APPLICATION NOTE

AC500 ETHERCAT
DIAGNOSIS GUIDELINE

Contents

1 Introduction

1.1 Scope of the document

For any application, diagnostics is one of the most important aspects, both in the area of commissioning and during operation.

Due to the nature of the EtherCAT bus, diagnostics cannot be standardized and must be individually adapted to the requirements of the plant.

This application note is intended for system engineers and integrators who are using or planning to use the AC500 series in combination with the EtherCAT system of ABB. It describes general EtherCAT diagnostic features that comes along the Automation Builder.

Furthermore, the document does not only give an overview about the EtherCAT diagnostic features but also a guideline for diagnosis during Commissioning and Operational within the integrated Automation Builder diagnosis and the IEC programming.

This guideline is based on experiences and does not cover all error scenarios which can occurs with EtherCAT. Depending on Applications, Machineries and Devices the diagnostic can be different.

1.2 Compatibility

The application note explained in this document have been used with the below engineering system versions. They should also work with other versions, nevertheless some small adaptations may be necessary, for future versions.

- AC500 V3 PLC
- Automation Builder 2.3.0 or newer

2 EtherCAT basics

The analysis and implementation of EtherCAT diagnostics requires the understanding of the general EtherCAT behavior. Therefore, this chapter will give a short overview about:

- EtherCAT topology and communication
- EtherCAT state machine
- EtherCAT slave – Telegram processing
- EtherCAT basic diagnostic terms

These chapters will explain the necessary part for the EtherCAT diagnostics with Automation Builder and the AC500 and does not fully cover the complete System description. The description is based on the AC500 products but can also be applied to other devices.

2.1 EtherCAT communication and topology

Communication of EtherCAT is based on the Ethernet Cables to serve a modern way for automation process. Medium for the communication is 100BASE-TX, 100BASE-FX and EBUS, whereby the latter is used for Beckhoff specific device internal communication.

AC500 uses the 100BASE variants which are physically implements into the EtherCAT-Slave-Controller (ESC). Media Independent Interface (MII) serves as general interface for the ESC with which consistent and topology-independent signals can be generated.

This allows different Topologies with EtherCAT, just like:

- Line – Topology
- Star – Topology (only with special EtherCAT switches or star connectors)
- Ring – Topology (only for masters with more than one EtherCAT port)

The AC500 EtherCAT System, does only supports the Line – Topology, this Topology is based on Daisy Chain. Daisy Chain describes in series connected devices whereby the first device is directly connected to a computer system. For EtherCAT this computer system is the EtherCAT master CM579-ETHCAT and the AC500 PLC.

The following picture shows a simple Daisy Chain Topology with the communication interfaces of ABB (**Figure 1**).

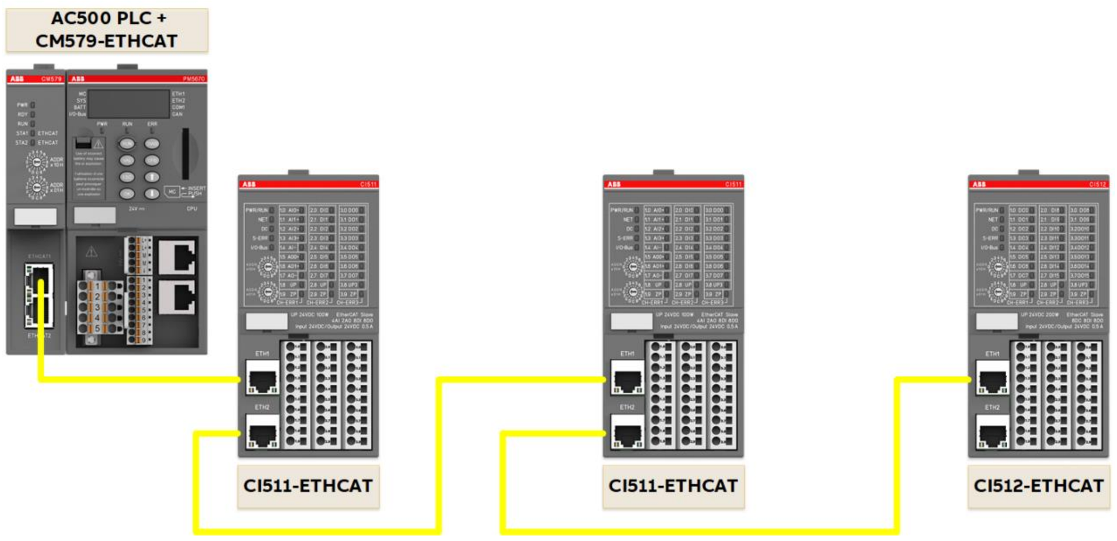


Figure 1 - Daisy Chain Topology with AC500 Products

In a combination with different devices by third party manufacturers additional Line-Topologies or Stub Line Topologies can be added to receive a combination of Line and Daisy Chain Topology (**Figure 2**).

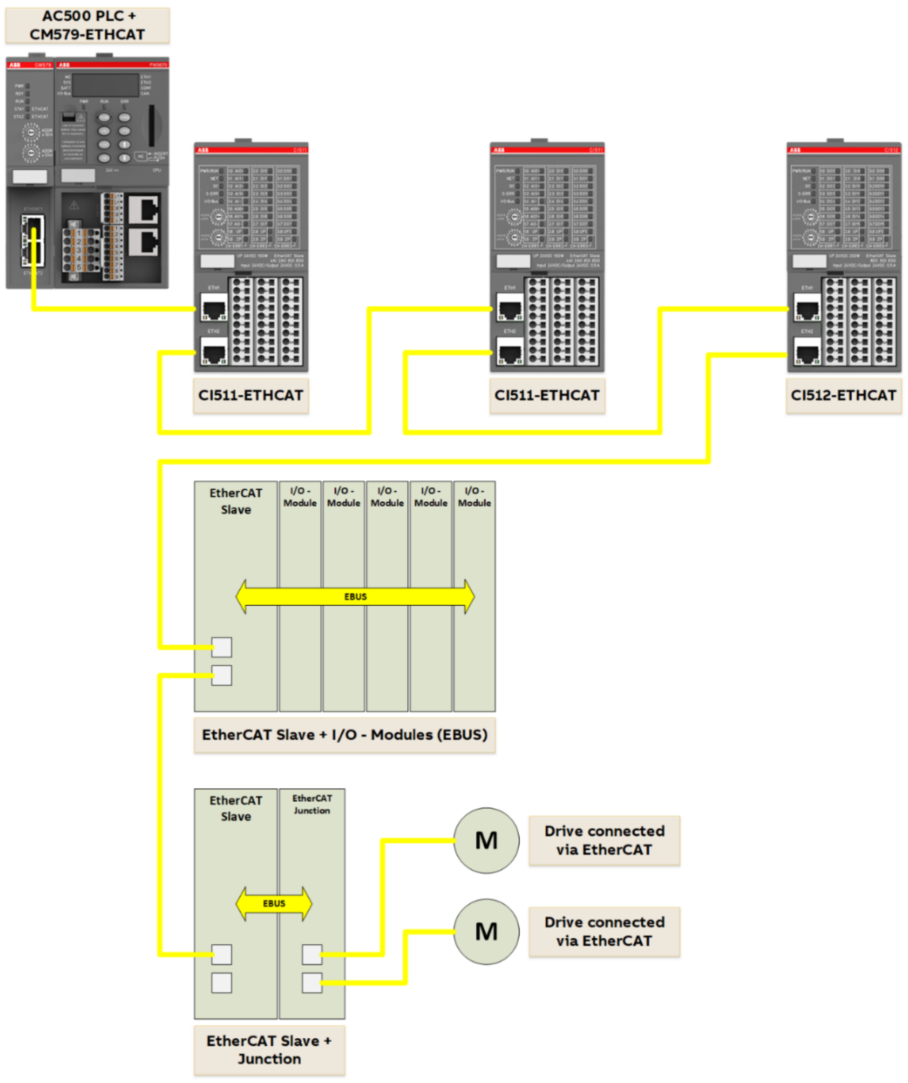


Figure 2 - Tree Structure Topology with AC500 Products and Third-Party Devices

For the real-time behavior of EtherCAT, the communication is based on a telegram which is predefined according to the configuration which exchanges the in- and output data on the fly with the slaves.
For this reason, a correct topology of EtherCAT is required to match the corresponding datagram with the correct slaves. The data will be exchanged with the slave during the telegram is passed by (**Figure 3**).

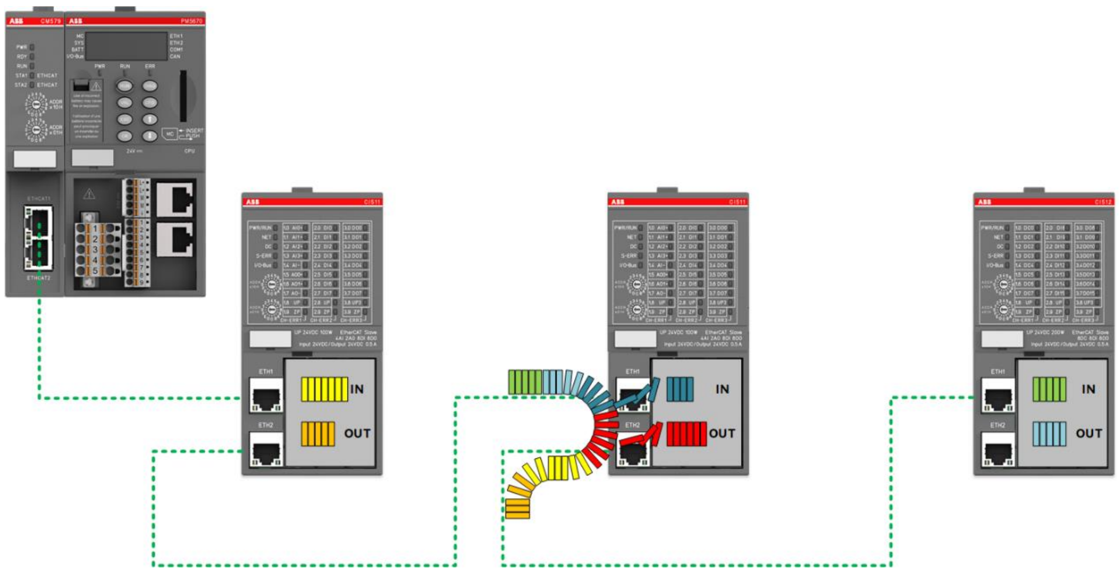


Figure 3 - EtherCAT Telegram behaviour

2.2 EtherCAT Slave – telegram processing

An EtherCAT Slave consists of an EtherCAT slave controller (ESC) for processing the telegram in a fast way and to route it to the next slave.

In general, the communication of EtherCAT is based on a Full-Duplex-Mode and the in- and outcoming data will be handled by the ESC. The processing direction takes place at one data lane, coming from the master, going to the slaves. Only in this direction, the telegram will be passed through the processing unit of the EtherCAT slave. The following picture shows an ESC with Link to all ports and the processing direction from port 0 to port 3, where the Processing Unit will be passed (**Figure 4**).

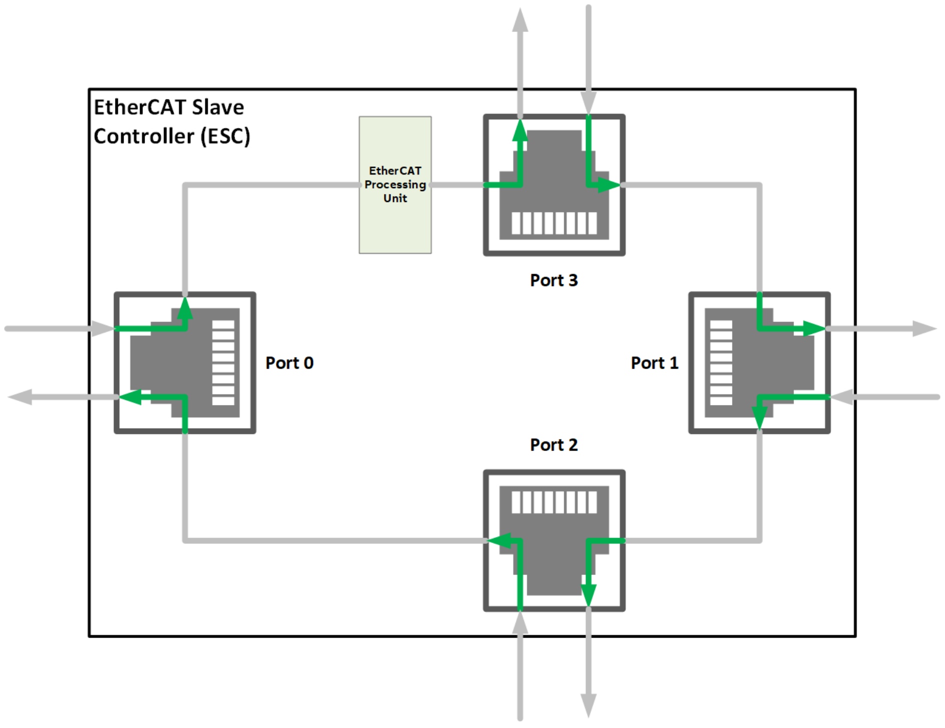


Figure 4 - EtherCAT Slave Controller

Looking at the Tree Structure – Topology:

Due to the Daisy Chain Topology, the last Slave does not have Link at port 1. The telegram will be passed through this port within the Loopback functionality (**Figure 5**) and therefore re-turned at this slave. Port 3 and port 2 are used for stub lines, where the last slave of this line has active Loopback functionality too. This is applying to any line and for every last slave of a Tree Structure Topology.

