# ECX-EC

EtherCAT®

## Slave Bridge


Manual

to Product E.3022.02



esd electronic system design gmbh
Vahrenwalder Str. 207 • 30165 Hannover • Germany
http://www.esd.eu
Phone: +49 (0) 511 3 72 98-0 • Fax: +49 (0) 511 3 72 98-68

## N O T E

The information in this document has been carefully checked and is believed to be entirely reliable. **esd electronics** makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document. In particular descriptions and technical data specified in this document may not be constituted to be guaranteed product features in any legal sense.

**esd electronics** reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.



esd electronics gmbh
Vahrenwalder Str. 207
30165 Hannover
Germany

| | |
|---|---|
| Phone: | +49-511-372 98-0 |
| Fax: | +49-511-372 98-68 |
| E-Mail: | info@esd.eu |
| Internet: | www.esd.eu |

This manual contains important information and instructions on safe and efficient handling of the ECX-EC. Carefully read this manual before commencing any work and follow the instructions.
The manual is a product component, please retain it for future use.

| | |
|---|---|
| **Document file:** | I:\Texte\Doku\MANUALS\ECX\ECX-EC\Englisch\ECX-EC-Manual_en_19.odt |
| **Date of print:** | 2020-01-14 |
| **Document type number:** | DOC0800 |

| | |
|---|---|
| **Firmware version:** | from Version 1.0.3 |

**Document History**

The changes in the document listed below affect changes in the hardware as well as changes in the description of the facts, only.

| Revision | Chapter | Changes versus previous version | Date |
|---|---|---|---|
| 1.5 | 6.2.4.1 | Fixed reference to wrong object | 2015-04-24 |
| | 6.2.6 | Chapter “UDP Gateway” added | |
| | 6.3.2.5 | Chapter for Object 0x2030 added | |
| | 9. | Declaration of Conformity updated | |
| 1.6 | - | Safety Information revised, Classification of safety Instructions inserted, Note to data safety inserted | 2016-09-19 |
| | 2.1 | Note to „Hardware Installation “ inserted | |
| | 3 | Warning message inserted | |
| | 5.1 | Note inserted | |
| | 5.3 | Chapter revised | |
| | 6.1, 6.2 | Note to UDDC Software | |
| | 6.2.6.2 | New chapter: „Configuration with UDDC - Universal Device Description Composer“ , text moved from chapter „Configuration“ | |
| | 6.4 | Note to Firmware-Update inserted | |
| | 6.4.1 | LED description supplemented | |
| 1.7 | 1.1 | Note on ECX-EC-CD inserted | 2016-09-26 |
| | 4.5, 6.1, 6.2, 6.4 | Note on EtherCAT Workbench on ECX-EC-CD inserted | |
| 1.8 | 1.1 | Block circuit diagram revised | 2017-02-15 |
| 1.9 | 9. | EU-Declaration of Conformity updated | 2020-01-14 |

Technical details are subject to change without further notice.

## Classification of Warning Messages and Safety Instructions

This manual contains noticeable descriptions, warning messages and safety instructions, which you must follow to avoid personal injuries or death and property damage.

This is the safety alert symbol.
It is used to alert you to potential personal injury hazards. Obey all safety messages and instructions that follow this symbol to avoid possible injury or death.

### DANGER, WARNING, CAUTION

Depending on the hazard level the signal words DANGER, WARNING or CAUTION are used to highlight safety instructions and warning messages. These messages may also include a warning relating to property damage.

**DANGER**
Danger statements indicate a hazardous situation which, if not avoided, will result in death or serious injury.

**WARNING**
Warning statements indicate a hazardous situation that, if not avoided, could result in death or serious injury.

**CAUTION**
Caution statements indicate a hazardous situation that, if not avoided, could result in minor or moderate injury.

### NOTICE

Notice statements are used to notify people on hazards that could result in things other than personal injury, like property damage.

**NOTICE**
This NOTICE statement contains the general mandatory sign and gives information that must be heeded and complied with for a safe use.

### INFORMATION

**INFORMATION**
Notes to point out something important or useful.

# Safety Instructions

- When working with the ECX-EC follow the instructions below and read the manual carefully to protect yourself from injury and the ECX-EC from damage.
- Do not use damaged or defective cables to connect the ECX-EC.
- In case of damages to the device, which might affect safety, appropriate and immediate measures must be taken, that exclude an endangerment of persons and domestic animals and property.
- Current circuits which are connected to the device have to be sufficiently protected against hazardous voltage (SELV according to EN 60950-1).
- The ECX-EC may only be driven by power supply current circuits, that are contact protected. A power supply, that provides a safety extra-low voltage (SELV) according to EN 60950-1, complies with this conditions.

