous Versions](#) on page 27). Select the installation type - **Typical** - only if no project saved from the previous version is used with this new version.



Clicking **Cancel** in any of the installation wizard dialogs interrupts the installation. When installation procedure is interrupted, all the previously installed components are disregarded.

### Installing the OPC Server for AC 800M

1. In the Install Software dialog, click **OPC Server for AC 800M**, to start the Installation Wizard.
2. Follow the installation instructions that is displayed in the wizard.

Running the OPC Server on the same PC as the Compact Control Builder does not require further settings.



For more information about setting up an OPC Server for multi-user engineering, see [Setting Up OPC Server](#) on page 108.

## Coexistence with Previous Versions

The Compact Control Builder projects saved from previous versions can be upgraded and used with the new installed version.

This is possible only if the required options are selected during the installation of Compact Control Builder:

1. In the Installation Wizard of Compact Control Builder AC 800M, select the installation type as **Custom**. The Select Features dialog is displayed.

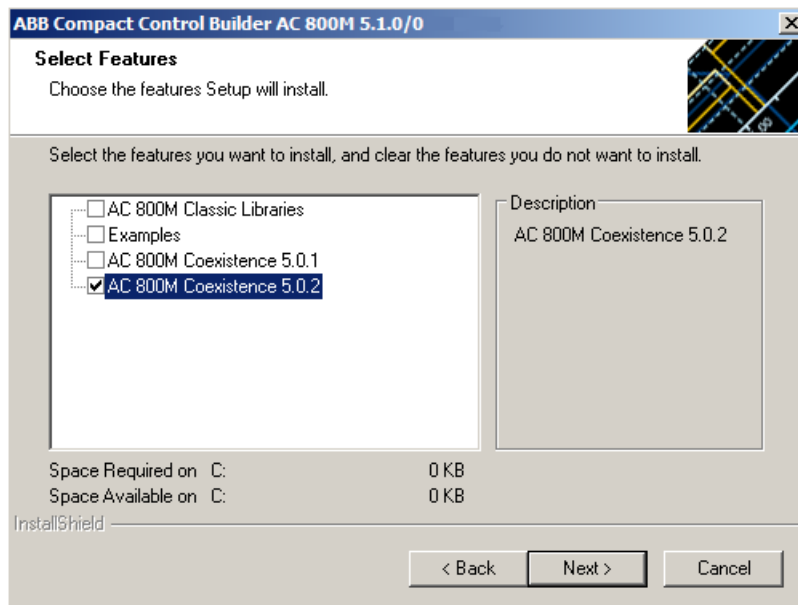


Figure 6. Select Features dialog for Custom install option

2. Select the version for coexistence. For example, if any project saved in Compact Control Builder 5.0.2 need to be used, select **AC 800M Coexistence 5.0.2**.
3. Click **Next** to proceed with the installation.



If the installation type **Typical** is selected during installation, it is later possible to change the features to include the coexistence feature. Go to Control Panel > Programs and Features, right click ABB Compact Control Builder AC 800M and select **Change**. Select **Modify** and click **Next**. The Select Features dialog is displayed.

## Starting Up

### Control Builder

#### Starting the Compact Control Builder

Double-click the Control Builder icon on the desktop (if selected during installation), or from the Start menu on the Windows Task Bar,

**Start > All Programs > ABB Industrial IT AC 800M > Compact Control Builder AC 800M.**

### SoftController

The SoftController is a simulation tool that runs with Base Software. A SoftController allows the download of projects from the Project Explorer even though the user may not have access to a real AC 800M PLC. Instead of downloading to a PLC, the projects can be downloaded to the SoftController.



In order to start the SoftController, it is necessary either to have administrator privileges in Windows, or be part of the local group *ABB Controller user group*. Contact the administrator and apply to be a member of this group. The *ABB Controller user group* is created automatically in Windows during the SoftController installation.

#### Starting the SoftController

The following steps help to start the SoftController and to locate its network address (the address must be set in Project Explorer and OPC Server panel).

1. Double-click the SoftController icon on the desktop (if desktop shortcut is selected during installation), or start the SoftController from the Start menu:

**Start > All Programs > ABB Industrial IT AC 800M > .... > SoftController 5.1.**

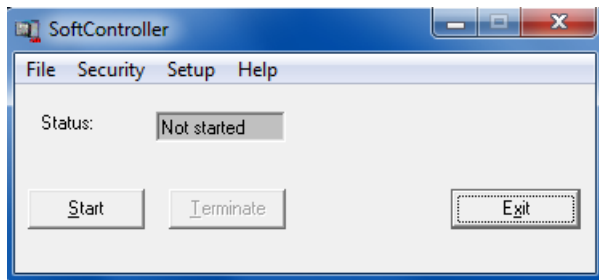


Figure 7. The SoftController start panel

2. Click **Start**. The Status field displays *Started* and the SoftController starts.
3. Select **File > View log file**. A Session.LOG file opens in Notepad.
4. Scroll down to find a network address. An example of an IP address:  
10.46.35.117:2.



To find out the IP address for a PC, open the command prompt (DOS editor) and run the command **ipconfig**.

5. Close the Notepad program.

To learn more about running an application in a SoftController, see [Section 5, Hardware Configuration](#) and [Section 6, Connecting the PLC and Go Online](#).

### Stopping the SoftController

1. Click **Terminate** to open the SoftController dialog.
2. Click **Yes**. The Status field displays *Not started* and the SoftController stops.
3. Click **Exit**.

## OPC Server

After the OPC Server is installed, it is easy to connect a PLC to the OPC Server from the OPC panel. Before doing this, ensure that the controller is connected to the network.

### Starting the OPC Server

Double-click the OPC Server icon on the desktop (if desktop shortcut is selected during installation), or start the OPC Server from the Start menu:

**Start > All Programs > ABB Industrial IT AC 800M > ..... > OPC Server for AC 800M 5.1.**

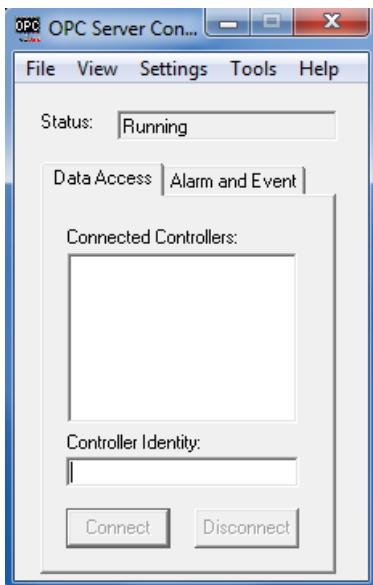


Figure 8. The OPC Server Configuration window

### Connecting the OPC Server

1. Open the Data Access tab screen.
2. In the Controller Identity field, enter the IP address of the PLC.
3. Click **Connect**.
4. Open the Alarm and Event tab screen, and repeat Step 2 and Step 3.



For information on creating a PLC Id, see [Setting an IP Address](#) on page 90 and [Setting the System Identity in Control Builder](#) on page 94.

## Configuration Issues

The Compact Control Builder supports multi-user engineering. For more information see [Appendix A, Compact Control Builder AC 800M Settings](#).

This manual also covers configuration issues like network redundancy and upgrading projects to 5.1 version. For more information, see [Appendix B, Network Redundancy](#) and [Appendix C, Upgrade](#).



# Section 3 Compact Control Builder User Interface

## Introduction

This subsection provides a brief introduction to the Compact Control Builder, and its core interface, the Project Explorer.

## About Programs and Projects

The following list describes the hierarchy among projects, applications, programs, and tasks in the Compact Control Builder.

- A project is the top level software unit and it contains the configuration data for libraries, applications, connected hardware, and so on. It also groups libraries, applications and the connected hardware in an hierarchical tree structure in the Project Explorer.
- Each application contains programs and additional objects (data types, function block types, control module types) that are used within the application.
- Each program is connected to a task, which decides how often the program is executed. It is also possible to connect individual function blocks and control modules to different tasks.

[Figure 9](#) shows the sequence from creating a new project to performing a download.

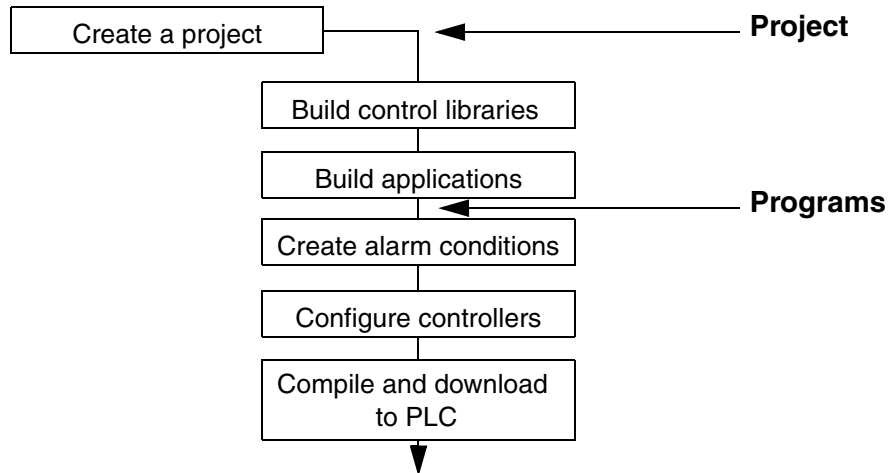


Figure 9. Sequence for building a project and the hierarchy between a Project and Programs

## Project Templates

While creating a new project, the Control Builder provides a set of predefined templates, typical for a control system. The templates contain predefined initial setup data that is suitable for different kind of projects. The Compact Control Builder provides the following project templates:

- AC800M, for normal use.
- SoftController, for development use, without access to a PLC.
- EmptyProject, consisting a minimum configuration with only the System folder inserted.

An empty project template contains only the mandatory system firmware functions, with no additional application or hardware functions.

## Project Explorer

Project Explorer is the core user interface in the Control Builder programming tool. It displays the currently active project. Only one project can be open at the same time.

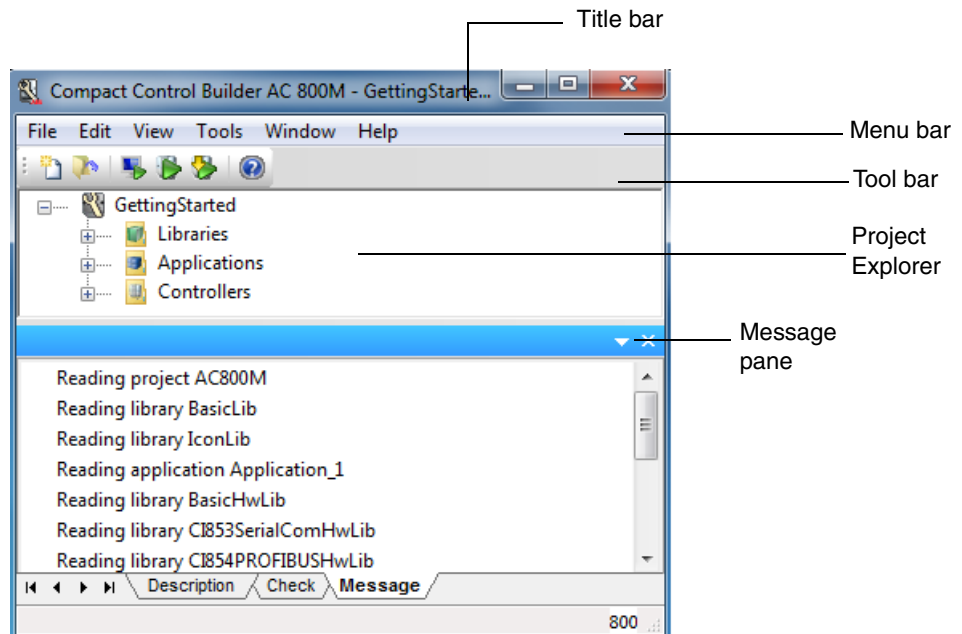


Figure 10. Project Explorer

### Title Bar, Menu Bar, and Tool Bar

The title bar shows the project name.

The menu bar contains the drop-down menus: File, Edit, View, Tools, Window, and Help.



When menu items on the menus are grayed out, they cannot be accessed (the function is not allowed in the current context).

The tool bar contains icons that serve as shortcuts to the most common Control Builder functions, such as online help and download.



For detailed information about the menu bar and tool bar, refer to the Control Builder online help.

## Project Explorer Pane

The Project Explorer pane contains three main folders, see [Figure 11](#):

- The Libraries folder, see [Libraries Folder](#) on page 38.
- The Applications folder, see [Applications Folder](#) on page 40.
- The Controllers folder, see [Controllers Folder](#) on page 43.

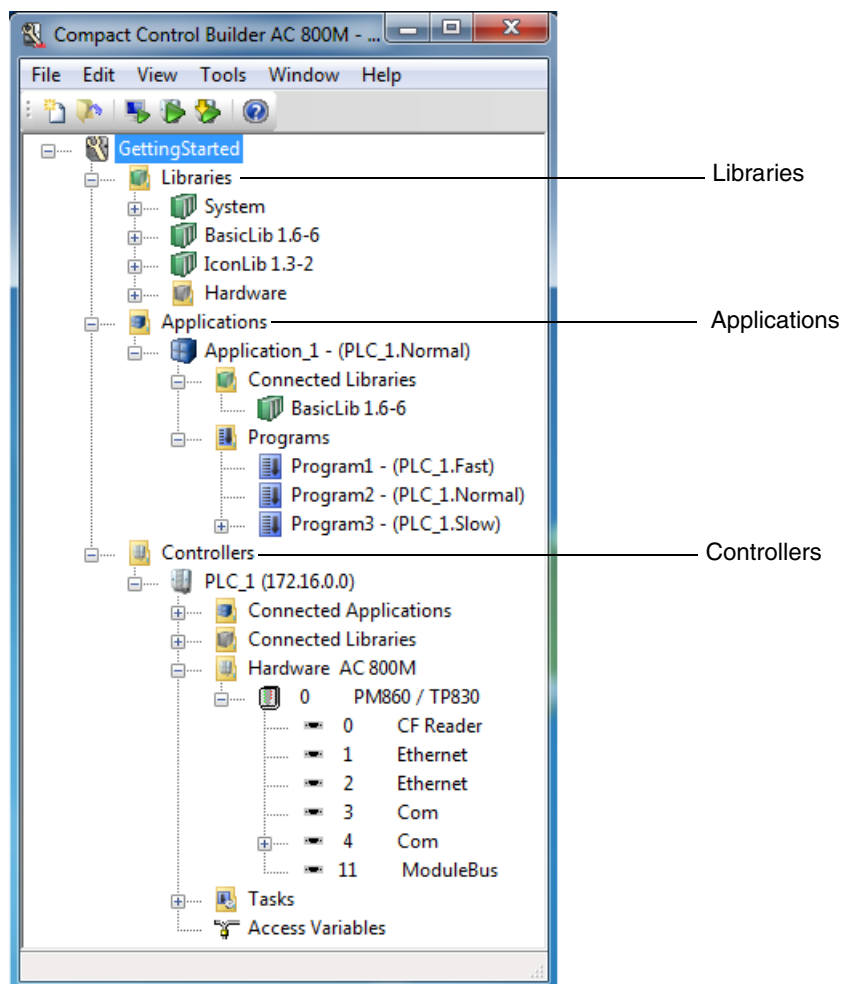


Figure 11. The Project Explorer pane, showing the three main folders, Libraries, Applications, and Controllers

## Libraries Folder

When a project is created, the Libraries folder contains the System folder (containing firmware functions that can be used throughout the applications) and the Basic and the Icon libraries (the two libraries that are always connected to a project).

Besides these three, the Libraries folder also contains a Hardware folder with a sub-folder containing Basic hardware types (BasicHWLib).

When the project is created, both the standard libraries and self-defined libraries can be added into the Libraries and Hardware folders.



When a new library is created, the subfolders – Data Types, Control Module Types, and Function Block Types – under the library are not visible because they do not contain any objects.

Right-click the library to open the context menu, and go to **New** which displays the option to create the different types (see [Figure 12](#)). Once the types are created, they are displayed under the corresponding subfolders in the library.

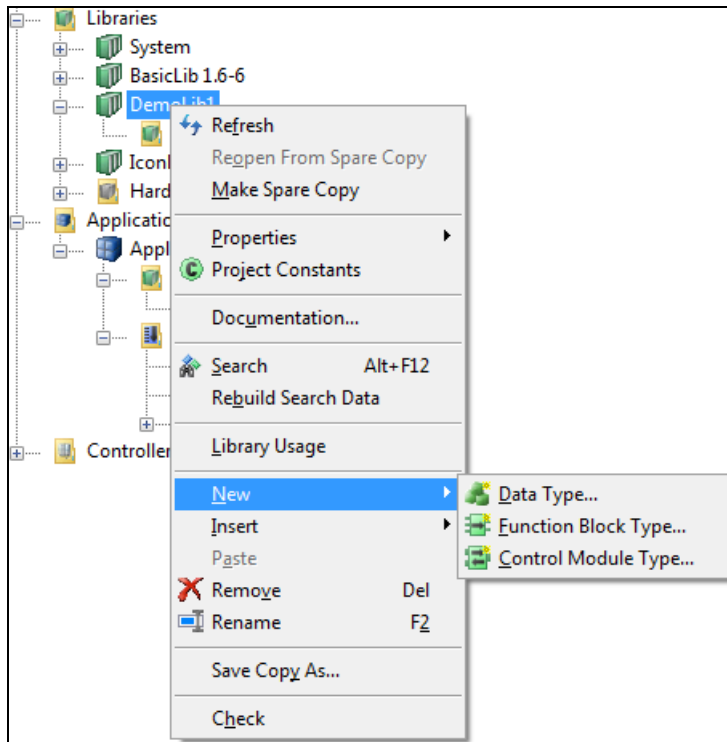


Figure 12. Creating types under a library



A library can only be added to an application if it has first been added to the Libraries folder. A hardware unit (type) can only be connected in a controller configuration if the corresponding hardware library has first been added to the Hardware folder.



For more information on libraries and library handling, see Online Help.

## Applications Folder

The Applications folder holds all the code that is downloaded to the PLC(s). This code can be stored either as program, or as control module types or single control modules. The chosen method depends on the requirements of the particular application.

The Applications folder contains applications and other applications folders. To create a new application folder under an application, right click the application and select **New Folder**. The new application folder may in turn contain both applications and application folders.

Due to this, it is possible to structure or group the applications in the Project Explorer. It is also possible to move applications and application folders in the folder structure using the drag and drop operation.

The Connected Libraries folder contains all libraries that are connected to this particular application. Libraries are connected by right-clicking this folder and selecting Connect Library. However, only libraries that have already been inserted to the project can be connected to an application. In order to access the types inside a library, it must be connected to the application. Connect a library to an application by right-clicking the Connected Libraries icon and select a library from the drop-down menu.

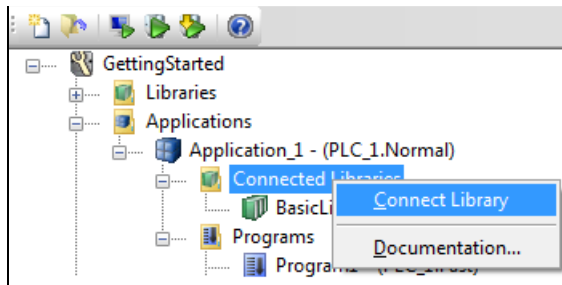


Figure 13. The context menu for connecting a library to an application

The Application folder contains three sub-folders for collecting types: Data Types, Function Block Types, and Control Module Types. The user can either insert an existing type (from another project) or create a new type within any of these three sub-folders.



There are two ways of organizing the code, in programs or in control modules. Control modules are stored in the Control Modules folder, while programs are stored in the Programs folder.

The Programs folder in the default application contains three programs. Each of these three programs are connected to a default task. The task connections can be changed, as well as the own tasks and programs can be added.



When a new application is created, the subfolders – Control Modules, Control Module Types, Data Types, Function Block Types, and Programs – under the application are not visible because they do not contain any objects. Right-click the application to open the context menu, and go to **New** which displays the option to create the different types, control modules, and programs (see [Figure 14](#)). Once these are created, they are displayed under the corresponding subfolders in the application.

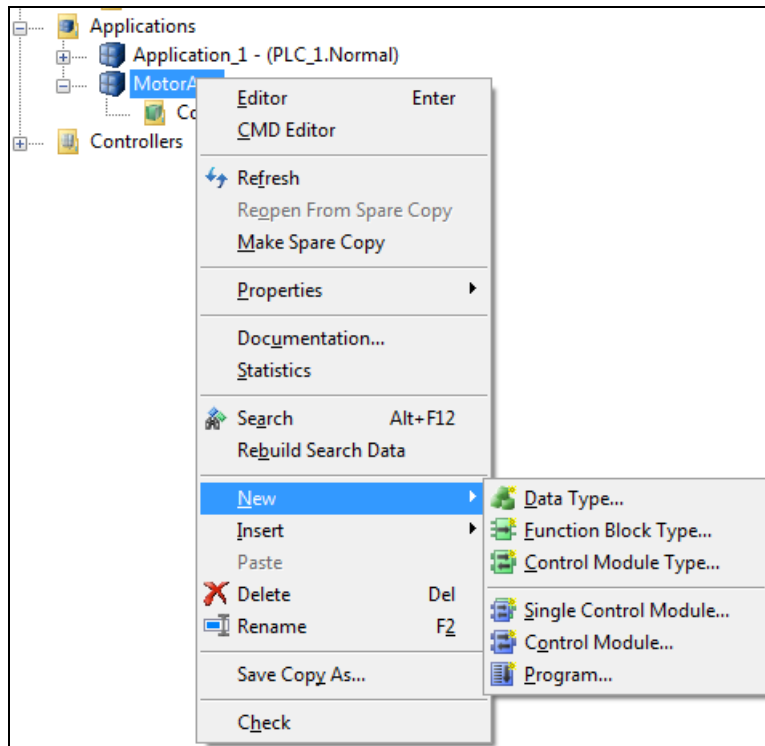


Figure 14. Creating types, control modules or programs under an application.



From the objects in the Applications folder, a number of software editors can be opened, see [Editors](#) on page 48.

The code can be checked for errors by clicking on the Check icon on the tool bar. Errors are indicated by a red triangle next to the object in question (in offline mode). Descriptions of the errors (if any) are displayed in the Check tab of the message pane.

## Controllers Folder

The Controllers folder contains all controllers that belong to the project.

Each controller has a Connected Applications folder with the application(s) running in the controller, and a Connected Hardware Libraries folder with all hardware types to be used when configuring the controller. The applications and Hardware Libraries are connected by right-clicking the folder and selecting **Connected Applications** and **Connected Hardware Libraries** respectively. It is only hardware libraries added to the project that can be connected to a controller.

For each controller, there is a CPU unit to which other hardware units, such as I/O units and communication interfaces can be added. Units can also be added to the controller on the same level as the CPU unit. The controller structure mirrors the physical structure, which means that all ports and buses have their own corresponding unit (icon) in Project Explorer.



For more information about hardware configuration and the Controllers folder, see [Configure Hardware](#) on page 79.

The Controllers folder also contains a sub-folder Tasks. The tasks folder contains tasks that are used to control the execution of the applications. By default, the Tasks folder contains three tasks: Fast, Normal, and Slow. However, the tasks can be added to the applications as needed. The Connected Applications folder contains all the applications that are connected to the PLC.

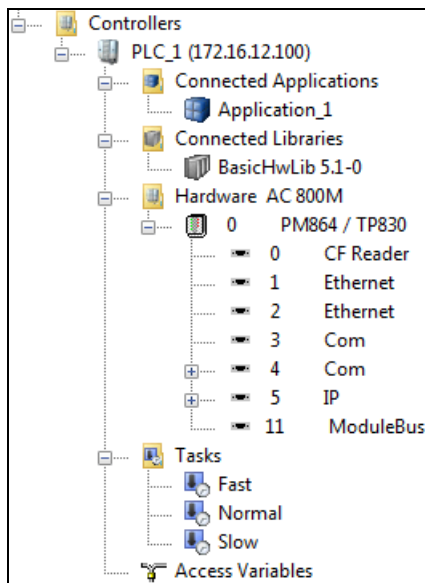


Figure 15. The PLC structure in Project Explorer, with corresponding icons for ports and buses

Double-click the 'Tasks' folder to open a task overview. Double-clicking an individual task, displays the Task Properties dialog for that particular task.



CPU units, I/O units, communication ports, communication interfaces, and so on, can be opened using editors, see [Editors](#) on page 48.

## Drag-and-Drop in Project Explorer

The Project Explorer supports drag-and-drop operations.

### Dragging to Text Input Fields

All objects can be dragged to an arbitrary text input field or text editor. When the object is dropped, the current name of the object becomes the text input. This helps in deriving names for variables, parameters, and function blocks, from the existing object names.

For example, in [Figure 16](#), the name of the parameter is the result of a drag-and-drop operation from the library *FBCReactorLib* to the Name column in the Function Block editor. The text can be modified in the Name column.

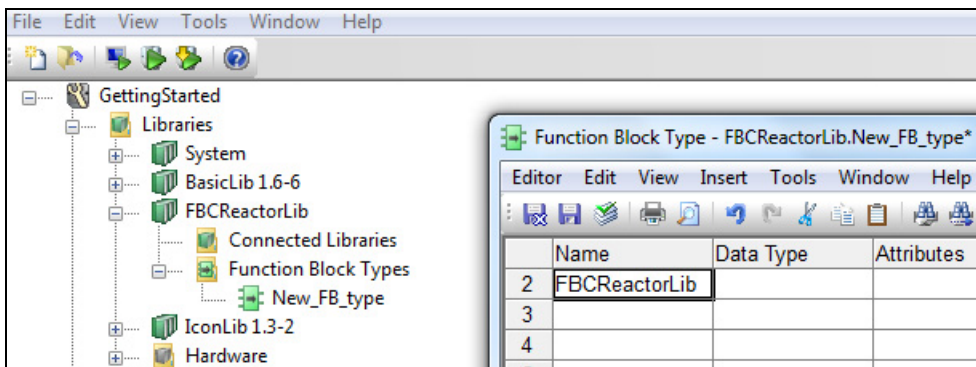


Figure 16. Parameter name derived by dragging to text input field

### Dragging to Objects

Some objects can be dragged to other objects. [Table 2](#) shows the supported actions.

*Table 2. Drag-and-Drop operations with objects*

Drag Source	Drop Target	Operation
Library	Application or another library	Connects the library to the application or the target library. The source library is then visible in the Connected Libraries folder in the target application or library.
Hardware Library	Controller	Connects the hardware library to the controller. The source hardware library is then visible in the Connected Libraries folder in the target controller.
Application	Controller	Assigns the application to the Controller. The source application is then visible in the Connected Applications folder in the target controller. <b>Note:</b> If the application is an application reference object (that is, an application shown below the "Assigned Applications" object for a Controller), then this is a Move operation that removes the previous assignment.
Application	Task	This results in two operations: <ul style="list-style-type: none"> <li>• Assigns the application to the task.</li> <li>• Assigns the application to the corresponding controller.</li> </ul> The source application is then visible in the Connected Applications folder in the corresponding controller.
Application	Application Folder	Moves the application to the target application folder.
Application Folder	Application Folder	Moves the application folder and its contents to the target application folder.

## Context Menu

Context menus can be used to edit the properties of various objects. Context menus are displayed by right-clicking an object in Project Explorer.

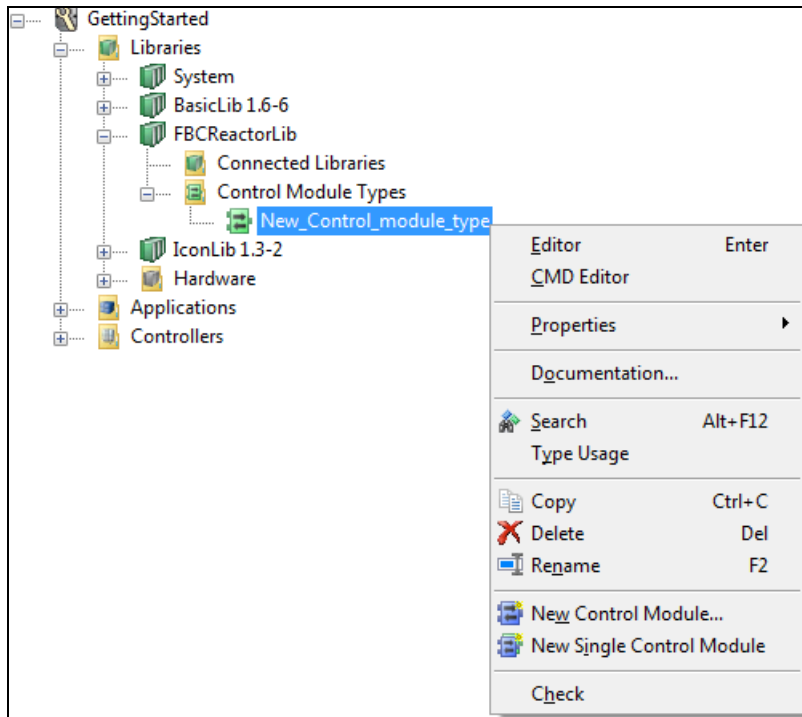


Figure 17. Context menu in Project Explorer

## Message Pane

The message pane contains three tabs:

- **Description**, shows a description of the selected type or hardware object.
- **Check**, shows the result of a code check, including error messages.
- **Message**, showing messages resulting from events in Control Builder, such as compiling and loading a new project.