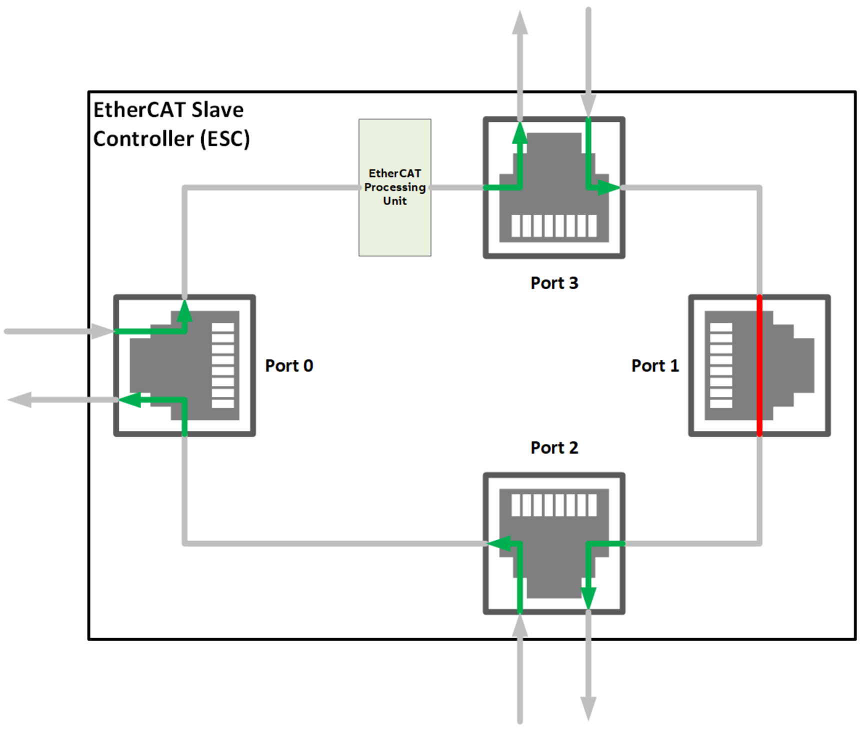


Figure 5 - EtherCAT Slave Controller with Loopback function

2.3 EtherCAT state machine

During the startup of the EtherCAT System, every EtherCAT device needs to go through the EtherCAT State Machine, to indicates available functionalities. These states and the corresponding functionalities are shown in the following picture (**Figure 6**):

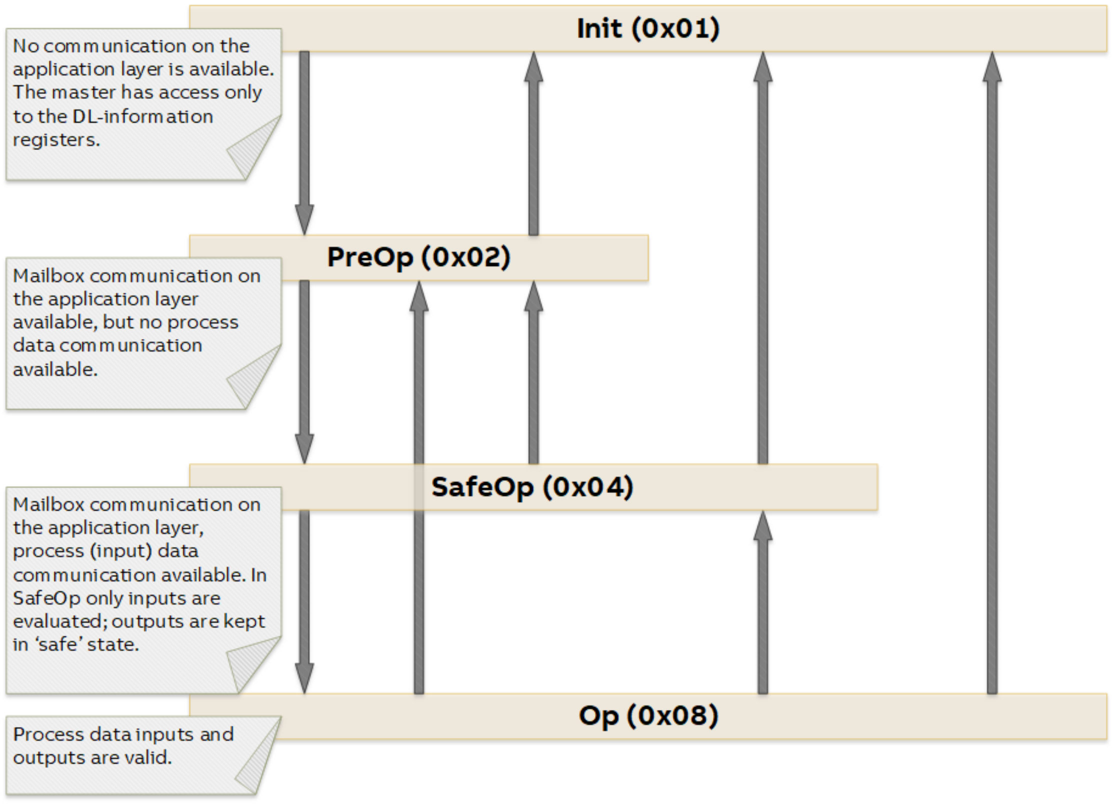

Init (0x01)
No communication on the application layer is available. The master has access only to the DL-information registers.
PreOp (0x02)
Mailbox communication on the application layer available, but no process data communication available.
SafeOp (0x04)
Mailbox communication on the application layer, process (input) data communication available. In SafeOp only inputs are evaluated; outputs are kept in 'safe' state.
Op (0x08)
Process data inputs and outputs are valid.

Figure 6 - EtherCAT State Machine[1]

Only when all devices are in Operational (Op), the EtherCAT system is running properly. A faulty configuration or an error during the startup at one of the states, can causes a not working EtherCAT system. The diagnostic can be used to determine the issue.

2.4 EtherCAT basic diagnostic Terms

EtherCAT diagnostic is specified by the EtherCAT Technology Group (ETG)[2], an official partner of IEC Standardization to define industry specific device profiles and guidelines for EtherCAT.

The following EtherCAT diagnostic terms are specified by the ETG and are explained for the understanding of this document based on the AC500 products or the Automation Builder.

Of course, this explanation can be transferred to other EtherCAT devices of other manufacturers as well.

[1] EtherCAT Slave Implementation Guide - https://www.ethercat.org/download/documents/ETG2200_V3i1i0_G_R_SlaveImplementationGuide.pdf

[2]The EtherCAT Technology Group (ETG) is an official partner of IEC Standardization to define industry specific device profiles and guidelines for EtherCAT. (https://www.ethercat.org/default.htm)

2.4.1 Working Counter (WKC)

The so-called **Working Counter (WKC)** is a counter that is implemented into each datagram as a 16 Bit variable. This WKC variable will be increased by each addressed and proper working slave. Based on a previously defined value of the EtherCAT master, the returned WKC will be compared and validated. The result of this comparison returns the state of the communication, functional or non-functional, i.e. FALSE or TRUE (**Figure 7**).

If the WKC is equal to the predefined value, the system is running correctly.

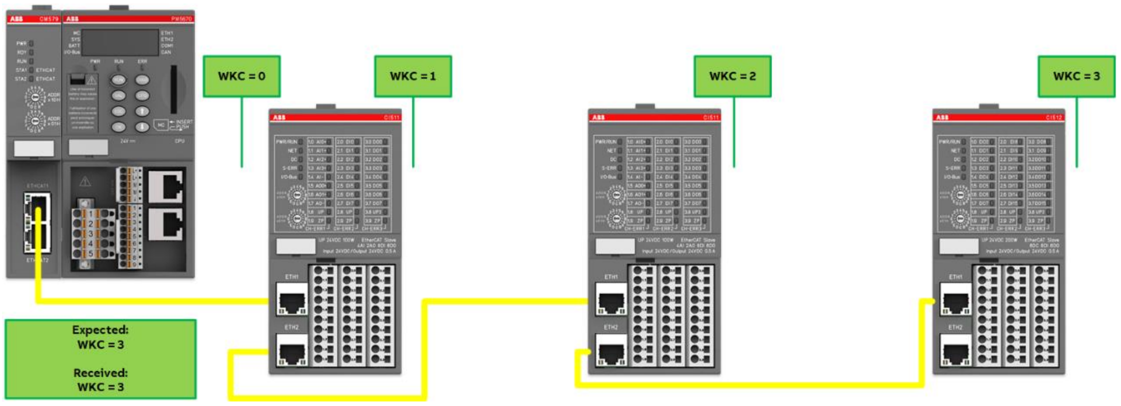


Figure 7 - Valid Working Counter

The other way around, if the WKC is not equal to the predefined value, there seems to be a mistake in the system (**Figure 8**).

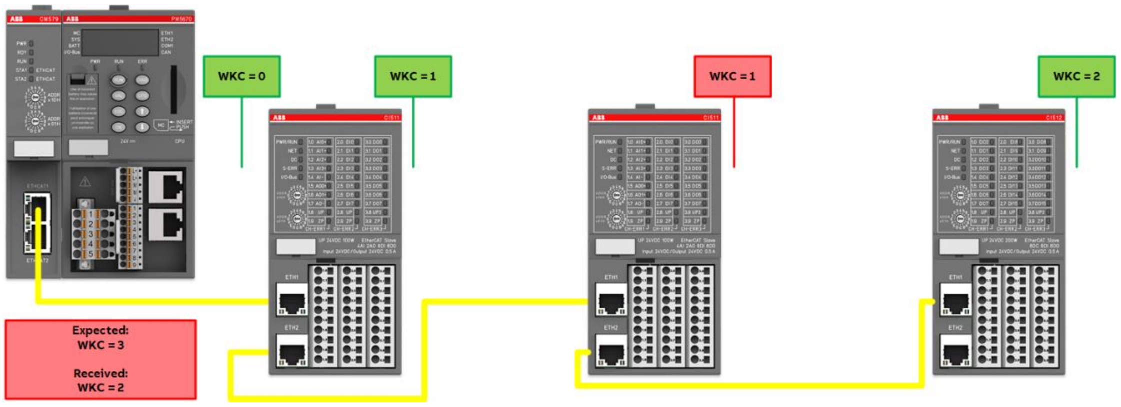


Figure 8 - Invalid Working Counter

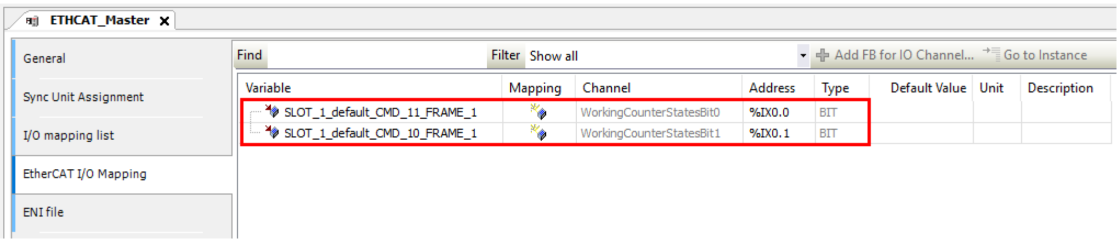
Resulting from this:

$$WKC = expected\ value \rightarrow valid, FALSE$$

$$WKC \neq expected\ value \rightarrow invalid, TRUE$$

As the WKC is implemented into the telegram and can be validate in every Interrupt Task, it is the most significant information to detect a faulty communication, localization is not possible. This WKC can be used as real-time capable diagnostic tool. **To obtain a reliable statement about the state of the system, the WKC must be queried in the interrupt Task of the IEC programming.**

In Automation Builder, two WKC's are created for each Sync Unit automatically if the Sync Unit contains In- and Output data. To use these WKC's inside IEC programming these variables can be mapped inside the I/O Mapping List (**Figure 9**).



Variable	Mapping	Channel	Address	Type	Default Value	Unit	Description
SLOT_1_default_CMD_11_FRAME_1		WorkingCounterStatesBit0	%IX0.0	BIT			
SLOT_1_default_CMD_10_FRAME_1		WorkingCounterStatesBit1	%IX0.1	BIT			

Figure 9 - Working Counter for V3 PLC

Summary:

1. Defined and implemented by the Specification into each EtherCAT datagram
2. Detection of EtherCAT communication errors
3. Should be observed inside the Interrupt Task to receive communication error in real-time behavior
4. Two WKC's for each Sync Unit available (In- and Out- Frame)

	Note: At each startup of the EtherCAT system, the slaves will be configured by the master and therefore the WKC is always invalid. For this reason, the WKC should only be monitored while the bus is running.

2.4.2 Master Diagnosis

2.4.2.1 Master State and communication error

EtherCAT is a Master-Slave System and typically the Master has the control over the whole System. During the boot up of the System, the Master distributes addresses to the Slave and configures them according to the configuration which is written down inside the EtherCAT Network Information (ENI).

The configuration of the EtherCAT bus is conform with the EtherCAT state machine. As the Master is the overhead of the System, the states of the Slaves will be determined by the Master. Reversed, the Master state is also depending on the Slaves states, it does only go to the next state as soon as each Slave is in the same state as the Master itself.

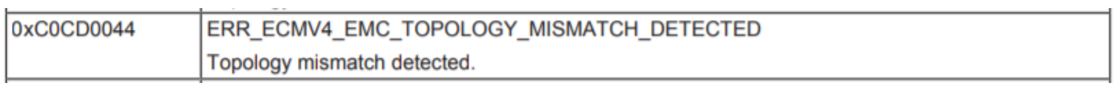
Therefore, a first indication of a working EtherCAT System can be the masters state according to the EtherCAT state machine.

The AC500 and the CM579-ETHCAT Module supports additional states to indicates faults and errors next to the default EtherCAT states. These states are:

Init + Error	0x11
PreOp + Error	0x12
SafeOp + Error	0x14

To receive further information about an error, the master serves information about actual communication statuses as a hexadecimal code. With the Online Help (PLC Automation with V2 CPUs > Diagnosis and debugging for AC500 V2 products > Diagnosis messages > Communication modules diagnosis > CM579-ETHCAT) of Automation Builder, this code can be decoded.

E.g. in case of a Topology error during start-up of the System, the master will throw the error code *0xC0CD0044* which says (**Figure 10**):



0xC0CD0044	ERR_ECMV4_EMC_TOPOLOGY_MISMATCH_DETECTED Topology mismatch detected.

Figure 10 - Communication error - Topology mismatch

2.4.2.2 Lost frame counter

Although EtherCAT uses the Ethernet frame, not every device that receives Ethernet telegrams can act as an EtherCAT slave. While the telegram passes through a functional EtherCAT station, it reads the data intended for it and writes some data back. At the end of the EtherCAT bus, the telegram is sent back and passes through each slave again according to the EtherCAT slave controller implementation which is described in chapter *EtherCAT Slave – telegram processing*. This means that the master expects a return telegram with certain information about the bus and the slaves. If a return telegram is missing, he will increase an internal counter, the so-called **Lost frame counter**.

Missing telegrams can be caused by:

- Faulty EtherCAT Slaves – single telegram or multiple telegrams will not be forwarded
- Non EtherCAT attendee is part of the bus – as Ethernet devices doesn’t return the telegrams at all, the master will not receive any telegram back

The loss of individual telegrams can cause faulty information during EtherCAT diagnostics. But it’s more likely that all telegrams are rejected. This is one of the worst cases as no more diagnostic can be performed due to a missing loopback functionality. The diagnostic information do not return to the master and therefore they cannot be evaluated.

To validate and analyze the Lost Frame Counter, it should be monitored over a period to get the growth of the counter. The number of lost frames in that period, is decisive for further diagnostic measures.

Summary:

1. Increasing for every missing telegram at the master
2. Analysis and evaluation of the speed of the rising counter
3. Detection of Non-EtherCAT devices or faulty Slaves

2.4.2.3 Bus and topology scan

As already mentioned in the *EtherCAT communication and topology* the correct topology is required for a proper working EtherCAT bus. During commissioning of failure search, the complete bus can be scanned for the connected devices. According to the EtherCAT Technology Group, each Slave has implemented several information that can be read by the master.

- Vendor ID
- Product Code
- Revision Number
- Serial Number
- Port State
- Available Ports
- Previous Device
- Previous Port

This allows to scan the bus for available devices according to the Vendors ID and Product Code.

With the additional information about the available ports and those states the topology can be scanned as well. The information about the previous ports and devices enables the possibility to check for reversed ports which are difficult to find with the visual troubleshooting of the physical Ethernet ports.