- Do not open the housing of the ECX-EC.
- The ECX-EC has to be securely installed before commissioning.
- The permitted operating position is specified as shown (Figure: 33). Other operating positions are not allowed.
- Never let liquids get inside the ECX-EC. Otherwise, electric shocks or short circuits may result.
- Protect the ECX-EC from dust, moisture and steam.
- Protect the ECX-EC from shocks and vibrations.
- The ECX-EC may become warm during normal use. Always allow adequate ventilation around the ECX-EC and use care when handling.
- Do not operate the ECX-EC adjacent to heat sources and do not expose it to unnecessary thermal radiation. Ensure an ambient temperature as specified in the technical data.

**Qualified Personnel**

This documentation is directed exclusively towards personnel qualified in control and automation engineering. The installation and commissioning of the product may only be carried out by qualified personnel, which is authorized to put devices, systems and electric circuits into operation according to the applicable national standards of safety engineering.

**Conformity**

The ECX-EC meets the demands of the EU regulations and EMC standards printed in the conformity declaration at the end of this manual.

**Data Safety**

This device is equipped with an Ethernet or other interface which is suitable to establish a connection to data networks. Depending on the software used on the device, these interfaces may allow attackers to compromise normal function, get illegal access or cause damage.
esd does not take responsibility for any damage caused by the device if operated at any networks. It is the responsibility of the device's user to take care that necessary safety precautions for the device's network interface are in place.

**Intended Use**

The intended use of the ECX-EC is the operation as EtherCAT Slave Bridge.
The guarantee given by esd does not cover damages which result from improper use, usage not in accordance with regulations or disregard of safety instructions and warnings.

- The ECX-EC is intended for indoor installation only.
- The operation of the ECX-EC in hazardous areas, or areas exposed to potentially explosive materials is not permitted.
- The operation of the ECX-EC for medical purposes is prohibited.

**Service Note**
The ECX-EC does not contain any parts that require maintenance by the user. The ECX-EC does not require any manual configuration of the hardware. Unauthorized intervention in the device voids warranty claims.

**Disposal**
Devices which have become defective in the long run have to be disposed in an appropriate way or have to be returned to the manufacturer for proper disposal. Please, make a contribution to environmental protection.

---

**Number Representation**
All numbers in this document are base 10 unless designated otherwise. Hexadecimal numbers have a prefix of 0x. For example, 42 is represented as 0x2A in hexadecimal.

**Abbreviations**

| | |
|---|---|
| ESI | Ether CAT Slave Information |
| PDO | Process Data Object |
| SDO | Service Data Object |
| XML | Extensible Markup Language |

# Table of contents

# 1. Overview

## 1.1 Description of the ECX-EC

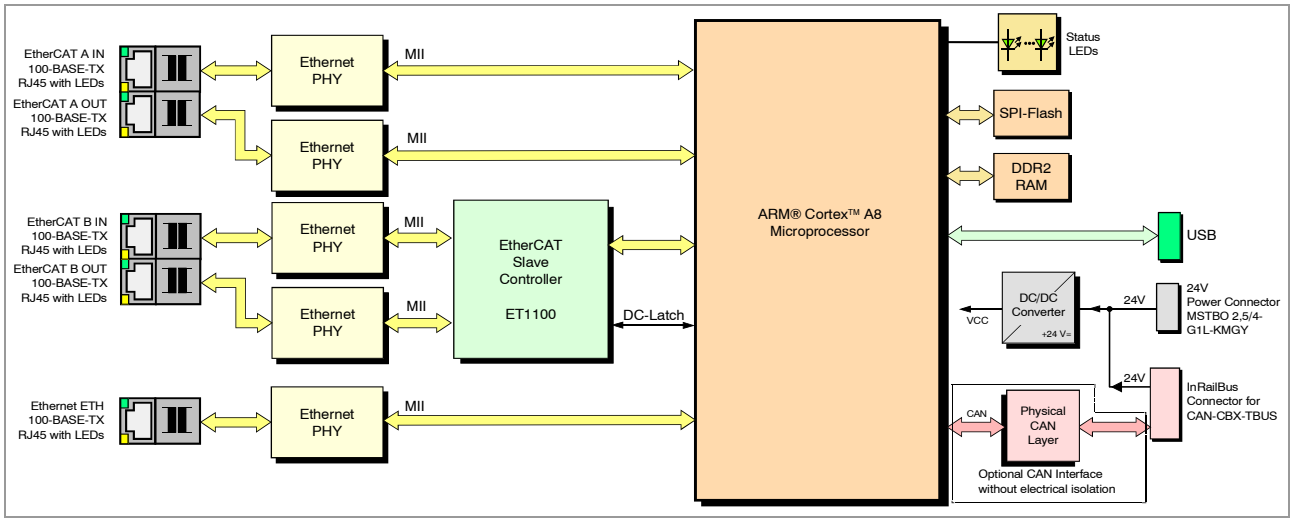



**Fig. 1:** Block circuit diagram

The ECX-EC EtherCAT Slave Bridge connects two EtherCAT segments. Therefore the EtherCAT slave bridge implements two separate EtherCAT slave interfaces for integration into two EtherCAT networks. The bridge allows EtherCAT process data exchange between the networks.