Editors

Control Builder contains a number of editors. The editors can be accessed from Project Explorer. To access an editor, right-click on the object (a PLC, another hardware unit, an application, a program, or a type) and select the editor from the context menu.

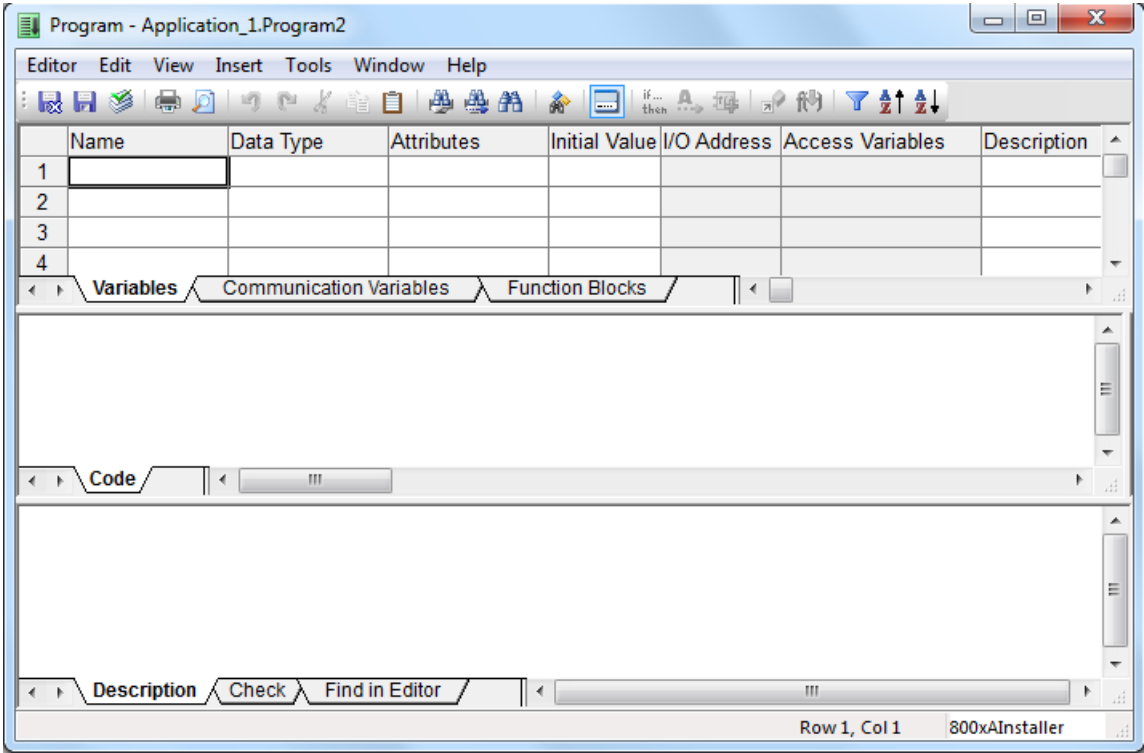


Figure 18. Program editor

Among many things, editors are used to declare project constants, and parameters, as well as to declare variables and connect them to I/O channels. There are also a number of programming language-specific editors, such as the Function Block Diagram (FBD) editor and the Control Module Diagram (CMD) editor.



## Project Documentation

Compact Control Builder facilitates a project documentation feature for libraries, applications, and PLCs or for single types. The project documentation is developed as a Microsoft Word file and can be connected to either a standard document template or user/company specific template. A table of contents is automatically generated for every project document.

### Creating Project Documentation

To create project documentation from the Project Explorer:

1. Right-click an object in the Project Explorer tree, and select **Documentation** to open the Documentation window.
2. Click **More** to open the Editor Properties window.
3. Select a tab in the window.
4. Click **OK**.



## Section 4 MyDoors Project

This section describes building a small project and getting familiarized with the Control Builder environment. An example project called MyDoors, to simulate the entrance to a store is created. While working with the MyDoors project, the concept of declaring variables, function blocks, and separating code by using code blocks, is also described.

At the end of the MyDoors project, the application is tested in the Control Builder Test mode. This helps to verify, in a secure way, how variable values and conditions are changing during a program execution.



If the access to a PLC or IO modules is not available, build this project with a SoftController. Look for SoftController specific instructions throughout the MyDoors project example.

However, to study the MyDoors project example, the Control Builder installation comes with an existing example called ShopDoors. See the subsection [Project Examples in Control Builder](#) on page 75, for locating the project examples.



For locating the ShopDoors example, or any other Control Builder examples, follow the instruction given under section [Project Examples in Control Builder](#) on page 75.

Once the guidelines are followed in this section, proceed with the further sections:

- [Section 5, Hardware Configuration](#) explains how to setup a hardware configuration according to the control system.
- [Section 6, Connecting the PLC and Go Online](#) covers all the essential steps for making it possible to download an application and go online.

## MyDoors Project

Before creating the project and writing the code, refer to the [Specifications](#) on page 52 and the [Defined Variables](#) on page 53.

### Specifications

This project simulates the entrance to a store. The following specifications are given:

- The entrance consists of two sliding doors that open when a customer activates a photocell.
- Each door is opened and closed by its own motor.
- The doors return to default position (closed) five seconds after the last activation of the photocell. Consequently, several customers arriving one after the other extends the time the door remains open.
- The number of customers is recorded for statistics. Manual reset of this counter should be possible.
- The total number of times the doors have opened since they were last serviced should be recorded.
- Each time the door opens, the counter should be incremented. When the counter reaches a preset limit, a flag indicates that service is required. Manual reset of the flag is possible.

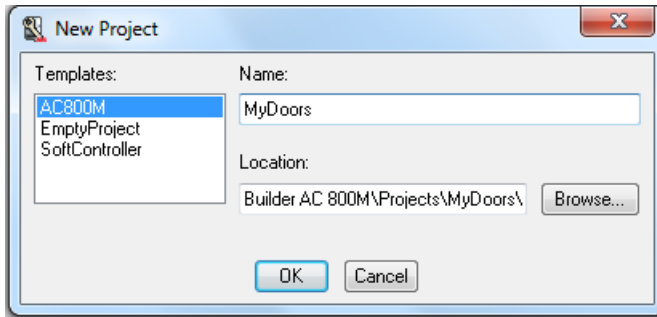
## Defined Variables

- **Photocell**  
The photocell has two states, active and inactive, typically represented by a Boolean variable. In this project, a Boolean variable named `Photo_Cell` (true = active, false = inactive) is used.
- **Door motors**  
The entrance itself consists of two doors facing each other. Each door is opened by a motor controlled by Boolean signals (`Motor_1` and `Motor_2`). The time the doors should remain open is declared in a variable `DoorsOpen_Time` of type time.
- **Number of customers**  
Each time the photocell is activated, a counter representing the total number of customers entering the shop should be incremented. The counter, `Customers_Qty`, is of type integer.
- **Reset the counter on certain dates**  
On certain dates, the shop manager records the total number of customers up to that date, and resets the counter. Consequently, a Boolean variable `Reset_Counter` is declared, which resets the counter.
- **Door service intervals**  
The doors should have regular service intervals, approximately after every 10,000 openings; also the number of openings from the previous service need to be recorded. The record is represented as the variable `Openings_Freq` of type dint.
- **Time for service**  
When the counter reaches the upper limit defined by `Openings_Total` of type dint, a flag (`Service_Req` of type Boolean) is set, indicating that service is required. This flag can be accessed by all PLCs in the project. Manual reset of the service counter is activated using a Boolean variable `Serviced`. The doors should continue to work even if service is not performed.

## Creating MyDoors Project

### Creating a New Project

1. From the Project Explorer, select **File > New Project**, or click the  icon. A New Project window opens.



*Figure 19. The New Project window for setting up a project*

2. Select the **AC800M** template and type *MyDoors* in the **Name** field. Ignore the given location path (for now).
3. Click **OK**. Project Explorer creates and opens MyDoors project, see [Figure 20](#).

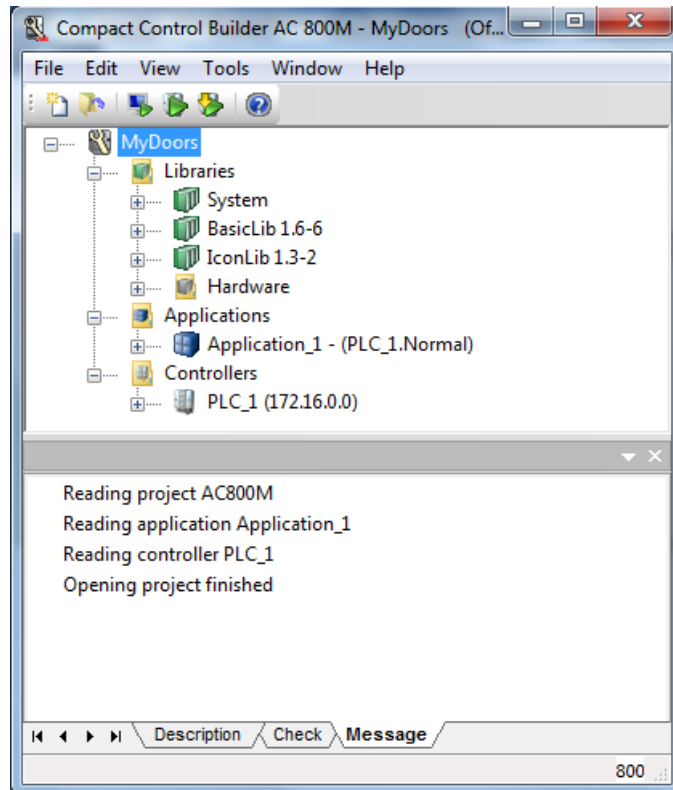


Figure 20. MyDoors project opened in Project Explorer

The Libraries folder contains the standard libraries Basic library (BasicLib) and Icon library (IconLib).



The System folder is always automatically inserted into a project. It contains firmware functions and cannot be removed from the project or changed by the user.

## Variables

There are different types of variables in Compact Control Builder for storing and computing values (local, global, communication, and access variables), where the local variables are the most frequently used in Control Builder. They always belong to the local code inside a function block, control module or a program.

Communication variables are used to communicate between applications in the same controller or between different controllers in the network. The name of the communication variable must be unique within the project. Within a project, Control Builder automatically finds the referenced communication variables. If the communication variable is accessed from another project, then the IP address needs to be specified.

In this example, you will declare 10 local variables and one communication variable in a program named **Program2**.

### Declaring Local Variables and Communication Variable

1. In the Project Explorer, expand the project tree to see **Program2**, (Figure 21). Double-click the icon to open the Program editor.

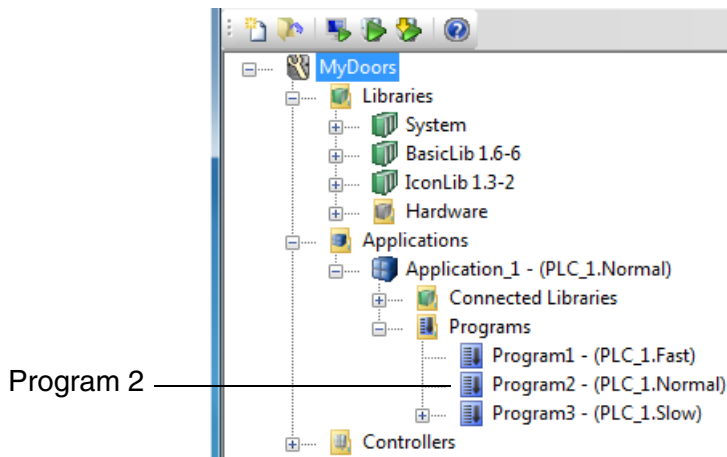


Figure 21. Programs folder expanded with three preset programs available

2. The program editor is divided into three panes: the declaration pane, the code pane, and the message pane, see Figure 22.



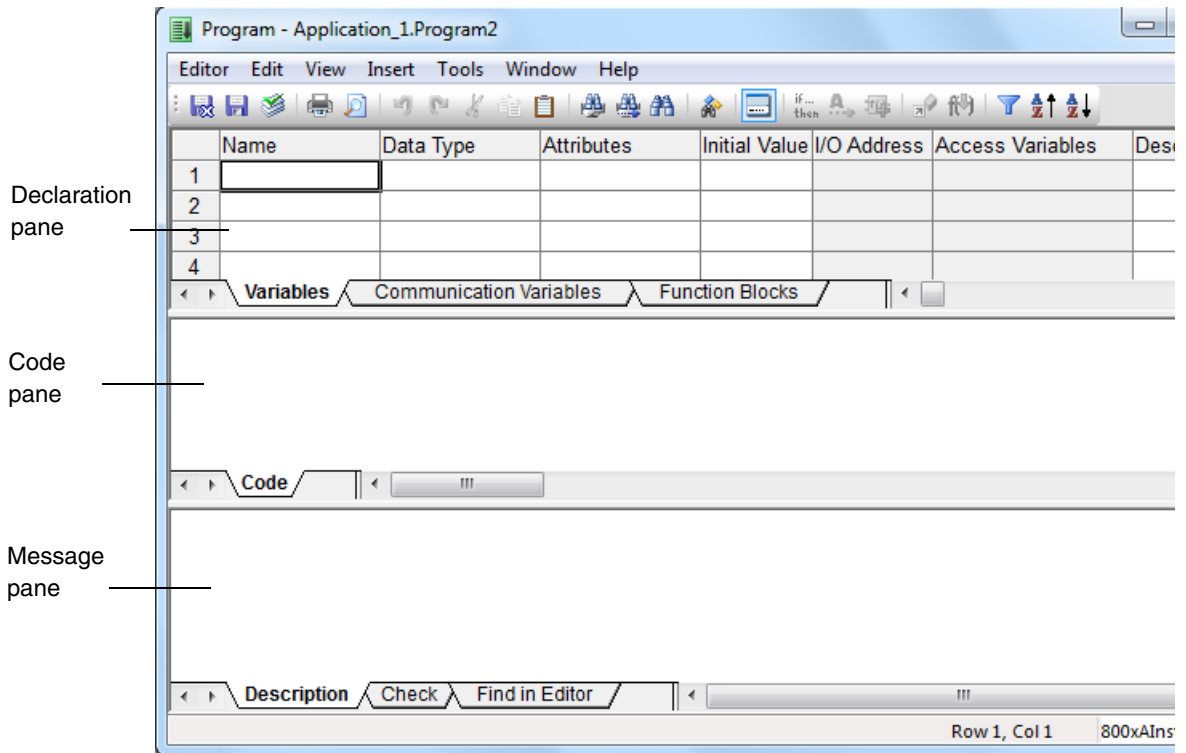


Figure 22. The editor for program **Program2**

3. Place the cursor in the upper left-hand cell in the declaration pane and enter `Photo_Cell` in the 'Name' column.
4. Move one cell to the right by pressing the tab key. Type `bool` in the "Data type" column. Move the cursor to next column labeled "Attributes".
5. Choose the default setting `retain` (which means that the variable keeps its value at a restart). Press the tab key to move to the next column.
6. Set the initial value to `false` to indicate that the doors are closed at start-up.
7. Skip the column I/O address. The address is automatically filled in later when configuring the hardware.

8. Description can be entered in the last column. The declaration of the Boolean variable is shown in [Figure 23](#).

	Name	Data Type	Attributes	Initial Value	I/O Address	Access Variables	Description
1	Photo_Cell	bool	retain	false			Photo cell to doors

*Figure 23. Declaration of the Boolean variable Photo\_Cell*

9. Declare a second variable named, DoorsOpen\_Time which represents duration of time the doors remain open. Complete the declaration of DoorsOpen\_Time according to row 2 in [Figure 24](#).

	Name	Data Type	Attributes	Initial Value	I/O Address	Access Variables	Description
1	Photo_Cell	bool	retain	false			Photo cell to doors
2	DoorsOpen_Time	time	constant	T#5s			Time duration that doors should be opened
3	DoorsOpen_ET	time	retain				Elapsed time after photo cell has been activated

*Figure 24. Declaration of the DoorsOpen\_Time and DoorsOpen\_ET variables*



For the attribute constant of the variable DoorsOpen\_Time, either explicitly enter **constant**, or scroll through the available formats using Alt-key together with the up and down arrow keys, or press Ctrl+J to display the list of attributes and then select constant.

10. Declare a variable named, DoorsOpen\_ET that records the time elapsed since the photocell was activated last time. Complete the declaration of DoorsOpen\_ET as shown in row 3 in [Figure 24](#).
11. Declare the remaining variables (starting from row 4) in the grid as shown in [Figure 25](#).

3	DoorsOpen_ET	time	retain				Elapsed time after photo cell has been activated
4	Motor_1	bool	retain	false			Output to motor opening door 1
5	Motor_2	bool	retain	false			Output to motor opening door 2
6	Openings_Total	dint	constant	10			Total number of openings until service
7	Openings_Freq	dint	retain				Number of openings since last service
8	Serviced	bool	retain	false			Flag that resets the number of openings
9	Customers_Qty	dint	retain	0			Total number of customers
10	Reset_Counter	bool	retain	false			Flag that resets the number of customers
11							

*Figure 25. Declaration of the remaining variables*




To specify the Data types in the column, choose **Insert > Variable, Type, Attribute** from the editor (or press CTRL+J) to access a list of possible data types.

12. Select the **Communication Variables** tab in the program editor, and declare the `Service_Req` communication variable as shown in [Figure 26](#).

	Name	Data Type	Attributes	Direction	Initial Value	ISP Value	Interval Time	IP Address	Description
1	Service_Req	bool	retain	out	false		normal	auto	Flag that is set when service is required
2									
3									
4									

Variables    Communication Variables    Function Blocks

*Figure 26. Declaration of the communication variable `Service_Req`*

13. Click **Check**  to check for errors.
14. Click **Save**  to save the variables.

## Function Blocks

Timers and counters in the Compact Control Builder are normally represented as function block types and located in the Basic library. This example declares one Timer (TOF), and two Counters (CTU) from the Basic library.

### Declaring Function Blocks

Make sure the program editor is open, (see [Figure 21](#), to open editor)

1. Select the **Function Blocks** tab in the declaration pane.
2. Place the cursor in the upper left-hand cell in the declaration pane and type `OpenDoors` in the Name column.
3. Move one cell to the right by pressing the tab key. Right-click the cell and select **Insert > Variable, Type, Attribute** from the context menu. A dialog list opens.

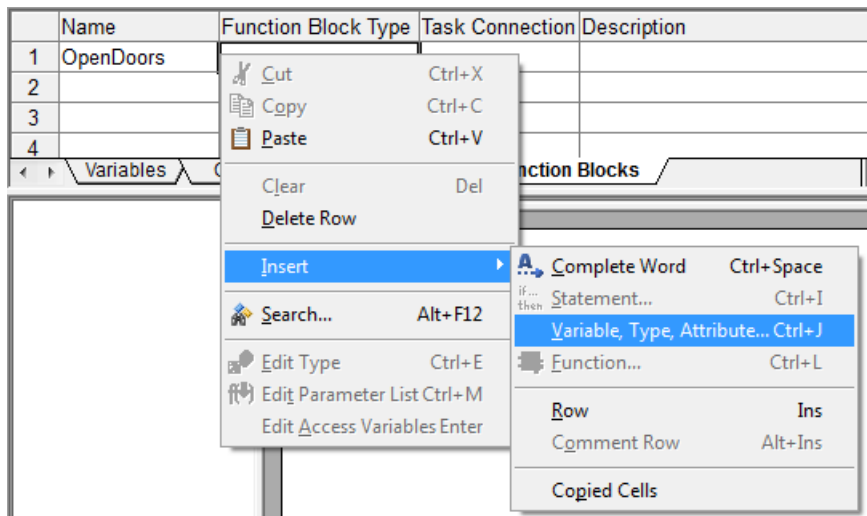


Figure 27. The path in the context menu for selecting (for example) function blocks



It is also possible to display the list of function block types by pressing Ctrl+J inside the cell.

- 4. Type **TO** (or TOF) to jump down to the TOF Function Block type in the list.

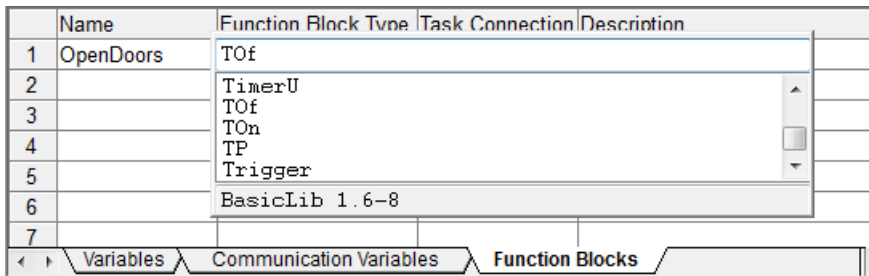


Figure 28. The TOF function block type is selected

- 5. Press the ENTER key to declare the TOF in the program editor. Write Description text according to [Figure 29](#).
- 6. Similarly, declare the two CTU function blocks in row 2 and row 3. Name them Customer\_Count\_Up and Service\_Count\_Doors, according to [Figure 29](#).

	Name	Function Block Type	Task Connection	Description
1	OpenDoors	TOF		Timer for motor
2	Customer_Count_Up	CTU		Counter for the number of customers
3	Service_Count_Doors	CTU		Counter for the number of openings of doors
4				
5				

Variables Communication Variables **Function Blocks**

Figure 29. Declaring the function blocks



For information about the TOF and the CTU function blocks, refer to the Control Builder Online Help. Place the cursor in the Function Block Type cell (for example TOF), and press **F1**.

7. Click **Check**  to check for errors.

## Code Blocks

Both programming editors (programs and control modules) support code blocks. All the code blocks, except the first code block, are always self-defined which means that user defined code blocks can be created for structuring code. Thus, code blocks are a way of enhancing the readability, traceability, and structure, of the programming code. However, the number of code blocks must be kept to a minimum, else there is a risk of badly arranged programming structure.

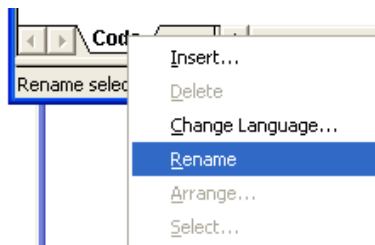
The code blocks are always executed in a predetermined order, and in this example from left to right (programs).



Some characters are not allowed in Code block names. See the Online help for more information on the characters that must not be used in code block names.

### Creating Code Blocks

1. Right-click the Code tab and select **Rename** from the context menu. A Rename Code Block window opens.



*Figure 30. Right-click the code block tab to access the context menu*

2. Write `Motors_Doors` in the Name field.
3. Click **OK**.
4. Right-click the `Motors_Doors` tab and select **Insert** from the context menu. A Insert New Code Block window opens.
5. Write `Number_Of_Customers` in the Name field and check that the Structured Text language is selected. Click **OK**.
6. Add a third code block in the same way and name it `Service_On_Doors`.

	Name	Data Type	Attributes	Initial Value	I/O Address	Access Variables	Description
1	Photo_Cell	bool	retain	false			Photo cell to doors
2	DoorsOpen_Time	time	constant	T#5s			Time duration that the doors should be opened
3	DoorsOpen_ET	time	retain				Elapsed time after photo cell has been activated
4	Motor_1	bool	retain	false			Output to motor opening door 1
5	Motor_2	bool	retain	false			Output to motor opening Door 2
6	Openings_Total	dint	constant	10			Total number of openings until service
7	Openings_Freq	dint	retain				Number of openings since last service
8	Serviced	bool	retain	false			Flag that resets the number of openings
9	Customers_Qty	dint	retain	0			Total number of customers
10	Reset_Couter	bool	retain	false			Flag that resets the number of customers
11							
12							

Variables

Communication Variables

Function Blocks

Motors\_Doors

Number\_Of\_Customers

Service\_On\_Doors

Row 1, Col 1

Code Blocks

*Figure 31. The program editor with three new code blocks created*

After creating the three code blocks representing motors, customers, and service issues in the program editor, the corresponding code block specific programming code can be written.



The code blocks are executed from left to right in the following sequence: Motors\_Doors, Number\_Of\_Customers, Service\_On\_Doors.

## Code Input

### Making a Function Block Call in Motors\_Doors

1. Select the code block tab **Motors\_Doors** and place the cursor in the code pane.
2. Select **Insert > Variable, Type, Attribute** from the Menu bar (or press CTRL+J). A dialog list opens.



The variables and function blocks declared under the **Function Blocks** tab in the declaration pane are shown in the dialog list.

3. Select the **OpenDoors** function block in the list and press the ENTER key.



Type the initial letters of the Function block (for example, **Op** for OpenDoors), to choose the required function block in the scroll list.

4. Make sure the cursor is located directly after `OpenDoors` in the code pane. Enter a left-hand parenthesis '`(`'.

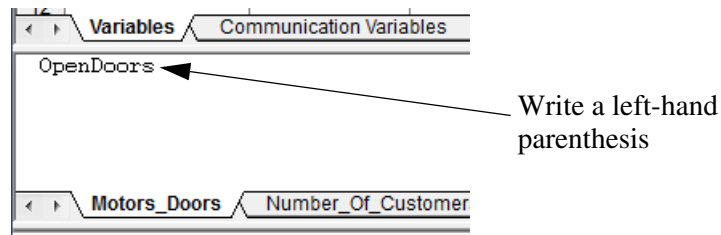


Figure 32. Write a left-hand parenthesis after `OpenDoors` in the code pane




5. When the leading left-hand parenthesis ‘ ( ‘ is entered, a Function block call editor opens, see [Figure 33](#).

	Name	Data Type	Initial Value	Parameter	Direction	Description
1	In	bool			in	Timer input
2	PT	time			in	Pre-set time
3	Q	bool			out	Timer output
4	ET	time			out	Elapsed time

Parameters

Row 1, Col 4

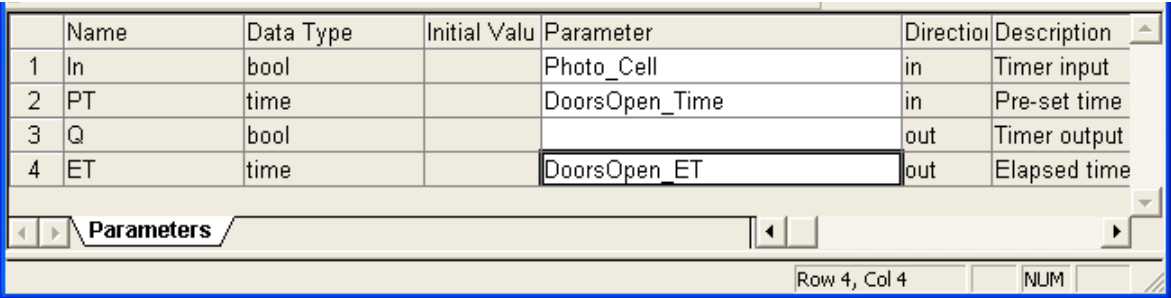
Figure 33. The Function block call editor

6. Place the cursor in the first empty white cell and select **Insert > Parameter From List** (or press CTRL+J, or click  -icon in the Menu bar). A variable list opens.
7. Select **Photo\_Cell** and press the ENTER key to accept the selection.

	Name	Data Type	Initial Value	Parameter	Direction	Description
1	In	bool		Photo_Cell		
2	PT	time		Photo_Cell		
3	Q	bool		Reset_Counter		
4	ET	time		Service_Count_Doors		
				Service_Req		
				Serviced		
				bool Photo cell to doors		


Figure 34. Variable list in the Function block call dialog

8. Fill in the other two variables in the parameter list according to [Figure 35](#).



	Name	Data Type	Initial Value	Parameter	Direction	Description
1	In	bool		Photo_Cell	in	Timer input
2	PT	time		DoorsOpen_Time	in	Pre-set time
3	Q	bool			out	Timer output
4	ET	time		DoorsOpen_ET	out	Elapsed time

Figure 35. Connecting function block parameters

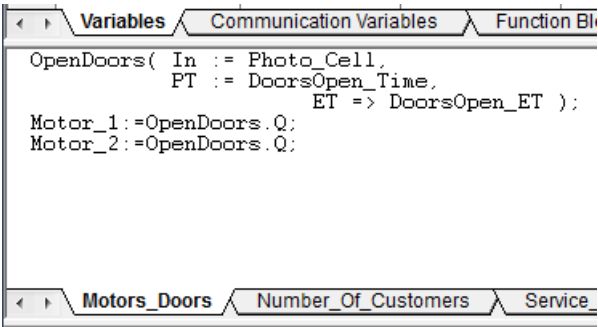
9. Click **Save and Close**  to insert the parameters into the code.