If the lost frame counter is increasing fast and every telegram is rejected by a faulty device or a non EtherCAT attendee, the scan of the Bus and topology is not possible. In this case the Emergency scan must be execute.

Summary:

1. Scan of available devices inside the EtherCAT network
2. Detection and localization of reversed ports
3. Not possible when all telegrams are rejected

⚠	CAUTION! Bus and topology scan will reset the system. As this is not a common function during operational and can causes unexpected issues, the scan is allowed at the Init state only.

2.4.2.4 Emergency scan

Coming with the name, the Emergency scan is needed for emergency scenario where no diagnostic can be done. One scenario was already described in *Bus and topology scan* – e.g. a Non EtherCAT slave was added to the EtherCAT network, all telegrams will be rejected and therefore no diagnostic is possible. With the emergency scan position of the Non EtherCAT slave can be determined.

The localization of the position with the emergency scan works as described in the following:

The master sends out an emergency telegram, which will disable port 0 of the devices. Starting with the last device in the Topology, going forward to the first one.

As soon as the Port 0 of the faulty or non EtherCAT device was disabled, the telegrams will be returned from the previous slave according to the EtherCAT standard and the position is located.

Summary:

1. Required when diagnostic is no longer possible (all telegrams will be rejected)
2. Disables port 0 of the Slave to localize the faulty device

⚠	CAUTION! Emergency scan will reset the system. As this is not a common function during operational and can causes unexpected issues, the scan is allowed at the Init state only.

2.4.3 Slave Diagnosis

The EtherCAT Technology Group (ETG) has defined that each slave has its own diagnostic information to obtain a supplier independent diagnostic system for EtherCAT.

Thereby the information contains:

- **Slave name** – according to the configuration and <u>not</u> according to the Product Code
- **Slave state** – according to the EtherCAT State Machine (ESM) with the additional States for Init/ PreOp/ SafeOp + Error
- **Communication state** – similar to the communication error of the Master diagnosis, the error code for the communication state can be decoded with the Online Help of Automation Builder
- **CoE emergencies –** Supplier specific diagnostic information about slave IO’s

2.4.3.1 CoE emergencies (Diagnosis Data)

CoE (CAN over EtherCAT) emergencies are messages that will be sent from the slave to the master in case that an undefined condition occurs.

For the Communication Interfaces CI511-ETHCAT and CI512-ETHCAT, these undefined conditions might be modular channel error’s, e.g. a cut wire at an analog Input of the slave. This also counts for additional I/O modules which will be extended to the CI511-ETHCAT or the CI512-ETHCAT.

One emergency message is made up of 8 BYTES:

- Error Code (**2 BYTES**)
- Error Register (**1 BYTE**)
- Error Data (**5 BYTES**)

A maximum number of 5 emergencies can be buffered in the master at the same time for each slave. Once the emergencies are read out, the message will be discarded. If the failure persists, the slave will send out the emergency again every 10 seconds.

	The CoE emergency message is supplier specific and must be decoded by the device specific manual.

For the S500 series of ABB, which includes the CI511-ETHCAT and CI512-ETHCAT modules as well, it is possible to display the emergencies as S500-Format. This Format consists of 5x 5Bytes that can be decoded via the Online Help of Automation Builder.

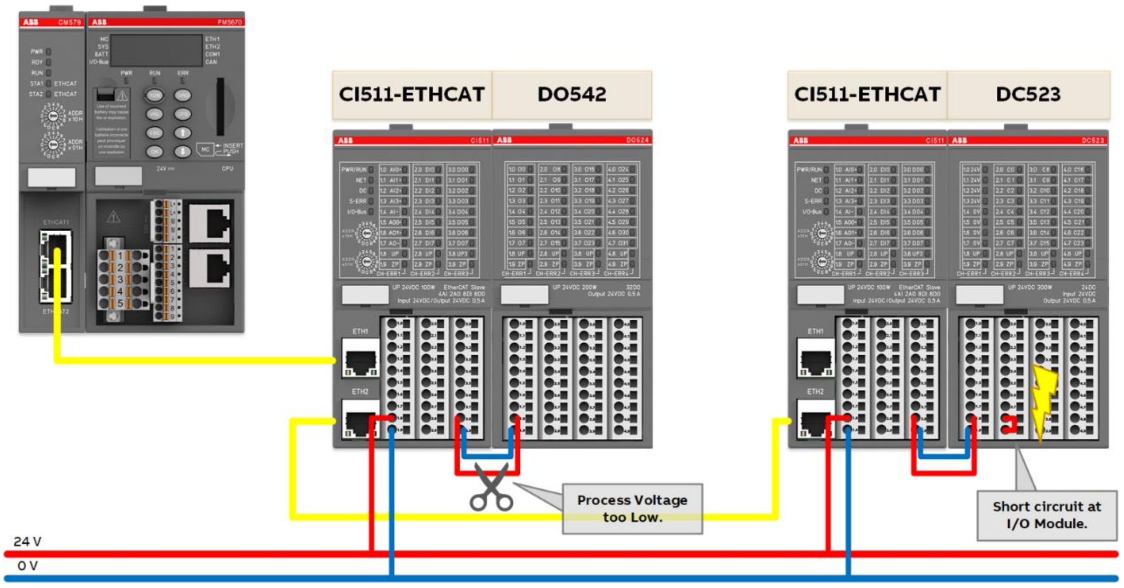


Figure 11 - CoE Emergencies for S500 series of ABB

One diagnosis data for S500 format consists of the following 5 Bytes:

Byte 1:	**Error class** (1…4)
Byte 2:	**Slave number** (Device number) within one cluster where 31 stands for the CI51x itself
Byte 3:	**Module number** (indicates whether Digital/ Analog in-/ output have some errors)
Byte 4:	**Channel number** which has the error of the specified device
Byte 5:	**Error number** that gives detailed information of the fault

Summary:

1. Detection of modular channel errors (e.g. Cut wire)
2. Supplier specific code (must be check in the device manual)
3. Up to 5 emergencies can be stored into the master for each slave

2.4.3.2 Lost link counter (optional)

Each time, the physical connection of a port at a slave is getting lost, the **Lost link counter** is increasing for this slave at the specific port number. During a “Link Down” the port states changes which causes the lost link counter to increase.

Port 0 checks for incoming Frame, while the other ports increasing the counter with the change from “Link” to “Loopback function”.

The functionality of the lost link counter is optional and therefore it is manufacturer specific if a slave supports this functionality or not. The CI511-ETHCAT and CI512-ETHCAT do not support the Lost Link Counter functionality but with the **Firmware version >2.8.1** the **RX Counter** will display the value **255** instead (**Figure 12**).

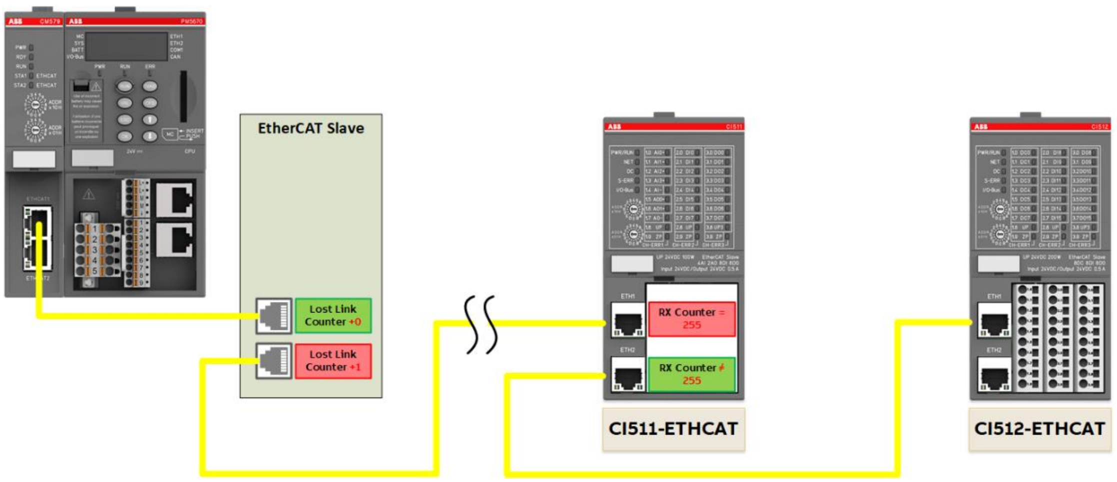


Figure 12 - Lost Link Counter with a general EtherCAT Slave and with CI511-ETHCAT/ CI512-ETHCAT

Summary:

1. Optional functionality
2. Increasing for every physical loss of connection
3. Detection of loose, broken, or physical disconnection of cables
4. Not support by CI511-ETHCAT and CI512-ETHCAT

Note:

The lost link counter is defined as an optional function in the EtherCAT specification. Not every slave supports this lost link counter.

CI511-ETHCAT and **CI512-ETHCAT** modules do **not** support the lost link counter but the RX counter will set to 255 instead.

2.4.3.3 RX counter

The RX counter increases in case of corrupt frames and thus frames discarded by the slave. Each port of a slave has its own RX counter. Loose cables or faulty telegrams, caused by EMC problems or defective devices, can be detected by the RX counter. The counters can be used not only for detection but also for localization of the error and fault.

This RX counter consists of two different counters, the **Physical layer error counter** and the **Frame error counter**. Whereby the Physical layer error counter indicates invalid information data inside and outside the frames. The Frame error counter shows that the Bit-Sequence is corrupted which does only occurs inside the frame (**Figure 13**).

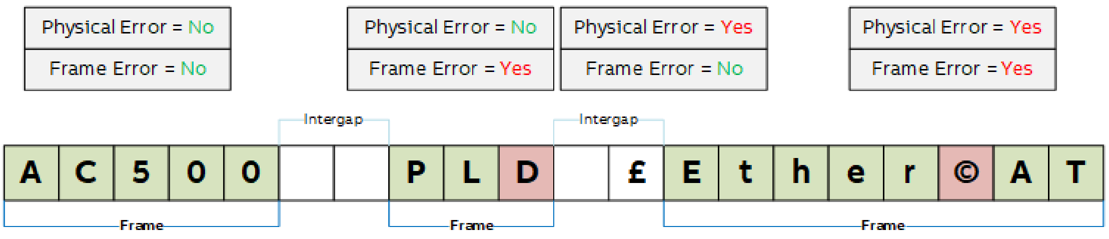


Figure 13 - EtherCAT Frame with RX errors

In general, following faults can be detected with the help of RX counter:

- EMC issues (usually sporadic increasing)
- Faulty devices or connections (usually fast and systematic increasing)

To localize the fault, the increasing RX counters must be validated. If two slaves in immediate succession show an error on the connected ports, there will most likely be an error between both slaves. This error then might be caused by external influences or faulty connections (**Figure 14**).

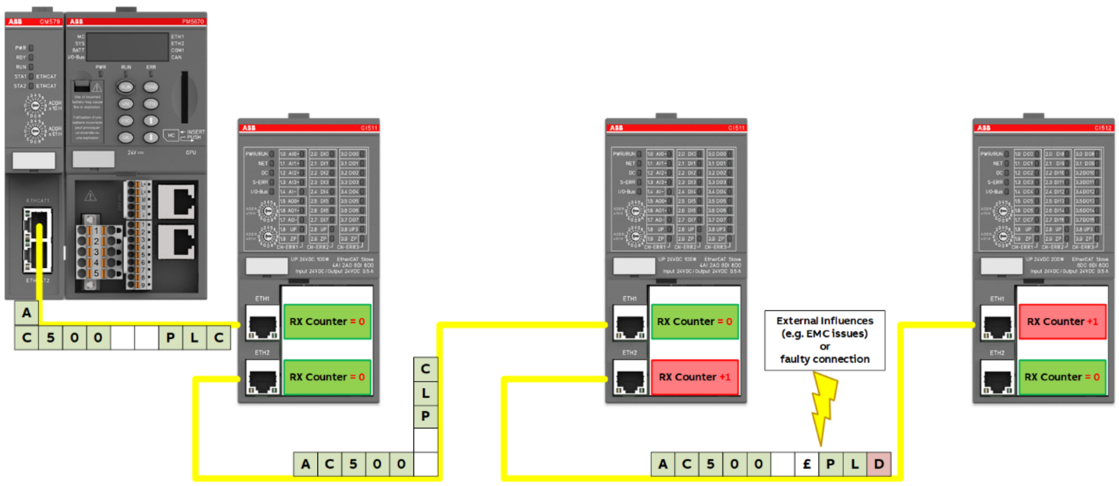


Figure 14 - RX counter with external influences or faulty connection

Another scenario might be that not the next slave but the second one counts RX errors, then the fault could be also caused by the slave in between (**Figure 15**).

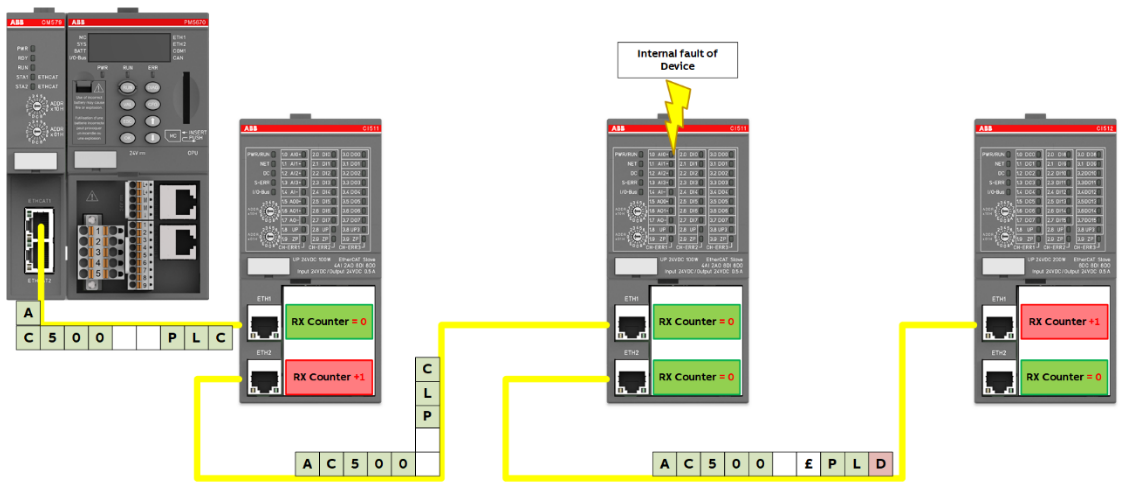


Figure 15 – RX counter with internal fault of device

As not every communication is completely faultless, sporadic increments of the counters in a period of days or weeks do not mean that a fault or error is pending. Systematic or fast increasing counters, which can be determined in minutes or seconds, indicates hardware or external faults and should be monitored by the system.

Summary:

1. Consists of **Physical-layer error counter** and **Frame error counter**
2. Detection and localization of EMC issues or faulty connection and hardware
3. Validation of systematic or sporadic increments in a period of minutes and seconds

Note:

When booting the system or immediately after restarting a slave, the frame error counter may also increase. The counter should only be monitored in a running system (operational).

3 Application scenarios of EtherCAT diagnostics

In this chapter, typical faults and errors are listed which can be determined, detected and located by the EtherCAT bus.