For synchronization of Distributed Clocks (DC) the ECX-EC offers the exact difference between the two slave time stamps as CoE object to allow one master adjusting its time to the other.
For DC synchronization in the redundancy case the ECX-EC bridge is used as first and last slave simultaneously, which allows the master to keep all slaves synchronized in both segments.

Additionally the ECX-EC comes with an Ethernet interface, that can be used as Ethernet over EtherCAT (EoE) Switch Port.

The ECX-EC is a compact hat-rail module designed for carrier rail mounting (TS 35).
The 24 V-power supply can be applied via a connector with spring-cage connection or optionally via the InRailBus connector (TBUS-connector) integrated in the mounting-rail.
Furthermore, an optional CAN interface can be realized, accessible via the InRailBus connector.

The 100BASE-T-EtherCAT interfaces are designed according to IEEE 802.3.

The ECX-EC can be configured by typical network configuration tools (e.g. esd Workbench, TwinCAT®, etc.).

**INFORMATION**
The enclosed ECX-EC-CD contains the documentation, the EtherCAT Workbench as network configuration and diagnostic tool for EtherCAT networks and the esd Software UDDC (Universal Description Composer).

# 2. Hardware Installation

## 2.1 Connecting Diagram

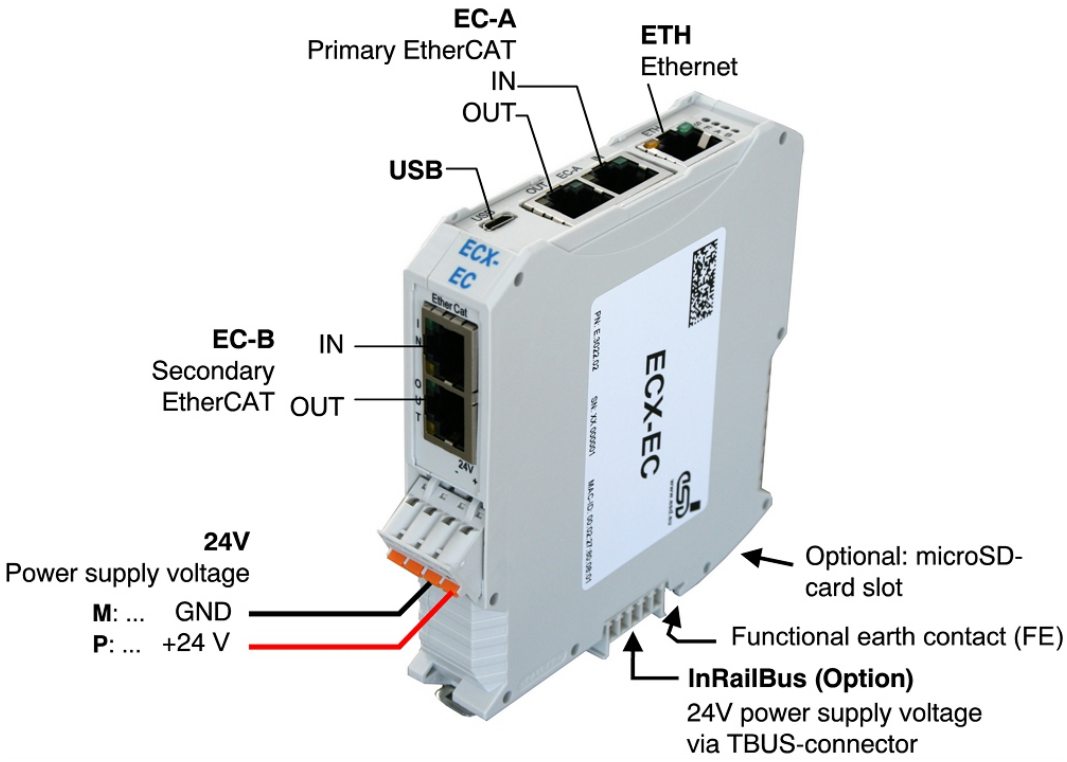



**Fig. 2:** Connections of the ECX-EC in ready-to-operate condition

**NOTICE**
Read chapter “Hardware Installation ” on page 14, before you start with the installation of the hardware!

**NOTICE**
Do not connect the USB interface!
It is currently without function.

**NOTICE**
The pins 1,2 and 3 of the mounting rail bus connector are reserved and must not be connected!

For signal assignment of the connectors see page 18 et seq.
For conductor connection and conductor cross section see page 20.