**Q parameter in TOF**

The output Q parameter is a Boolean signal, which represents the status on the door position (open or closed) and is passed on to the motors. For both doors to open, the Q signal must be passed to both motors. To achieve this, write the following code in the code pane:

```
Motor_1 := OpenDoors.Q;  
Motor_2 := OpenDoors.Q;
```

The output Q is now addressed directly to the function block and a value assigned to both motors to open the doors, see [Figure 36](#).



```
OpenDoors( In := Photo_Cell,  
          PT := DoorsOpen_Time,  
          ET => DoorsOpen_ET );  
Motor_1:=OpenDoors.Q;  
Motor_2:=OpenDoors.Q;
```

Figure 36. The code block Motors\_Doors



The code can be written structured with or without tabs and spaces.

10. Click **Save**  to save the code.

### **Making a Function Block Call in Number\_Of\_Customers**

1. Select the code block tab `Number_Of_Customers` and place the cursor on the first line in the code pane.
2. Select **Insert > Variable, Type, Attribute** from the Menu bar (or press CTRL+J). A dialog list opens.



The variables and function blocks declared under the **Function Blocks** tab in the declaration pane are shown in the dialog list.

3. Select the `Customer_Count_Up` function block in the list.



Type the initial letters of the Function block (for example, **Cu** for `Customer_Count_Up`) to choose the required function block in the scroll list.

4. Accept the selection by pressing the ENTER key. Type the leading left-hand parenthesis '('. A Function block call editor opens.
5. Place the cursor in the first empty white cell and select **Insert > Parameter From List** (or press CTRL+J). A variable list opens.

6. Select `Photo_Cell` in the list and press the ENTER key to accept the selection.

	Name	Data Type	Initial Value	Parameter	Direction	Description
1	CU	bool		<u>Photo_Cell</u>		
2	Reset	bool		Photo_Cell		
3	PV	dint		Reset_Counter		
4	Q	bool		Service_Count_Doors		
5	CV	dint		Service_Req		
				Serviced		
				bool Photo cell to doors		


7. Fill in the other two variables in the parameter list according to [Figure 37](#). Ignore the parameter for PV and Q.

	Name	Data Type	Initial Value	Parameter	Direction	Description
1	CU	bool		Photo_Cell	in	Count up inp
2	Reset	bool		Reset_Counter	in	Resets the c
3	PV	dint			in	Preset value
4	Q	bool			out	Indicates tha
5	CV	dint		Customers_Qty	out	The present i

Parameters

Row 5, Col 4 NUM

Figure 37. Connecting CTU function block parameters

8. Click **Save and Close**  to insert the parameters into the code. See [Figure 38](#).

Customer_Count_Up( CU := Photo_Cell, Reset := Reset_Counter, CV => Customers_Qty );						
Motors_Doors <b>Number_Of_Customers</b> Service_On_Doors						

Row 1, Col 2 NUM

Figure 38. The code block `Number_Of_Customers`

### Making a Function Block Call in Service\_On\_Doors

1. Select the code block tab `Service_On_Doors` and place the cursor on the first line in the code pane.
2. Select **Insert > Variable, Type, Attribute** from the Menu bar (or press CTRL+J). A dialog list opens.



The variables and function blocks declared under the **Function Blocks** tab in the declaration pane are shown in the dialog list.

3. Select the `Service_Count_Doors` function block in the list.
4. Accept the selection by pressing the ENTER key. Enter the leading left-hand parenthesis '(', to open the parameter editor.

	Name	Data Type	Initial Value	Parameter	Direction	Description
1	CU	bool			in	Count up in
2	Reset	bool			in	Resets the c
3	PV	dint			in	Preset value
4	Q	bool			out	Indicates the
5	CV	dint			out	The present

Parameters

Row 1, Col 4

Figure 39. The Function Block Call Editor


5. Place the cursor in the first empty white cell and select **Insert > Parameter From List** (or press CTRL+J). A variable list opens.
6. Select `Motor_1` in the list and press the ENTER key to accept the selection.

	Name	Data Type	Initial Value	Parameter	Direction	Description
1	CU	bool		Motor_1		
2	Reset	bool		Motor_1		
3	PV	dint		Motor_2		
4	Q	bool		OpenDoors		
5	CV	dint		Openings_Freq		
				Openings_Total		
				bool		

7. Fill in the other variables in the parameter list according to [Figure 40](#).

	Name	Data Type	Initial Value	Parameter	Direction	Description
1	CU	bool		Motor_1		
2	Reset	bool		Serviced		
3	PV	dint		Openings_Total		
4	Q	bool		Service_Req		
5	CV	dint		Openings_Freq		

Figure 40. Connecting CTU function block parameters

8. Click **Save and Close**  to insert the parameters into the code. See [Figure 41](#).

```
Service_Count_Doors( CU := Motor_1,
                    Reset := Serviced,
                    PV := Openings_Total,
                    Q => Service_Req,
                    CV => Openings_Freq );
```

◀ ▶

Motors\_Doors

Number\_Of\_Customers

**Service\_On\_Doors**

Figure 41. The parameters connected in the code block, *Service\_On\_Doors*



If an error message should be displayed in the message pane, double-click the error line to move the cursor to the error location in the code. Also, a brief description of the type of error is displayed in the message pane.

## Testing MyDoors Project

Before downloading the application to a PLC and going online, it is necessary to first test the application in an offline mode to ensure that everything is working properly. This mode is called the Test Mode, where the Control Builder compiles and executes the code locally in the PC as if it is an AC 800M PLC.

The test mode is an easy way to test the application many times. However, external communication is disabled during the test mode, thus reading and writing variables connected to IO units cannot be validated in test mode.



A communication variable used between applications or between tasks within the applications does not work in Test Mode.

In Test Mode, the communication variable does only work within the same task in an application.



Before running the program in Test mode, there is an option to enable the Difference Report window. However, the Difference Report function is not important for this example since it does not generate a report in Test mode. For details on how to enable this function, see [Disable/Enable Difference Report](#) on page 121. This example assumes that the Difference Report has the default setting (not enabled).

1. In Project Explorer, click **Test Mode** . The Test Mode Analysis window opens.
2. Click **Cold Restart All**.
3. Click **Continue**.
4. Double-click **Program2** to display the editor.
5. Select **Motors\_Doors** tab. All variables in **Program2** are listed in the upper pane and the code in the lower pane, see [Figure 42](#).

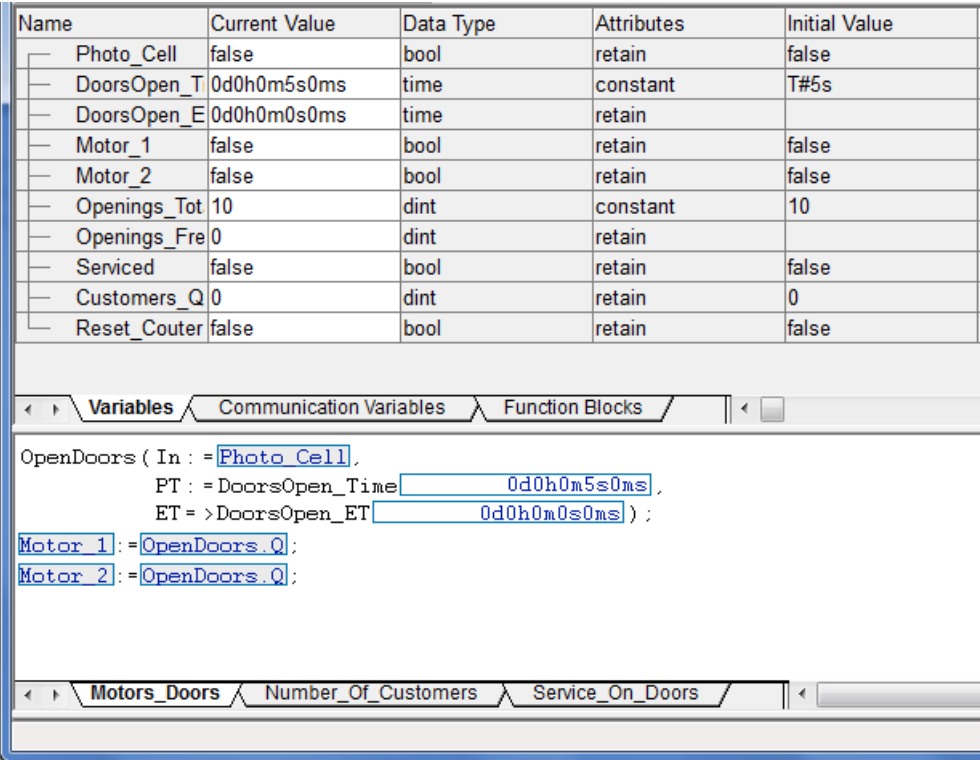


Figure 42. The Program2 editor in Test mode

Analyzing the Code During Program Executions

The test mode helps in testing and analyzing the project, without having a PLC ready in the Project Explorer tree. The variable values can be changed to study the program response.



Analyzing the variable conditions requires right-clicking of variables. These can be accessed either from the parameter list or from the code pane.

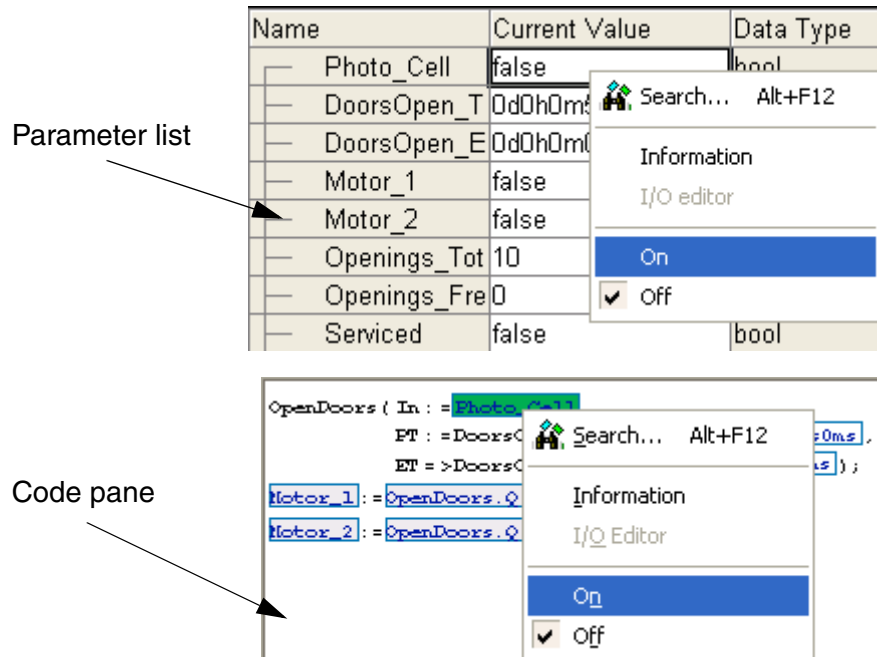


Figure 43. Changing the current value on a variable from the Parameter list, or in the code pane

1. Right-click `Photo_Cell` and select **On** in the context menu.

The motors change to True (start) and the number of openings since last service increases by one, as does the number of customers.

2. Right-click `Photo_Cell` and select **Off** in context menu.

Simulating that no customer is activating the photocell. The clock starts and counts up to five seconds at which point the motors are set to False (stop) and the doors close.

3. Right-click `Photo_Cell` and select **On**, then QUICKLY select **Off** again.

Simulating that a customer has activated the photocell. The number of openings is increased, as does the number of customers.

4. Wait until the doors close. Right-click `Photo_Cell` and QUICKLY select **On, Off, On, Off, On, Off**.

Simulating that three customers are passing the photocell one by one. Notice that the clock starts when the first customer passes the photocell and resets to 0 when the next customer passes. Consequently, the opening time is extended for a new period of 5 seconds, and so on. The number of times the doors open increases by one, whereas the number of customers increases by three.

This results in three openings of the doors and five customers.

5. In the variables list, right-click `Reset_Counter` and select **On**, then select **Off** again. Reset the customer counter as well.
6. Activate the photocell so that when the number of openings (`Openings_Freq`) passes, `Openings_Total`. `Service_Req` becomes *True*.
7. Right-click `Serviced` and select **On**, then select **Off** again.  
Study the reaction of the counters and flags. The variable `Openings_Freq` resets.
8. Close Program editor.
9. From Control Builder Menu bar, select **Tools > Stop Test Mode**.

## Project Examples in Control Builder

All the examples are Read-only, which means they cannot be altered. To study these, see [Opening the ShopDoors Example \(Read Only\)](#) on page 77.

If these are used as a template or a skeleton for other projects, then the projects attribute needs to be changed from read only as described below.

### Preparing the ShopDoors Example for a Project

From the disk where Compact Control Builder<sup>1</sup> is installed:

1. Start the Search tool in Windows and search for *ShopDoors\_ST.prj* . The project file is located inside the example folder for the ShopDoors project.
2. Copy the complete **ShopDoors\_ST** folder into the **Projects** folder<sup>2</sup>.
3. Right-click the **ShopDoors\_ST** folder and select **Properties** from the context menu.
4. Clear the folders properties Attribute Read-only. Apply changes to the folder, sub folder and files.
5. Follow the next steps under [Opening a Project](#) on page 76, to open the converted ShopDoors\_ST project.

---

1. Compact Control Builder is normally installed on the Local Disk (C:)

2. The Projects folder is normally located at; C:\ABB Industrial IT Data\Engineer IT Data\ Control Builder AC 800M\Projects

## Opening a Project

1. From the Project Explorer, select **File > Open Project**. An Open Project window opens.

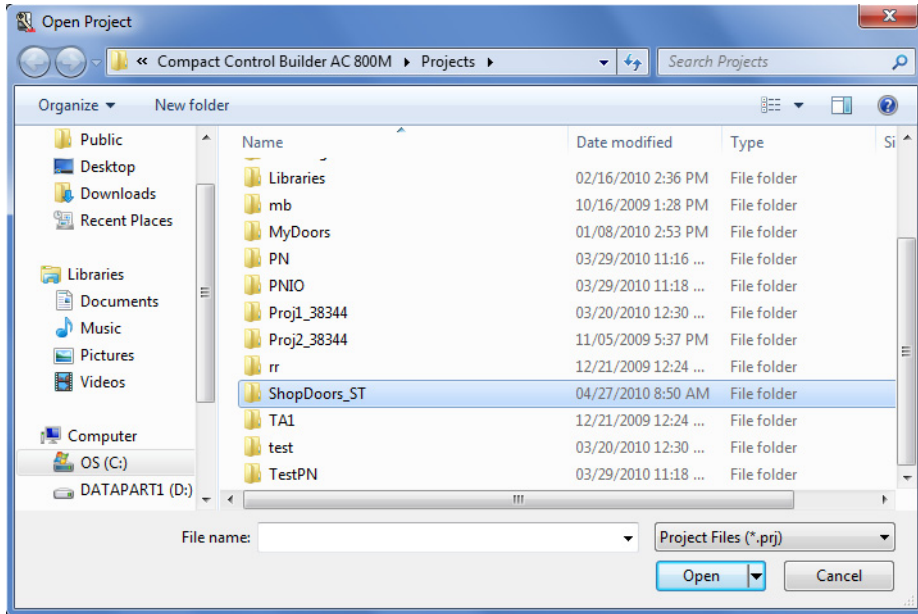
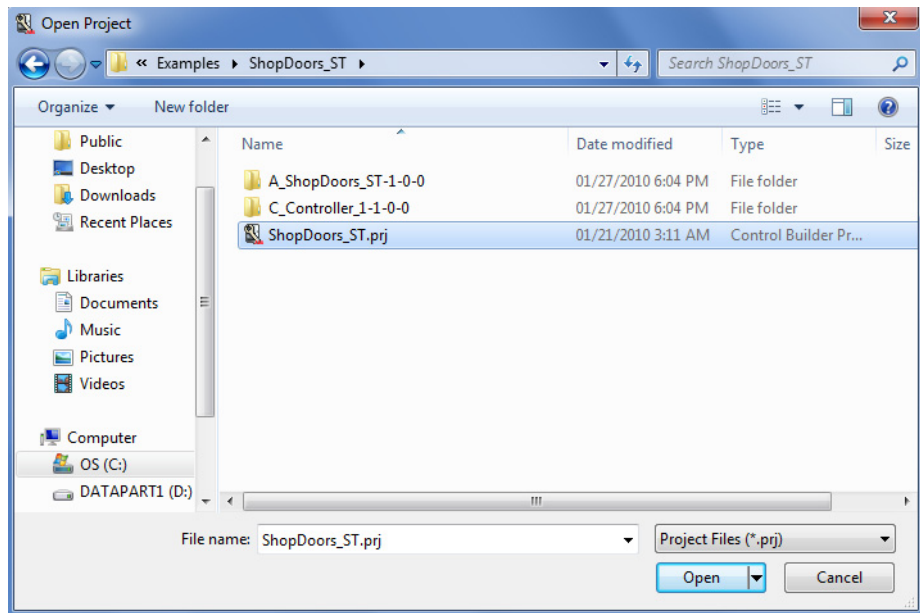


Figure 44. The Open project window

2. Select the **ShopDoors\_ST** folder and click **Open**. The project ShopDoors\_ST folder opens.
3. Select the **ShopDoors\_ST.prj** file and click **Open**. The Project Explorer opens the ShopDoors\_ST project.

**Opening the ShopDoors Example (Read Only)**

1. From the Project Explorer, select **File > Open Project**. An Open Project window opens
2. In the Open Project window, go to the **Examples** folder in the Compact Control Builder installation folder in the drive (typically, C:\Program Files\ABB Industrial IT\Engineer IT\Compact Control Builder AC 800M 5.1).
3. Open the ShopDoors\_ST folder and select **ShopDoors\_ST.prj**.



*Figure 45. Selecting ShopDoors\_ST project (Read Only)*

4. Click **Open**. The ShopDoors example (read only) opens in Project Explorer.



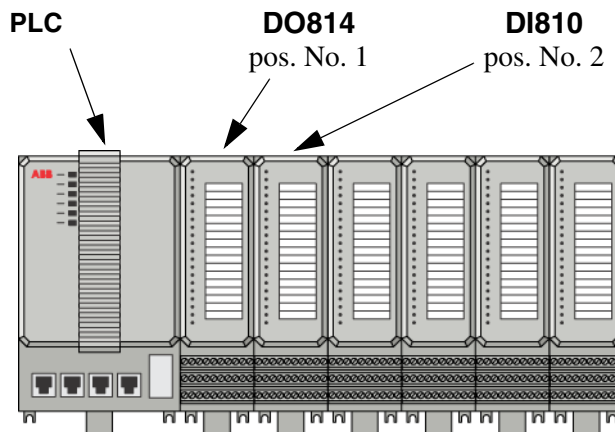
## Section 5 Hardware Configuration

This section describes about adding or removing hardware units from the tree structure in the Project Explorer. It covers the necessary steps for building a software model that represents a limited part of a hardware configuration in the plant.

The Controller object in Project Explorer is the root to which hardware objects are added.

### Configure Hardware

Study the hardware configuration in [Figure 46](#). Assume a PLC, together with six I/O modules. In this example, two of them are added to the tree structure in Project Explorer. Add DO814 and DI810, at positions 1 and 2 respectively.



*Figure 46. Hardware position for IO modules (for example DO814 at position 1 and DI810 at position 2)*

### Changing a CPU

The CPU connected in the Project Explorer must be same as in the physical PLC, else the application cannot be downloaded to the PLC.

However, if the application is run with a SoftController, the choice of the CPU model is optional.



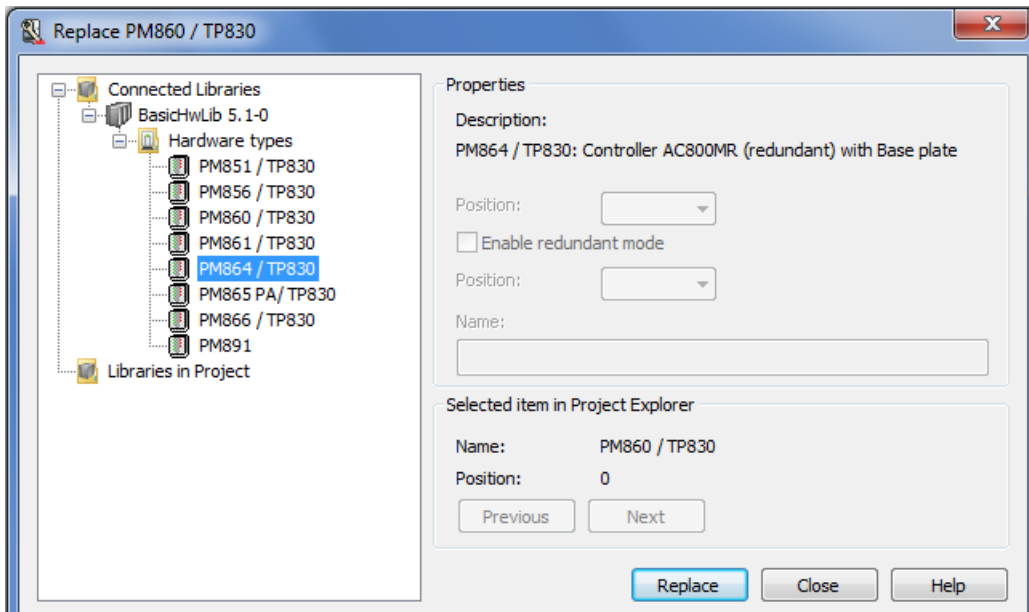
In this example, a default CPU PM860 is replaced with a CPU PM864.



To remove a hardware unit, right-click the object in the tree structure and select Delete.

To replace a CPU:

1. Expand **Controllers > PLC\_1 > Hardware AC 800M** to view the **PM860 / TP830** item at position 0 in the Project Explorer tree.
2. Right-click the **PM860 / TP830** item and select **Replace Unit** in the context menu. A 'Replace' window opens.
3. Expand **Connected Libraries** and select, for example **PM864/TP830**.





- Click **Replace** and then **Yes** to accept the change.

### Adding the IO Modules DO814 and DI810

The S800 IO modules are represented in Control Builder as hardware types located in the hardware library **S800IoModulebusHwLib**. Thus, before adding the IO modules, first insert the hardware library to the project. Once the library is inserted to the project, connect the library to the hardware configuration and then access the IO modules and add them to the controller configuration.

To insert and connect a hardware library:

- Expand **Libraries** folder to see the **Hardware** folder in the Project Explorer tree.

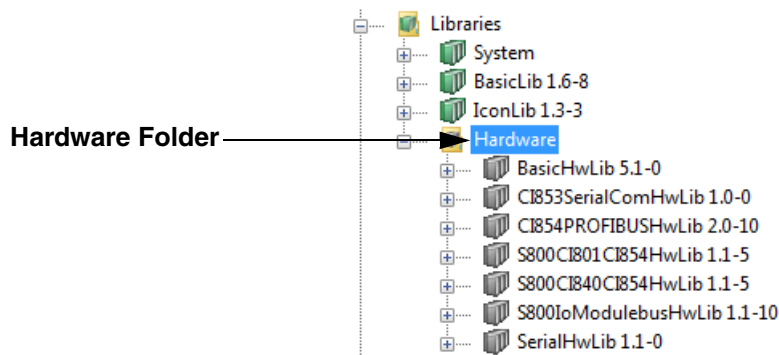


Figure 47. The hardware folder located inside Libraries folder in the Project Explorer



Among the hardware libraries listed under the Hardware folder, the **S800IoModulebusHwLib** library contains S800 IO units for the Modulebus.

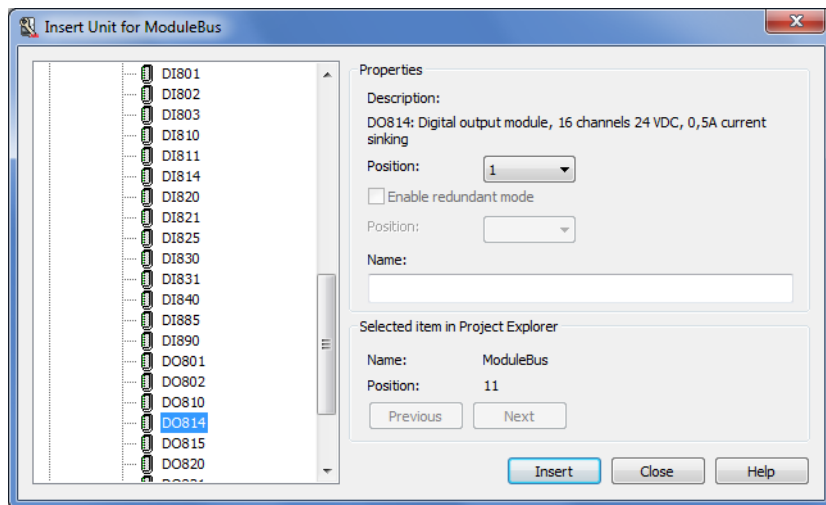
- Expand **Controllers > PLC\_1** to see the **Connected Hardware Libraries** folder in the Project Explorer tree.
- Right-click the **Connected Libraries** folder, select **Connect Library**, and select **S800IoModulebusHwLib** from the window.
- Click **OK**.



Figure 48. The new S800IO library has been connected to the controller

Adding the IO modules from the hardware library:

1. Expand **Controllers > PLC\_1 > Hardware AC 800M > PM860/TP830** to see the **ModuleBus** item in the Project Explorer tree.
2. Right-click the **ModuleBus** item and select **Insert Unit** in the context menu. The 'Insert Unit for ModuleBus' window opens.
3. Expand **Connected Hardware Libraries > S800Io ModulebusHwLib > Hardware types** and select **DO814**.



4. Keep default position **1** from the *Position* drop-down menu and click **Insert**.
5. Select **DI810** from the list.
6. Keep default position **2** from the *Position* drop-down menu and click **Insert**.

When the two IO modules are added, the “hardware tree” looks like the configuration shown in [Figure 49](#).

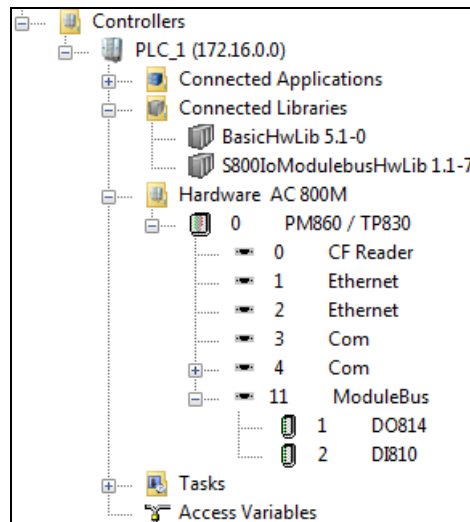


Figure 49. Hardware setup with the new IO Units added on the module bus



Information on an I/O unit, for example DO814 or DI810, can be accessible from the On-line Help; select the I/O unit in the Project Explorer and press F1.

## Connect Variables to I/O Channels

Communication between I/O channels and code is established by connecting variables to I/O channels. Therefore, Control Builder provides two different connection methods. Both the methods can be followed for better understanding.



The MyDoors project is not completed unless both methods are followed.

[Method 1 - Using Dot Notation](#) on page 84 connects the variable Photo\_Cell to the IO module DI810 by using dot notation. [Method 2 - Using a Path Selector](#) on page 84, connects the variables Motor\_1 and Motor\_2, to the IO module DO814 via a path selector menu.

Method 1 - Using Dot Notation

Connecting a Variable to DI810 Channel

From the Controllers in Project Explorer:

- 1. Double-click **DI810** I/O module. The DI810 hardware editor opens.
- 2. Select the **Connections** tab and place the cursor in the first empty white cell.
- 3. Type **A** (for Application\_1) and observe how the editor fills in the rest.
- 4. Press “.” (dot) to move to the next level. All three Programs are displayed.
- 5. Select **Program2**, and press “.” (dot) again. A list of variables opens.
- 6. Select **Photo\_Cell** in the list.
- 7. Press the ENTER key. The **Photo\_Cell** variable has been connected to the first channel in DI810.

Channel	Name	Type	Variable
IXD.11.5.1	Input 1	bool	Application_1.Program2.Photo_Cell
IXD.11.5.2	Input 2	bool	

Settings

Connections

Properties

Status

Unit Status

Figure 50. The variable **Photo\_Cell** connected to first IO Channel


Method 2 - Using a Path Selector

Connecting a Variable to DO814 Channel

From the Controllers in Project Explorer:

- 1. Double-click the **DO814** I/O module. The hardware editor for DO814 opens.
- 2. Select the **Connections** tab and place the cursor in the first empty white cell (Channel QX0.11.3.1 in the **Variable** column).
- 3. Right-click **Insert > Insert Path From Tree** from the context menu. A list box opens.



The Menu bar action **Insert > Insert Path From Tree**, can also be done by clicking the  icon located in the hardware editor, or by pressing CTRL+T inside the cell.