The errors and faults are classified by **Commissioning** and **Operational** of the machinery. The list is based on practical experience with EtherCAT and consists predefined faults and errors derived from the ETG specification.

Note:

This document lists only a small selection of errors that can occur in an EtherCAT system. It is only intended to provide an exemplary overview of the various error groups. Depending on the application and installation, further errors and error groups may occur.

3.1 Commissioning

Especially during commissioning, the topology errors can occur frequently. With completion of the hardware setup, the topology errors are minimized and communication errors can emerge. Of course, some topology errors might be also caused by communication error, e.g. a loose connection of an EtherCAT device which leads to an incomplete system.

3.1.1 Topology error

During the transition from ***Bus Off*** to ***Init***, i.e. during the start-up of the EtherCAT bus, the topology of the system will be checked by the master according to the configuration. The correct topology is mandatory for the proper operation of the bus. Normally, a topology error only occurs during commissioning.

In case of a specific application where single devices must shut down and replaced, these errors can occur as well.

Error or fault	Description
Additional devices	More devices installed than configured.
Missing devices	Less devices installed than configured.
Missing cable between two devices	This is basically the same behavior as the "Missing devices" error. All devices behind the missing cable are physically not available and therefore there are less devices installed than configured.
Reversed devices	Different types of EtherCAT Slave module, e.g. CI512-ETHCAT instead of a CI511-ETHCAT.
Reversed devices within one cluster	Two additional IO Modules in a CI51x-ETHCAT cluster are swapped.
Reversed devices of same type but with predefined station address	CI511 module with predefined station address 1001 is swapped with a CI511 with the address of 1002.
Reversed connection at In/Out Ports	During commissioning a reversed connection of the IN and OUT ports will cause that this device will be set as the last slave of the topology due to the different telegram processing inside the slave (see chapter 2.2)

Table 1 - Topology errors

3.1.2 Communication error

A communication error is an error when writing or reading the telegram and can therefore occurs at startup or operation. The correct operation of the communication is decisive for a functioning bus. In case of a communication error, further errors can be caused.

Error and fault	Description
Telegram error	E.g. Defect cable or port, EMC interferences.
Loose connection	Loose connection of EtherCAT cable or supply voltage.
Inadmissibly long or non-deterministic forwarding times	Faulty EtherCAT Slave Controller implementation or non-EtherCAT switch
Faulty device/slave	Loosing telegrams (e.g. telegrams will be rejected due to a broken line inside the slave) or telegram are not executed correctly.
Non-EtherCAT device	Loosing telegrams (The telegrams will not be returned, and therefore the telegrams are rejected).

Table 2 - Communication errors

3.2 Operational

Usually, topology errors do not occur during a running system unless the application requires a hardware change during operation or a person willfully changes the topology.

Accordingly, only a communication error or a device, module or channel error can lead to a necessary diagnostic during operation.

3.2.1 Device-, Module-, Channel- error

A Device-, Module- or Channel error can be defined as an internal error of a EtherCAT slave cluster. For example, the communication interface CI511-ETHCAT acts as EtherCAT slave itself with input and output data and serves as an interface to additional input/output modules. Each of these devices and modules can have faults at each of their channels which of course needs to be diagnosed. Typically, every vendor has its own diagnostic principle. Therefore, these kinds of error are partly predefined in the ETG specifications as emergency messages and can be read if the slave supports CoE (Can over EtherCAT) communication.

If an emergency error occurs during the startup of the bus, the state change might be interrupted and a proper starting bus is not guaranteed. During operational the emergency messages do not affect the EtherCAT communication at all.

Error and fault	Description
Missing module	E.g. Cluster with two additional IO modules are configured but only one module is installed.
Faulty module	Faulty, broken, defect IO module.
Wrong module	Wrong type of IO module is installed.
Missing process voltage	Process voltage for one or more IO modules is missing.
Cut wire, short circuit or wire break at IO module	Cut wire, short circuit or wire break at one or more than one channels of the IO module.

3.2.2 Communication error

A communication error is an error when writing or reading the telegram and can therefore occurs at startup or operation. The correct operation of the communication is decisive for a functioning bus. In case of a communication error, further errors can be caused.

Error and fault	Description
Telegram error	E.g. Defect cable or port, EMC interferences.
Loose connection	Loose connection of EtherCAT cable or supply voltage.
Inadmissibly long or non-deterministic forwarding times	Faulty EtherCAT Slave Controller implementation.
Faulty device/slave	Loosing telegrams (e.g. telegrams will be rejected due to a broken line inside the slave) or telegram are not executed correctly.
Non-EtherCAT device	Loosing telegrams (The telegrams will not be returned, and therefore the telegrams are rejected).

4 Diagnostic with Automation Builder

4.1 Diagnostic tools

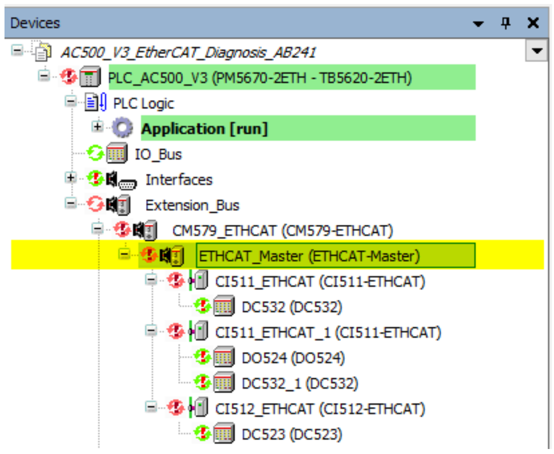
Since the release of Automation Builder 2.2.4 (631), the Engineering Tool for the AC500, includes interfaces to read basic diagnosis information for commissioning purposes without any application effort.

This diagnostic has been improved over different Automation Builder Versions and is now completely implemented. This Guideline will explain the diagnostic interfaces since **Automation Builder 2.3.0** and above.

Note:

Some of the diagnostic parts may be different or not available in previous Automation Builder versions. To use the implemented diagnostics for EtherCAT, at least Automation Builder 2.3.0 is required.

Independent of the PLC type, the diagnosis can be found at the tab *ETHCAT_Master (ETHCAT-Master)* and is only available in **Online** mode (**Figure 16**).



```
Devices
AC500_V3_EtherCAT_Diagnosis_AB241
  PLC_AC500_V3 (PM5670-2ETH - TB5620-2ETH)
    PLC Logic
      Application [run]
    IO_Bus
    Interfaces
    Extension_Bus
      CM579_ETHCAT (CM579-ETHCAT)
        ETHCAT_Master (ETHCAT-Master)
          CI511_ETHCAT (CI511-ETHCAT)
            DC532 (DC532)
          CI511_ETHCAT_1 (CI511-ETHCAT)
            DO524 (DO524)
            DC532_1 (DC532)
          CI512_ETHCAT (CI512-ETHCAT)
            DC523 (DC523)
```

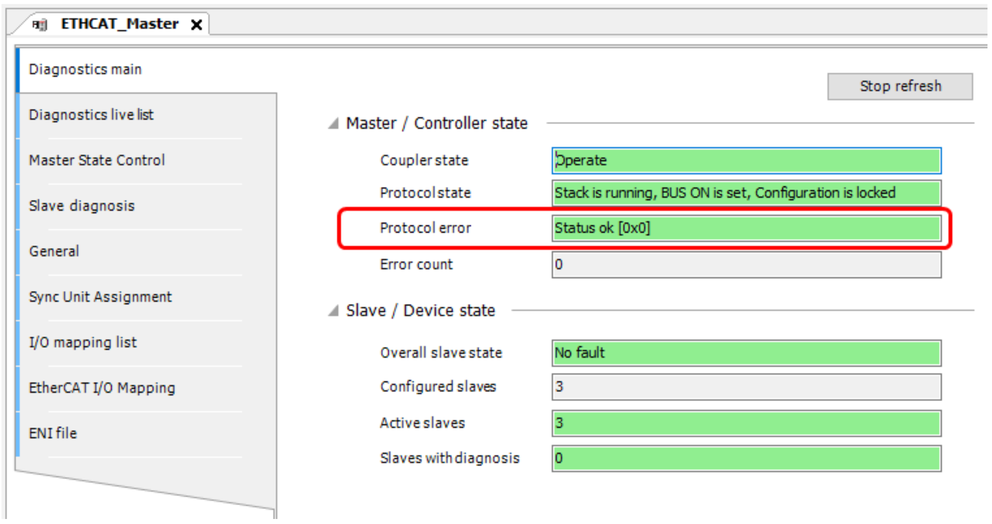
Figure 16 - Automation Builder diagnostic at ETHCAT_Master

The following chapter will give an overview about the different diagnosis tools in Automation Builder.

4.1.1 Diagnostics main

As general overview of the System, the Diagnostics main tab can be used. Accordingly, to the EtherCAT state machine (see also Chapter *2.3*) the master and all slaves have to pass through all states for an operational bus. In the **Diagnostics main** tab, the Master and the Slave states are displayed and can be used to determine an error but not to locate the faulty device.

If the system goes to operational (OP), without having any error at the Master or the Slave, all display indications must be green.



Master / Controller state

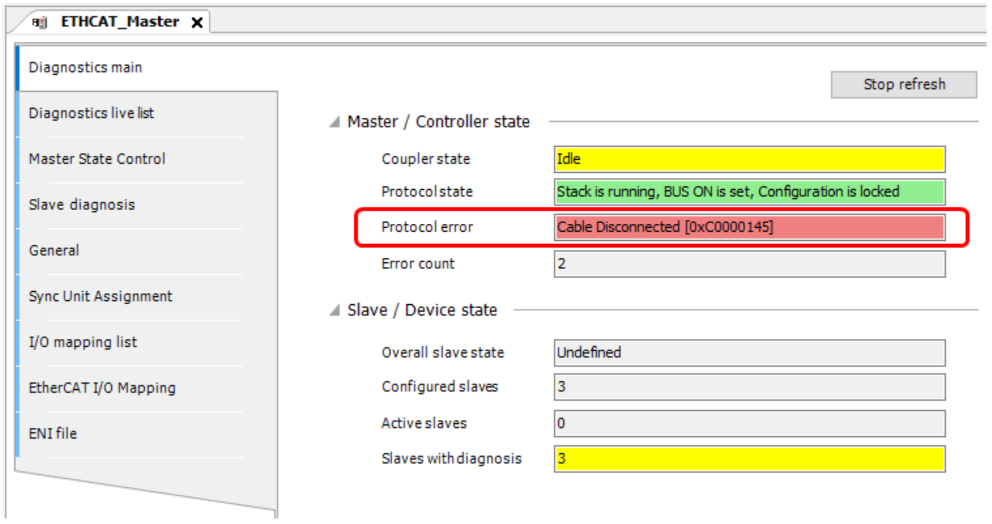
Coupler state	Operate
Protocol state	Stack is running, BUS ON is set, Configuration is locked
Protocol error	Status ok [0x0]
Error count	0

Slave / Device state

Overall slave state	No fault
Configured slaves	3
Active slaves	3
Slaves with diagnosis	0

Figure 17 - Diagnostics main when System is in OP

In case of an error, the display indications change their color and the communication fault can be seen inside the **Protocol error** tab.



Master / Controller state

Coupler state	Idle
Protocol state	Stack is running, BUS ON is set, Configuration is locked
Protocol error	Cable Disconnected [0xC0000145]
Error count	2

Slave / Device state

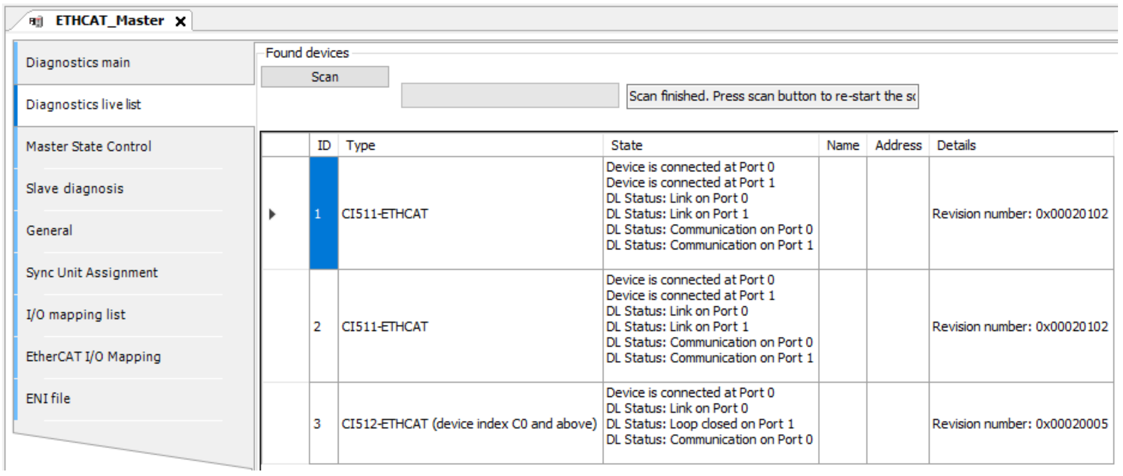
Overall slave state	Undefined
Configured slaves	3
Active slaves	0
Slaves with diagnosis	3

Figure 18 - Diagnostics main when System detects error

Having a look to the Slave/ Device state, it is noticeable that the number of configured and actives slaves does not match each other. It is obvious that the connection from the master to the first slave already has an error.

4.1.2 Diagnostics live list

In case of a topology error the **Diagnostics live list** will list all found devices in their actual topology after pressing “Scan”. Beside the name, it will also display the state of the ports and the revision number of the devices.