## 2.2 LED Display

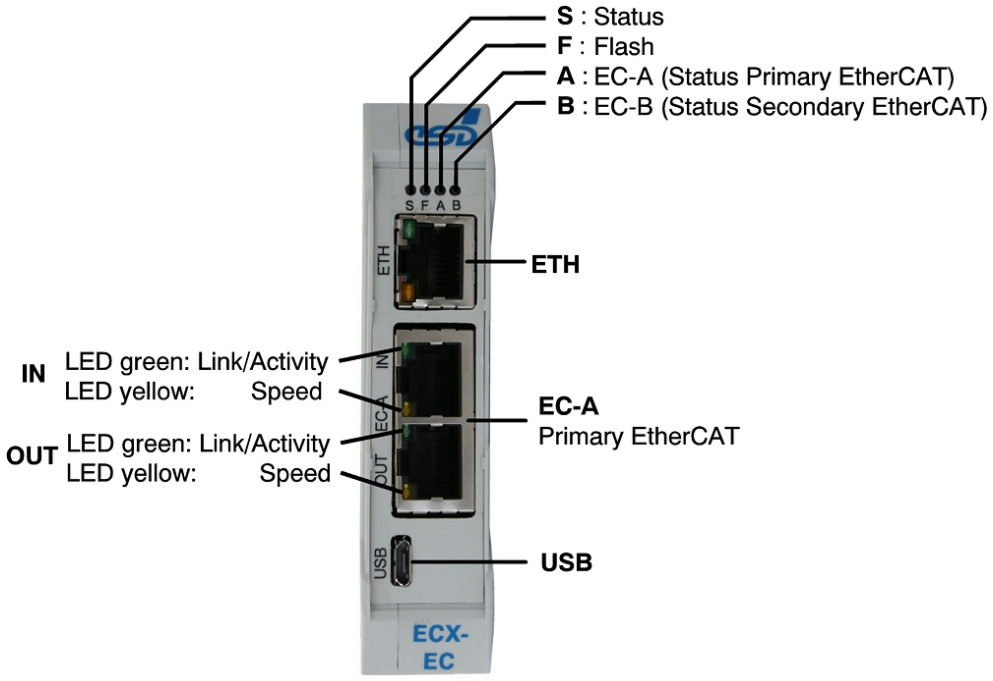



**Fig. 3:** Position of the LEDs

The position of the LEDs of the secondary EtherCAT interface EC-B in the RJ45 sockets *IN* and *OUT* and its display functions is equal to those of interface EC-A. The position of the sockets of EC-B can be taken from the connecting diagram (page 11).

**LEDs of the RJ45 sockets *IN* and *OUT* of the EtherCAT interfaces**

| LED | Colour | Indication | Description |
|---|---|---|---|
| **Link/ Activity** | green | off | no Ethernet link |
| | | blinking | Ethernet link is established,<br>Ethernet Activity (Receiving Ethernet data packages) |
| **Speed** | yellow | off | no link |
| | | on | 100 Mbit/s-link is established |

**Table 1:** Display function of the Ethernet LEDs (in RJ45 socket)

**EtherCAT-LEDs *S, F, A, B***

| Indicator states | Description |
|---|---|
| blinking | LED blinking cycle: 200 ms on, 200 ms off. |
| flickering | LED blinking cycle: 50 ms on, 50 ms off. |
| single flash | LED blinking cycle: 200 ms on, 1000 ms off. |
| double flash | LED blinking cycle: 200 ms on, 200 ms off, 200 ms on, 1000 ms off. |

**Table 2:** LED states (according to ETG.1300-documentation)

| LED | Function | | Color | Indicator state | Description | Name in schematic diagram |
|---|---|---|---|---|---|---|
| **S** | Status | | green | on | power supply voltage on | LED4 |
| | | | | flickering | firmware update finished, device has to be restarted | |
| **F** | Flash | | yellow | on | write access to the internal flash is active | LED3 |
| **A** | EC-A<br><br>(Status Primary EtherCAT) | Run* | green | off | Init | LED2A/B |
| | | | | flickering | BootStrap | |
| | | | | blinking | Pre-Operational | |
| | | | | single flash | Safe-Operational | |
| | | | | on | Operational | |
| | | Error | red | off | no error | |
| | | | | blinking | “EtherCAT state”- change failed | |
| | | | | single flash | “EtherCAT state”-change because of configuration error | |
| | | | | double flash | SM watchdog is triggered | |
| | | | | flickering | firmware update failed | |
| **B** | EC-B<br><br>(Status Sekundary EtherCAT) | Run* | green | off | Init | LED1A/B |
| | | | | blinking | Pre-Operational | |
| | | | | single flash | Safe-Operational | |
| | | | | on | Operational | |
| | | Error | red | off | no error | |
| | | | | blinking | “EtherCAT state”- change failed | |
| | | | | single flash | “EtherCAT state”- change because of configuration error | |
| | | | | double flash | SM watchdog is triggered | |

* “Error” overwrites “Run”, i.e. “Run” is always off if Error is not in “off”’ status.