4. Expand **Application\_1 > Program2**. Double-click **Motor\_1** to insert the full path, see [Figure 51](#).

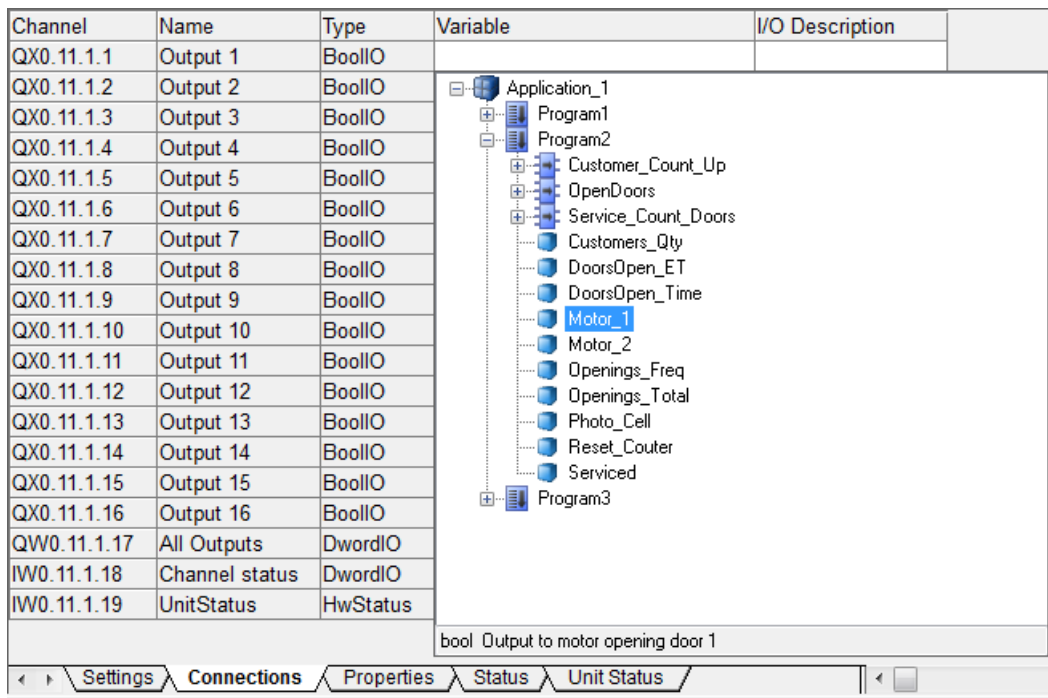



Figure 51. The path for **Motor\_1** variable selected in the tree



5. Place the cursor in the second empty white cell (Channel QX0.11.3.2 in the **Variable** column) and connect **Motor\_2**. Use the  icon.

After connecting the variables, the hardware editor looks as in [Figure 52](#).

Channel	Name	Type	Variable	I/O
QXD.11.1.1	Output 1	bool	Application_1.Program2.Motor_1	
QXD.11.1.2	Output 2	bool	Application_1.Program2.Motor_2	
QXD.11.1.3	Output 3	bool		
QXD.11.1.4	Output 4	bool		

At the bottom of the table are tabs: Settings, **Connections**, Properties, Status, Unit Status.

Figure 52. The **Motor\_1** and **Motor\_2** connected to DO814

- 6. Click **Check**  to check the declaration.
- 7. Click **Save and Close** .

Reading I/O addresses from the Application

An easy way to read the I/O address is to open **Program2** in the program editor and check the column labeled **I/O Address**. This column shows the address for the photocell and the motors, as shown in [Figure 53](#).

	Name	Data Type	Attributes	Initial Value	I/O Address	
1	Photo_Cell	bool	retain	false	Controller_1.0.11.2.1	DI810
2	DoorsOpen_Time	time	constant	T#5s		
3	DoorsOpen_ET	time	retain			
4	Motor_1	bool	retain	false	Controller_1.0.11.1.1	DO814
5	Motor_2	bool	retain	false	Controller_1.0.11.1.2	
6	Openings_Total	dint	constant	10		
7	Openings_Freq	dint	retain			

*Figure 53. The I/O Address column shows how variables are connected to I/O channels*

Changes made to I/O connections in the hardware editor are reflected in both editors.

The project has now been tested offline and the hardware configuration is complete.

## Section 6 Connecting the PLC and Go Online

This section describes the prerequisites for connecting a PLC and the general procedure for downloading a project to the PLC.

Before completing the MyDoors project by downloading the application to a PLC, create a project according to [Section 4, MyDoors Project](#). Then, follow the instructions in [Section 5, Hardware Configuration](#).

If there is no access to a PLC or IO modules, use a **SoftController**. In that case, refer to [Setting the System Identity in Control Builder](#) on page 94.

### Firmware Upgrade

The PLC firmware version and the Compact Control Builder version must be identical. If the firmware version is different, perform the steps in this section to upgrade the firmware.



Firmware upgrade can be performed from the Compact Control Builder via the Ethernet network (see the Control Builder Online Help). The Serial line upgrade procedure is also available.



Serial Firmware Upgrade Tool cannot be used for firmware upgrade of PM891. The firmware upgrade of PM891 can be done using an SD card or from the Remote System dialog in Control Builder.

#### Firmware Upgrade via the Serial Cable (TK212A)

1. Connect the serial cable between the Control Builder PC and the PLC, as specified in [Table 3](#). For the type of cable, see [Appendix D, Communication Cables](#).

Table 3. Cable connection for the PLC

PLC	Tool Port	Connector	Cable Name
AC 800M	COM 4	RJ 45	TK212A



During the upgrade, do not run any program capable of blocking the selected COM port. This applies in particular to the MMS Server program.

2. Turn on the power to the PLC.
3. From the Windows Start menu select **All Programs > ABB Industrial IT AC 800M > Utilities > Serial Firmware Upgrade**. The following dialog box appears.

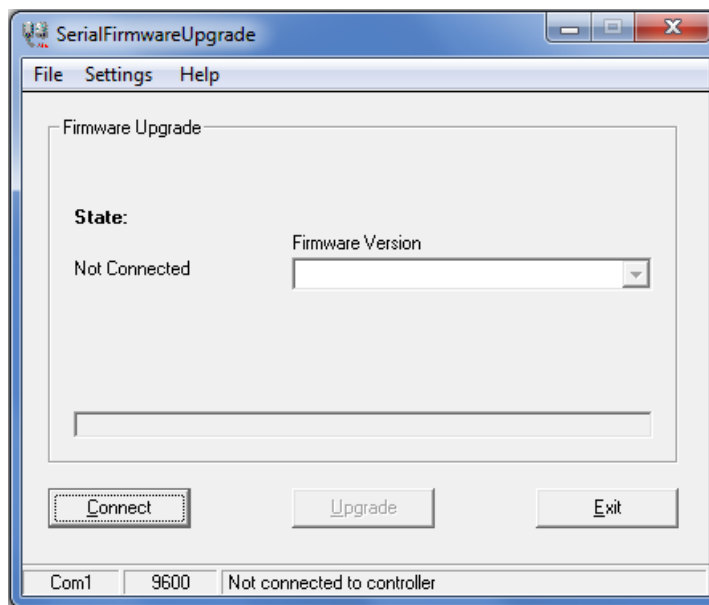


Figure 54. The Serial Firmware Upgrade dialog box

4. Select **Settings > COM Port** from the drop-down menu. Make sure the settings correspond to the physical COM port (on the PC) to which the cable is connected.



- Click **Connect** button and press the **Init** push-button on the PLC until the **Run** LED starts to blink. Wait for a minute until a message appears. If the connection is successful, a confirmation text appears in the Firmware Version text field (Figure 55).

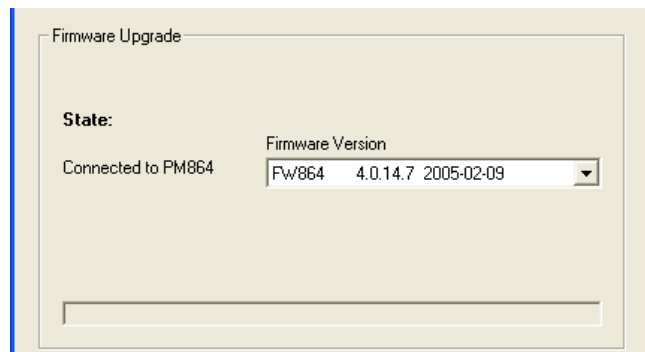


Figure 55. A Firmware version displayed in the text field



If an error message “Connection failed” appears, check the cables and repeat the steps again.

- Select Firmware version<sup>1</sup> from the drop-down menu and click **Upgrade**. File transmission starts to the PLC. A confirmation window opens when the PLC is upgraded, see Figure 56.

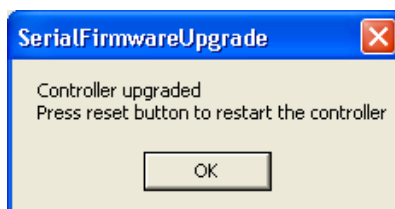


Figure 56. The Serial Firmware Upgrade window

- Click **OK**.
- Click **Exit**.
- Press the **Init** push-button on the PLC until the **Run** LED starts to blink.

---

1. The firmware version must be supported by the installed Control Builder version.

## Setting an IP Address

A unique PLC IP address must be set to avoid conflict with other devices on the Control Network. This subsection describes setting up an IP address for the PLC using the serial cable (TK212A), without any connection to the network. Furthermore, it provides instructions to setup the PC that runs the Control Builder. However, these instructions are strictly Microsoft Windows specific. A configuring tool, named IPConfig, is used to set the IP address for the PLC.

### Setting IP Address for PLC

#### Preparations

Connecting the cable between the Control Builder and the PLC are exactly the same as described in [Firmware Upgrade](#) on page 87.

1. Connect a serial cable between the Control Builder PC and the PLC, as specified in [Table 4](#). For the type of cable, see [Appendix D, Communication Cables](#).

*Table 4. Cable connection for the PLC*

PLC	Tool Port <sup>(1)</sup>	Connector	Cable Name
AC 800M	COM 4	RJ 45	TK212A

(1) The tool port COM4 is part of PLCs PM85x and PM86x. COM4 is not part of PM891 PLC.



During the upgrade, any program that is capable of blocking the selected COM port should not be run. This applies in particular to the MMS Server program.

2. Turn on the power to PLC.

### Starting the IPConfig Tool

1. From the Windows Start menu select **All Programs > ABB Industrial IT AC 800M > Utilities > IPConfig**. An IP Config dialog opens.

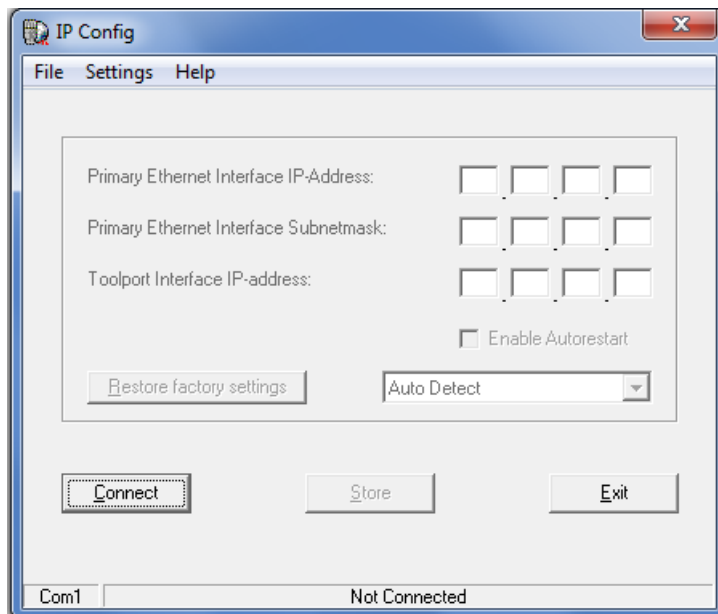


Figure 57. The IP Config dialog box

2. Click the **Connect** button and press the **Init** push-button on the PLC until the **Run** LED starts to blink. Wait for a minute until a message appears, see [Figure 58](#).

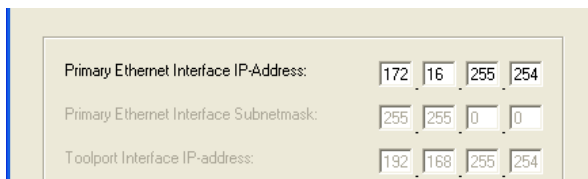


Figure 58. The IPConfig dialog box with factory default setting



If the error message “Connection failed” appears, check the cables and repeat the steps again.

3. From the IP Config dialog menu, select **Settings > Advanced Mode**.
4. Enter a unique IP address (obtainable from the Control Network Administrator). Example: 172.16.84.124, see [Figure 59](#).

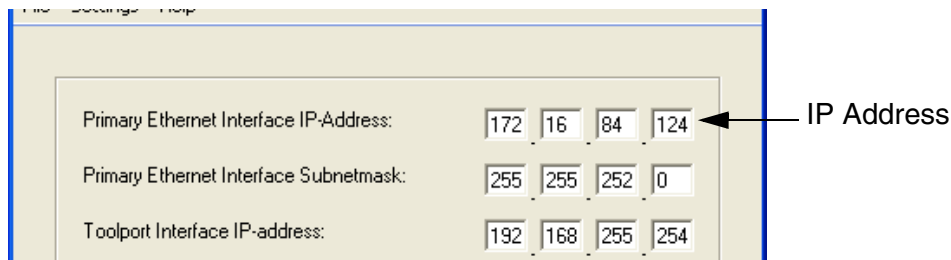


Figure 59. IP Config window for setting unique IP address

5. Type Primary Ethernet Interface Subnetmask (255.255.252.0) and click **Set IP**. The new address is sent to the PLC and an IP Config window opens, see [Figure 60](#).

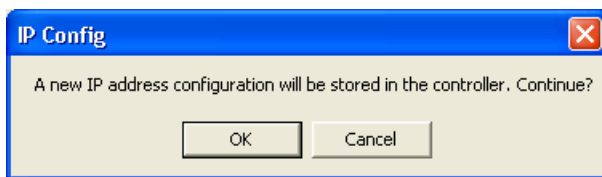


Figure 60. IP Config confirmation dialog window

6. Click **OK**.
7. Press the **Init** push-button on the PLC until the **Run** LED starts to blink. The new IP address is not valid until the PLC is restarted.

## Setting IP Address for PC

The following instructions help in setting up the IP address (in Windows) for the Control Builder PC.

1. Go to **Start > Control Panel > Network and Internet > Network and Sharing Center**.
2. Select **Change Adapter Settings**.



The Control Builder uses the IP address of the last network adaptor. Hence, setup the correct order of network adapters in the Control Builder PC, so that Control Network adaptor comes last.

3. Right-click **Local Area Connection** and select **Properties**. The Local Area Connection Properties dialog opens.
4. Select **Internet Protocol Version 4 (TCP/IPv4)** and click **Properties**. The 'Internet Protocol Version 4 (TCP/IPv4) Properties' dialog opens.
5. Select **Use the following IP address**.



The PC and PLC NetID must be the same for the first three positions (start from left to right). For example, if the PLC has the IP address 172.16.84.124, then the PC must have the IP address 172.16.84.Q. The number represented by Q must not be same as in the PLC (124 in this example).

6. Enter an IP address, in this example (172.16.84.120) and then enter Subnet mask (255.255.252.0).
7. Click **OK** to close all dialog windows.
8. Connect a network cable. The port and channel positions are shown in [Table 5](#).



To check that the IP configuration works; open the command prompt DOS window and ping the PLC by writing the following command: ping 172.16.84.n, where "n" is the address selected for the PLC.



If the PLC is to be connected to a PC via a switch or hub, then a *straight-through* Ethernet cable should be used. If there is a direct connection between the PLC and the PC, then use a *cross-over* Ethernet cable.

Table 5. Channel positions for connecting the Ethernet cable in the PLC

PLC	Communication Interface	Position	Channel
AC 800M	Built-in	-	CN1



CN2 port on the PLC must not be connected to the network. This port is used for connecting the PLC to a secondary network.

## Downloading the Project via Ethernet

The connection with the PLC is established, once the upgraded firmware (Serial Firmware Upgrade) is downloaded and the IP addresses are given. This means that the application is ready to download projects to the PLC and Go Online.

### Setting the System Identity in Control Builder

To download projects to the PLC, first set the system identity in Project Explorer.

#### Setting the IP address for PLC\_1

1. In the Project Explorer, expand **Controllers**.

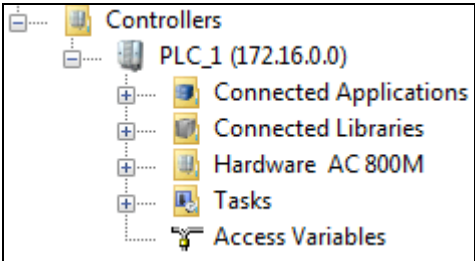


Figure 61. The Controllers expanded in Project Explorer

2. Right-click **PLC\_1** and select **Properties > System Identity** from the context menu. The System Identity window opens.

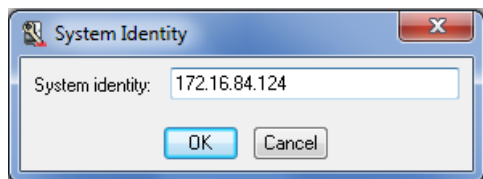


Figure 62. The System Identity window for setting the IP address

3. Enter the IP address of the PLC (for example: 172.16.84.124) and click **OK**. The System Identity window closes.



If the application is run with a **SoftController**, then enter the computer's IP address and add a colon and the digit 2. Example: 10.46.35.117:2

4. Expand **Hardware AC 800M** to view **1 Ethernet**. Right-click the **Ethernet** icon (at position 1) and select **Editor** to open the editor.
5. Select the **Settings** tab (lower left corner, see Figure 63) and enter the IP address in the IP address Value field.



Note that the IP address of the first Ethernet port has to be the same as the IP address of the PLC (system identity). The second Ethernet port (at position 2) is only used if the PLC is connected to a redundant network. For more information about redundant networks, study the subsection [Setting Up Redundant Network](#) on page 123

Parameter	Value	Type	Unit	Min	Max
IP address	172.16.84.124	string			
IP subnet mask	255.255.252.0	string			
Network Area	0	dint		0	35
Path Number	0	dint		0	3
Node Number	0	dint		0	500
Network Area Local	false	bool			
Send Period	1	dint	s	1	60
Max Lost Messages	3	dint		1	10
Proxy router	0.0.0.0	string			
Target address	0.0.0.0	string			

Settings Connections Unit Status

Figure 63. IP address for PLC Ethernet port at position 1 (in this case 172.16.84.124 which is the same IP address as given in System Identity)

Click **Save and Close** .

## Downloading the Project to a PLC

Ensure that the project has no errors, before downloading the application to a PLC.



If the application is run with a SoftController, see [Downloading to a SoftController](#) on page 97.



Ensure that the PLC and all other hardware units have the right firmware. For instructions on how to check and upgrade firmware versions, see [Firmware Upgrade](#) on page 87.

### Downloading to the PLC

The following instructions address the project MyDoors, which was previously created in [Section 4, MyDoors Project](#). However, these instructions are common for downloading any project application.

1. Ensure that the MyDoors project is in Offline mode.
2. Click **Download Project and Go Online** . The Online analysis window opens.
3. Click **Cold Restart All**.
4. Click **Continue**.
5. In the Task Analysis dialog, click  to accept the download, and go online.

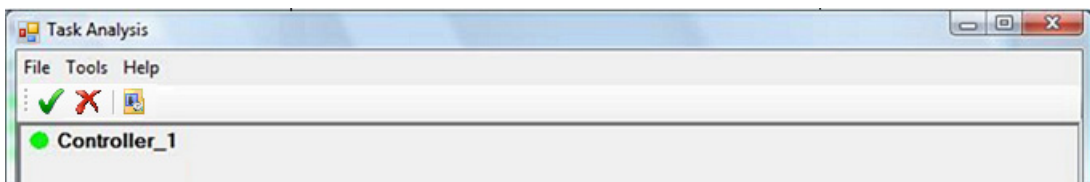


Figure 64. Task Analysis dialog



### Downloading to a SoftController

Make sure that the project (in this example, MyDoors) is in Offline mode (not running in test mode).

1. Enable the Hardware Simulation feature in Control Builder. Go to **Tools > Setup > Station > Application Download** to open the Setup - Application Download dialog, and then set the parameter *HWSimulationAllowed* to **true**.
2. From Project Explorer, expand the Controllers folder.
3. Right-click **PLC\_1** and select **Simulate Hardware** from context menu.
4. Start the SoftController. Double-click the SoftController icon on the desktop (if desktop shortcut is selected during installation), or from the Start menu on the Windows Task Bar: **Start > All Programs > ABB Industrial IT AC 800M > SoftController > SoftController**. The SoftController start panel opens.
5. Click the **Start** button. The Status field displays *Started* and the SoftController starts.

From Project Explorer:

6. Click **Download Project and Go Online** . The Online analysis window opens.
7. Click **Cold Restart All**.
8. Click **Continue**.

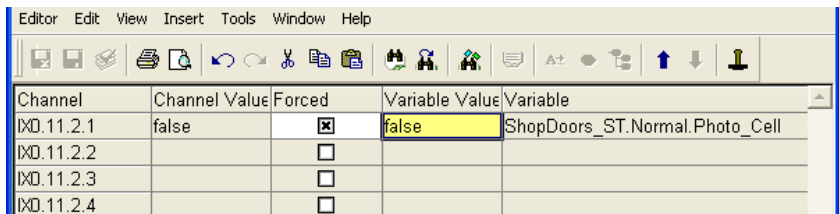
## Test the Program Online

This subsection describes how to force the variable Photo cell that is connected to the IO unit DI810 in the MyDoors sample project.

The *forcing* function can be used to activate/deactivate an I/O.

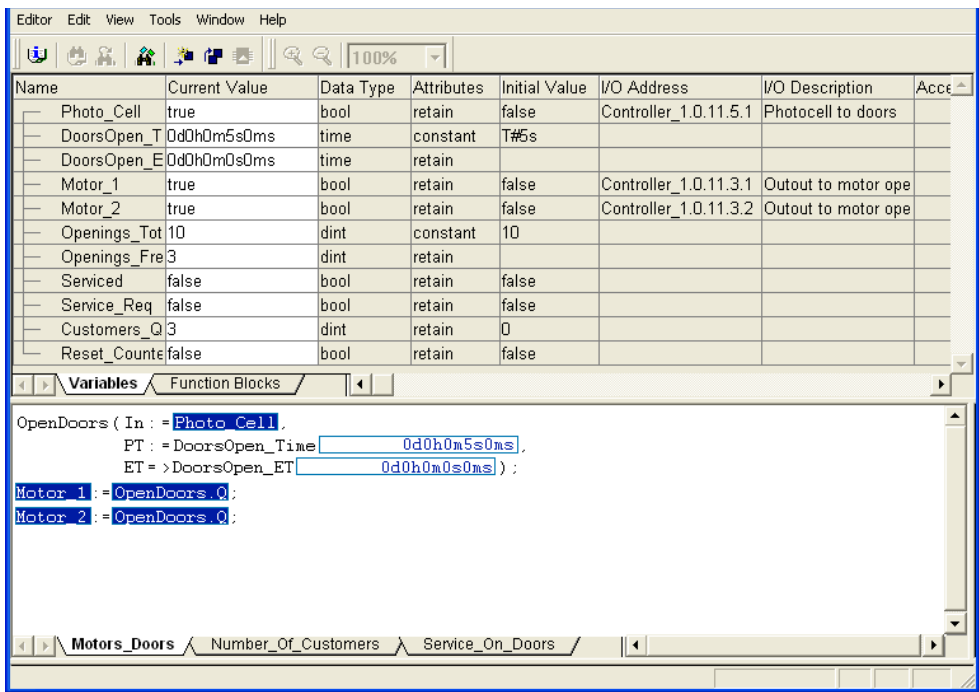
1. In Project Explorer, right-click **Program2** and select **Online Editor** to open the online editor.
2. Right-click I/O module **DI810** and select **Editor**.
3. In the Status tab, check the box in the **Forced** column, see [Figure 65](#). Change the variable `Photo_Cell` value to 1 (true), return quickly to 0 (false) and

inspect the motor’s values in the online editor (values change to 1 for five seconds and then return to 0).



Channel	Channel Value	Forced	Variable Value	Variable
I0.11.2.1	false	<input checked="" type="checkbox"/>	false	ShopDoors_ST.Normal.Photo_Cell
I0.11.2.2		<input type="checkbox"/>		
I0.11.2.3		<input type="checkbox"/>		
I0.11.2.4		<input type="checkbox"/>		

Figure 65. The status of the photocell and the motors can be forced in the I/O editor



Name	Current Value	Data Type	Attributes	Initial Value	I/O Address	I/O Description	Access
Photo_Cell	true	bool	retain	false	Controller_1.0.11.5.1	Photocell to doors	
DoorsOpen_T	0d0h0m5s0ms	time	constant	T#5s			
DoorsOpen_E	0d0h0m0s0ms	time	retain				
Motor_1	true	bool	retain	false	Controller_1.0.11.3.1	Outout to motor ope	
Motor_2	true	bool	retain	false	Controller_1.0.11.3.2	Outout to motor ope	
Openings_Tot	10	dint	constant	10			
Openings_Fre	3	dint	retain				
Serviced	false	bool	retain	false			
Service_Req	false	bool	retain	false			
Customers_Q	3	dint	retain	0			
Reset_Counte	false	bool	retain	false			

Variables

Function Blocks

```
OpenDoors ( In : = Photo_Cell ,
            PT : = DoorsOpen_Time 0d0h0m5s0ms ,
            ET = >DoorsOpen_ET 0d0h0m0s0ms ) ;
Motor_1 := OpenDoors.Q ;
Motor_2 := OpenDoors.Q ;
```

Motors\_Doors | Number\_Of\_Customers | Service\_On\_Doors

Figure 66. The changing motor values can be inspected in the online editor of the program

The MyDoors project is now completed.

## Appendix A Compact Control Builder AC 800M Settings

This appendix contains information about the Setup Wizard for Compact Control Builder. If a single-user configuration is set up with minor adjustments, then the wizard guides the user through the settings. However, if a multi-user configuration is to be setup, then the file path in the last dialog, 'File Location' (under Product settings), needs to be changed.

This appendix contains:

- [Starting the Setup Wizard for Compact Control Builder](#) on page 100.
- [Product Settings for Compact Control Builder](#) on page 100.
- [Multi-User Configuration](#) on page 102.
- [Download](#) on page 115.
- [Disable/Enable Difference Report](#) on page 121.



If the default settings for a single-user configuration are to be used, there is no need to configure the settings in the two Setup Wizards in this appendix.



Exclude the ABB Industrial IT Data folder and any used shared network disk from a virus scan, if the files are scanned at access. The user must configure the anti-virus program to scan these files and folders on demand or at a scheduled scan.

### Starting the Setup Wizard for Compact Control Builder

From **Start** select **All Programs > ABB Industrial IT AC 800M > Utilities > Setup Wizard**.



Some tabs have an Apply button. Specified settings are not implemented until this button is clicked.

All *Setup Wizard* dialog boxes contain a Show Settings button. Click this button to open a log file on screen containing all available Wizard settings. It also contains a list of system environment variables.

## Product Settings for Compact Control Builder

### Language

The language to be used for the programming tool, the libraries, and the Online Help files is selected under the Language tab (see [Figure 67](#)). By default, the installation program selects English.

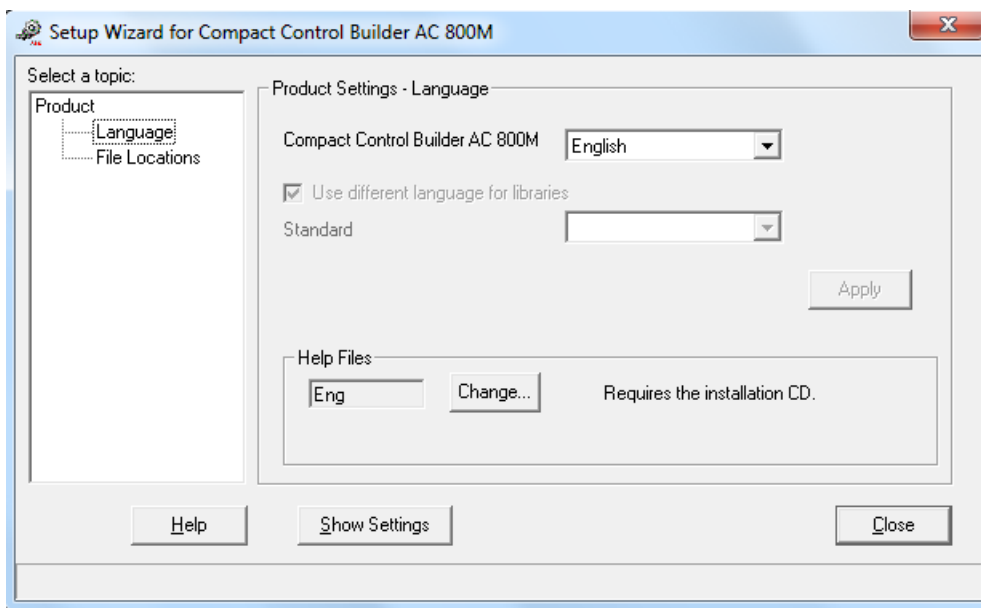


Figure 67. The Product setting dialog for Language

## File Locations

Compact Control Builder has three file locations which can be managed by the Setup Wizard. The Working folder and the MMS server working folder are handled by the system, thus should never be modified. The Project folder should only be changed, if the Compact Control Builder station should be part of a multi-user configuration.