Scan finished. Press scan button to re-start the sc

ID	Type	State	Name	Address	Details
1	CI511-ETHCAT	Device is connected at Port 0 Device is connected at Port 1 DL Status: Link on Port 0 DL Status: Link on Port 1 DL Status: Communication on Port 0 DL Status: Communication on Port 1			Revision number: 0x00020102
2	CI511-ETHCAT	Device is connected at Port 0 Device is connected at Port 1 DL Status: Link on Port 0 DL Status: Link on Port 1 DL Status: Communication on Port 0 DL Status: Communication on Port 1			Revision number: 0x00020102
3	CI512-ETHCAT (device index C0 and above)	Device is connected at Port 0 DL Status: Link on Port 0 DL Status: Loop closed on Port 1 DL Status: Communication on Port 0			Revision number: 0x00020005

Figure 19 - Diagnostics live list (AB2.2)

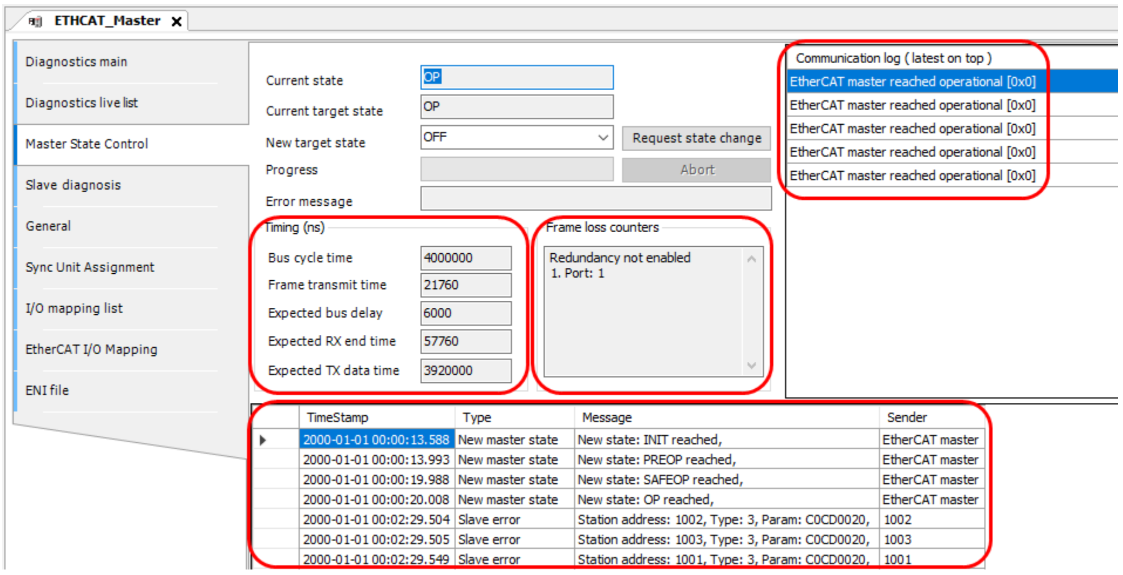
By comparing the scanned list with the configuration in the device tree, the incorrectly installed device and thus the topology error can be detected.

CAUTION!

Bus and Topology Scan will reset the System. As this is not a common function during operational and can causes unexpected issues, the scan is allowed at the **Init** state only.

4.1.3 Master State Control

The state machine of the EtherCAT bus can be changed by the **Master State Control** register. Within this state control, the Master as well as the Slaves can be set to a desired state to start, initialize, or stop the entire system.



Current state	OP
Current target state	OP
New target state	OFF
Progress	
Error message	

Timing (ns)

Bus cycle time	4000000
Frame transmit time	21760
Expected bus delay	6000
Expected RX end time	57760
Expected TX data time	3920000

Frame loss counters

Redundancy not enabled
1. Port: 1

Communication log (latest on top)

EtherCAT master reached operational [0x0]
EtherCAT master reached operational [0x0]
EtherCAT master reached operational [0x0]
EtherCAT master reached operational [0x0]
EtherCAT master reached operational [0x0]

TimeStamp	Type	Message	Sender
2000-01-01 00:00:13.588	New master state	New state: INIT reached,	EtherCAT master
2000-01-01 00:00:13.993	New master state	New state: PREOP reached,	EtherCAT master
2000-01-01 00:00:19.988	New master state	New state: SAFEOP reached,	EtherCAT master
2000-01-01 00:00:20.008	New master state	New state: OP reached,	EtherCAT master
2000-01-01 00:02:29.504	Slave error	Station address: 1002, Type: 3, Param: C0CD0020,	1002
2000-01-01 00:02:29.505	Slave error	Station address: 1003, Type: 3, Param: C0CD0020,	1003
2000-01-01 00:02:29.549	Slave error	Station address: 1001, Type: 3, Param: C0CD0020,	1001

Figure 20 - Master State Control (AB2.2)

Next to the control pane for the EtherCAT system, the Master State Control register has additional windows to display diagnostic information:

1. **Communication log:**

   The latest EtherCAT communication information is displayed on top to indicate the status of the EtherCAT network. In total the last five communication information of the master is shown.

2. **Frame loss counters:**

   In case of lost frames, the counter will increase and therefore indicates whether some telegrams are lost in the network. The lost frame counter might increase during startup, therefore it must be read out for each startup and compared with the value during the fault.

3. **Timing:**

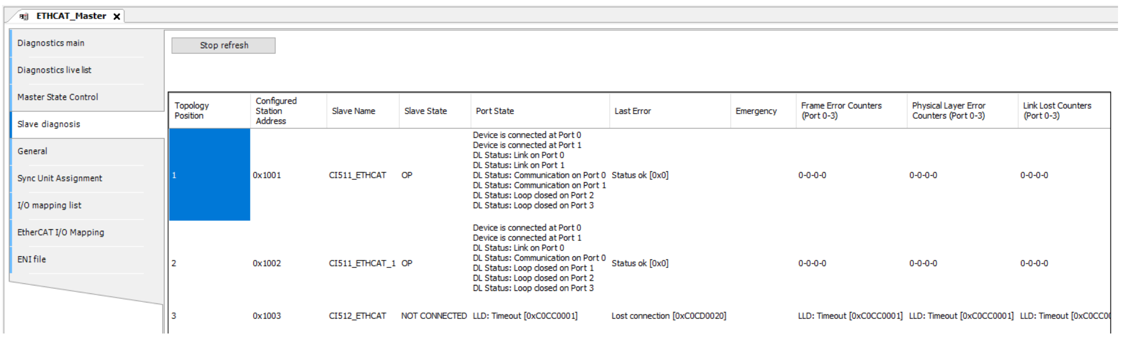
Bus cycle time	Bus cycle time in nanoseconds
Expected RX end time	Time from start of bus cycle transmission until completion of receiving the bus cycle back (in nanoseconds)
Expected TX data time	Time from start of bus cycle transmission until a new data update is expected to be signaled to stack (in nanoseconds)

4. **Log entry:**

   The existing Log file does not only show state changes of the bus, but also slave error with the hexadecimal code and the station address of the corresponding device. The additional slave errors can give deeper information about its faults (e.g. SDO error during startup of the system).

4.1.4 Slave diagnosis

New in Automation Builder 2.3 is the **Slave diagnosis** for the EtherCAT system. While the communication log of the Master State Control tab indicates whether an error is present or not, the Slave diagnosis can give additional information about the slaves directly and therefore about the positioning of the fault.



Topology Position	Configured Station Address	Slave Name	Slave State	Port State	Last Error	Emergency	Frame Error Counters (Port 0-3)	Physical Layer Error Counters (Port 0-3)	Link Lost Counters (Port 0-3)
1	0x1001	CI511_ETHCAT	OP	Device is connected at Port 0 Device is connected at Port 1 DL Status: Link on Port 0 DL Status: Link on Port 1 DL Status: Communication on Port 0 DL Status: Communication on Port 1 DL Status: Loop closed on Port 2 DL Status: Loop closed on Port 3	Status ok [0x0]		0-0-0-0	0-0-0-0	0-0-0-0
2	0x1002	CI511_ETHCAT_1	OP	Device is connected at Port 0 Device is connected at Port 1 DL Status: Link on Port 0 DL Status: Communication on Port 0 DL Status: Loop closed on Port 1 DL Status: Loop closed on Port 2 DL Status: Loop closed on Port 3	Status ok [0x0]		0-0-0-0	0-0-0-0	0-0-0-0
3	0x1003	CI512_ETHCAT	NOT CONNECTED	LLD: Timeout [0xC0CC0001]	Lost connection [0xC0CD0020]		LLD: Timeout [0xC0CC0001]	LLD: Timeout [0xC0CC0001]	LLD: Timeout [0xC0CC0

Figure 21 - Slave diagnosis (AB2.3)

The slave diagnosis combines the following diagnosis information within one table:

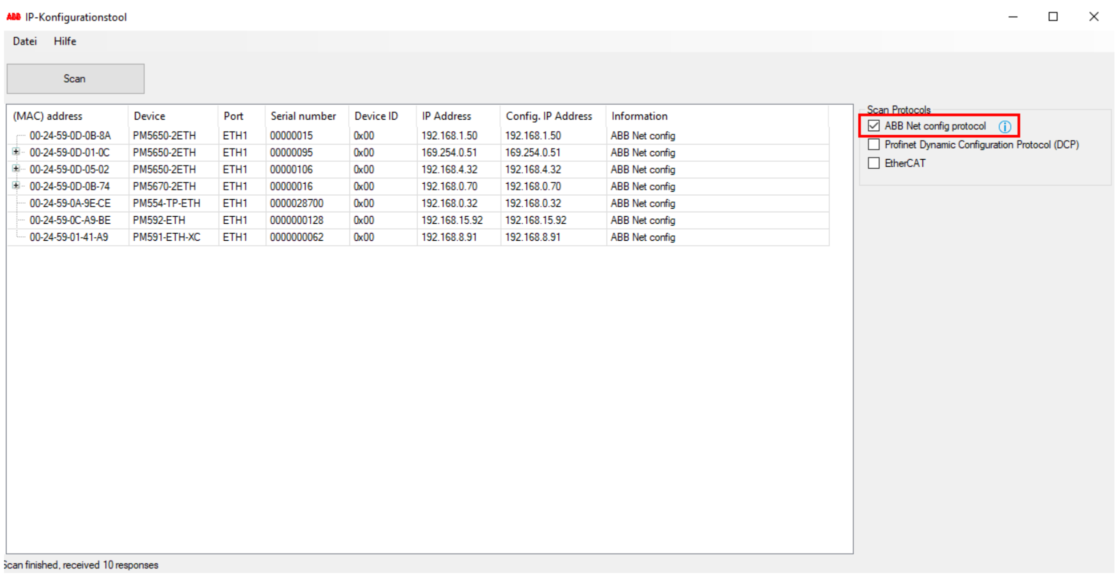
Topology Position:	Position of the configured slave device.
Configured Station Address:	Address that is defined by configuration. This address is not topology dependent.
Slave Name:	Configured name of the slave device.
Slave State:	The state of the slave. Possible slave states are: • Not Connected • Init • Preop • Safeop • Op • Init Err • Preop Err • Safeop Err
Port State:	The state of all ports (0-3) of the given slave. Shows how many connections this slave has to other slaves and if the connections are working fine: • Connected – Cable is plugged in • + Link – Physically connected to another slave • + Communication – Communication works fine
Last Error:	The last error that occurred in this slave. As text, if available, and error number. If this is any topology error, the editor view will show a hint to perform a bus scan.
Frame error counter:	Counts transmission errors on frame layer, detected by CRC check of frames. Fast growing values show a serious problem. Possible root causes include damaged cables, high electromagnetic noise or misbehavior of EtherCAT slave devices. Four counter values are shown, one for each port 0-3. Column has red background in case of any value other than 0.
Physical error counter:	Counts transmission errors on physical layer. Possible root causes include electromagnetic disturbance or faulty devices. Four counter values are shown, one for each port 0-3. Column has red background in case of any value other than 0.
Link lost counter:	Optional feature of EtherCAT slave devices, not supported by every device. Counts loss of physical connection (no link, LED off). Even short interruptions can be detected. Possible root causes include power dips, device reset, poor cables or connectors, loose contact. Four counter values are shown, one for each port 0-3. Column has red background in case of any value other than 0.

4.1.5 Emergency scan

Another additional diagnosis feature which was released with Automation Builder 2.3 is the **Emergency scan** for EtherCAT slaves.

This scan is implemented in the standalone “ABB IP-Configuration” Tool of Automation Builder. This standalone tool can be downloaded and installed via the Automation Builder Installer executable.

The IP-Configuration Tool is an already known tool in previous Automation Builder versions to set the IP Address and Gateway of the AC500-PLCs. Next to the known interface, where the devices in the network will be displayed, it is now possible to select for different **Scan Protocols**. With the *ABB Net config protocol* the PLC’s can be found.



(MAC) address	Device	Port	Serial number	Device ID	IP Address	Config. IP Address	Information
00-24-59-0D-0B-8A	PM5650-2ETH	ETH1	00000015	0x00	192.168.1.50	192.168.1.50	ABB Net config
00-24-59-0D-01-0C	PM5650-2ETH	ETH1	00000095	0x00	169.254.0.51	169.254.0.51	ABB Net config
00-24-59-0D-05-02	PM5650-2ETH	ETH1	00000106	0x00	192.168.4.32	192.168.4.32	ABB Net config
00-24-59-0D-0B-74	PM5670-2ETH	ETH1	00000016	0x00	192.168.0.70	192.168.0.70	ABB Net config
00-24-59-0A-9E-CE	PM554-TP-ETH	ETH1	0000028700	0x00	192.168.0.32	192.168.0.32	ABB Net config
00-24-59-0C-A9-BE	PM592-ETH	ETH1	0000000128	0x00	192.168.15.92	192.168.15.92	ABB Net config
00-24-59-01-41-A9	PM591-ETH-XC	ETH1	0000000062	0x00	192.168.8.91	192.168.8.91	ABB Net config

Scan Protocols
- [✓] ABB Net config protocol
- [ ] Profinet Dynamic Configuration Protocol (DCP)
- [ ] EtherCAT

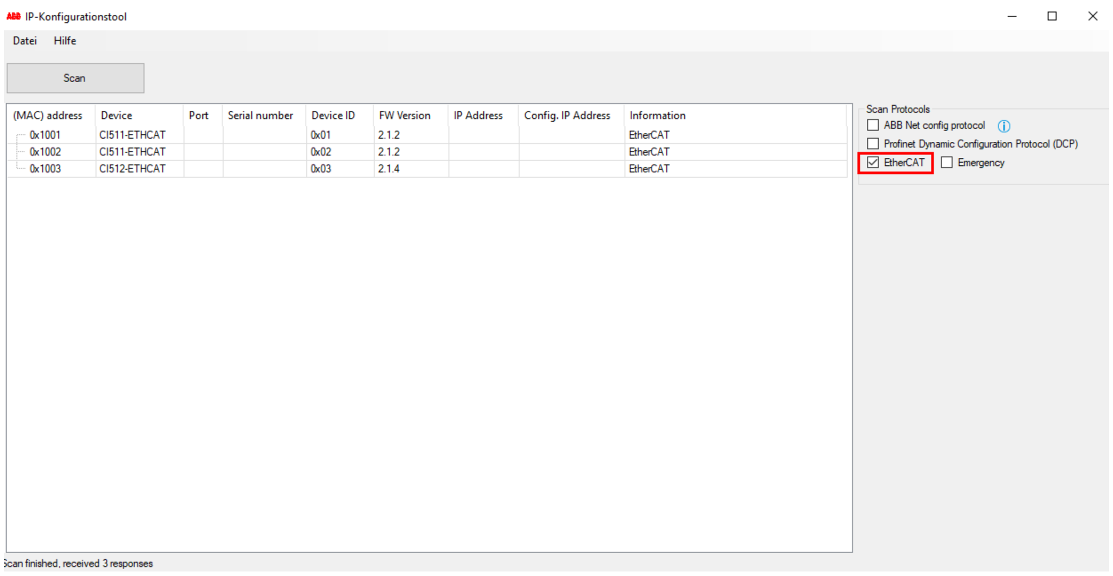
Scan finished, received 10 responses

Figure 22 - Scan Result with IP-Configuration and ABB Net config protocol (AB2.3)

If you select the Scan Protocol for EtherCAT, the *ABB Net config protocol* will be deselected, and a Scan request can be started to detect all EtherCAT devices in the network.

	Note: To scan the EtherCAT bus, a straight connection from the Engineering System to the first EtherCAT slave is required. The scan is not working if it is executing within a connection to the PLC.