**Table 3:** Display function of the LEDs

**Other states**

| Status | Description |
|---|---|
| All LEDs on | device is booting |
| All LEDs flickering | boot procedure failed |

# 3. Hardware Installation

To put the ECX-EC into operation, please follow the installation notes.

| Step | Procedure | see page |
|---|---|---|
|  | **NOTICE**<br>**Read the safety instructions at the beginning of this document carefully, before you start with the hardware installation!** | 4 |
|  | **WARNING**<br>Hazardous voltage - **Risk of electric shock**<br>All current circuits which are connected to the device have to be sufficiently protected against hazardous voltage (SELV according to EN 60950-1) before you start with the installation. | |
| 1. | Mount the ECX-EC module and connect the interfaces (power supply voltage, EtherCAT A and B). | 11 |
| 2. | If you use the InRailBus, please make sure to have read the chapter: “Appendix InRailBus (Option)”. | 67 |
| 3. | Switch on the 24 V-power supply voltage of the ECX-EC. | - |
| 4. | Please go on with the software configuration | 21 et seqq. |

# 4. Technical Data

## 4.1 General Technical Data

| | | |
|---|---|---|
| Power supply voltage | Nominal voltage: typical: 24 V/DC, (min.: 18 V, max.: 32 V)<br>Current consumption: (24 V, 20 °C): 100 mA<br>reverse voltage protection | |
| Connectors | 24V | 24 V-power supply voltage (X1, 4-pol. Phoenix Contact connector with spring-cage connection) |
| | EC-A | Primary EtherCAT I/F (EtherCAT0)<br>(X3a/b, 8-pin Dual-RJ45 with LEDs) |
| | EC-B | Secondary EtherCAT I/F (EtherCAT1)<br>(X2a/b, 8-pin Dual-RJ45 with LEDs) |
| | InRailBus | Power supply voltage and optional CAN-Bus interface via InRailBus (X6, 5-pin TBUS-connector, accessories) |
| | ETH | Ethernet interface, optional (X4, 8-pin RJ45 socket) |
| | USB | reserved<br>(USB device (X5, Micro USB type-B socket)) |
| Temperature range | 0 °C ... 60 °C ambient temperature | |
| Humidity | max. 90 %, non-condensing | |
| Protection class | IP20 | |
| pollution degree | maximum 2 (according to DIN EN 61131-2) | |
| Housing | Plastic housing for carrier rail mounting NS35/7,5 DIN EN 60715 | |
| Dimensions | Width: 22.5 mm, height: 99 mm, depth: 114.5 mm | |
| Weight | approx. 130 g | |

**Table 4:** General data of the module

## 4.2 Microprocessor and Memory

| | |
|---|---|
| CPU | TI, AM3357, min. 600 MHz |
| Main memory | 64 MB DDR2, 16 bit |
| Flash memory (NOR) | Max. 8 MB (via SPI) |
| Flash memory (NAND) | - |
| EEPROM | 24C32 |
| Watchdog | CPU internal |

**Table 5:** Microprocessor and memory

### 4.2.1 EtherCAT Slave EC-A (Primary)

| | |
|---|---|
| Number | 1 |
| Controller | PRUs of the CPU with Micrel KSZ8051 Ethernet Phys |
| Physical interface | 2 x RJ45 with internal transformer and LEDs |
| Connector | 2 x RJ45 with LEDs |

**Table 6:** Data of the EC-A EtherCAT interface

### 4.2.2 EtherCAT Slave EC-B (Secondary)

| | |
|---|---|
| Number | 1 |
| Controller | ET1100 with 2 x MII Phys (Micrel KSZ8051) |
| Physical interface | 2 x RJ45 with internal transformer and LEDs |
| Connector | 2 x RJ45 with LEDs |

**Table 7:** Data of the EC-B EtherCAT interface

## 4.3 Ethernet Interface

| | |
|---|---|
| Number of Ethernet interfaces | 1 |
| Controller | CPU internal EMAC with Micrel KSZ8051 Ethernet Phy |
| Bit rate | 100BASE-TX, 10/100 Mbit/s |
| Connection | Twisted pair (compatible to IEEE 802.3) |
| Electrical isolation | via transformer |
| Physical interface | IEEE 802.3 via copper |
| Connector | RJ-45-socket with integrated LEDs in the front panel |

**Table 8:** Data of the Ethernet interface

## 4.4 USB Interface

| | |
|---|---|
| Number | 1 |
| Controller | CPU internal controller |
| Bit rate | Max 480Mbit/s (USB High Speed) |
| Physical interface | CPU intern |
| Connector | Micro USB type-B socket |

**Table 9:** Data of the USB interface

## 4.5 Software Support

For the configuration of the ECX-EC common EtherCAT configuration tools, as e.g. esd Workbench or TwinCAT®, can be used. (For examples see chapter “Software” , page 21 et seqq.)

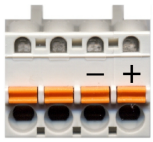
**INFORMATION**
The enclosed ECX-EC-CD contains the EtherCAT Workbench as network configuration and diagnostic tool for EtherCAT networks and the esd Software UDDC (Universal Description Composer).

# 5. Connector Assignments

## 5.1 24 V-Power Supply Voltage

**Device socket:** Phoenix Contact MSTBO 2,5/4-G1L-KMGY
**Line connector:** Phoenix Contact FKCT 2,5/4-ST, 5.0 mm pitch, spring cage connection,
Phoenix Contact order No.: 19 21 90 0 (included in the scope of delivery)
For conductor connection and conductor cross section see page 20.