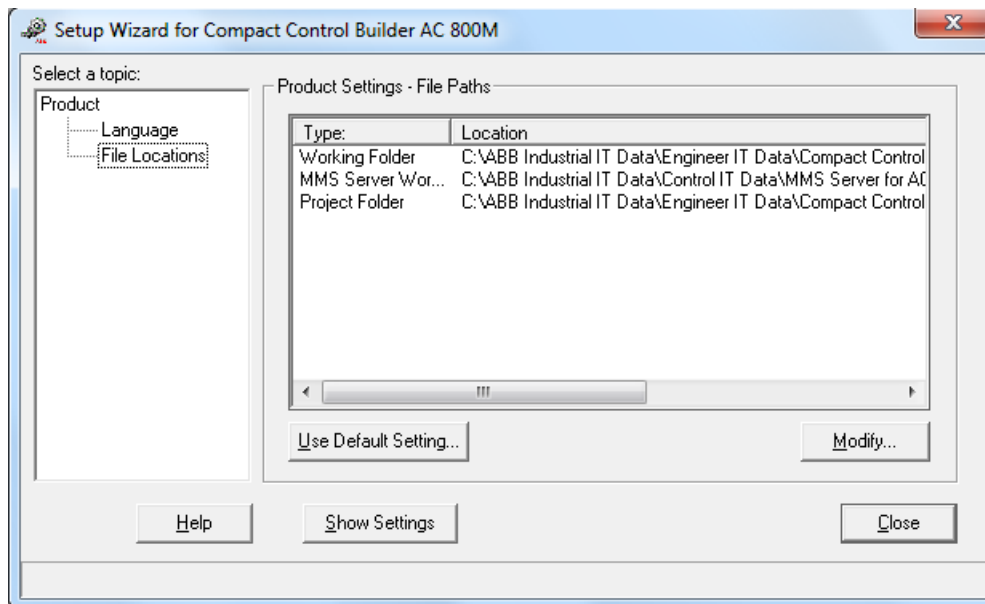


Figure 68. Product topic for file locations.

- Working folder; contains Compact Control Builder station log files, settings etc.
- MMS Server working folder; contains MMS Server log files.
- Project folder; contains projects.

Clicking *Use Default Settings*, resets the manual settings to the general default settings.



How to change the file location for the Project Folder, see [Multi-User Configuration](#) on page 102.

## Multi-User Configuration

The Compact Control Builder must be installed on every PC before setting up a multi-user environment. The OPC Server can be installed on one of the Compact Control Builder stations or on a standalone PC. All PCs must be connected to the same network.

A multi-user configuration requires that all Compact Control Builder stations and an OPC Server have access to the common project files<sup>1</sup>. During multi-user engineering, all Control Builder stations must write/read engineering changes to the common project files located in a shared project folder.

The OPC Server reads run-time data directly from the PLC over the network, by accessing the configuration data located in the common project folder. Thus, the project folder must be both shared and placed on a network server, before the team starts multi-user engineering.

Multi-user configuration consists of the following sections:

- [Creating a Shared Project Folder](#) on page 103.
- [Setting Up Compact Control Builder Stations](#) on page 106.
- [Setting Up OPC Server](#) on page 108.
- [Configuration Example](#) on page 112.
- [Guide lines for Multi-User Engineering](#) on page 114.

---

1. A Compact Control Builder project contains several files. These project files hold configuration data for libraries, applications, hardware, project constants etc.

## Creating a Shared Project Folder

Select a PC station as the network File server (from now on MyServer). Start Windows and login with administrator role.

1. Create a project folder (from now on **AC800Mprojects**) on MyServer.
2. In Windows Explorer, right-click **AC800Mprojects** and select **Properties** from the context menu. A Properties dialog opens.
3. Select the **Sharing** tab.

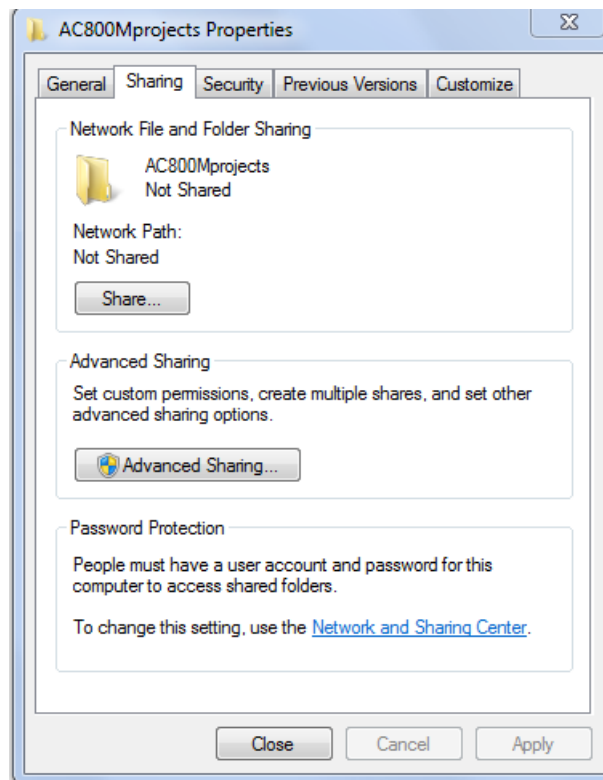


Figure 69. Properties dialog with the Sharing tab active.

4. Click **Advanced Sharing...** to open the Advanced Sharing dialog.
5. Select the **Share this folder** check box.

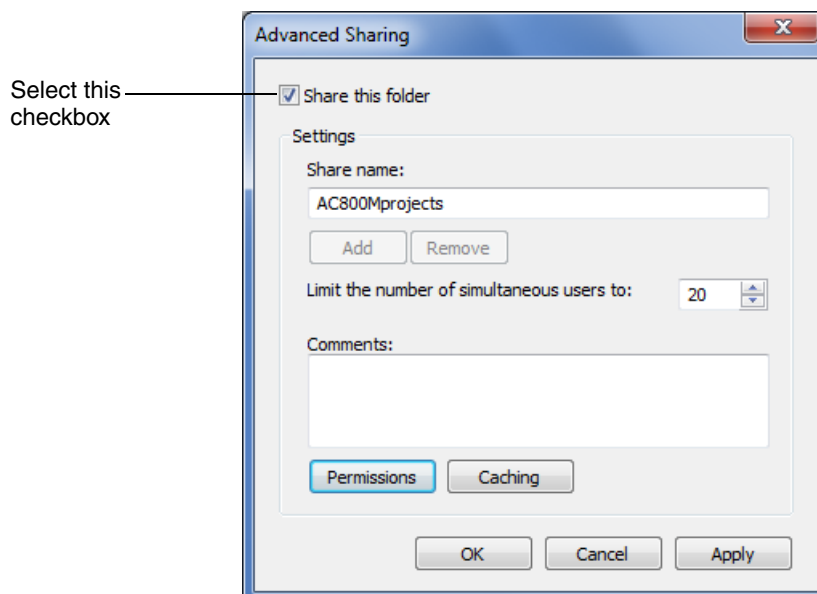


Figure 70. Advanced Sharing dialog

6. Click **Permissions** to open the Permissions for AC800Mprojects dialog.
7. Select the Everyone group, and select the **Allow** checkbox corresponding to Change.



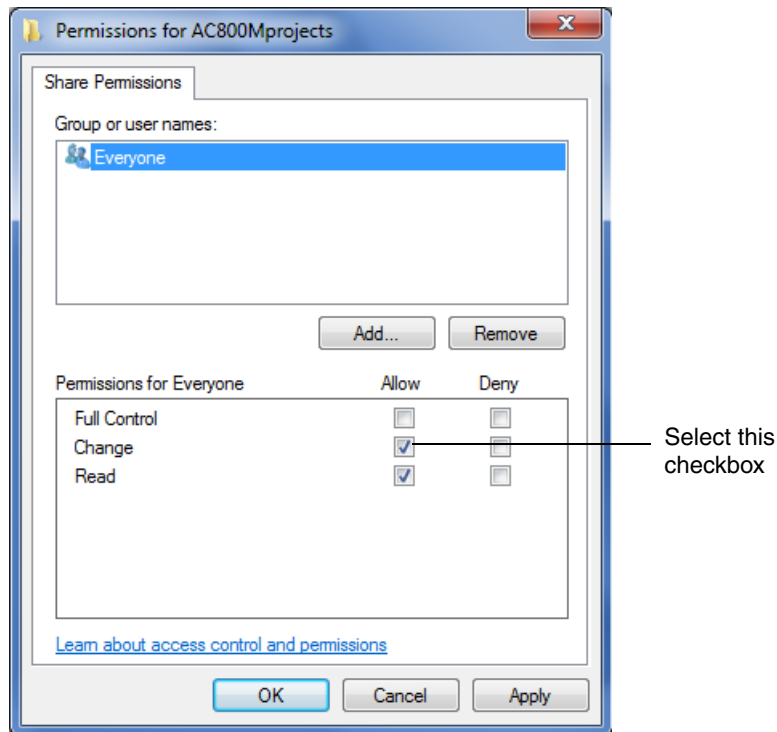


Figure 71. Permissions for Everyone

8. Click **Apply** and **OK**.
9. Click **OK**.
10. Click **Apply** in the Properties dialog.



It is preferable to create a new user group (for example MyTeam) in Windows and add the project members to MyTeam group. Then, select the permission as **Change** for MyTeam, instead of for Everyone.

## Setting Up Compact Control Builder Stations

Ensure that the Compact Control Builder is installed and the PC station is connected to Ethernet. For installation instructions see [Section 2, Installing Software](#).

1. Start the **Setup Wizard** according to [Starting the Setup Wizard for Compact Control Builder](#) on page 100.
2. Select **File Locations > Project Folder**, and click **Modify** to open the Browse to Project Folder dialog.

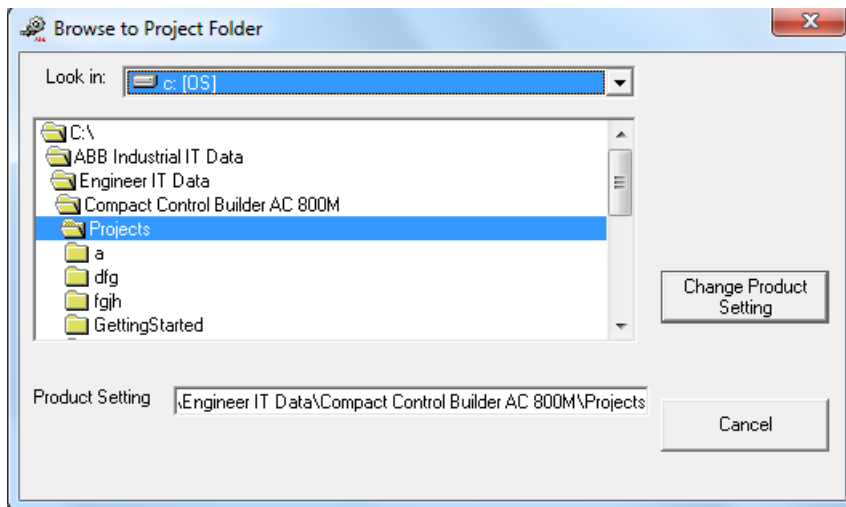


Figure 72. Browse to Project Folder dialog for locating the shared project folder



Before browsing to the (UNC) path, map a network drive in Windows. If the browse dialog does not permit browse navigation on network places, type the path in the Product Setting field ([Figure 72](#)).

The path must be specified with an UNC path that contains the server name and the shared folder name \\MyServer\AC800Mprojects.



Exclude the ABB Industrial IT Data folder and any used shared network disk from a virus scan, if the files are scanned at access. Configure the anti-virus program to scan these files and folders on demand or at a scheduled scan.

3. Locate the network File server and browse to the shared **AC800Mprojects** folder.
4. Click **Change Product Setting** to close the Browse to Project Folder dialog.

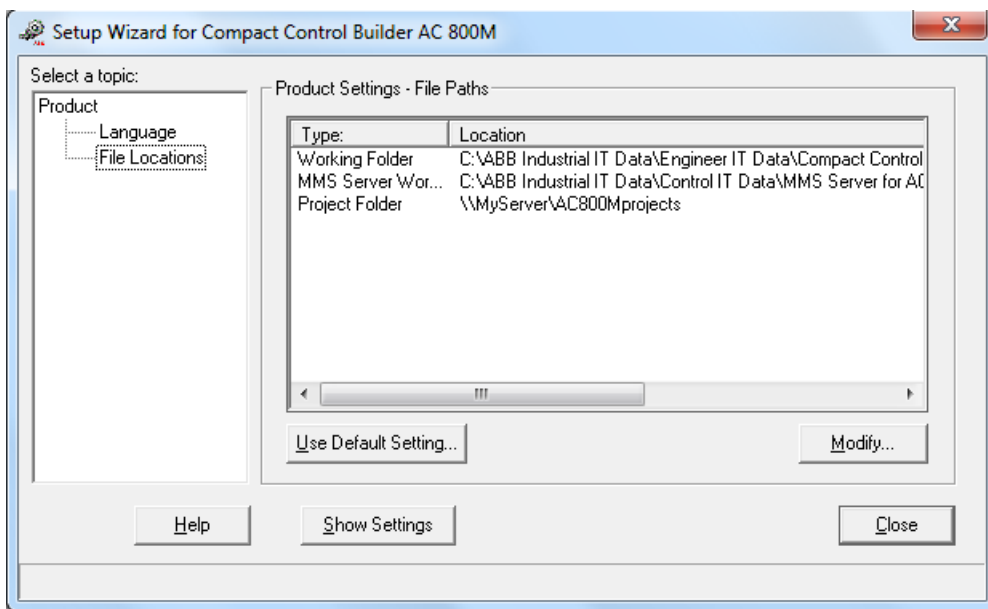


Figure 73. The new file location for the common project folder.



For more information about the differences between the Working folder and Project folder, see [File Locations](#) on page 101.

5. Click **Close** to close the Setup Wizard.
6. Repeat these steps for all Compact Control Builder stations.

## Setting Up OPC Server

An OPC Server configuration for multi-user engineering requires the following:

- The path to the common project folder. This path is to be specified under the topic 'File Locations' in the Setup Wizard.
- A Service Account.

Ensure that the OPC Server is installed. For installation instructions see [Installing the OPC Server for AC 800M](#) on page 27.

1. From **Start** select **All Programs > ABB Industrial IT AC 800M > OPC Server for AC 800M 5.1 > Setup Wizard**.

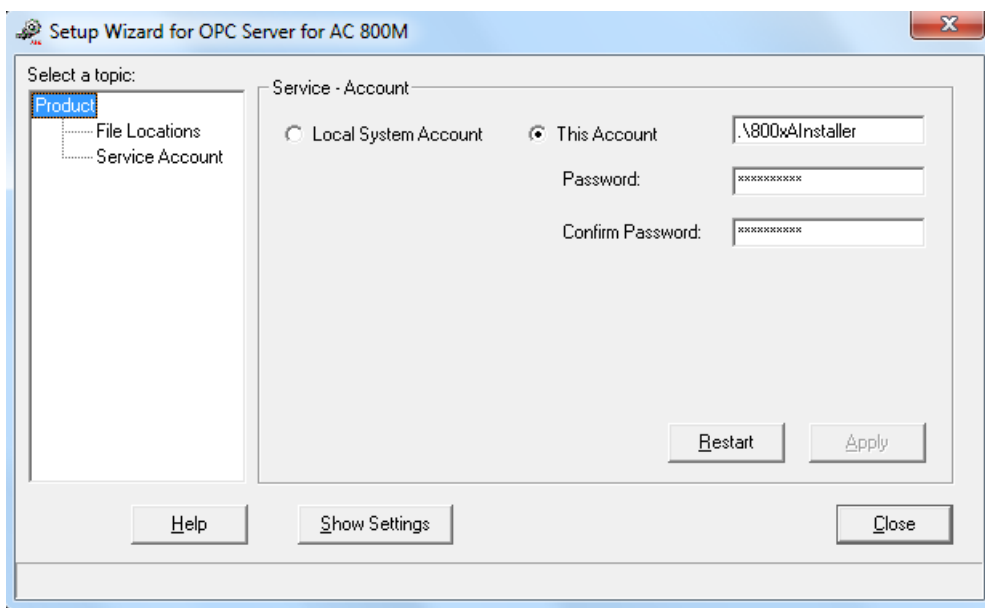


Figure 74. Setup Wizard start menu for OPC Server.

2. Select **File Locations > Project Folder**, and click **Modify** to open the Browse to Project Folder dialog.

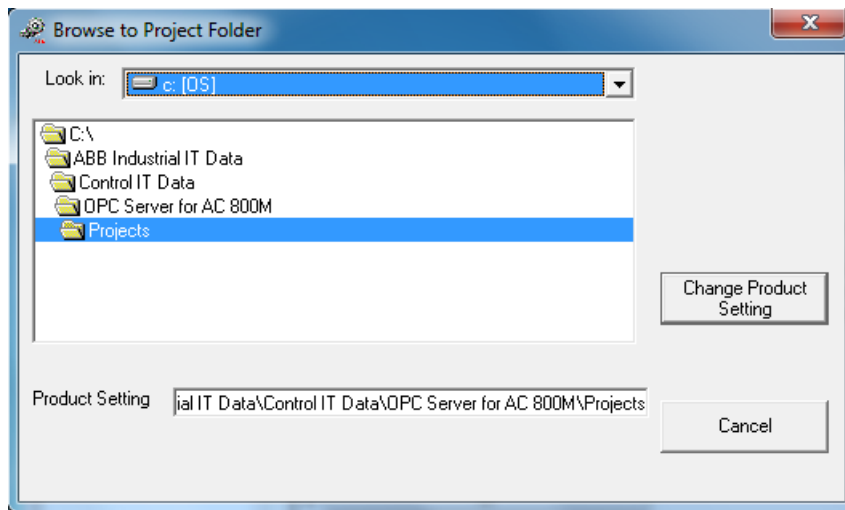


Figure 75. A Browse dialog for locating the shared project folder



Before browsing to the (UNC) path, map a network drive in Windows. If the browse dialog does not permit browse navigation on network places, type the path in the Product Setting field (Figure 75).

The path must be specified with an UNC path that contains the server name and the shared folder name \\MyServer\AC800Mprojects

3. Locate the network File server and browse to the shared **AC800Mprojects** folder.
4. Click **Change Product Setting** to close the Browse to Project Folder dialog.

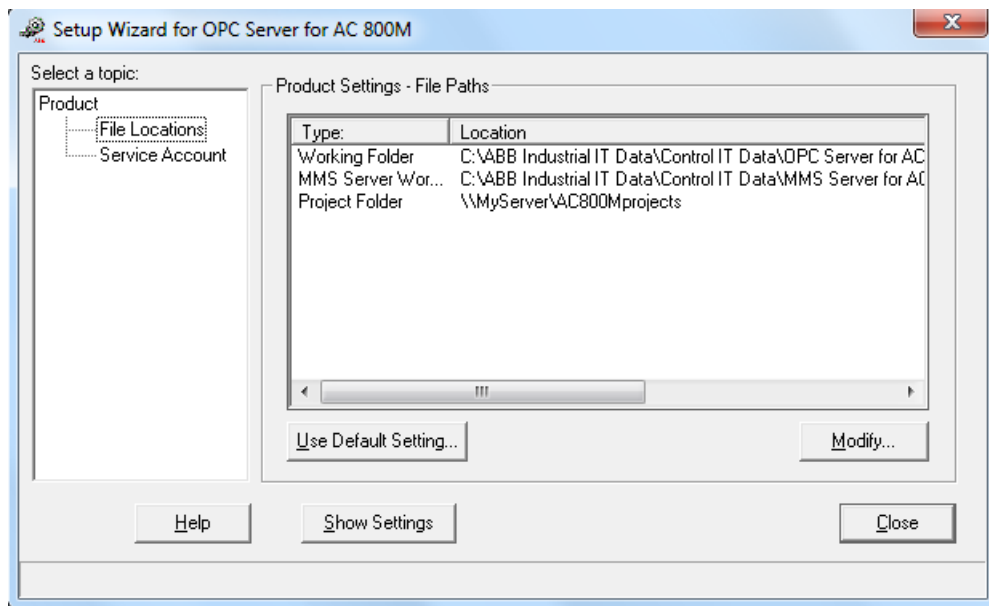


Figure 76. An example of a path to a shared project folder located on a network server.

5. Select the topic **Service Account**.

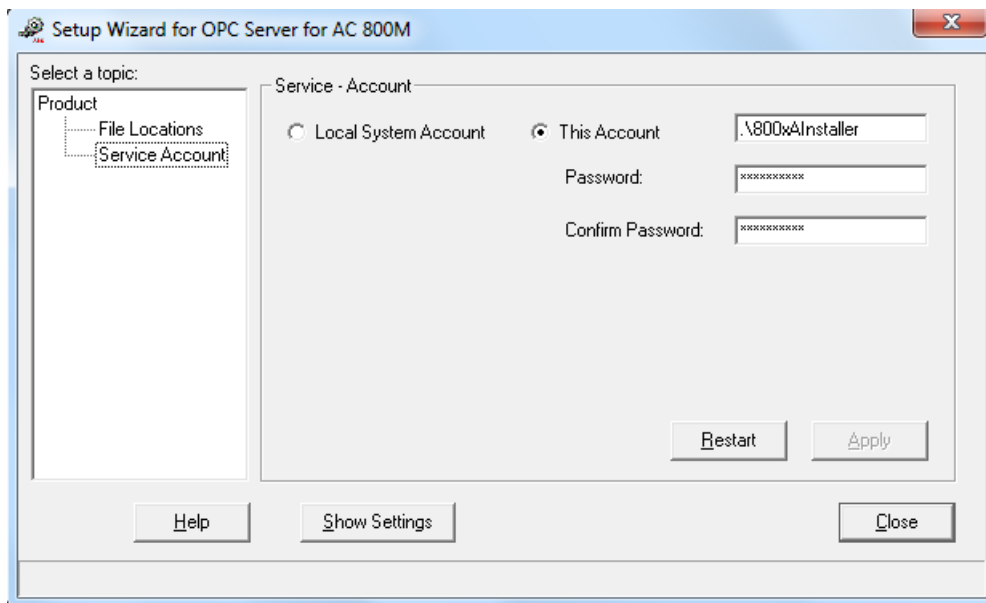


Figure 77. The Service Account topic for OPC server showing the current login name

### Choosing the Service Account

The choice between '*Local System Account*' and '*This Account*' depends on the OPC Server location and the other software products that are installed on the same PC (as the OPC Server). The two account options are:

- **Local System Account** cannot read from/write to a network file server. The project folder must be created locally on the same PC as the OPC server runs. This is irrespective of the user that is logged in.
- If a network file server holds the project folder the OPC service must run with **This Account** and the User account (user, password) must have change privileges to the project folder.

### Communication Failure between OPC Server and the OPC Panel

If a communication failure occurs between the OPC Server and the OPC Panel, then a pop-up menu appears with the message 'Access denied', and the OPC Panel stops responding. In such a scenario, proceed with the following steps:

- a. In Windows, search for the **OPCServerPanel.exe** file<sup>1</sup>.
- b. Right-click **OPCServerPanel.exe** and select **Run as** in the context menu.
- c. Select the OPC Server user account (with administrator role) for running the OPC Server Panel application.

For more information about setting up user account in Windows, refer to Windows User Documentation.

6. Click **Close** to close the Setup Wizard.



Create a small project in one of the Compact Control Builder stations and verify that the project can be opened from all the other Compact Control Builder stations.

## Configuration Example

Assume the following network configuration:

- The OPC Server is installed on a PC together with an operator interface. The OPC Server has the Project folder path set to the File Server.
  - OPC Server copies the configuration data from the project folder to its own local working folder. It uses the configuration data to translate the live data traffic from the PLC.
  - OPC Server writes cold retain values to the shared project folder. This is one of the reasons to set the permission *Change* in Windows.
- If *Local System Account* (see Figure 77) has been selected previously in the Setup Wizard, only members of MyTeam group should login to the PC machine, or the OPC Server is interrupted.
- If *This Account* (see Figure 77) has been selected previously in the Setup Wizard, any user including an operator can login to the PC machine without interrupting the OPC Server traffic.

---

1. Normally located at C:\Program Files\ABB Industrial IT\Control IT\OPC Server for AC 800M 5.1\Bin\



- Two Compact Control Builder stations with their project folder shared on the network File Server.
- One File Server with the shared project folder AC800Mprojects.

Figure 78 shows this multi-user configuration. The connected control system is a PLC.

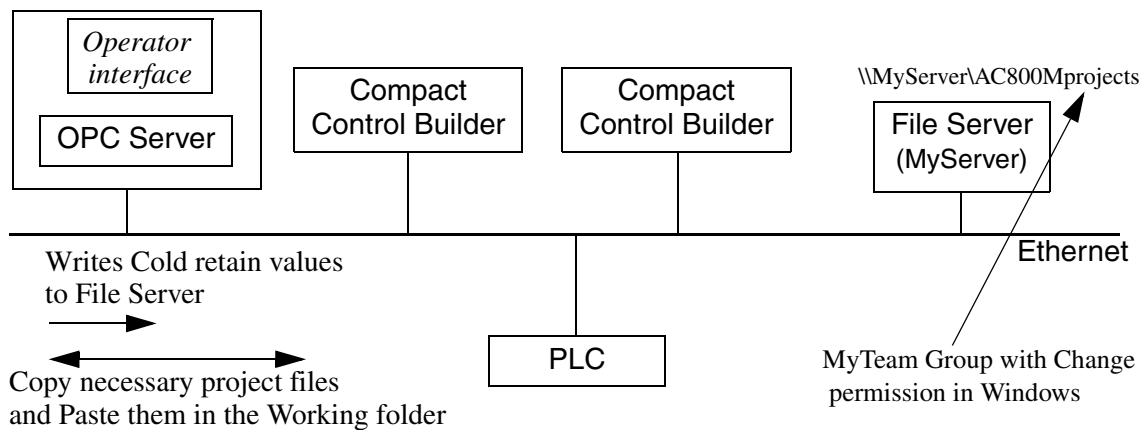


Figure 78. An example of a multi-user configuration.

## Guide lines for Multi-User Engineering

A project contains a number of files, whereas every member in a multi-user environment has a decisive impact on these project files. This means that a project folder shared by several Compact Control Builders may be subjected to multiple changes at the same time.

To avoid unwanted read/write results on the project files, follow these guidelines:

1. Several members may work with different libraries etc. without difficulties, but always strive to assign one member to a specific library, application or PLC at the time.
2. If several members must work with the same library or application, then allow only one member to work with a specific Type (Program, Function block type etc.) at the time.
3. Allow only one member to work with control modules in an application at the same time.
4. If several members must work with a specific PLC, then permit only one member to work with a specific Hardware unit, Task or Access variables, at the same time.



If a rename operation affects several files, Control Builder shows the alert message and displays the corresponding files before proceeding with the rename operation.

## Download

This section describes download and the checks and reports associated with download.

### General Download

Select **Tools > Download Project and Go Online** to download the project in Control Builder and enter the Online mode.

#### Version and Online Analysis

During the version check, the project in the Control Builder is analyzed and compared with the project downloaded in the controller (if any).

The version check detects the following conditions and provide solutions:

- If the project versions are identical, and neither the application nor the controller configuration has been changed, then the project is not downloaded. The effect is the same as going online without download.
- If there is no project version mismatch, but the application or controller configuration is changed, then the changed parts are downloaded. This download depends on the next two conditions.
- If the project in the controller contains an application that is not part of the project to be downloaded, but has been downloaded as part of the same project, then this application is deleted during the download process.
- If there is another project in the controller, different from the one to be downloaded, the user must delete the project and restart the controller, before downloading the new project. See [Download New Project to Controller](#) on page 118.

The Online Analysis dialog displays the present applications and controller configurations, and whether or not they have been changed in the Control Builder. An application or controller configuration is considered changed if the version in the Control Builder is different from the version running in the controller.

## Compilation

Compilation is performed in Control Builder. If any warnings or errors are detected during the compilation, a Compilation Summary dialog shows a summary of the warnings and errors. Then choose (if there are no errors) to continue or cancel the compilation.



Compiler switches can be used to set additional restrictions for the code. For more information, see the manual *Compact 800 Engineering, Compact Control Builder AC 800M, Planning (3BSE040935\* )*.

## Change Analysis

Control Builder performs a change analysis if any of the following are changed:

- Variables, function blocks, or control modules.
- Data types, function block types or control module types.
- Libraries.
- Applications.