The result of this scan is a list of available devices sorted in their topology position, recognizable by the assigned address. The device name and firmware versions will be shown as well.


IP-Konfigurationstool

Datei Hilfe

Scan

(MAC) address	Device	Port	Serial number	Device ID	FW Version	IP Address	Config. IP Address	Information
0x1001	CI511-ETHCAT			0x01	2.1.2			EtherCAT
0x1002	CI511-ETHCAT			0x02	2.1.2			EtherCAT
0x1003	CI512-ETHCAT			0x03	2.1.4			EtherCAT

Scan Protocols
- ☐ ABB Net config protocol
- ☐ Profinet Dynamic Configuration Protocol (DCP)
- ☑ EtherCAT ☐ Emergency

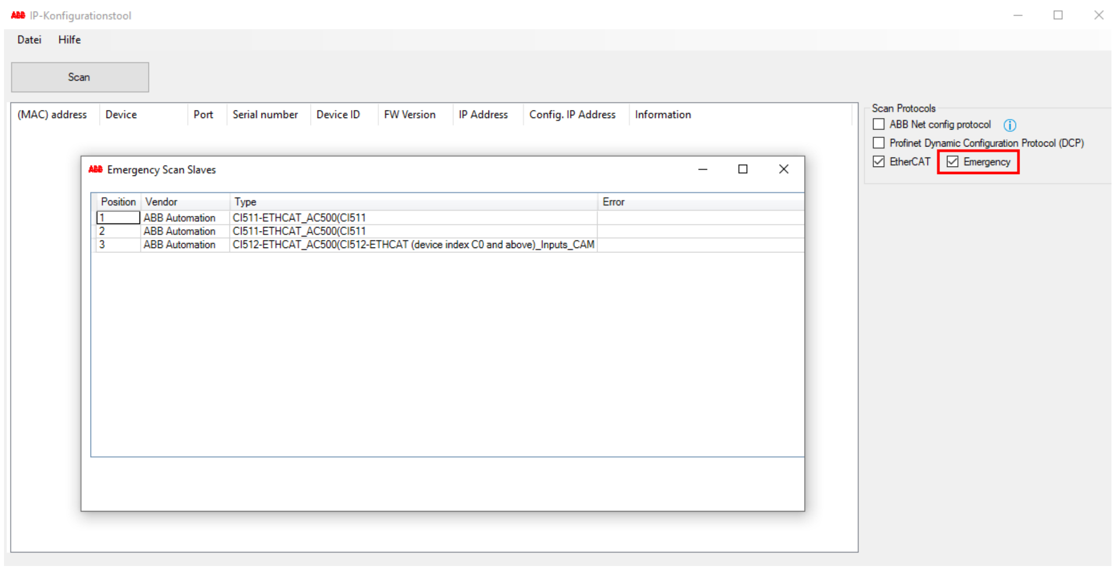
Scan finished, received 3 responses

Figure 23 - Scan Result with IP-Configuration and EtherCAT protocol (AB2.3)

To extend the scan to an Emergency one, the Checkbox at Emergency needs to be selected as well. A new window will occur which shows the result of the emergency scan.

In case that a non EtherCAT device (e.g. a Switch) is connected to the network, all telegrams will be rejected by this device and the diagnostic isn't working anymore. For those scenarios, the emergency scan is required, to find the faulty device.

In general, the Emergency scan is used for such scenarios, where no diagnostic is possible anymore.


IP-Konfigurationstool

Datei Hilfe

Scan

(MAC) address | Device | Port | Serial number | Device ID | FW Version | IP Address | Config. IP Address | Information

Emergency Scan Slaves

Position	Vendor	Type	Error
1	ABB Automation	CI511-ETHCAT_AC500(CI511	
2	ABB Automation	CI511-ETHCAT_AC500(CI511	
3	ABB Automation	CI512-ETHCAT_AC500(CI512-ETHCAT (device index C0 and above)_Inputs_CAM	

Scan Protocols
- ☐ ABB Net config protocol
- ☐ Profinet Dynamic Configuration Protocol (DCP)
- ☑ EtherCAT ☑ Emergency

Figure 24 – Scan Result of EtherCAT Emergency Scan

4.2 Process guideline for typical faults and errors during commissioning

4.2.1 Topology error

Error	Register	Description/ Explanation
Reversed devices		
Detection	Diagnostics main	• Coupler state: **Idle** (≠ Operate) • Protocol error: **Topology mismatch detected [0xC0CD0044]**
Localization	Diagnostics live list	Scan the bus and compare Types to find the faulty hardware position
Reversed devices within a cluster		
Detection	Diagnostics main/ Slave diagnosis	Protocol error/ Last error: **AL Control Timeout happened i.e. a slave ESM state change was not completed in time [0xC0CD006B]**
Localization	Slave diagnosis	Find topology position of faulty device by checking state • Manually comparison of cluster configuration
	Master state control	Check Log entries – Stations address with corresponding error message will localize the faulty slave. • Manually comparison of cluster configuration
Reversed devices of same type but with predefined station address		
Detection	Diagnostics main/ Slave diagnosis	Protocol error/ Last error: **Device identification via register failed [0xC0CD008E]**
Localization	Slave diagnosis	Last error displays “Device identification via register failed” and state remains in INIT + ERROR. • Manually comparison of identification wheels
	Master state control	Check Log entries – Stations address with corresponding error message will localize the faulty slave. • Manually comparison of identification wheels
Reversed connection at IN/OUT Ports		
Detection	Diagnostics main/ Slave diagnosis	Protocol error/ Last error: **Topology error detected [0xC0CD0043]**
Localization	Diagnostics live list	Scan the bus and compare Types to find the faulty hardware position. Check the List of devices with the configuration. A reversed port causes that the slave will be set to the end of the bus assumed that only one reversed port is available.
Missing cable between two devices		
Detection	Diagnostics main/ Slave diagnosis	Protocol error/ Last error: **Topology error detected [0xC0CD0043]**
Localization	Slave diagnosis	Localization by topology position - Last error: **Missing slave at port 1. [0xC0CD004C]** Lost connection to the next slave.
	Diagnostics live list	Result of the scan does not match the correct topology. All slaves after the missing cables are not listed.

Additional devices		
Detection	Diagnostics main	• Coupler state: **Idle** (≠ Operate) • Protocol error: **Topology mismatch detected [0xC0CD0044]**
Localization	Slave diagnosis	Slave diagnosis will only be displayed at configured devices. • **Last Error** at last configured device: **Unexpected slave at port 1 of slave. [0xC0CD0047]**
	Diagnostics live list	Result of the scan does not match the correct topology. The result must be compared manually to the configuration one.
Missing devices		
Detection	Diagnostics main/ Slave diagnosis	Protocol error/ Last error: **Topology error detected [0xC0CD0043]**
Localization	Slave diagnosis	Localization by topology position - Last error: **Missing slave at port 1 of slave. [0xC0CD004C]** Lost connection to the next slave.
Telegram error		
Detection	Diagnostics main	• **Current state** at Master Control ≠ **OP** • Protocol error: **Topology error detected [0xC0CD0043]** A telegram error can affect the system in various ways. The protocol error "**Topology error detected**" is just an example on how the system might react to this kind of error.
	Master State Control	• Frame loss counters ≠ 0 • Message Log – **Type**, **Index, Sub-index, Result**
Localization	Slave diagnosis	Localization by topology position of Slave – Error counter: • Frame Error Counter ≠ 0 • Physical Error Counter ≠ 0 Depending on the increasing error counters of the beside slaves, the root cause might be the cable or device itself.
Loose contact		
Detection	Diagnostics main	• CurrentState: **PREOP** (Bus state) or **PreOpErr** (Slave State) • CommErno/ LastErr: **Topology error detected [0xC0CD0043]** A loose contact can affect the system in various ways. The protocol error "**Topology error detected**" is just an example on how the system might react to this kind of error.
	Master State Control	• Frame loss counters ≠ 0 • Counter is increasing sporadically
Localization	Slave diagnosis	Localization by topology position of Slave – Error counter: • Frame Error Counter ≠ 0 • Physical Error Counter ≠ 0

		Depending on the increasing error counters of the beside slaves, the root cause might be the cable or device itself.

4.2.2 Communication error

Error	Function block	Description/ Explanation
Telegram error		
Detection	Diagnostics main	• **Current state** at Master Control = **PREOP** • Protocol error: **SDO Abort Code: Protocol Timeout [0xC0CF8002]** A telegram error can affect the system in various ways. The protocol error "**SDO Abort Code**" is just an example on how the system reacts to this kind of error.
	Master State Control	• Frame loss counters ≠ 0 • Message Log – **Type**, **Index, Sub-index, Result**
Localization	Slave diagnosis	Localization by topology position of Slave – Error counter: • Frame Error Counter ≠ 0 • Physical Error Counter ≠ 0 Depending on the increasing error counters of the beside slaves, the root cause might be the cable or device itself.
Loose contact		
Detection	Diagnostics main	• CurrentState: **PREOP** (Bus state) or **PreOpErr** (Slave State) • CommErno/ LastErr: **SDO protocol timeout [0xC0CF8002]** A loose contact can affect the system in various ways. The protocol error "**SDO protocol timeout**" is just an example on how the system reacts to this kind of error.
	Master State Control	• Frame loss counters ≠ 0 • Counter is increasing sporadically
Localization	Slave diagnosis	Localization by topology position of Slave – Error counter: • Frame Error Counter ≠ 0 • Physical Error Counter ≠ 0 Depending on the increasing error counters of the beside slaves, the root cause might be the cable or device itself.
Inadmissibly long or non-deterministic forwarding times		
Detection	Diagnostics main/ Slave diagnosis	CommErno/ LastErr: **DC RX TimeStamp Error [0xC0CD0026]**
Localization	Slave diagnosis	Localization by topology position of Slave - LastErr: **DC RX TimeStamp Error [0xC0CD0026]**
Faulty device/slave		
Detection	Diagnostics main	• Coupler state: **Idle** • Protocol error: **Cable Disconnected [0xC0000145]**
	Master State Control	• Frame loss counters ≠ 0

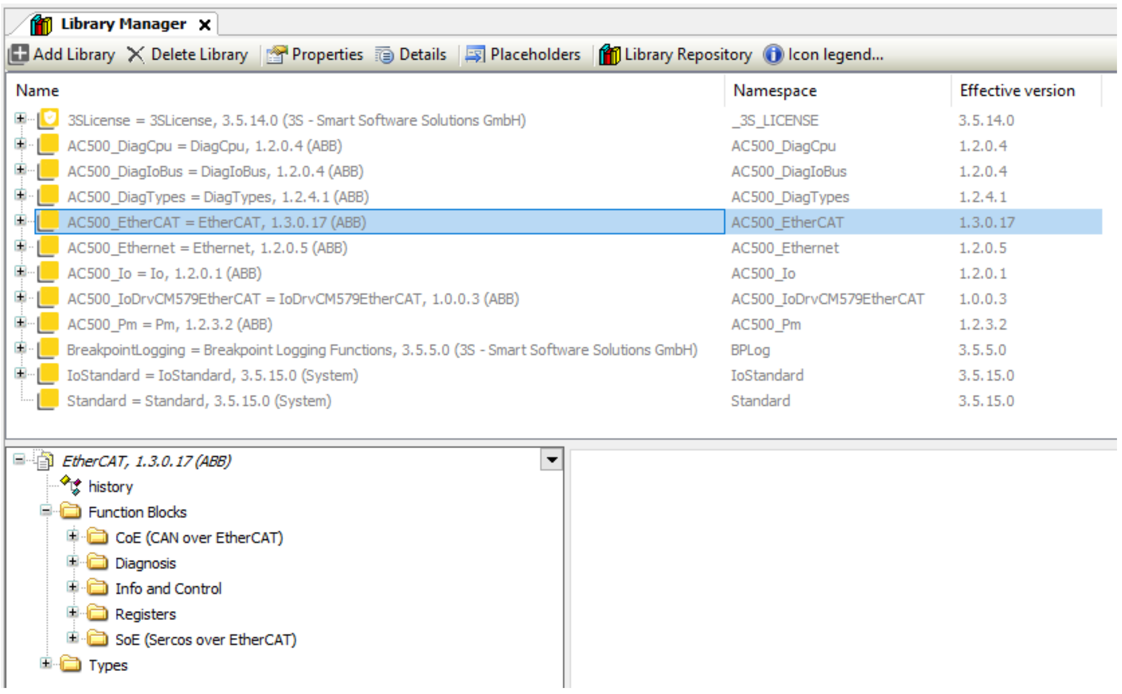
Localization	Emergency Scan (Tools > IP-Configuration)	The emergency scan lists the proper working devices according to the topology. Therefore, the device at the end of the list, indicates the last working slave. The device after is the non EtherCAT device.
Non-EtherCAT device		
Detection	Diagnostics main	• Coupler state: **Idle** • Protocol error: **Cable Disconnected [0xC0000145]**
	Master State Control	• Frame loss counters ≠ 0
Localization	Emergency Scan (Tools > IP-Configuration)	The emergency scan lists the proper working devices according to the topology. Therefore, the device at the end of the list, indicates the last working slave. The device after is the non EtherCAT device.

5 Diagnostic with IEC programming

Automation Builder serves libraries with function blocks for extended EtherCAT diagnostic. System integrators have access to the whole diagnostic System of EtherCAT and can implement it individually to provide diagnostic for different purposes. Therefore, diagnostic during commissioning is not only available in Automation Builder but also in IEC code.

To implement EtherCAT diagnostic inside IEC code for a V3 PLC, the Library AC500_EtherCAT is required. The library will be added to the Library Manager after adding the CM579-ETHCAT to a PLC slot. At least **Version 1.3.0.17** of the Library is required to use Automation Builder 2.3.0 and a PM5670-ETH PLC with the Firmware Version 3.3.1.0.

The Library consists of different folders with Function blocks to read/ write CoE- and SoE data or to read/ write registers, to control the EtherCAT system and to get diagnostic of the same (**Figure 25**).



Name	Namespace	Effective version
3SLicense = 3SLicense, 3.5.14.0 (3S - Smart Software Solutions GmbH)	_3S_LICENSE	3.5.14.0
AC500_DiagCpu = DiagCpu, 1.2.0.4 (ABB)	AC500_DiagCpu	1.2.0.4
AC500_DiagIoBus = DiagIoBus, 1.2.0.4 (ABB)	AC500_DiagIoBus	1.2.0.4
AC500_DiagTypes = DiagTypes, 1.2.4.1 (ABB)	AC500_DiagTypes	1.2.4.1
AC500_EtherCAT = EtherCAT, 1.3.0.17 (ABB)	AC500_EtherCAT	1.3.0.17
AC500_Ethernet = Ethernet, 1.2.0.5 (ABB)	AC500_Ethernet	1.2.0.5
AC500_Io = Io, 1.2.0.1 (ABB)	AC500_Io	1.2.0.1
AC500_IoDrvCM579EtherCAT = IoDrvCM579EtherCAT, 1.0.0.3 (ABB)	AC500_IoDrvCM579EtherCAT	1.0.0.3
AC500_Pm = Pm, 1.2.3.2 (ABB)	AC500_Pm	1.2.3.2
BreakpointLogging = Breakpoint Logging Functions, 3.5.5.0 (3S - Smart Software Solutions GmbH)	BPLog	3.5.5.0
IoStandard = IoStandard, 3.5.15.0 (System)	IoStandard	3.5.15.0
Standard = Standard, 3.5.15.0 (System)	Standard	3.5.15.0

```
EtherCAT, 1.3.0.17 (ABB)
  history
  Function Blocks
    CoE (CAN over EtherCAT)
    Diagnosis
    Info and Control
    Registers
    SoE (Sercos over EtherCAT)
  Types
```

Figure 25 - AC500_EtherCAT Library structure

5.1 Process guideline for typical faults and errors during commissioning

This chapter shows the process how to detect and localize faults and errors that might occurs during commissioning (chapter ***3.1***) by using the library in the IEC code.