**Pin Position:**




**Pin Assignment:**

| Labelling of the ECX-EC | | | 24V | |
|---|---|---|---|---|
| | . | . | M | P |
| Connector label | (free) | (free) | - | + |

| Pin No. | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Signal | **Do not connect!** | **Do not connect!** | M24 (GND) | P24 (+ 24 V) |

Please refer to the connecting diagram page 11.

**NOTICE**
Feeding through the +24V power supply voltage can cause damage on the modules.
It is not permitted to feed through the power supply voltage through the 24V connector and to supply the power supply voltage to another CAN module station!

**Signal Description:**

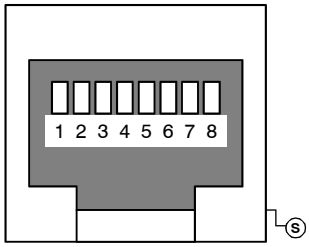
P24... power supply voltage +24 V (min.: 18 V, max.: 32 V)
M24... reference potential

## 5.2 EtherCAT EC-A, EC-B

Both EtherCAT interfaces have the same pin-assignment, each for the corresponding EtherCAT port.

**Device connector:** RJ45 socket, 8-pin
Ethernet 100BASE-TX, assigned according to IEEE 802.3-2008,
Table 25-3 “UTP MDI Contact Assignment”

**Pin Position:**




**Pin Assignment:**

| Pin | Signal | Meaning |
|---|---|---|
| 1 | EC-X TxD+ | Transmit Data + |
| 2 | EC-X TxD- | Transmit Data - |
| 3 | EC-X RxD+ | Receive Data + |
| 4 | - | - |
| 5 | - | - |
| 6 | EC-X RxD- | Receive Data - |
| 7 | - | - |
| 8 | - | - |
| S | Shield | |

The pins 1 to 8 are connected to termination.

**Signal Description:**

EC-X TxD+/-,
EC-X RxD+/- ... EtherCAT data lines of EtherCAT port X (X = A or B)
- ... reserved for future applications, do not connect!
Shield... case shield, connected with the front panel of the ECX-EC.

**NOTICE**
Permissible cable types: Cables of category 5e or higher have to be used to grant the function in networks with up to 100 Mbits/s.
esd grants the EC conformity of the product if the wiring is carried out with shielded twisted pair cables of class SF/UTP or higher.

## 5.3 Conductor Connection/Conductor Cross Sections

The following table contains an extract of the technical data of the line connectors.

| Interface | 24 V Power Supply Voltage [1] |
|---|---|
| Connector type plug component (Range of articles) | FKCT 2,5/..-ST KMGY |
| Connection method | spring-cage connection |
| Stripping length | 10 mm |
| Conductor cross section solid min. / max. | 0.2 mm² / 2.5 mm² |
| Conductor cross section stranded min. / max. | 0.2 mm² / 2.5 mm² |
| Conductor cross section stranded, with ferrule without plastic sleeve min./ max. | 0.25 mm² / 2.5 mm² |
| Conductor cross section stranded, with ferrule with plastic sleeve min. / max. | 0.25 mm² / 2.5 mm² |
| Conductor cross section AWG/kcmil min. / max. | 24 / 12 |
| 2 conductors with same cross section, stranded, TWIN ferrules with plastic sleeve, min. / max. | 0.5 mm² / 1.0 mm² |
| Minimum AWG according to UL/CUL | 26 |
| Maximum AWG according to UL/CUL | 12 |

1 Technical Data from Phoenix Contact website, printed circuit board connector, plug component

# 6. Software

The EtherCAT Slave Bridge ECX-EC features the data exchange between two independent EtherCAT segments. Therefore two separate EtherCAT slaves are contained in the hardware.
Each of this slaves is contained in one of the EtherCAT segments that have to be connected. The data exchange is done via the corresponding process data. The input data of one slave are the output data of the other slave, and vice versa.
For simplification the size of the process data is defined via variables. This is described in detail in chapter 6.2.3, the following chapter “Quick Start / First Steps” shows it exemplary with two EtherCAT configurators.

**INFORMATION**
Please be sure to perform the first steps in chapter “Quick Start / First Steps”, because it will make it much easier to understand the functionality and later troubleshooting will be considerably easier.

Additionally the ECX-EC features the synchronisation of the “Distributed Clocks” in both EtherCAT segments, see chapter 6.2.4.

## 6.1 Quick Start / First Steps

**INFORMATION**
The ECX-EC can be configured with the EtherCAT Workbench. The esd EtherCAT Workbench is contained on the supplied ECX-EC-CD.
For quick configuration of the ECX-EC the esd software **UDDC** (“Universal Device Description Composer”) can be used to generate a complete EtherCAT ESI file that already includes the variables.
The tool is included on the enclosed ECX-EC-CD. For further information read the UDDC Software Manual on the ECX-EC-CD.