The change analysis is performed before downloading, to check the possibility of maintaining variable values after restart.

The change analysis detects mismatches between the application version in the controller and the application version to be downloaded.

A mismatch can occur if:

- A variable is assigned another data type.
- A variable, function block, or control module is renamed.
- A data type, function block type, or control module type is missing, renamed, or moved to another library.
- A library is given a new name (this results in a mismatch for all data types, function blocks types, and control module types from this library).
- An application has been renamed (this results in a mismatch for all data types and variables, function blocks, and control modules in the application).

For variables with attributes *Retain* or *ColdRetain*, the change analysis is performed in the following way:

1. All data types, function block types, and control module types, which existed before the change, are checked for name matching.
2. All variables, function blocks, and control modules are checked for name and type matching.

If the change analysis detects mismatches, Control Builder cannot determine how to retain variable values. A warning dialog displays information about detected mismatches. Then the mismatching names should be corrected, by giving the renamed object the new name (click **Rename** in the dialog).

### Download and Go Online

The changed parts of the project (application and/or controller configuration) are downloaded to the controller(s). The controller(s) stops the running application(s), and restarts with the new/changed versions, and with variable values maintained (depending on the type of attribute and restart).

After the download is completed, Control Builder enters Online mode. In Online mode, Control Builder communicates with the controller(s), and the user can view variables and execution of the application in the controller(s) using online editors. Furthermore, the user can issue operations to the controller.



During download, if an error is detected in the downloaded controller configuration, then the message “Download aborted. See the controller log for further information.” appears. A common cause is that there is no sufficient controller memory. The details are found in the controller log. If the controller is still running, try to compile and download again. Refer to *3BSE040935\**, *Compact 800 Engineering*, *Compact Control Builder*, *AC 800M Configuration* manual to locate the log file.

## Download New Project to Controller

When **Tools > Download Project and Go Online** is selected, the Confirm Deletion of Project dialog is displayed if there is another project in the controller. To reset and restart the controller, click **Complete Reset** in the dialog. See [Figure 79](#).



Another way to reset and restart the controller is to press controller’s INIT button for more than 3 seconds.

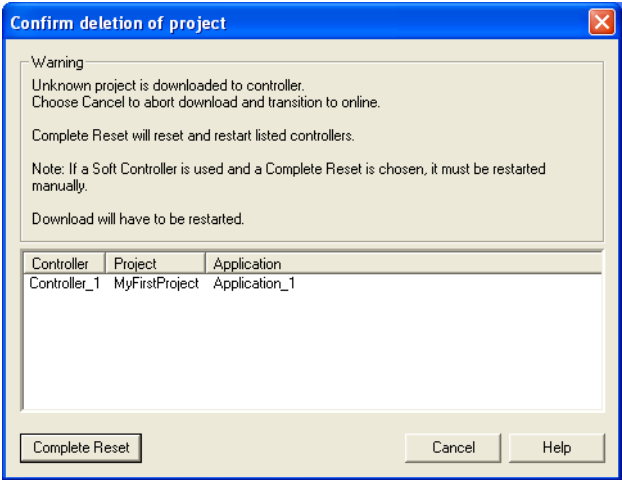


Figure 79. Confirm Deletion of Project dialog

The controller is reset, all existing applications in the controller are deleted, and the controller is restarted. The download of the new project can then be continued (see [General Download](#) on page 115).

## Download Project to Selected Controllers

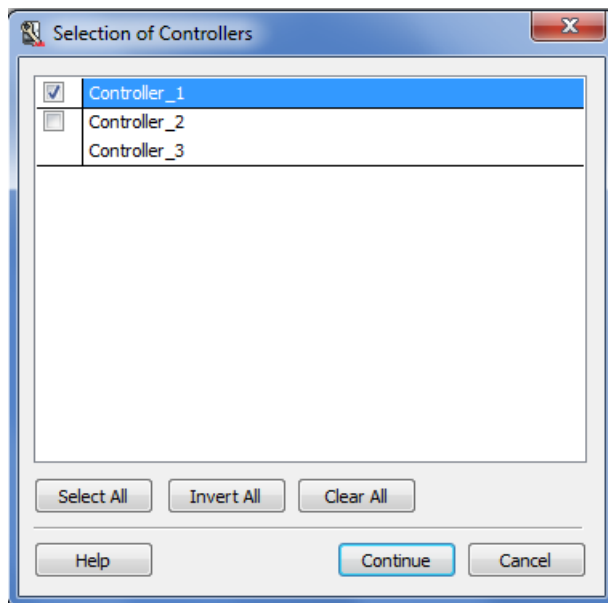
It is possible to select the controllers to download and go online, for a project that contains more than one controller. In this way, certain applications and controller configurations can be excluded from the download, and can go online with only a subset of the project to a selected controller. This reduces the compilation time.

When working in a multi-user environment, one user can work with some parts of the project, while other users can work with other parts. If the other parts of the project cannot be compiled because they are not completed, the user can go online with parts that are completed.



If an application is connected to several controllers, it is not possible to select only one controller. The remaining controllers are added automatically.

When **Tools > Download Project and Go Online** is selected for a project with more than one controller, the Selection of Controllers dialog is displayed. For example, in [Figure 80](#), “Controller\_2” and “Controller\_3” cannot be separately selected or excluded since one application is connected to both these controllers.



*Figure 80. Selection of Controllers dialog*

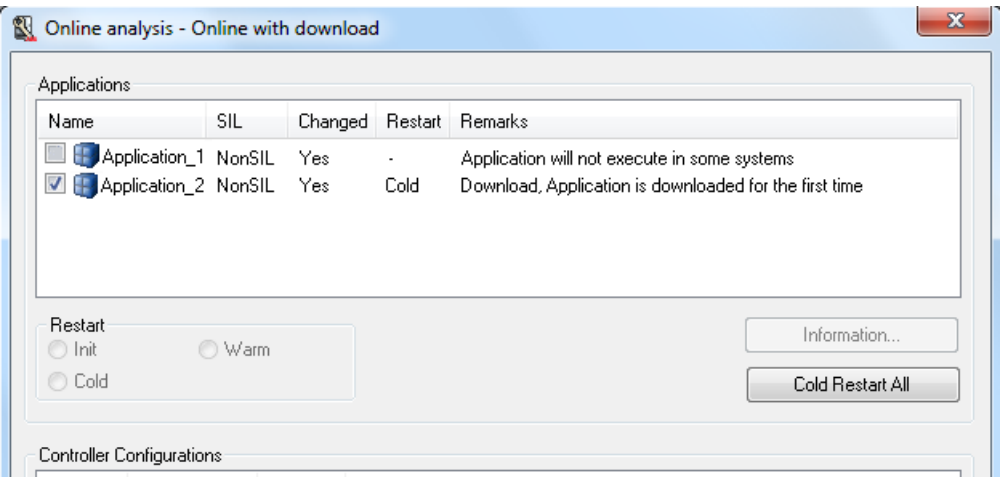
Another way to select a specific controller is to right-click the controller in Project Explorer, and select **Download and Online Mode**.

The Selective Download function is by default enabled, but can be disabled by selecting **Tools > Setup > Station > Application Download**, and in the Setup dialog setting the parameter *SuppressOnlineSelectionDialog* to true.

**Selecting an Application to Download**

Besides connecting several applications to one controller and downloading all of them, a single application can be selected for download. This is convenient when some parts, for example, in Application\_1, are not ready but Application\_2 is completed and ready for testing. If both applications are connected to Controller\_1, then there is an option to select if both applications should be downloaded or just Application\_2.

- 1. Click **Continue**-button in Figure 80, an Online Analysis window opens.
- 2. Select the application to be downloaded. See the example in Figure 81.



*Figure 81. Connected applications to a controller. Only Application 2 will be downloaded.*



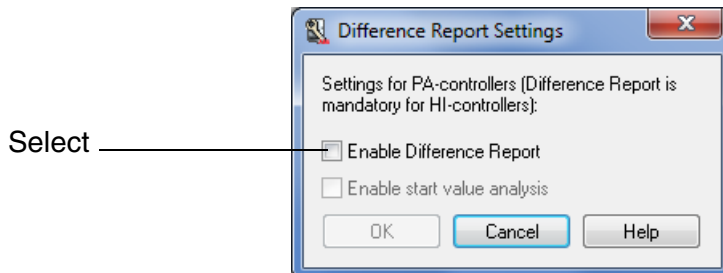
## Disable/Enable Difference Report

The following instructions enable the Difference Report, thus displaying the Difference Report dialog to pop-up when selecting *Test Mode*, *Online*, *Download Project* and *Go Online*.

### Enable Difference Report

From the Project Explorer tree:

1. Right-click the Project icon (root level) and select **Settings > Difference Report** from the context menu. A 'Difference Report Settings' dialog opens.
2. Select the **Enable Difference Report** option.



3. Click **OK**.



# Appendix B Network Redundancy



The information given in this Appendix applies only to users who intend to setup Redundant Networks.

For more information about Redundant Networks and clock synchronization, refer to the online help and the *AC 800M Communication Protocols (3BSE035982\*)* manual (in particular, the MMS section in the manual).

## Setting Up Redundant Network

The following example explains how to set up two separate redundant networks with the so called *implicit IP addressing* method. It also explains configuring IP addresses for the two PLCs and two PCs. The following sub-sections have step-by-step instructions which can be applied in the projects. After completing this example, the user must be able to add another PLC or Compact Control Builder station.

### Two Separate Redundant Networks

The example consists of PLC\_1 and PLC\_2 on one redundant Control Network, and PC\_1 with a Control Builder on another redundant Client/Server Network. PC\_2, with for example an OPC Server to access the right network components, connects the two separate redundant networks according to [Figure 82](#).

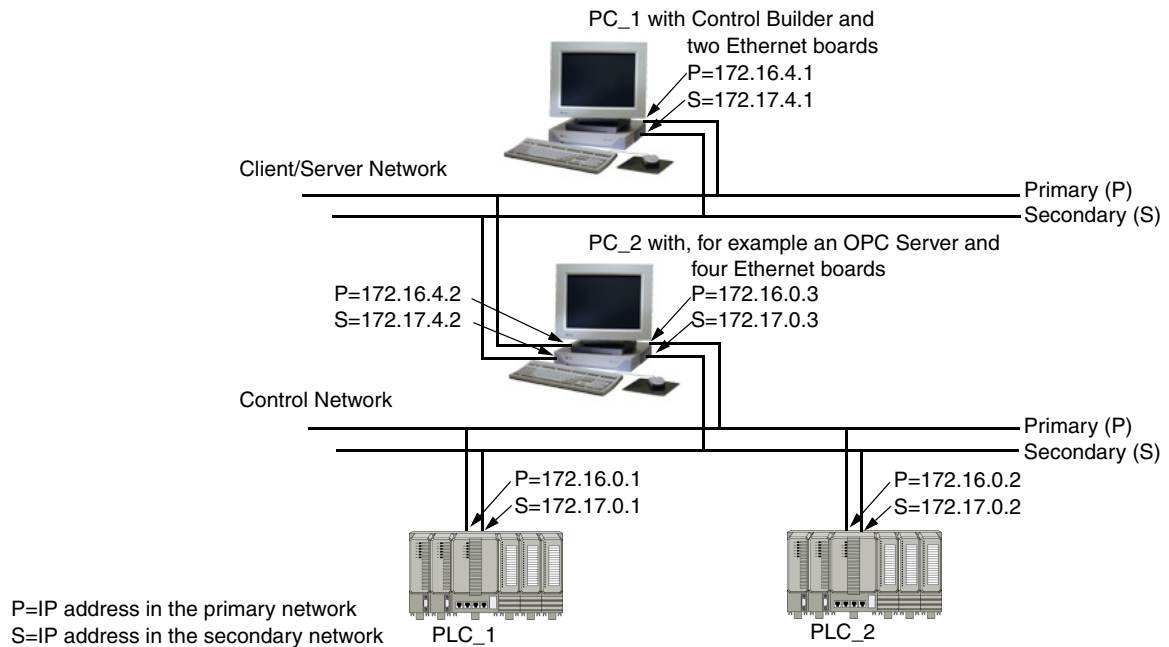


Figure 82. Redundant Control Network and redundant Client/Server Network.

### Changing in RNRP Setup Wizard

Run the RNRP Setup Wizard for PC\_2, otherwise the routing from PC\_1, via PC\_2, to the PLCs does not work.

1. Right-click the **ABB RNRP**-icon located at the lower right-side of the Windows desktop. The RNRP Wizard opens.

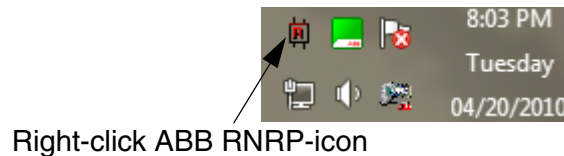


Figure 83. ABB RNRP-icon for opening the Setup wizard.

2. Make sure the **Base Parameters** tab are active.

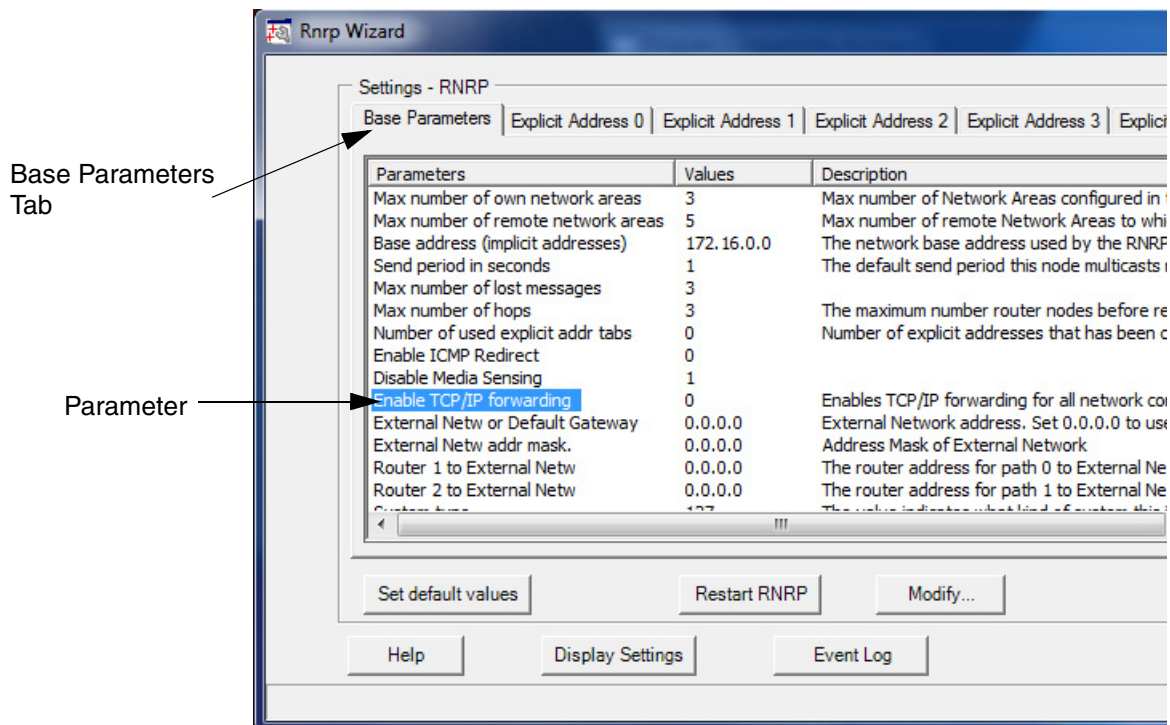


Figure 84. Rnrp Wizard

3. Select the **Enable TCP/IP forwarding** parameter and click the **Modify** button. A value dialog opens.

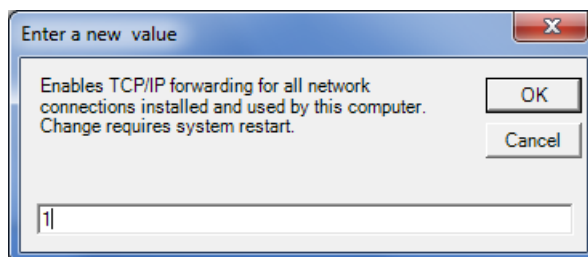


Figure 85. Parameter dialog for changing parameter values.

4. Change the parameter value to **1** instead of the default value 0 and click **OK**.
5. Click **Close** to close the RNRP Wizard.

## Decide IP Addresses

First the IP addresses must be decided, and then the IP subnet mask to use for each Ethernet connection port. These redundant networks must have two IP addresses for each PLC and in PC\_1 with the Control Builder. In PC\_2, with the OPC Server, four IP addresses are required to the four ports in the Ethernet PC boards.

In this example, select the IP addresses displayed in [Table 6](#) below and in [Figure 82](#) as follows.

### Subnet mask

Use the following subnet mask for **all** IP addresses: **255.255.252.0**.

### IP addresses (X.Y.Z.Q)

- Use the recommended IP addresses in the **X.Y.** positions of X.Y.Z.Q for both redundant networks; **172.16.Z.Q** for the primary network, and **172.17.Z.Q** for the secondary network.
- Due to the subnet mask value, select the **Z.** position of X.Y.Z.Q as a multiple of four. Choose two values between; 172.16.**0.Q**, 172.16.**4.Q**, 172.16.**8.Q**, 172.16.**12.Q** etc. up to 172.16.**124.Q**. Note that the **Z.** value must also be different for Control Network and Client/Server Network.
- Select the **Q** position of X.Y.Z.**Q** as a free serial number in the range 1 - 254, for each **node** on each separate network. Thus one serial **Q** number for the primary and the secondary network ports of each PLC and PC.

Table 6. Selected settings of the IP addresses with the 255.255.252.0 subnet mask.

Network	PLC_1 (X.Y.Z.Q)	PLC_2 (X.Y.Z.Q)	PC_1 with Control Builder M (X.Y.Z.Q)	PC_2 with OPC Server (X.Y.Z.Q)
Primary Control Network	172.16.0.1	172.16.0.2		172.16.0.3
Secondary Control Network	172.17.0.1	172.17.0.2		172.17.0.3
Primary Client/Server Network			172.16.4.1	172.16.4.2
Secondary Client/Server Network			172.17.4.1	172.17.4.2

## Setup Using the IPConfig Tool

To get initial access to the PLCs, assign them with a first primary IP address.

1. Connect a PC, running the program *IPConfig* tool, to the PLC\_1 Tool port (via a serial cable).
2. Follow IPConfig online help instructions and set the primary IP address to 172.16.0.1 (see [Table 6](#)).
3. Repeat Step 1 and Step 2, for PLC\_2.

## Configure PLC Ports from Project Explorer

1. In the Project Explorer of PC\_1, configure the project with PLC\_1 and PLC\_2.
2. In PLC\_1, double-click Ethernet port number 1 to open its editor.
3. In the editor Settings tab, set the IP address parameter to 172.16.0.1 (see [Table 6](#)), and the IP subnet mask to 255.255.252.0. No other parameter setting is required.
4. Repeat step 2 and 3 for Ethernet port number 2 of PLC\_1, set its IP address to 172.17.0.1 (see [Table 6](#)).
5. From Ethernet port number 2, select **Settings** tab and set the **Enable Ethernet channel** parameter to true ([Figure 86](#)).

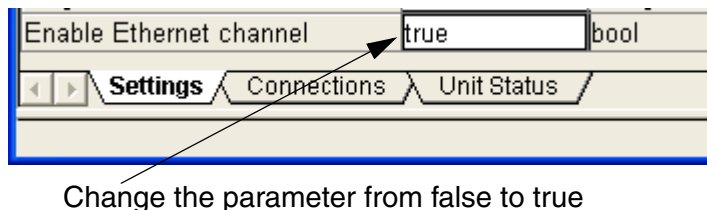


Figure 86. Hardware editor for Ethernet port No. 2 in Project Explorer. The parameter must be true for Network redundancy.

6. Repeat step 2 to 5 for PLC\_2.

## Configure PC Ports in Windows 7

Configure the Ethernet board ports for PC\_1 and PC\_2, using the following steps:

1. In PC\_1, go to **Start > Control Panel > Network and Internet > Network and Sharing Center**.
2. Click **Change adapter settings**.
3. Right-click Local Area Connection and select **Properties**. The Local Area Connection Properties dialog is displayed.
4. In the Connect Using list, select the Ethernet board that is to be connected to the primary network.
5. Select **Internet Protocol Version 4 (TCP/IP)** and click **Properties**. The Internet Protocol Version 4 (TCP/IP) Properties dialog is displayed.
6. Select **Use the following IP address** and enter the IP address (172.16.4.1 for the primary network in PC\_1, see [Table 6](#)), and the subnet mask 255.255.252.0. Click **OK**.
7. Repeat steps 1 to 7 for the secondary network (using the IP address 172.16.4.2).
8. Click **OK**.
9. For PC\_2, repeat steps 1 to 9 for **all four** Ethernet boards to be configured.
10. Click **OK**. The PC IP addresses and subnet masks have now been set for all Ethernet ports connected to the two networks.



## Download Project and Go Online

When all IP addresses and subnet masks are set, download the project and go online in the Control Builder. After a while, redundant network communication is enabled.

## Setting Clock Synchronization using the CNCP Protocol

This subsection describes how to setup a clock synchronization using the CNCP protocol. CNCP offers relative clock synchronization down to milliseconds, which means that the real time clocks in the controllers will never be more than a few milliseconds apart. Hence, absolute time will not necessarily be totally correct. See also the *AC 800M Communication Protocols (3BSE035982\*)* manual for other available clock synchronization methods.



CNCP requires that RNRP is properly configured in the PLCs.

When using AC 800M controllers, CNCP is the recommended protocol for time synchronization to all nodes on the Control Network that support CNCP.

### Setting the Local Time for CNCP Using Control Builder

The Time Set function in the Control Builder helps to set the local time and sends a Set-Time message with CNCP. The message is received by all nodes that run CNCP.

The Time Set function can only be used to set the time in the AC 800M controllers if the Control Builder is connected to the Control Network with no routers between itself and the controllers to be synchronized.

To set a new local time:

1. In the Control Builder, go to **Tools > Maintenance > Clock Synch > Time Set** to open the Time Set window.
2. Enter the new local time to set.
3. Click **OK**.

The Set-Time message takes effect only if the setting *CS Time Set Enabled*, in the hardware editor of the controller, is set to `True`.

Configure the CNCP clock master

Assign one controller to be the clock master (any type of controller is adequate). From the Project Explorer tree:

- 1. Right-click the assigned PM unit and select **Editor** from the context menu. The hardware editor opens.
- 2. Select the **Settings** tab and locate the following parameters.
- 3. Set the following (four) parameters according to Figure 87. The setting *CS Time Set Enabled* can be set to `True` if local time source is used.

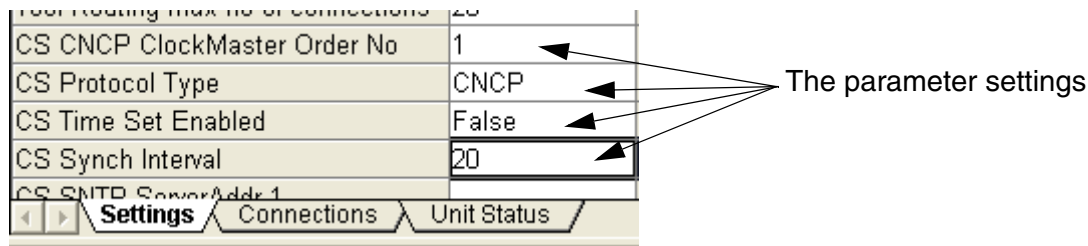


Figure 87. The parameter settings under the Settings tab in the hardware editor.

- 4. Save the settings and close the editor (**Ctrl+U**).
- 5. Perform a download to the (clock master) controller.

The CNCP Clock Master periodically sends time updates to the clock slaves with multicast messages.

Configure the CNCP clock slaves

Assign all the other controllers to be clock slaves (they should all be configured in the same way). From the Project Explorer tree:

- 1. Right-click the PM unit and select **Editor** from the context menu. The hardware editor opens.
- 2. Select the **Settings** tab and locate the following parameters.
- 3. Set the following (four) parameters according to Figure 88. The setting *CS Time Set Enabled* can be set to `True` if local time source is used.

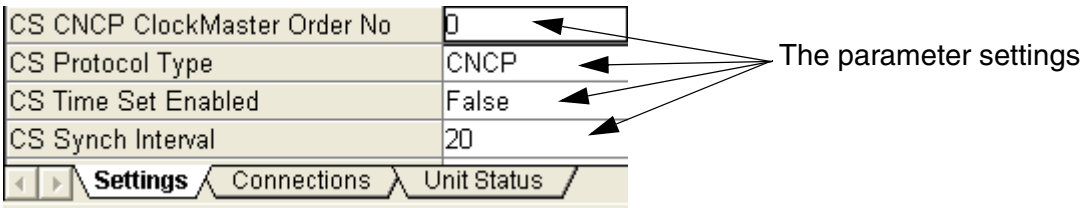


Figure 88. The parameter settings under the Settings tab in the hardware editor.

4. Save the settings and close the editor (**Ctrl+U**).
5. Perform a download to the (clock slave) controller.

# Design

## IP Address

A communication channel IP address is a 32-bit word (4×8 bits) that can be entered as a string X.Y.Z.Q of four decimal numbers 0-255, separated by periods. The IP standard uses the terms *NetID* and *HostID*. The *subnet mask* specifies the boundary between the NetID part and the HostID part of the IP address (the zero bits indicate the HostID part). Depending on the value of X, IP addresses are divided mainly into three classes, A–C:

Table 7. IP address classes.

IP address in bit format		XXXXXXXX.YYYYYYYY.ZZZZZZZZ.QQQQQQQQ			
Class A		←NetID→	←HostID→		
Class B		←NetID→	←HostID→		
Class C		←NetID→	←HostID→		
Class	Value of X	NetID	HostID	Possible host IP addresses	Default subnet mask
A	1-126	X	Y.Z.Q	X.0.0.1-X.255.255.254	255.0.0.0
B	128-191	X.Y	Z.Q	X.Y.0.1-X.Y.255.254	255.255.0.0
C	192-223	X.Y.Z	Q	X.Y.Z.1-X.Y.Z.254	255.255.255.0

The *Redundant Network Routing Protocol (RNRP)* developed by ABB handles alternative paths between nodes and automatically adapts to topology changes. RNRP uses more terms than the standard IP, namely *network area* and *node number*. By selecting 225.255.252.0 as the subnet mask, the last 10 bits constitute the node number (i.e. host ID, 0-1023). Note that the largest permitted node number is 500. The remaining NetID part is used for network ID (16 bits), local flag (1 bit), and network area number (5 bits). The last two bits of the network ID make up the path number, where 0 indicates the primary network and 1 the redundant secondary network.

Consequently, RNRP requires a different interpretation of the IP address to the IP standard.

It is recommended that the RNRP interpretation of the IP address be used. If the decimal numbers are converted to binary numbers, the address can be interpreted as follows:

XXXXXXXX.XXXXXXPP.LAAAAA.NNNNNNNN

Each position represents a binary digit (0 or 1). The different parts of the address signify the following:

Table 8. IP address converted into binary.

Binary number	Number of bits	IP term	RNRP term
XXXXXXXX.XXXXXXPP	16	Network ID	Network ID
LAAAAA		Subnetwork ID	
PP	2		Path number
L	1		Local flag
AAAAA	5		Network area number
NNNNNNNNNN	10	Host ID	Node number

The subnet mask sets the boundary between the host ID part and the subnet ID part and is selected as 11111111.11111111.11111100.00000000 (=255.255.252.0 in decimal notation).



The network ID must be identical for all nodes on the same network. It is recommended that an address be selected from the private IP address space, which has the following advantages:

- There is no requirement to apply to the licensing authorities for an IP address.
- Some protection is gained against illegal access, since private addresses are not permitted on the public Internet.
- The firm connection between IP and RNRP parameters reduces the risk of inconsistency.



The following primary network IP addresses are recommended (class B):

172.16.0.0  
172.20.0.0  
172.24.0.0  
or 172.28.0.0

By converting the second decimal number into binary, the path number is converted to 0 (see also [Table 6](#) and [Table 8](#)).

### Example 1:

Convert the subnet ID 11111111.11111111.11111100.00000000 to decimal.

By writing the first part in groups of four binary digits (1111), the hexadecimal equivalent is FF. The decimal equivalent is 255. The second part is identical, that is, 255. The third part 1111 1100 equals FC (hexadecimal) and 252 (decimal). The fourth part equals 0 in both decimal and hexadecimal notation. Consequently the subnet ID is written 255.255.252.0 in decimal notation.

### Example 2:

IP address 192.168.255.25 written in binary notation:

It can be written C0.A8.FF.19 in hexadecimal form, which means 11000000.10101000.11111111.00011001 in binary notation.

## Separating Client/Server and Control Network

The Control Network must be protected from foreign traffic that can be a security risk and also cause undesired load on both the nodes and network. To avoid these risks the Control Network should be physically separated from the Internet and protected by servers and/or fire walls. In large configurations such separation may also be desirable between the Control Network and client/server networks.