5.1.1 Topology error

Error	Function block	Description/ Explanation
Reversed devices		
Detection	EcatBusDiag	CurrentState: **INIT** (≠ Operate) CommErno: **Topology mismatch detected [0xC0CD0044]**
Localization	EcatScanTopology & EcatSlvDiag	• Scan the bus and compare Types to find the faulty hardware position • LastErr: **Wrong slave at position [0xC0CD0035]**
Reversed devices within a cluster		
Detection	EcatBusDiag/ EcatSlvDiag	CommErno/ LastErr: **AL Control Timeout [0xC0CD006B]**
Localization		DiagData: **Wrong type or model.**
Reversed devices of same type but with predefined station address		
Detection	EcatBusDiag/ EcatSlvDiag	Protocol error/ Last error: **Explicit device identification failed (register) [0xC0CD008E]**
Localization		Find topology position of faulty device by checking state • Manually comparison of cluster configuration
Reversed connection at IN/OUT Ports		
Detection	EcatBusDiag	CommErno: **Topology mismatch detected. [0xC0CD0044]**
Localization	EcatScanTopology (CheckTopology = True)	• Check the List of Devices with the configuration. A reversed port causes that the slave will be set to the end of the bus assumed that only one reversed port is available. • Input “CheckTopology” is available at the EcatScan-Topology function block. If this input is set to True, the function block is checking the bus for reversed ports. → ReversedPortsPosition
Missing cable between two devices		
Detection	EcatBusDiag/ EcatSlvDiag	CommErno/ LastErr: **Topology mismatch detected [0xC0CD0044]**
Localization	EcatSlvDiag	Localization by topology position - LastErr: **Missing slave at port 1. [0xC0CD004C]** Lost connection to the next slave.
Additional devices		
Detection	EcatBusDiag	CommErno: **Topology mismatch detected. [0xC0CD0044]**
Localization	EcatSlvDiag	Slave diagnosis will only be displayed at configured devices. **LastErr** at last configured device: **Unexpected slave at port 1 of slave. [0xC0CD0047]**

	EcatScanTopology	Result of the scan does not match the correct topology. The result must be compared manually to the configuration one.
Missing devices		
Detection	EcatBusDiag/ EcatSlvDiag	CommErno/ LastErr: **Topology mismatch detected [0xC0CD0044]**
Localization	EcatSlvDiag	Localization by topology position - LastErr: **Missing slave at port 1. [0xC0CD004C]** Lost connection to the next slave.
Telegram error		
Detection	EcatBusDiag	• **Current state** at Master Control ≠ **OP** • Protocol error: **Topology error detected [0xC0CD0043]** A telegram error can affect the system in various ways. The protocol error "**Topology error detected**" is just an example on how the system might react to this kind of error.
	EcatMasterGetTresholdCnt **or** EcatMasterGet-FrameLossCount	• Frame loss counters ≠ 0 • Message Log – **Type**, **Index, Sub-index, Result**
Localization	EcatSlvDiag	Localization by topology position of Slave – Error counter: • Frame Error Counter ≠ 0 • Physical Error Counter ≠ 0 • Depending on the increasing error counters of the beside slaves, the root cause might be the cable or device itself.
Loose contact		
Detection	EcatBusDiag/ EcatSlvDiag	• CurrentState: **PREOP** (Bus state) or **PreOpErr** (Slave State) • CommErno/ LastErr: **Topology error detected [0xC0CD0043]** A loose contact can affect the system in various ways. The protocol error "**Topology error detected**" is just an example on how the system might react to this kind of error.
	EcatMasterGetTresholdCnt **or** EcatMasterGet-FrameLossCount	• Frame loss counters ≠ 0 • Counter is increasing sporadically
Localization	EcatSlvDiag	Localization by topology position of Slave – Error counter: • Frame Error Counter ≠ 0 • Physical Error Counter ≠ 0 Depending on the increasing error counters of the beside slaves, the root cause might be the cable or device itself.

5.1.2 Communication error

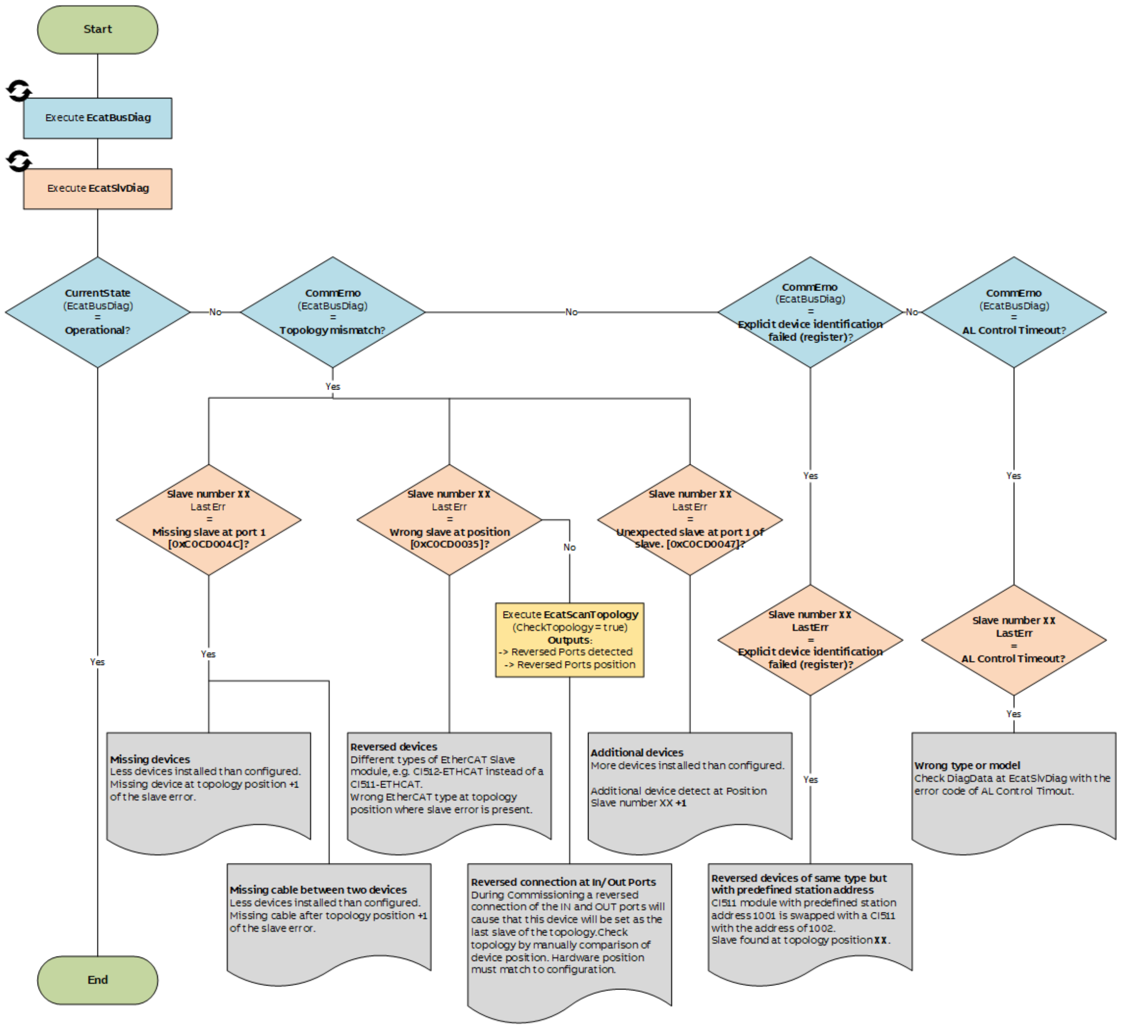
Error	Function block	Description/ Explanation
Telegram error		
Detection	EcatBusDiag/ EcatSlaveDiag	• CurrentState: **PREOP** (Bus state) or **PreOpErr** (Slave State) • CommErno/ LastErr: **SDO protocol timeout [0xC0CF8002]** A telegram error can affect the system in various ways. The protocol error "**SDO protocol timeout**" is just an example on how the system might react to this kind of error.
	EcatMasterGetTresholdCnt **or** EcatMasterGet-FrameLossCount	• TresholdCounters/ LostFrames ≠ 0 • Counter is increasing sporadically
Localization	EcatSlvReadRxErrorCnt	Localization by topology position of Slave – Error counter: • Frame Error Counter ≠ 0 • Physical Error Counter ≠ 0 Depending on the increasing error counters of the beside slaves, the root cause might be the cable or device itself.
Loose contact		
Detection	EcatBusDiag/ EcatSlaveDiag	• CurrentState: **PREOP** (Bus state) or **PreOpErr** (Slave State) • CommErno/ LastErr: **SDO protocol timeout [0xC0CF8002]** A telegram error can affect the system in various ways. The protocol error "**SDO protocol timeout**" is just an example on how the system might react to this kind of error.
	EcatMasterGetTresholdCnt **or** EcatMasterGet-FrameLossCount	• TresholdCounters/ LostFrames ≠ 0 • Counter is increasing sporadically
Localization	EcatSlvReadRxErrorCnt/ EcatSlavReadLostLinkCnt (optional)	Localization by topology position of Slave – Error counter: • Frame Error Counter ≠ 0 • Physical Error Counter ≠ 0 • Lost Link Error Counter ≠ 0 Depending on the increasing error counters of the beside slaves, the root cause might be the cable or device itself.
Inadmissibly long or non-deterministic forwarding times		
Detection	EcatBusDiag/ EcatSlaveDiag	CommErno/ LastErr: **DC RX TimeStamp Error [0xC0CD0026]**
Localization	EcatSlaveDiag	Localization by topology position of Slave – LastErr: **DC RX TimeStamp Error [0xC0CD0026]**
Faulty device/slave		
Detection	EcatBusDiag	• CurrentState: **INIT** (≠ Operate) • CommErno: **Cable Disconnected [0xC0000145]**

	EcatMasterGetTresholdCnt or EcatMasterGet-FrameLossCount	• TresholdCounters/ LostFrames ≠ 0
	Special case: It might happen that a faulty slave does not executes telegrams correctly which affects that the **Working counter is invalid** while the **TresholdCounters/ LostFrames remains 0**.	
Localization	EcatEmergencyScan	The emergency scan lists the proper working devices according to the topology. Therefore, the device at the end of the list, indicates the last working slave. The device after is the non EtherCAT device.
Non-EtherCAT device		
Detection	EcatBusDiag	• CurrentState: **INIT** (≠ Operate) • CommErno: **Cable Disconnected [0xC0000145]**
	EcatMasterGetTresholdCnt or EcatMasterGet-FrameLossCount	• TresholdCounters/ LostFrames ≠ 0
Localization	EcatEmergencyScan	The emergency scan lists the proper working devices according to the topology. Therefore, the device at the end of the list, indicates the last working slave. The device after is the non EtherCAT device.

5.1.3 Float Chart for commissioning diagnostic with IEC programming

This float Chart is an overview about the general diagnostic structure for EtherCAT within the IEC Programming with an AC500 System.

The cyclic repetition of calling the **EcatBusDiag** and **EcatSlvDiag** function block is shown at the beginning only and can be used to use additional diagnostic on demand.


Start
Execute EcatBusDiag
Execute EcatSlvDiag
CurrentState (EcatBusDiag) = Operational?
No
CommErno (EcatBusDiag) = Topology mismatch?
No
CommErno (EcatBusDiag) = Explicit device identification failed (register)?
No
CommErno (EcatBusDiag) = AL Control Timeout?
Yes
Slave number XX LastErr = Missing slave at port 1 [0xC0CD004C]?
Slave number XX LastErr = Wrong slave at position [0xC0CD0035]?
Slave number XX LastErr = Unexpected slave at port 1 of slave. [0xC0CD0047]?
No
Execute EcatScanTopology (CheckTopology = true) Outputs: -> Reversed Ports detected -> Reversed Ports position
Yes
Slave number XX LastErr = Explicit device identification failed (register)?
Yes
Slave number XX LastErr = AL Control Timeout?
Yes
Yes
Yes
Yes
End

Missing devices
Less devices installed than configured.
Missing device at topology position +1 of the slave error.

Reversed devices
Different types of EtherCAT Slave module, e.g. CI512-ETHCAT instead of a CI511-ETHCAT.
Wrong EtherCAT type at topology position where slave error is present.

Additional devices
More devices installed than configured.

Additional device detect at Position Slave number XX **+1**

Wrong type or model
Check DiagData at EcatSlvDiag with the error code of AL Control Timout.

Missing cable between two devices
Less devices installed than configured.
Missing cable after topology position +1 of the slave error.

Reversed connection at In/Out Ports
During Commissioning a reversed connection of the IN and OUT ports will cause that this device will be set as the last slave of the topology.Check topology by manually comparison of device position. Hardware position must match to configuration.

Reversed devices of same type but with predefined station address
CI511 module with predefined station address 1001 is swapped with a CI511 with the address of 1002.
Slave found at topology position **XX**.