As an introduction it will be described here how an EtherCAT configuration tool creates variables for the data exchange and how these variables are written and read.
For simplification of the description only one EtherCAT segment is used here, i.e. both slaves, and following both sides of the bridge, are connected with each other.
The result is one EtherCAT segment that consists of two slaves, whereas only one ECX-EC is used, see Fig. 4.

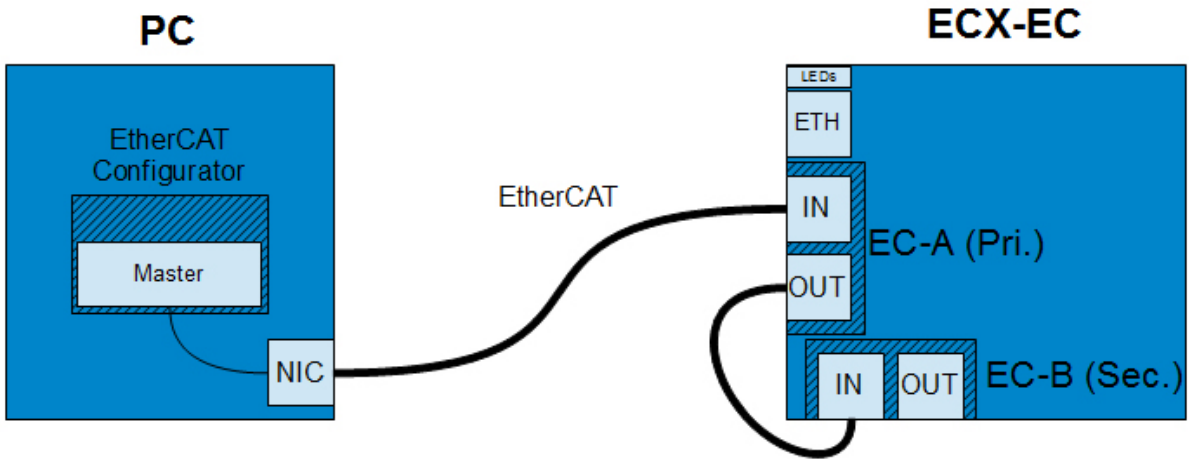



**Fig. 4:** Principle Quick Start / First Steps

In the following the procedure will be described for different EtherCAT configurators.
In principle the following steps have to be proceeded as follows:

1. Import the ESI (.xml description file of the ECX-EC) into the configurator
2. Scan the EtherCAT segment
3. Create input and output variables
4. Start up the EtherCAT segment (slaves in “Operational”)
5. Write variables and upload them into the other slave

## 6.1.1 Requirements

1. **Start-up of the hardware**
   See chapter 3.
2. **EtherCAT wiring**
   As in practically every EtherCAT slave the input is connected with the previous slave or master and the output is connected with the next slave.
   In the example in Fig. 4 the master (the PC's network card) must be connected with the input (EC-A IN) of the primary side of the ECX-EC at first. Then the output (EC-A OUT) of this slave has to be connected to the input (EC-B IN) of the secondary slave.
   (For the functionality it is irrelevant if the primary or secondary side is connected first – only in- and output ports of a slaves must be used correctly)
3. **Installation of the configurator**
   This is not subject of this description, for further information please refer to the corresponding documentation.

## 6.1.2 esd EtherCAT Workbench

The procedure herein is shown exemplary for the EtherCAT Workbench Version 1.2.0. The figures might differ for other versions, but the basic procedure is the same.

### *6.1.2.1 ESI Installation*

Either

- Start the EtherCAT Workbench and choose *Copy ESI file(s) to slave library* in the upper *Tools* menu.
- In the following dialogue choose the file: `ESD ECX-EC.xml`

or:

- Copy the file `ESD ECX-EC.xml` into the subdirectory *SlaveLibrary* of the Workbench installation.
  (For example:
  `C:\Program Files\esd\EtherCAT\EtherCAT Workbench\SlaveLibrary\`)

afterwards

- Restart the EtherCAT Workbench

### 6.1.2.2 Scan the EtherCAT Segment

To establish a connection to the master click the button *Online.*
A dialogue with a request opens. Confirm here to perform a scan. (In newer versions of the Workbench this step will be carried out automatically if the slaves were not yet assigned.)

If the ECX-EC is successfully detected, the following dialogue is shown.