### Client Node and Server Node

In a large network topology, only nodes with real-time communication requirements may be connected to the Control Network. *Client nodes* (such as engineering stations, operator stations, etc.) not belonging to the Control Network may go through a *server node* that performs authority control and can disable routing (transfers) to nodes on the Control Network.

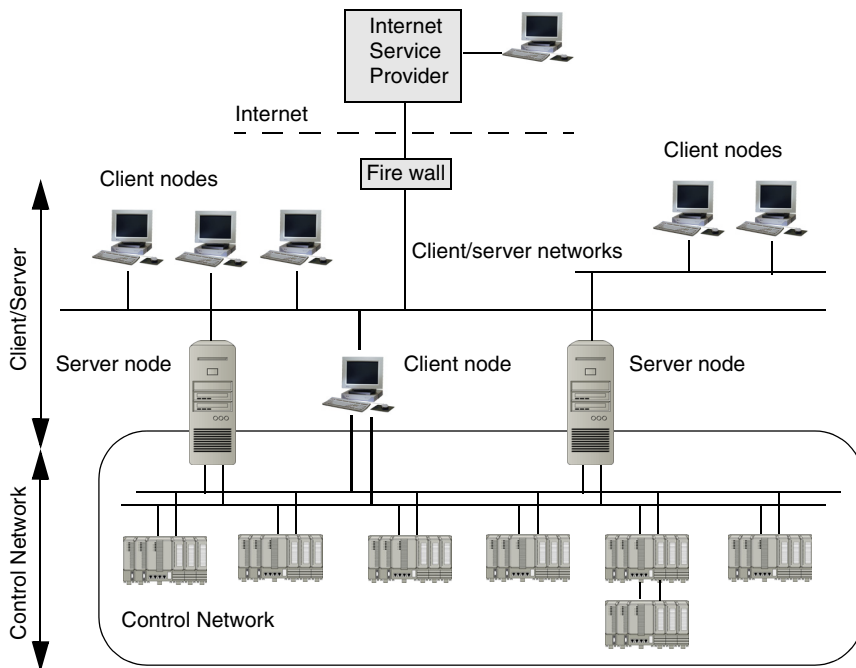


Figure 89. Separation of Control Network and Client/server networks.

It is, however, possible to have client nodes such as operator stations and engineering stations connected directly to the Control Network. If such a client needs access to the client/server network, a separate network connection is possible.



Traffic other than that between Compact Control Builder products may jeopardize performance.

Only IP traffic generated by Industrial<sup>IT</sup> products is allowed to reach the PLCs on the Control Network.

## Summary of Configuration Steps

1. Specify network ID (IP address and subnet mask).
2. Split the control network into network areas.
3. Specify local network areas.
4. Specify network area numbers.
5. Assign path numbers.
6. Decide which devices should run the routing functionality.
7. Assign IP addresses to each device.
8. Assign redundant functionality to each system.





## Appendix C Upgrade

This appendix describes how to upgrade Compact Control Builder (Compact CB or CCB) 4.1 and 5.0.2 projects to a CCB 5.1 project version.

Figure 90 shows the possible options for upgrade to Compact CB version 5.1.

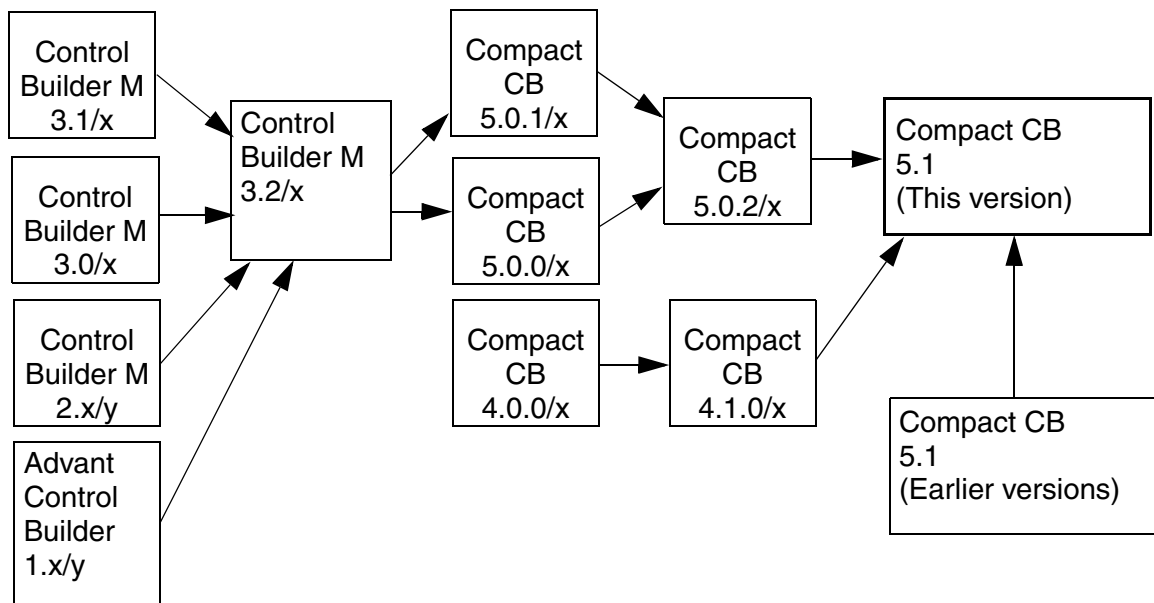


Figure 90. Options for upgrade to Compact CB Version 5.1



This appendix covers only the upgrade from CCB 5.0.2 and CCB 4.1 to CCB 5.1.




It is not possible to upgrade a project created in Control Builder Professional to Compact Control Builder 5.1.

# Introduction

Compact Control Builder AC 800M comes with features and performance similar to Control Builder Professional 5.1 version.

An upgrade involves the following:

- 1. Licences, see [Licenses](#) on page 138.
- 2. Products, see [Products](#) on page 139.
- 3. Applications, see [Applications](#) on page 140.



Upgrading the Compact Control Builder requires a shutdown of the plant. To guarantee the functionality of the upgraded system, follow the upgrade instructions carefully.

# Licenses

Contact the sales representative to perform the license upgrade. Collect all Industrial IT licenses and have them available when contacting ABB.

[Table 9](#) describes the Control Builder license structure.



Upgrading from CCB 4.1 or CCB 5.0.2 to CCB 5.1 does not require any additional number of licenses.

Table 9. License Structure between version 4.1/5.0.2 and 5.1

License in Version 4.1/5.0.2	License in Version 5.1
Compact Control Builder for AC 800M	Compact Control Builder for AC 800M <sup>(1)</sup>
OPC Server for AC 800M	OPC Server for AC 800M

(1) All the product licenses that were collected under the Compact Control Builder license in version 4.1/5.0.2 are fully supported in the 5.1 version.

# Products



All PLCs, Control Builder and OPC Server nodes must be shut down during the upgrade. A plant shutdown is needed, because a software upgrade of the PLC firmware and control software stops PLC execution.

Table 10 lists and describes specific considerations to take into account before and after upgrading.

Table 10. Product Upgrade Considerations

Issue	Considerations
Compatibility	Control Builder, AC 800M Firmware, and OPC Server for AC 800M must all be upgraded to the new version in order to work together. However, PLC peer-to-peer communication is possible with other PLCs running version 2.x, 3.x or 4.x by means of Access Variables.
Privilege Handling	All privilege/user handling done in Control Builder M in the previous version (PrivUsers/PrivClasses) will be lost during the upgrade. The Compact Control Builder does not have any privilege handling.

## Applications

### Saving Application and Configuration Data in CCB 4.1/5.0.2

The application and configuration data must be saved on a safe media before upgrading. It is also possible to take a backup of the entire project (which saves the application and configuration data) and restore it in the new version.

#### Settings

It is very important to save the Cold Retain values from the previous version (4.1 or 5.0.2), on files, before performing the upgrade.

To save the application and configuration data:

1. Open the project in the Compact Control Builder.
2. Go to the Online mode.
3. From Control Builder menu bar, select **Tools > Save ColdRetain Values**.  
Failure to make new files can cause some or all Cold Retain marked variables to revert to older values after upgrading.
4. Go to Offline mode.
5. From Control Builder menu bar, select **Tools > Maintenance > Project Backup**, select **Complete backup** as the option, and save the project to a safe media.
6. Repeat the procedure for each Control Builder project.
7. From the OPC Server Configuration window, select **File > Save Configuration**.  
The OPC configurations are saved in:  
...\\ABB Industrial IT Data\\Control IT Data\\OPC Server for AC 800M\\Files.

### Copy Files to Safe Media

Copy the following OPC Server files from the OPC Server to a safe media:

- systemsetup.sys, located in:  
`... \ABB Industrial IT Data \Control IT Data \`  
`OPC Server for AC 800M`
- Configuration files (\*.cfg), located in:  
`... \ABB Industrial IT Data \Control IT Data \`  
`OPC Server for AC 800M \Files`

## Remove Products

Stop all third party OPC Clients and shut down the Control Builder on all nodes.

In order to upgrade, the previous version or revision must be **removed** before installing the new version. If several products (that is, Control Builder and OPC Server, SoftController, User Documentation, Firmware and RNRP) are installed on the same PC, all should be removed before the new versions are installed.

The following services must be manually stopped before the products are removed/upgraded:

- MMS Server for AC 800M.
- OPC Server for AC 800M.
- RNRP Service.



It also applies for any other service that automatically starts any of above services.

The RNRP Service should be stopped in the Windows Services Overview, which is found at **Start > Control Panel > System and Security > Administrative Tools > Services**.



When the PC is upgraded to Windows 7 or Windows Server 2008, all existing Control Builder projects (in the previous version) will be deleted from the local disc.

In case of re-install of Control Builder on Windows 7 or Windows Server 2008, no existing projects will be deleted from disc when the Control Builder version is removed. In this case, the new Control Builder will be able to open the projects left on disc.

## Install Products

Install the CCB 5.1 products by follow the instructions given in [Section 2, Installing Software](#).

## Restore Application and Configuration Data in CCB 5.1

### OPC Server for AC 800M

1. Restore the previously saved systemsetup.sys file to the following location:  
`ABB Industrial IT Data\Control IT Data\OPC Server for AC 800M`
2. Restore the configuration files (\*.cfg) to the Files folder in the same location.

### Compact Control Builder

1. From Compact Control Builder menu bar, select **Tools > Maintenance > Restore**, select **Complete restore** as the option, and restore the project from the safe media to a new folder in the following destination:  
`...\ABB Industrial IT Data\Engineer IT Data\Compact Control Builder AC 800M\Projects`
2. Repeat the procedure for each Control Builder project.
3. Ensure that all the files in the Projects folder are not marked as Read-only (especially, if the files have been stored on a CD-ROM).

4. Upgrade the CCB 5.0.2 or CCB 4.1 projects to CCB 5.1 by performing one of the following methods:
  - Open the project in CCB 5.1.
  - In CCB 5.1, select **Tools > Maintenance > Upgrade Project**.

### Upgrade by Opening the Project

1. Open the CCB 5.0.2 or CCB 4.1 project in CCB 5.1 by selecting the option **File > Open Project**.
2. If the project needs to be upgraded, a dialog pops-up and informs the user about the upgrade of project to the current version.

During the upgrade, a check is made for the version of BasicHwLib. If any old version is connected to the project, the user is asked if the old libraries shall be kept or replaced by the newest library in the system

When the BasicHwLib is replaced, the firmware in the controllers also needs to be upgraded.



If other hardware libraries exist in newer major versions and the versions are dependent on the BasicHwLib version, then they are also replaced if the user selects to replace BasicHwLib. This is the case for the new versions of CI-HwLibraries delivered in CCB 5.1

### Upgrade by Selecting the Upgrade Project Option



Libraries, Applications, and Controllers originally placed in folders different from the project folder will not be upgraded if the project backup on safe media are restored using Windows Explorer.



The CI851 and CI854 HWD-files will be automatically placed in hardware libraries after the upgrade, unless the hardware libraries do not contain these files already.

1. From the Compact Control Builder, select **Tools > Maintenance > Upgrade Project**.
2. Browse to the project file (\*.prj) and click **Open**.  
The 'Save Upgraded Project As' dialog opens.

3. Select location and click **Save** in the project folder.
4. Wait for the upgrade to finish.
  - a. During the upgrade process, some messages may be displayed. Read the messages and choose among the available options.
  - b. The upgrade may take up to 30 minutes. The Compact Control Builder may stop responding during that time. If the process is interrupted, it must be restarted. The Windows Task Manager can be used to supervise the completeness.
5. Repeat [Step 1](#) to [Step 4](#) for all other projects.

### Upgrade PLC Firmware

Upgrade PLC firmware to the new version in order to be compatible with other products of version 5.1. Firmware in all communication interfaces must also be upgraded to the latest version.

## Handling a download



Before handling a download, refer to the compatibility issues given in [Table 11](#) and [Table 12](#).

1. Open the project.
2. Ignore the following warning messages that may be displayed in the message pane of the Project Explorer when opening the project.
  - At line xxx: Unknown parameter name: xxx
  - At line xxx: Parameter type mismatch: xxx

Make a dummy change in the hardware configuration for all hardware units if such messages are shown.
3. Go online with *download* and answer any mismatch dialog with the corresponding new name, in order to retain any Cold Retain values (for example, settings of PID loops).



It is very important to complete the mismatch handling during the first download. If not, some or all Cold Retain marked variables relinquish their saved values. See [Handling Mismatch for cold retain values](#) on page 145.





It is NOT possible to download a project to a controller if a project with a different name is already running in the controller. This rule applies if the name of the project is changed while upgrading (in the Save As dialog), and the project (with the previous version) is still present in the controller.

### Handling Mismatch for cold retain values

Upgrade to version 5.1 may give mismatch for cold retain values.

This means that Compact Control Builder displays the version mismatch dialog when downloading the project after upgrading to version 5.1. To prevent that cold retain values are lost during upgrading, examine the solutions provided below.

Solutions:

- An Application not found mismatch dialog may warn for the Source code unit SystemLib when an upgraded project is downloaded for the first time. Click **Rename** in the dialog and choose to rename the SystemLib to BasicLib. If this does not work, then the application does not have any type that has been moved to BasicLib. Cancel the Rename dialog and select **Next Mismatch**. Refer to Control Builder Online Help for additional information.
- If the version mismatch dialog displays 'CommunicationLib not found', rename the old CommunicationLib to any of the newly divided communication libraries in the version mismatch dialog.
- 4. From the hardware tree in Project Explorer, select the PLC and right-click **Properties > Heap Utilization**. A Heap Utilization dialog opens.
- 5. Check the spare memory needed for online changes in each PLC.
- 6. Repeat these steps for all other projects.

## Compatibility Issues

### CCB 5.0.2 to CCB 5.1 Compatibility Issues

[Table 11](#) lists the Compact Control Builder compatibility issues, and the suggested solutions, when upgrading from 5.0.2 to 5.1.

Table 11. CCB 5.0.2 to CCB 5.1 Compatibility Issues

Condition	Issue	Solution
Mixing formal and informal parameter passing in a function call is not allowed.	<p>Prior to CCB 5.1 it was allowed to mix formal and informal parameter passing in a function call.</p> <p>The Control Builder compiler in CCB 5.1 gives a compile error for mixed parameter passing.</p> <p><b>Allowed:</b></p> <pre>AddSuffix( String := StrVar, Suffix := StrSuffix );</pre> <p>or</p> <pre>AddSuffix( StrVar, StrSuffix );</pre> <p><b>Not allowed:</b></p> <pre>AddSuffix( String := StrVar, StrSuffix );</pre>	Correct the code to use either formal or informal parameter passing for all parameters in the function call.
Writing to components of a structured function block parameter with direction [in] via control module parameter is prohibited.	If a function block has a parameter p1 with direction [in] of a structured data type, it was possible to write to its components by connecting it to a control module that writes to the parameter. This is prohibited in CCB 5.1	Correct the code to not use this illegal construction.
New attributes for function block, control module, and data types have been introduced.	<p>Some new attributes have been introduced in CCB 5.1 that affect function block, control module, and data types.</p> <p>For example:</p> <ul style="list-style-type: none"> <li>- Direction attribute on control module parameters.</li> <li>- Reverse and ISP attribute on datatype components.</li> </ul>	For detailed information on the new attributes and their consequences, refer to <i>Compact Control Builder AC 800M Configuration (3BSE040935*)</i> .

Table 11. CCB 5.0.2 to CCB 5.1 Compatibility Issues (Continued)

Condition	Issue	Solution
Upgrade of CCB 5.0.2 projects containing CI851 and CI852 units.	The hardware libraries for CI851 and CI852 do not work together with CCB 5.1 controller firmware.	When Upgrade Project is performed in Control Builder the user must select to not upgrade to CCB 5.1 firmware in the controllers. This is not compatible with the hardware libraries for CI851 and CI852. After Upgrade Project is performed, verify that the controller is using BasicHWLib 5.0-2 and the hardware libraries for CI851 and CI852 as applicable.

### CCB 4.1 to CCB 5.1 Compatibility Issues

Table 12 lists and describes Compact Control Builder compatibility issues, and the suggested solutions, when upgrading from CCB 4.1 to CCB 5.1.

Table 12. CCB 4.1 to CCB 5.1 Compatibility Issues

Condition	Issue	Solution
Mixing formal and informal parameter passing in a function call is not allowed.	<p>Prior to CCB 5.1 it was allowed to mix formal and informal parameter passing in a function call.</p> <p>The Control Builder compiler in CCB 5.1 gives a compile error for mixed parameter passing.</p> <p><b>Allowed:</b></p> <pre>AddSuffix( String := StrVar, Suffix := StrSuffix );</pre> <p>or</p> <pre>AddSuffix( StrVar, StrSuffix );</pre> <p><b>Not allowed:</b></p> <pre>AddSuffix( String := StrVar, StrSuffix );</pre>	Correct the code to use either formal or informal parameter passing for all parameters in the function call.
Writing to components of a structured function block parameter with direction [in] via control module parameter is prohibited.	If a function block has a parameter p1 with direction [in] of a structured data type, it was possible to write to its components by connecting it to a control module that writes to the parameter. This is prohibited in CCB 5.1	Correct the code to not use this illegal construction.

Table 12. CCB 4.1 to CCB 5.1 Compatibility Issues (Continued)

Condition	Issue	Solution
New attributes for function block, control module, and data types have been introduced.	Some new attributes have been introduced in CCB 5.1 that affect function block, control module, and data types. For example: - Direction attribute on control module parameters. - Reverse and ISP attribute on datatype components.	For detailed information on the new attributes and their consequences, refer to <i>Compact Control Builder AC 800M Configuration (3BSE040935*)</i> .
Upgrade of CCB 4.1 projects containing CI851 and CI852 units.	The hardware libraries for CI851 and CI852 are not part of AC 800M Connect in CCB 5.1 and does not work together with CCB 5.1 controller firmware.	To upgrade a Control Builder Project from CCB 4.1 that has controllers containing CI851 or CI852 units, the following features must be installed and loaded after the restore to the CCB 5.1 is performed, but before Upgrade Project is performed in Control Builder: - AC 800M Classic Libraries (contains the hardware libraries for CI851 and CI852). - AC 800M Coexistence 5.0.2 (contains BasicHWLib 5.0-2). After Upgrade Project is performed, verify that the controller is using BasicHWLib 5.0-2 and the hardware libraries for CI851 and CI852 as applicable.
Force I/O from SIL2 applications prohibited	In order to better enforce the counting of maximum number of forces in High Integrity Controllers, the possibility to force I/O signals directly from code has been removed. This means that no writing will be allowed to the .Forced component of an IO data type in a SIL1-2 or SIL3 application. It will result in a compile error.	The only possibility to force I/O in a SIL application will be via the Safe Online Write function (confirmed write) from operator graphics. This restriction will also improve the response time when resetting forces in the controller. The reset of individual forced signals from code will still be possible by using the new firmware function ResetForcedValue(), which is introduced in CCB 5.0.2. This implies that constructions where variables of IO data type (i.e. BoolIO, RealIO, DWordIO, or DintIO) are copied or connected via out parameters in function blocks, will no longer work. However, parameter passing via in or in_out parameters is not affected.

Table 12. CCB 4.1 to CCB 5.1 Compatibility Issues (Continued)

Condition	Issue	Solution
Discontinued functions for Safe MMS communication (function blocks)	<p>Due to the restriction described in Force I/O from SIL2 applications prohibited, the SIL2 marking of function blocks writing to IO data types has been removed. The affected function blocks are:</p> <ul style="list-style-type: none"> <li>• MMSRead4BoolIO.</li> <li>• MMSRead4DintIO.</li> <li>• MMSRead4RealIO.</li> </ul> <p>Consequently, these function blocks can not be used in SIL applications but will still be available for use in non-SIL applications.</p> <p>The corresponding definition function blocks (MMSdef4xxxxIO) will still be available from SIL applications and consequently reading IO from SIL to non-SIL applications will still be possible.</p>	In SIL applications it is recommended to replace the MMSRead4xxxxIO with the MMSRead4xxxx function blocks and only transfer the .Value component of the IO data type.
Safety Operator User Group	An operator must be a member of the Safety Operator User Group to be able to write data to objects running in a SIL application in the controller.	Make the operator a member of the Safety Operator User Group.
No Time Sync warning in controller	Some controllers might get a No time sync warning in the controller log after upgrade to CCB 5.0.2.	Set the CS Protocol Type for controllers that are not intended to be synchronized <b>No Synch</b> in order to avoid the warning.

Table 12. CCB 4.1 to CCB 5.1 Compatibility Issues (Continued)

Condition	Issue	Solution
Accessing Local Variables in Function Blocks.	<p>It has been possible to access local variables in Function Blocks from surrounding code.</p> <p>According to the IEC 61131-3 standard, local variables shall only be accessible within the containing software element.</p> <p>A compilation error is given if local variables are accessed from the outside.</p>	Correct any such warnings before performing the download. Local variables used in this fashion need to be changed to parameters instead.
Code sorting loops treated as errors.	<p>Code sorting loops in applications are by default considered as errors in CCB 5.0.2.</p> <p>It is not possible to compile and download an application if it contains code sorting loops.</p>	<p>First, try to correct the sorting loops. Refer to Interpret and Correct Code Loop Errors in <i>Compact Control Builder AC 800M Planning (3BSE044222*)</i>.</p> <p>Another alternative is to change the compiler switch for Code Sorting Loops:</p> <ol style="list-style-type: none"> <li>1. Mark the project in Project Explorer.</li> <li>2. Right-click and select <b>Settings &gt; Compiler Switch</b>.</li> <li>3. Set the global Loops in Control Modules switch to <b>Warning</b>.</li> </ol>
Applications having integer literal values too large as initial values.	<p>Compile error 1040 might appear in Control Builder projects that previously did not contain any compile errors.</p> <p>Control Builder now makes more stringent compiler checks on initial values for variables. It previously allowed illegal (too large) integer literal values as initial values. The actual used value was zero.</p>	<p>Correct the compile error.</p> <p>e.g. by entering a smaller, legal value on the literal.</p>

Table 12. CCB 4.1 to CCB 5.1 Compatibility Issues (Continued)

Condition	Issue	Solution
Self defined serial communication using the Serial Communication Library.	The Serial Communication Library has undergone a major internal redesign that in some cases may lead to compatibility issues.	<p><b>Use of firmware functions:</b></p> <p>Firmware functions used for serial communication handling are no longer supported.</p> <p>This means that user-built libraries where these firmware functions have been used, can no longer be used.</p> <p>The no longer supported firmware functions are:</p> <ul style="list-style-type: none"> <li>• OpenDevice</li> <li>• UpdateDeviceSetup,</li> <li>• SetDeviceClearRead</li> <li>• CloseDevice.</li> <li>• ReadStringDevice</li> <li>• ReadLineDevice</li> <li>• WriteStringDevice</li> </ul> <p><b>Improvement were made in the following areas:</b></p> <ul style="list-style-type: none"> <li>• Clear buffer when entering the listen operation of the <i>SerialListen</i> function block.</li> <li>• If the <i>SerialWriteWait</i> function block is triggered when the timeout has elapsed, the read/listen buffer is cleared in between the retry operations in connection to the write operation.</li> <li>• <b>Behavior of the SerialWriteWait function block in previous versions:</b> If the function block is triggered after the application stopping phase is entered, the function block propagates the status code -15 via the Status -parameter.</li> </ul>



Table 12. CCB 4.1 to CCB 5.1 Compatibility Issues (Continued)

Condition	Issue	Solution
Self defined serial communication using the Serial Communication Library. (continued)	The Serial Communication Library has undergone a major internal redesign that in some cases may lead to compatibility issues. (continued)	<p>If the power fails, and a function block was in a pending state, the Status -5331 is derived from it after the controller is powered up again.</p> <ul style="list-style-type: none"> <li> <b>Behavior of the <i>SerialWriteWait</i> function block in previous versions:</b> If the function block was triggered after the application stopping phase was entered, the function block should normally propagate status code -15. But this was masked, and instead, after the finished application change, the function block automatically retriggers the write operation. </li> </ul>
	The Serial Communication Library has undergone a major internal redesign that in some cases may lead to compatibility issues. (continued)	<p><b>Printing a string longer than 140 characters:</b></p> <p>The behavior of the <i>SerialWrite</i> operation is no longer synchronous. Therefore, the possibility to call one and the same function block time after time to print a longer string than 140 characters is no longer supported.</p> <p>Instead, do like the following:</p> <p>To be able to print a string longer than the maximum length of 140 characters, call subsequent function blocks in order:</p> <pre>Write1( Req := TRUE,       Id := Id,       EndChar := EndChar_Write1,       Done =&gt; Done_Write1,       Error =&gt; Error_Write1,       Status =&gt; Status_Write1,       Sd := Sd_Write1 ); Write2( Req := TRUE,       Id := Id,       EndChar := EndChar_Write2,       Done =&gt; Done_Write2,       Error =&gt; Error_Write2,       Status =&gt; Status_Write2,       Sd := Sd_Write2 );</pre> <p>Printing the two strings Sd_Write1 + Sd_Write2 by calling the two function blocks will be queued up, and it will be printed in a series.</p>

Table 12. CCB 4.1 to CCB 5.1 Compatibility Issues (Continued)

Condition	Issue	Solution
Compiler warnings occur if there is a risk for task collisions.	If there is a risk that tasks can collide in the controller, a warning will be displayed during compilation. The compilation warning will look like:  Warning 9155: Controller_1:HW Task Normal and Fast may have colliding start times	Apply proper task offsets so that the tasks can no longer collide.
Applications using SattBus on TCP/IP.	SattBus on TCP/IP will not work after the upgrade. The COMLI communication function blocks have been used also for SattBus on TCP/IP, but this is no longer the case. A new library including a set of new function blocks should be used instead.	Change the application to use function blocks and data types from SattBusCommLib as follows: Function Blocks BeforeAfter COMLIConnectComliSBConnect COMLIReadComliSBRead COMLIReadCycComliSBReadCyc COMLIReadPhysComliSBReadPhys COMLIWriteComliSBWrite COMLIWriteDTComliSBWriteDT Data Types BeforeAfter Comm_Channel_COMLIComm_Channel_ComliSB

Table 12. CCB 4.1 to CCB 5.1 Compatibility Issues (Continued)

Condition	Issue	Solution
Some project constants have been removed from SupervisionBasicLib (cSinit.*), BasicLib (cEnable.*), and SupervisionLib (cInit.*). Refer to <a href="#">Removed Project Constants</a> on page 164 for a list of the removed project constants.	<ol style="list-style-type: none"> <li>1. If these project constants have been used in code there will be compile errors after upgrading the project.</li> <li>2. If the value of the project constants was changed in the project in CCB 5.0 SP1a, it will lead to changed behavior of the code after the upgrade, because these project constants have been used as default values on parameters.</li> </ol>	<ol style="list-style-type: none"> <li>1. Change to the correct literal value in the places where the project constants are used.</li> <li>2. Check if the value of the removed project constant was changed in the project in CCB 5.0 SP1a. This can be done by opening the Project Constants dialog box after the upgrade and seeing if the project constants are listed.</li> </ol> <p>In that case the parameters previously using the project constant as a default value must be manually connected to a literal with the corresponding value.</p>

### Control Builder M 3.2/x to CCB 5.0.2 Compatibility Issues

Table 13 lists and describes Control Builder compatibility issues, including solutions to the issues, when upgrading from Control Builder M 3.2/x to CCB 5.0.2.

Table 13. Control Builder M 3.2/x to CCB 5.0.2 Compatibility Issues

Condition	Issue	Solution
No Time Sync warning in controller	Some controllers might get a <code>No time sync</code> warning in the controller log after upgrade to CCB 5.0.2.	Set the CS Protocol Type for controllers that are not intended to be synchronized <b>No Synch</b> in order to avoid the warning.
Valid for users that have copied a template object from, for example, a Process Object Library, and are using its Display Element Reduced Icon.	The Visual BASIC® resize code of the Display Element Reduced Icon becomes corrupt after deploy in Graphics Builder.	Remove the source comments around the <code>UserControl_Resize</code> method before deploying the Display Element Reduced Icon. Edit the Display Element Reduced Icon, removing the comments, and then deploy the Display Element Reduced Icon. The source comments are: <pre>' \$\$ABB_BEGIN_USERCONTROL_RESIZE --- ' \$\$ABB_END_USERCONTROL_RESIZE</pre> <b>NOTE:</b> Delete only the comment lines. Do not delete the source code between the comment lines.
Internal MMS Communication between applications residing in the same controller.	The behavior of internal communication between two applications in the same controller has changed.	This kind of communication is now asynchronous; all data may no longer always be received in the same scan.