Figure 26 - Flow Chart for topology error diagnostic during commissioning within IEC code

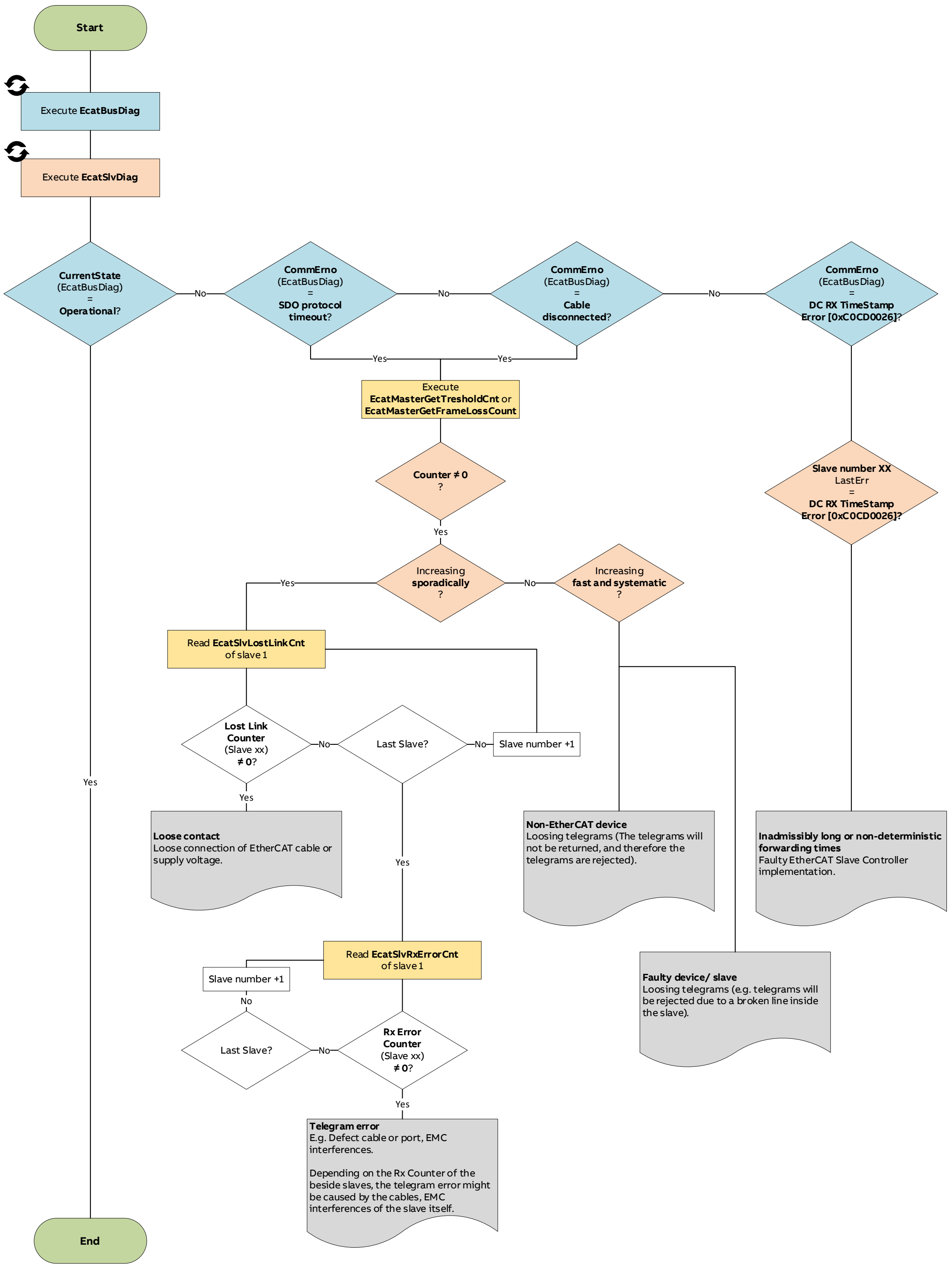

Start
Execute EcatBusDiag
Execute EcatSlvDiag
CurrentState (EcatBusDiag) = Operational?
No
CommErno (EcatBusDiag) = SDO protocol timeout?
No
CommErno (EcatBusDiag) = Cable disconnected?
No
CommErno (EcatBusDiag) = DC RX TimeStamp Error [0xC0CD0026]?
Yes
Yes
Execute EcatMasterGetTresholdCnt or EcatMasterGetFrameLossCount
Counter ≠ 0 ?
Yes
Increasing sporadically ?
Yes
No
Increasing fast and systematic ?
Slave number XX LastErr = DC RX TimeStamp Error [0xC0CD0026]?
Read EcatSlvLostLinkCnt of slave 1
Lost Link Counter (Slave xx) ≠ 0?
No
Last Slave?
No
Slave number +1
Yes
Yes
Loose contact
Loose connection of EtherCAT cable or supply voltage.
Yes
Non-EtherCAT device
Loosing telegrams (The telegrams will not be returned, and therefore the telegrams are rejected).
Inadmissibly long or non-deterministic forwarding times
Faulty EtherCAT Slave Controller implementation.
Read EcatSlvRxErrorCnt of slave 1
Slave number +1
No
Faulty device/ slave
Loosing telegrams (e.g. telegrams will be rejected due to a broken line inside the slave).
Last Slave?
No
Rx Error Counter (Slave xx) ≠ 0?
Yes
Telegram error
E.g. Defect cable or port, EMC interferences.
Depending on the Rx Counter of the beside slaves, the telegram error might be caused by the cables, EMC interferences of the slave itself.
End

Figure 27 - Flow Chart for communication error diagnostic during commissioning within IEC code

5.2 Process guideline for typical faults and errors during operational

This chapter shows the processes how to detect and localize the typical faults and errors of the chapter *3.2 Operational* by using the libraries in the IEC code. The following list is exemplary only and valid for S500 EtherCAT slaves and its additional I/O Modules by ABB.

5.2.1 Device-, Module-, Channel- Error

Error	Function block	Description/ Explanation
Missing module		
Detection	EcatSlvDiag (S500 Format = TRUE)	Output **NumErr** ≠ 0
Localization		Output **DiagData** (Emergency message) • Byte information: **3-<<*Dev. Nr.*>>-31-<<*Ch. Nr.*>>-17** • Decoded example: “**E3: Module itself at Position 1 – No communication with the I/O module.**”
Faulty module		
Detection	EcatSlvDiag (S500 Format = TRUE)	**NumErr** ≠ 0
Localization		A defective module can manifest itself in a variety of ways and therefore an exact description of it cannot be done as such that easy. It might happen that the faulty module affects that the module itself is not recognized as available. Another scenario could be that the inline connection to a channel is broken and therefore a channel fault is detected.
Wrong module		
Detection	EcatSlvDiag (S500 Format = TRUE)	**NumErr** ≠ 0
Localization		Output **DiagData** (Emergency message) • Byte information: **3-<<*Dev. Nr.*>>-31-<<*Ch. Nr.*>>-32** • Decoded example: “**E3: Module itself at Position 1 - Unknown type or model.**”
Missing process voltage		
Detection	EcatSlvDiag (S500 Format = TRUE)	**NumErr** ≠ 0
Localization		Output **DiagData** (Emergency message) • Byte information: **3-<<*Dev. Nr.*>>-31-<<*Ch. Nr.*>>-11** **4-<<*Dev. Nr.*>>-31-<<*Ch. Nr.*>>-45** • Decoded example: “**E3: Module itself at Position 1 – Process voltage too low (Check process voltage).**” “**E4: Module itself at Position 31 (CI51x) – Process voltage switched off (ON -> OFF).**”

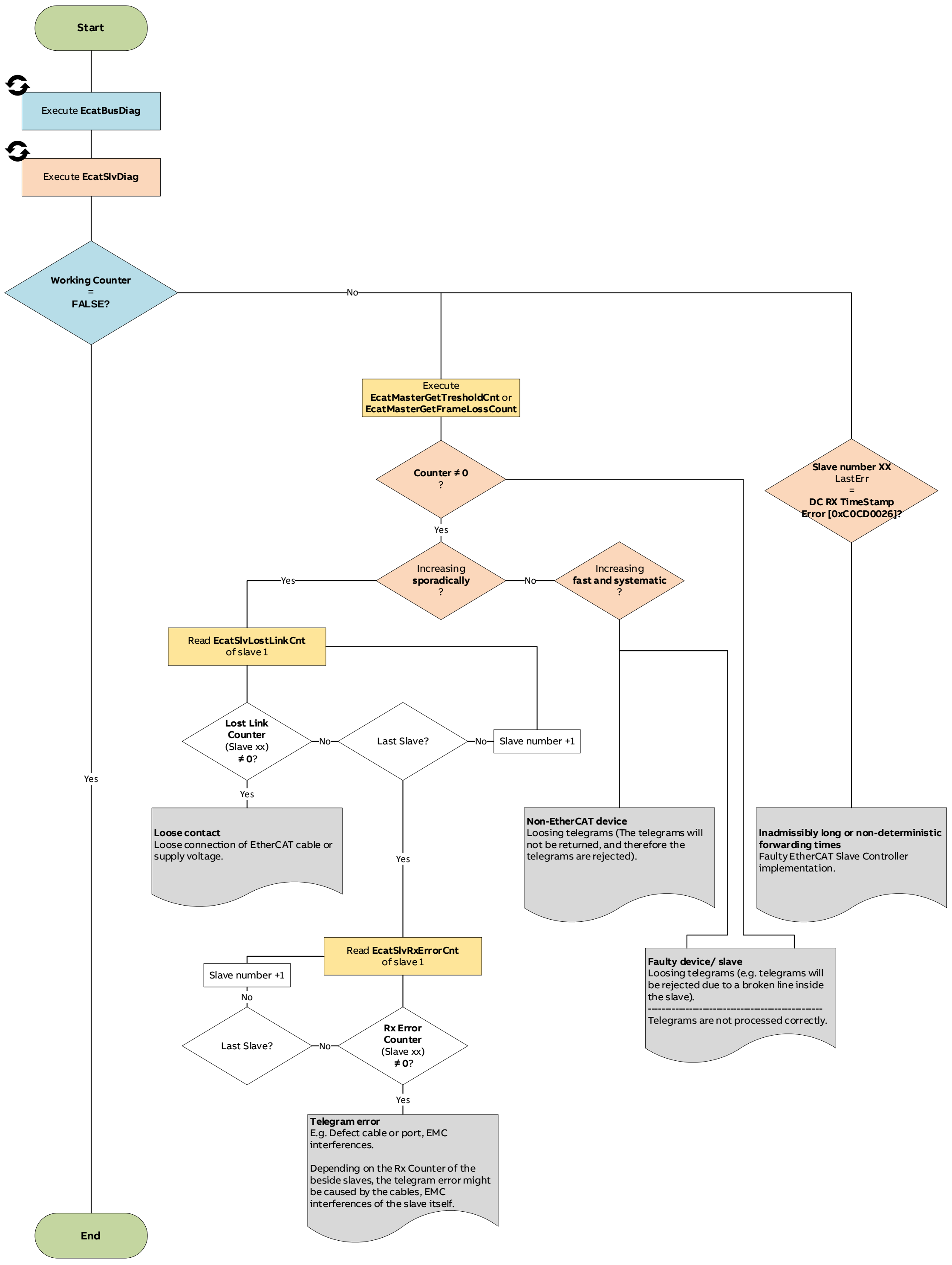
Cut wire, short circuit or wire break at IO module		
Detection	EcatSlvDiag (S500 Format = TRUE)	**NumErr** ≠ 0
Localization		Output **DiagData** (Emergency message) • Byte information: **4-*<<Dev. Nr.>>*-1-*<<Ch. Nr.>>*-48** • Decoded example: “**E4: Analog input Channel 2 at Position 1 – Over-load/ Cut wire.**”

5.2.2 Communication Error

Error	Function block	Description/ Explanation
Telegram error		
Detection	Working Counter	Working counter is invalid
	EcatMasterGetTresholdCnt **or** EcatMasterGet-FrameLossCount	• TresholdCounters/ LostFrames ≠ 0 • Counter is increasing sporadically
Localization	EcatSlvReadRxErrorCnt	• tsEcmSlvRxErrorCnt ≠ 0 • Monitoring of Rx Error Counter of the beside Slaves as well • Counter is increasing sporadically
Loose contact		
Detection	Working Counter	Working counter is invalid
Localization	EcatSlvReadRxErrorCnt	• tsEcmSlvRxErrorCnt ≠ 0 • Monitoring of Rx Error counter of the beside Slaves as well • Counter is increasing sporadically
	EcatSlvReadRxErrorCnt (op-tional)	• tsEcmSlvLostLinkCnt ≠ 0 • Counter is increasing sporadically
Inadmissibly long or non-deterministic forwarding times		
Detection	EcatBusDiag/ EcatSlaveDiag	CommErno/ LastErr: **DC RX TimeStamp Error [0xC0CD0026]**
Localization	EcatSlaveDiag	Localization by topology position of Slave – LastErr: **DC RX TimeStamp Error [0xC0CD0026]**
Faulty device/slave		
Detection	EcatMasterGetTresholdCnt **or** EcatMasterGet-FrameLossCount	• TresholdCounters/ LostFrames ≠ 0 • Counter is sporadically **or** fast and systematic in-creasing
	Special case: It might happen that a faulty slave does not executes telegrams correctly which affects that the **Working counter is invalid** while the **TresholdCounters/ LostFrames remains 0**.	
Localization	EcatSlvReadRxErrorCnt	Sporadically increasing LostFrame counter • tsEcmSlvRxErrorCnt ≠ 0 • Monitoring of Rx Error counter of the beside Slaves as well will indicates whether a device or ca-ble is broken

	EcatEmergencyScan	Fast and systematic increasing (Telegrams will be re-jected completely) • Last functional device will be display • According to the topology, the device behind the last functional slave is faulty
Non-EtherCAT device		
Detection	EcatMasterGetTresholdCnt **or** EcatMasterGet-FrameLossCount	• TresholdCounters/ LostFrames ≠ 0 • Counter is fast and systematic increasing
Localization	EcatEmergencyScan	Fast and systematic increasing (Telegrams will be re-jected completely) • Last functional device will be display • According to the topology, the device behind the last functional slave is a Non-EtherCAT device

5.2.3 Flow Chart for Operational diagnostic with IEC programming



Start
Execute **EcatBusDiag**
Execute **EcatSlvDiag**
Working Counter = FALSE?
No
Yes
End
Execute **EcatMasterGetTresholdCnt** or **EcatMasterGetFrameLossCount**
Counter ≠ 0 ?
Yes
Increasing **sporadically** ?
Yes
No
Increasing **fast and systematic** ?
Read **EcatSlvLostLinkCnt** of slave 1
Lost Link Counter (Slave xx) **≠ 0**?
No
Last Slave?
No
Slave number +1
Yes
Loose contact Loose connection of EtherCAT cable or supply voltage.
Yes
Read **EcatSlvRxErrorCnt** of slave 1
Slave number +1
No
Last Slave?
No
Rx Error Counter (Slave xx) **≠ 0**?
Yes
Telegram error E.g. Defect cable or port, EMC interferences. Depending on the Rx Counter of the beside slaves, the telegram error might be caused by the cables, EMC interferences of the slave itself.
Non-EtherCAT device Loosing telegrams (The telegrams will not be returned, and therefore the telegrams are rejected).
Faulty device/ slave Loosing telegrams (e.g. telegrams will be rejected due to a broken line inside the slave). -- Telegrams are not processed correctly.
Slave number XX LastErr = **DC RX TimeStamp Error [0xC0CD0026]?**
Inadmissibly long or non-deterministic forwarding times Faulty EtherCAT Slave Controller implementation.

—

ABB AG
Eppelheimer Straße 82
69123 Heidelberg, Germany
Phone: +49 62 21 701 1444
Fax: +49 62 21 701 1382
E-Mail: plc.support@de.abb.com
www.abb.com/plc

—

We reserve the right to make technical changes or modify the contents of this document without prior notice. With regard to purchase orders, the agreed particulars shall prevail. ABB AG does not accept any responsibility whatsoever for potential errors or possible lack of information in this document.

We reserve all rights in this document and in the subject matter and illustrations contained therein. Any reproduction, disclosure to third parties or utilization of its contents – in whole or in parts – is forbidden without prior written consent of ABB AG.
Copyright© 2022 ABB. All rights reserved