Table 13. Control Builder M 3.2/x to CCB 5.0.2 Compatibility Issues (Continued)

Condition	Issue	Solution
User-Defined Hardware Definition Files when 1131-application variables of type dInt and dWord are written from the application to I/O-channels.	The behavior has been undefined when the 1131-variable has a very large value (larger than the maximum value possible to write on a specific hardware unit). The very large value has really been truncated, a hardware unit receiving 8-bit data has received the 8 lowest bits in the dInt or dWord.	The behavior is redefined as follows: <ul style="list-style-type: none"> <li>The hardware unit will be written with the largest, which is different depending on hardware, possible value if 1131-variable has a very large positive value.</li> <li>The hardware unit will be written with the largest, which is different depending on hardware, negative value if 1131-variable has a very large negative value.</li> </ul>
FF HSE input I/O channels.	The mapping of FF-status in input I/O channels (BoolIO, RealIO, etc.) connected to CI860 channels has been changed.	Previously, the FF-status of in-channels has been mapped into both the MSB and the LSB of the status word. This has changed so that the LSB now is remapped into OPC-status (since this byte now is sent to the OPC-server). <b>Example:</b> The FF status 16#80 was previously mapped as 16#80000080 into the status word in the I/O data type. It is from now mapped as 16#800000C0. The result is that the OPC Server sends 16#C0 to its OPC clients.
Accessing Local Variables in Function Blocks.	It has been possible to access local variables in Function Blocks from surrounding code. According to the IEC 61131-3 standard, local variables shall only be accessible within the containing software element. A compilation error is given if local variables are accessed from the outside.	Correct any such warnings before performing the download. Local variables used in this fashion need to be changed to parameters instead.

Table 13. Control Builder M 3.2/x to CCB 5.0.2 Compatibility Issues (Continued)

Condition	Issue	Solution
Code sorting loops treated as errors.	Code sorting loops in applications are by default considered as errors in CCB 5.0.2. It is not possible to compile and download an application if it contains code sorting loops.	First, try to correct the sorting loops. Refer to Interpret and Correct Code Loop Errors in <i>Compact Control Builder AC 800M Planning (3BSE044222*)</i> . Another alternative is to change the compiler switch for Code Sorting Loops: 1. Mark the project in Project Explorer. 2. Right-click and select <b>Settings &gt; Compiler Switch</b> . 3. Set the global Loops in Control Modules switch to <b>Warning</b> .
Applications having integer literal values too large as initial values.	Compile error 1040 might appear in Control Builder projects that previously did not contain any compile errors. Control Builder now makes more stringent compiler checks on initial values for variables. It previously allowed illegal (too large) integer literal values as initial values. The actual used value was zero.	Correct the compile error. e.g. by entering a smaller, legal value on the literal.

Table 13. Control Builder M 3.2/x to CCB 5.0.2 Compatibility Issues (Continued)

Condition	Issue	Solution																		
Applications using SattBus on TCP/IP.	<p>SattBus on TCP/IP will not work after the upgrade.</p> <p>The COMLI communication function blocks have been used also for SattBus on TCP/IP, but this is no longer the case. A new library including a set of new function blocks should be used instead.</p>	<p>Change the application to use function blocks and data types from SattBusCommLib as follows:</p> <p>Function Blocks</p> <table><tr><td>Before</td><td>After</td></tr><tr><td>COMLIConnect</td><td>ComliSBConnect</td></tr><tr><td>COMLIRead</td><td>ComliSBRead</td></tr><tr><td>COMLIReadCyc</td><td>ComliSBReadCyc</td></tr><tr><td>COMLIReadPhys</td><td>ComliSBReadPhys</td></tr><tr><td>COMLIWrite</td><td>ComliSBWrite</td></tr><tr><td>COMLIWriteDT</td><td>ComliSBWriteDT</td></tr></table> <p>Data Types</p> <table><tr><td>Before</td><td>After</td></tr><tr><td>Comm_Channel_COMLI</td><td>Comm_Channel_ComliSB</td></tr></table>	Before	After	COMLIConnect	ComliSBConnect	COMLIRead	ComliSBRead	COMLIReadCyc	ComliSBReadCyc	COMLIReadPhys	ComliSBReadPhys	COMLIWrite	ComliSBWrite	COMLIWriteDT	ComliSBWriteDT	Before	After	Comm_Channel_COMLI	Comm_Channel_ComliSB
Before	After																			
COMLIConnect	ComliSBConnect																			
COMLIRead	ComliSBRead																			
COMLIReadCyc	ComliSBReadCyc																			
COMLIReadPhys	ComliSBReadPhys																			
COMLIWrite	ComliSBWrite																			
COMLIWriteDT	ComliSBWriteDT																			
Before	After																			
Comm_Channel_COMLI	Comm_Channel_ComliSB																			
Compiler warnings occur if there is a risk for task collisions.	<p>If there is a risk that tasks can collide in the controller, a warning will be displayed during compilation. The compilation warning will look like:</p> <p>Warning 9155: Controller_1:HW Task Normal and Fast may have colliding start times</p>	<p>Apply proper task offsets so that the tasks can no longer collide.</p>																		

Table 13. Control Builder M 3.2/x to CCB 5.0.2 Compatibility Issues (Continued)

Condition	Issue	Solution
Self defined serial communication using the Serial Communication Library.	The Serial Communication Library has undergone a major internal redesign that in some cases may lead to compatibility issues.	<p><b>Use of firmware functions:</b></p> <p>Firmware functions used for serial communication handling are no longer supported.</p> <p>This means that user-built libraries where these firmware functions have been used, can no longer be used.</p> <p>The no longer supported firmware functions are:</p> <ul style="list-style-type: none"><li>• OpenDevice</li><li>• UpdateDeviceSetup,</li><li>• SetDeviceClearRead</li><li>• CloseDevice.</li><li>• ReadStringDevice</li><li>• ReadLineDevice</li><li>• WriteStringDevice</li></ul> <p><b>Improvement were made in the following areas:</b></p> <ul style="list-style-type: none"><li>• Clear buffer when entering the listen operation of the <i>SerialListen</i> function block.</li><li>• If the <i>SerialWriteWait</i> function block is triggered when the timeout has elapsed, the read/listen buffer is cleared in between the retry operations in connection to the write operation.</li></ul>



Table 13. Control Builder M 3.2/x to CCB 5.0.2 Compatibility Issues (Continued)

Condition	Issue	Solution
Self defined serial communication using the Serial Communication Library. (continued)	The Serial Communication Library has undergone a major internal redesign that in some cases may lead to compatibility issues. (continued)	<ul style="list-style-type: none"> <li>• <b>Behavior of the <i>SerialWriteWait</i> function block in previous versions:</b> If the function block was triggered after the application stopping phase was entered, the function block should normally propagate status code -15. But this was masked, and instead, after the finished application change, the function block automatically retriggered the write operation.</li> <li>• <b>Behavior of the <i>SerialWriteWait</i> function block in previous versions:</b> If the function block is triggered after the application stopping phase is entered, the function block propagates the status code -15 via the Status -parameter.</li> </ul> <p>If the power fails, and a function block was in a pending state, the Status -5331 is derived from it after the controller is powered up again.</p>

Table 13. Control Builder M 3.2/x to CCB 5.0.2 Compatibility Issues (Continued)

Condition	Issue	Solution
Self defined serial communication using the Serial Communication Library. (continued)	The Serial Communication Library has undergone a major internal redesign that in some cases may lead to compatibility issues. (continued)	<p><b>Printing a string longer than 140 characters:</b></p> <p>The behavior of the <i>SerialWrite</i> operation is no longer synchronous. Therefore, the possibility to call one and the same function block time after time to print a longer string than 140 characters is no longer supported.</p> <p>Instead, do like the following:</p> <p>To be able to print a string longer than the maximum length of 140 characters, call subsequent function blocks in order:</p> <pre> Write1( Req := TRUE,       Id := Id,       EndChar := EndChar_Write1,       Done =&gt; Done_Write1,       Error =&gt; Error_Write1,       Status =&gt; Status_Write1,       Sd := Sd_Write1 ); Write2( Req := TRUE,       Id := Id,       EndChar := EndChar_Write2,       Done =&gt; Done_Write2,       Error =&gt; Error_Write2,       Status =&gt; Status_Write2,       Sd := Sd_Write2 ); </pre> <p>Printing the two strings Sd_Write1 + Sd_Write2 by calling the two function blocks will be queued up, and it will be printed in a series.</p>

Table 13. Control Builder M 3.2/x to CCB 5.0.2 Compatibility Issues (Continued)

Condition	Issue	Solution
ProcessObjectAE Function block parameter name change	<p>The <b>Name</b> parameter on the ProcessObjectAE Function Block in AlarmEventLib has a changed name. The parameter is now called CondNameObjErr.</p> <p>This means that applications using the ProcessObjectAE will not be able to be downloaded after the upgrade. There will be compilation errors for unknown parameter names.</p>	Applications using this Function Block need to be changed to use the CondNameObjErr parameter instead.
Some project constants have been removed from SupervisionBasicLib (cSinit.*), BasicLib (cEnable.*), and SupervisionLib (cInit.*). Refer to <a href="#">Removed Project Constants</a> on page 164 for a list of the removed project constants.	<ol style="list-style-type: none"> <li>1. If these project constants have been used in code there will be compile errors after upgrading the project.</li> <li>2. If the value of the project constants was changed in the project in CCB 5.0.2/1, it will lead to changed behavior of the code after the upgrade, because these project constants have been used as default values on parameters.</li> </ol>	<ol style="list-style-type: none"> <li>1. Change to the correct literal value in the places where the project constants are used.</li> <li>2. Check if the value of the removed project constant was changed in the project in CCB 5.0.2/1. This can be done by opening the Project Constants dialog box after the upgrade and seeing if the project constants are listed.</li> </ol> <p>In that case the parameters previously using the project constant as a default value must be manually connected to a literal with the corresponding value.</p>

## Removed Project Constants

Some project constants have been removed from SupervisionBasicLib (cSinit.\*), BasicLib (cEnable.\*), and SupervisionLib (cInit.\*).

### **SupervisionBasicLib (cSinit.\*)**

- cSinit.Latch shall be replaced by 'true'.
- cSinit.LevelHH shall be replaced by '80.0'.
- cSinit.LevelH shall be replaced by '60.0'.
- cSinit.LevelL shall be replaced by '40.0'.
- cSinit.LevelLL shall be replaced by '20.0'.

**BasicLib (cEnable.\*)***Table 14. BasicLib (cEnable)*

cEnable.InputOverUnderRange		bool	Default value in library BasicLib 1.4-11 (bool): false
cEnable.OutputOverUnderRange		bool	Default value in library BasicLib 1.4-11 (bool): false

**SupervisionLib (cInit.\*)***Table 15. SupervisionLib (cInit.\*)*

cInit.AckreqAtErr		bool	Default value in library SupervisionLib 2.3-7 (bool): false
cInit.AckReqToReset		bool	Default value in library SupervisionLib 2.3-7 (bool): false
cInit.ActOnError		bool	Default value in library SupervisionLib 2.3-7 (bool): false
cInit.AEClass		dint	Default value in library SupervisionLib 2.3-7 (dint): 1
cInit.AEConfigCableBreak		dint	Default value in library SupervisionLib 2.3-7 (dint): 0
cInit.AEConfigDetectorFault		dint	Default value in library SupervisionLib 2.3-7 (dint): 0
cInit.AEConfigDiffInput		dint	Default value in library SupervisionLib 2.3-7 (dint): 1
cInit.AEConfigDiffOutput		dint	Default value in library SupervisionLib 2.3-7 (dint): 3
cInit.AEConfigErr		dint	Default value in library SupervisionLib 2.3-7 (dint): 1
cInit.AEConfigH		dint	Default value in library SupervisionLib 2.3-7 (dint): 1
cInit.AEConfigHH		dint	Default value in library SupervisionLib 2.3-7 (dint): 1
cInit.AEConfigL		dint	Default value in library SupervisionLib 2.3-7 (dint): 0
cInit.AEConfigLoopFault		dint	Default value in library SupervisionLib 2.3-7 (dint): 0
cInit.AEConfigMaintenance		dint	Default value in library SupervisionLib 2.3-7 (dint): 0
cInit.AEConfigPrewarn		dint	Default value in library SupervisionLib 2.3-7 (dint): 1
cInit.AEConfigShortCircuit		dint	Default value in library SupervisionLib 2.3-7 (dint): 0
cInit.AESevCableBreak		dint	Default value in library SupervisionLib 2.3-7 (dint): 800
cInit.AESevDetectorFault		dint	Default value in library SupervisionLib 2.3-7 (dint): 800

Table 15. *SupervisionLib (cInit.\*) (Continued)*

clnit.AESevDiffInput		dint	Default value in library SupervisionLib 2.3-7 (dint): 800
clnit.AESevDiffOutput		dint	Default value in library SupervisionLib 2.3-7 (dint): 300
clnit.AESevErr		dint	Default value in library SupervisionLib 2.3-7 (dint): 800
clnit.AESevFeedback		dint	Default value in library SupervisionLib 2.3-7 (dint): 300
clnit.AESevH		dint	Default value in library SupervisionLib 2.3-7 (dint): 800
clnit.AESevHH		dint	Default value in library SupervisionLib 2.3-7 (dint): 800
clnit.AESevL		dint	Default value in library SupervisionLib 2.3-7 (dint): 800
clnit.AESevLoopFault		dint	Default value in library SupervisionLib 2.3-7 (dint): 800
clnit.AESevMaintenance		dint	Default value in library SupervisionLib 2.3-7 (dint): 800
clnit.AESevPrewarn		dint	Default value in library SupervisionLib 2.3-7 (dint): 800
clnit.AESevReleased		dint	Default value in library SupervisionLib 2.3-7 (dint): 300
clnit.AESevShortCircuit		dint	Default value in library SupervisionLib 2.3-7 (dint): 800
clnit.AlarmLimit		real	Default value in library SupervisionLib 2.3-7 (real): 40.0
clnit.ConfirmedLimitH		dint	Default value in library SupervisionLib 2.3-7 (dint): 2
clnit.ConfirmedLimitHH		dint	Default value in library SupervisionLib 2.3-7 (dint): 0
clnit.ConfirmedLimitOr		bool	Default value in library SupervisionLib 2.3-7 (bool): true
clnit.DelayTime		time	Default value in library SupervisionLib 2.3-7 (time): TIME#10s
clnit.EnableCommonReset		bool	Default value in library SupervisionLib 2.3-7 (bool): true
clnit.EnableRangeConversion		bool	Default value in library SupervisionLib 2.3-7 (bool): false
clnit.Latch		bool	Default value in library SupervisionLib 2.3-7 (bool): true
clnit.LevelCableBreak		real	Default value in library SupervisionLib 2.3-7 (real): 1.0
clnit.LevelDetectorFault		real	Default value in library SupervisionLib 2.3-7 (real): 3.0
clnit.LevelDiff		real	Default value in library SupervisionLib 2.3-7 (real): 10.0
clnit.LevelLoopFault		real	Default value in library SupervisionLib 2.3-7 (real): 3.5
clnit.LevelMaintenance		real	Default value in library SupervisionLib 2.3-7 (real): 2.5

Table 15. *SupervisionLib (cInit. \*) (Continued)*

cInit.LevelShortCircuit		real	Default value in library SupervisionLib 2.3-7 (real): 19.0
cInit.OneLevelH		real	Default value in library SupervisionLib 2.3-7 (real): 10.0
cInit.OneLevelL		real	Default value in library SupervisionLib 2.3-7 (real): 5.0
cInit.OutputResetDelay		time	Default value in library SupervisionLib 2.3-7 (time): TIME#1s
cInit.PrewarningLimit		real	Default value in library SupervisionLib 2.3-7 (real): 5.0
cInit.RedIncDecLim		real	Default value in library SupervisionLib 2.3-7 (real): 10.0
cInit.ResetTime		time	Default value in library SupervisionLib 2.3-7 (time): TIME#5s
cInit.ResponseTime		time	Default value in library SupervisionLib 2.3-7 (time): TIME#5s
cInit.RText		string	Default value in library SupervisionLib 2.3-7 (string): 'R'
cInit.TwoLevelH		real	Default value in library SupervisionLib 2.3-7 (real): 20.0
cInit.TwoLevelHH		real	Default value in library SupervisionLib 2.3-7 (real): 60.0
cInit.UseLevelPar		bool	Default value in library SupervisionLib 2.3-7 (bool): True





# Appendix D Communication Cables

Serial communication between Control Builder and the PLC is performed by using the TK212A cable.

Connect the DB9 Female connector to a Control Builder PC COM port, thus the RJ45 (8P8C) plug to the PLC COM4 port. The [Figure 91](#) illustrates the TK212A pin-out configuration.

## Connecting Control Builder PC to a PLC

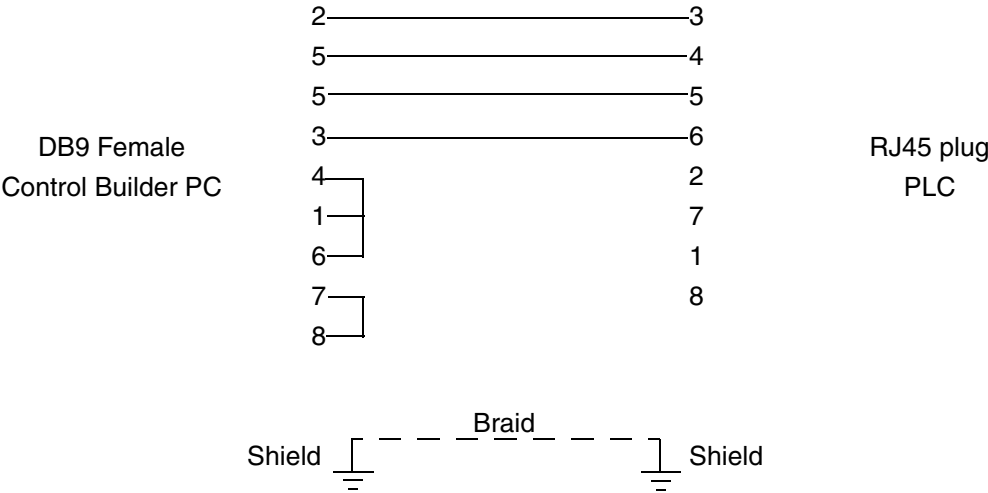


Figure 91. Cable TK212A Pinout configuration.



## Appendix E Programming Languages

Compact Control Builder provides five different programming languages according to IEC 61131-3. These are Function Block Diagram (FBD), Structured Text (ST), Instruction List (IL), Ladder Diagram (LD) and Sequential Function Chart (SFC). The specific rules and syntax of the programming languages will not be discussed in detail in this manual. For more information, refer to the Control Builder Online Help.



SFC Viewer is not supported by Compact Control Builder AC 800M.

### General

The IEC 61131-3 standard defines five of the most commonly used programming languages on the market. Depending on previous experience, programmers often have their own personal preference for a certain language. All the languages have advantages and disadvantages, and no single one of them is suitable for all control tasks. We start with three basic statements and then proceed to some important characteristics of each language.

- In small applications with relatively few logical conditions, the demand for good structure and re-use of code is not as great as in larger systems.
- ST and IL are textual languages, while FBD, LD, and SFC are based on graphical metaphors.
- LD and IL are not as powerful as ST or FBD.



The definition of function block is allowed in all five languages, not only in FBD. A function block is a method of encapsulating the code in a “black box” with inputs and outputs.

All instructions and functions viewed in this section are explained in detail in Control Builder online help.

For more information on the different languages, refer to the manual *IEC 61131 Control Languages, Introduction*.

Some important characteristics of the languages are listed in [Table 16](#).

*Table 16. Compact Control Builder programming languages.*

Language	Function
Function Block Diagram (FBD)	A graphical language for depicting signal and data flows through function blocks and re-usable software elements. Function blocks and variables are interconnected graphically, which makes the resulting control diagrams easy to read.
Structured Text (ST)	A high-level programming language. ST is highly structured and has a comprehensive range of constructs for assignments, function/function block calls, expressions, conditional statements, iterations, etc.  It is easy to write advanced, compact, but clear ST code, due to its logical and structured layout.
Instruction List (IL)	A traditional PLC language. It has a structure similar to simple machine assembler code.
Ladder Diagram (LD)	Ladder diagram (LD) is a graphical language based on relay ladder logic.
Sequential Function Chart (SFC)	Sequential function chart (SFC) is a graphical language for depicting the sequential behavior of a control program.

## Appendix F Glossary

The following is a list of terms associated with Compact Control Builder. The list contains some terms and abbreviations that are unique to ABB or have a usage or definition that is different from standard industry usage.

Term/Acronym	Description
Application	The application contains the program code to be compiled and downloaded for execution in the PLC. Applications are displayed in the Project Explorer.
Control Builder	A programming tool with a compiler for control software. Control Builder is accessed through the Project Explorer interface.
Control Module (Type)	A program unit that supports object-oriented data flow programming. Control modules offer free-layout graphical programming, code sorting and static parameter connections. Control module instances are created from control module types.
Firmware	The system software in the hardware.
Hardware Description	The tree structure in the Project Explorer, that defines the hardware's physical layout.
IEC 61131-3	A standard for control system languages defined by the IEC.
Industrial <sup>IT</sup>	ABB's vision for enterprise automation.
Industrial <sup>IT</sup> 800xA System	A computer system that implements the Industrial IT vision.
Interaction Window	A graphical interface used by the programmer to interact with an object. Available for many library types.

Term/Acronym	Description
MMS	Manufacturing Message Specification, a standard for messages used in industrial communication.  This is the application layer used within MAP (Manufacturing Automation Protocol), a specification for open communication based on the OSI model.
OPC	OLE for Processing Control.
OPC/DA	An application programming interface defined by the standardization group OPC Foundation. The standard defines how to access large amounts of real-time data between applications. The OPC standard interface is used between automation/control applications, field systems/devices and business/office application.
Process Object	A process concept/equipment such as valve, motor, conveyor or tank.
Project Explorer	The Control Builder interface. Used to create, navigate and configure libraries, applications and hardware.  All objects such as data types, functions and function block types can be selected and displayed in an editor. The software and hardware is configured in the Project Explorer.
RNRP	Redundant Network Routing Protocol.

Term/Acronym	Description
Security	<p>Security controls a user's authority to perform different operations on (Aspect) Objects, depending on several parameters:</p> <ul style="list-style-type: none"><li>• The user's credentials, as provided by Windows</li><li>• The node where the user is logged in. This makes it possible to give users different authority depending on where they are located, for example close to the process equipment, in a control room, or at home accessing the system through Internet.</li><li>• The task that the user wants to perform.</li></ul>
Type	<p>A type solution that is defined in a library or locally, in an application. A type is used to create instances, which inherit the properties of the type.</p>





---

# INDEX

## A

ABB Common 3rd Party Install 23  
AC 800M  
    cable configuration 169  
    License upgrade  
        CCB 4.1 or CCB 5.0.2 to CCB 5.1 138  
    Product upgrade  
        CCB 4.1 or CCB 5.0.2 to CCB 5.1 139  
AC800M 34  
addresses  
    convert 132  
Alarm and Event Library 15  
analysis 115  
    changes at download 116  
    test mode 71  
application 33  
Applications 37

## B

Basic Library 15, 55

## C

Cable Configuration  
    AC 800M 169  
CCB 4.1 or CCB 5.0.2 to CCB 5.1  
    AC 800M considerations  
        License upgrade 138  
CCB 4.1 or CCB 5.0.2 to CCB 5.1  
    Copying files to safe media  
        OPC server for AC 800M 141  
change analysis 116  
Check 47  
Check icon 43  
class A  
    IP address 131

class B  
    IP address 131  
class C  
    IP address 131  
client  
    node 134  
client/server networks 134  
code pane 56  
Cold Retain 140  
Communication Library 15  
Communication Variable 56  
compilation 116  
configuration  
    MMS 135  
Connected Libraries 40  
Contents Topic 20  
Context-Sensitive Help 20  
Control Module Types 40  
Control Network 134  
Controllers 37  
controllers  
    select at download 118  
convert  
    addresses 132  
Copying files to safe media  
    OPC server for AC 800M  
        SV 4.1 or SV 5.0.2 to SV 5.1 141  
counters 59  
cross-over 93

## D

Data Types 40  
declaration pane 56  
Description 47  
download

- new project 118
- select controller(s) 118
- to selected controller(s) 118

Download and Go Online 117

Downloading

- via Ethernet 94

## E

editor

- programs 56

EmptyProject 34

Enable Ethernet channel 127

Ethernet

- download via 94
- ports 95
- set IP address 95

Ethernet cable 93

## F

FBD 171

File Location

- select 101

File server 103

Firmware

- upgrade via serial line 87

Firmware Version 89

Forced 97

Function Block Types 40

## H

Heap Utilization 145

host ID

- IP address 132

HostID 131

## I

I/O Address 86

I/O channels 83

Icon library 55

IL 171

implicit IP addressing 123

Index 20

index

- online help 21

Industrial IT licenses 138

IP address 90, 92

- class A 131
- class B 131
- class C 131
- host ID 132

IPConfig 90

## K

Keyword Search 20

## L

Language

- select 100

LD 171

Libraries 37

- AlarmEventLib 15
- BasicLib 15, 55
- CommunicationLib 15
- Control Library 15

local flag 132

## M

manuals 20

Message 47

message pane 56, 70

mismatch 144

MMS

- configuration 135

ModuleBus 82

multi-user configuration 19

## N

NetID 131

network area 132

network ID 132

## node

- client 134
- server 134

## node number 132

- RNRP 132

**O**

## online analysis 96 to 97

## Online Help 20

## online help

- index 21
- text search 21

## Online Manuals 20

## online mode 96 to 97

## OPC panel 30

## OPC Server 23

## OPC server for AC 800M

- Copying files to safe media  
SV 4.1 or SV 5.0.2 to SV 5.1 141

**P**

## parameters

- RNRP 132

## path number

- RNRP 132

## permission Full Control 105

## PLC Control Builder 23

## PLC control builder version 87

## PLC firmware 87

## ports

- Ethernet 95
- IP address 95

## program 33

## program editor 56

## project 33

## project documentation 49

## Project Explorer 14

## Project folder 19, 101

**R**

## real-time communication 134

## refresh 35

## RNRP

- node number 132
- parameters 132
- path number 132

## RNRP Service 141

## RNRP Setup Wizard 124

## RNRP-icon 124

**S**

## search

- online help 21

## Selecting

- file location 101
- language 100

## server

- node 134

## Service Account 110 to 111

## session log file 30

## Settings tab 95

## Setup Wizard 100

## SFC 171

## single-user configuration 23

## SoftController 34

## software model 79

## ST 171

## standard libraries 15

## straight-through 93

## structuring code 62

## subnetwork ID 132

## System folder 34

**T**

## Tasks 43

## templates 34

## test mode 71

- analysis 71

## Timers 59

TK212A cable 88, 169

## U

UNC path 106, 109

Upgrade firmware via serial line 87

upgrading projects 137

## V

variable conditions 73

version and online analysis 115

versions 115



# Contact us

## **ABB AB**

### **Control Systems**

Västerås, Sweden

Phone: +46 (0) 21 32 50 00

Fax: +46 (0) 21 13 78 45

E-Mail: [processautomation@se.abb.com](mailto:processautomation@se.abb.com)

[www.abb.com/controlsystems](http://www.abb.com/controlsystems)

Copyright © 2003-2010 by ABB.  
All Rights Reserved

3BSE041584-510

## **ABB Inc.**

### **Control Systems**

Wickliffe, Ohio, USA

Phone: +1 440 585 8500

Fax: +1 440 585 8756

E-Mail: [industrialitsolutions@us.abb.com](mailto:industrialitsolutions@us.abb.com)

[www.abb.com/controlsystems](http://www.abb.com/controlsystems)

## **ABB Industry Pte Ltd**

### **Control Systems**

Singapore

Phone: +65 6776 5711

Fax: +65 6778 0222

E-Mail: [processautomation@sg.abb.com](mailto:processautomation@sg.abb.com)

[www.abb.com/controlsystems](http://www.abb.com/controlsystems)

## **ABB Automation GmbH**

### **Control Systems**

Mannheim, Germany

Phone: +49 1805 26 67 76

Fax: +49 1805 77 63 29

E-Mail: [marketing.control-products@de.abb.com](mailto:marketing.control-products@de.abb.com)

[www.abb.de/controlsystems](http://www.abb.de/controlsystems)

Power and productivity  
for a better world™