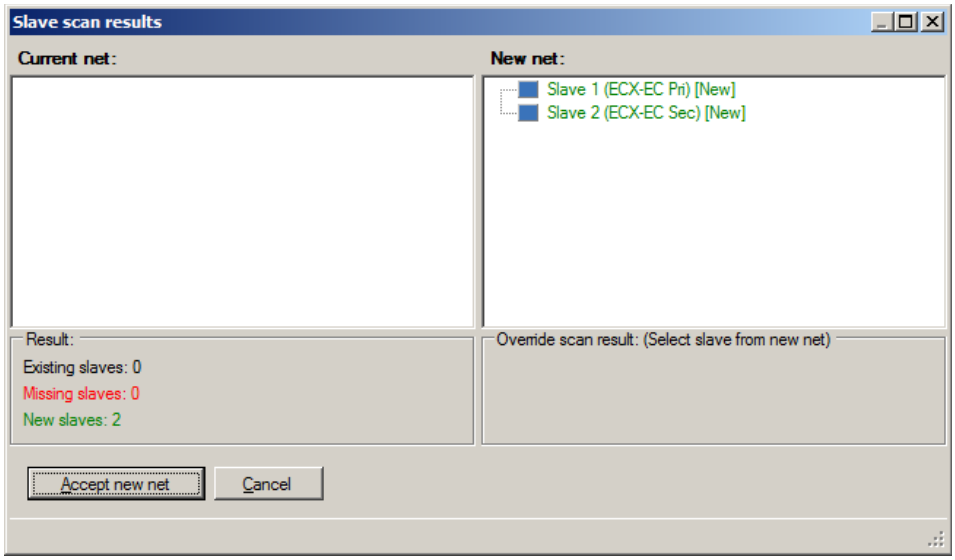



**Fig. 5:** ECX-EC at network scan of the EtherCAT Workbench

Confirm with *Accept new net.*

A successfully detected ECX-EC is shown in the following structure:

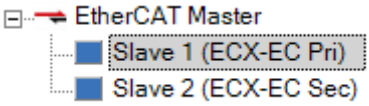



**Fig. 6:** ECX-EC in the network overview of the EtherCAT Workbench

**Potential problems**

- Error message “No slaves found [...]” and Workbench warning: “Master: Link lost on primary network interface”, status of the master:
  → ECX-EC without power supply voltage
  → Faulty wiring

- Workbench output “Master: Scanning slaves failed with [Request timed out] (Failed to get slave count)”, status of the master:
  → In-/Output ports of the slave mixed up

- Warning “Found unknown slave (no ESI file). Working with slave's EEPROM data” in Log Range:
  → Install ESI file, restart Workbench and begin again (see 6.1.2.1, ESI Installation)

### 6.1.2.3 Creating Variables

For the ECX-EC special tabs are created under *Device Specific* in the EtherCAT Workbench, see Fig. 7 and Fig. 8.
These tabs simplify the creation of variables and allow the transmission to the corresponding opposite side. (In principle this is also possible without this special configuration tab, see 6.2.3 for details.)

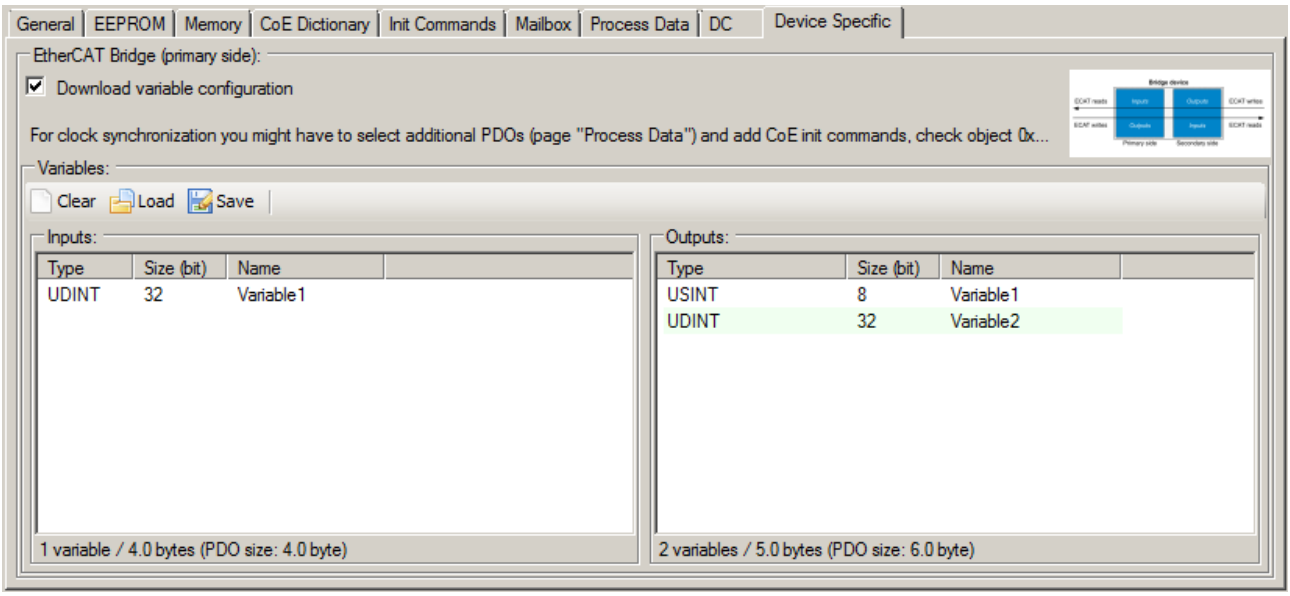



**Fig. 7:** Workbench configuration tab for the ECX-EC (Primary)

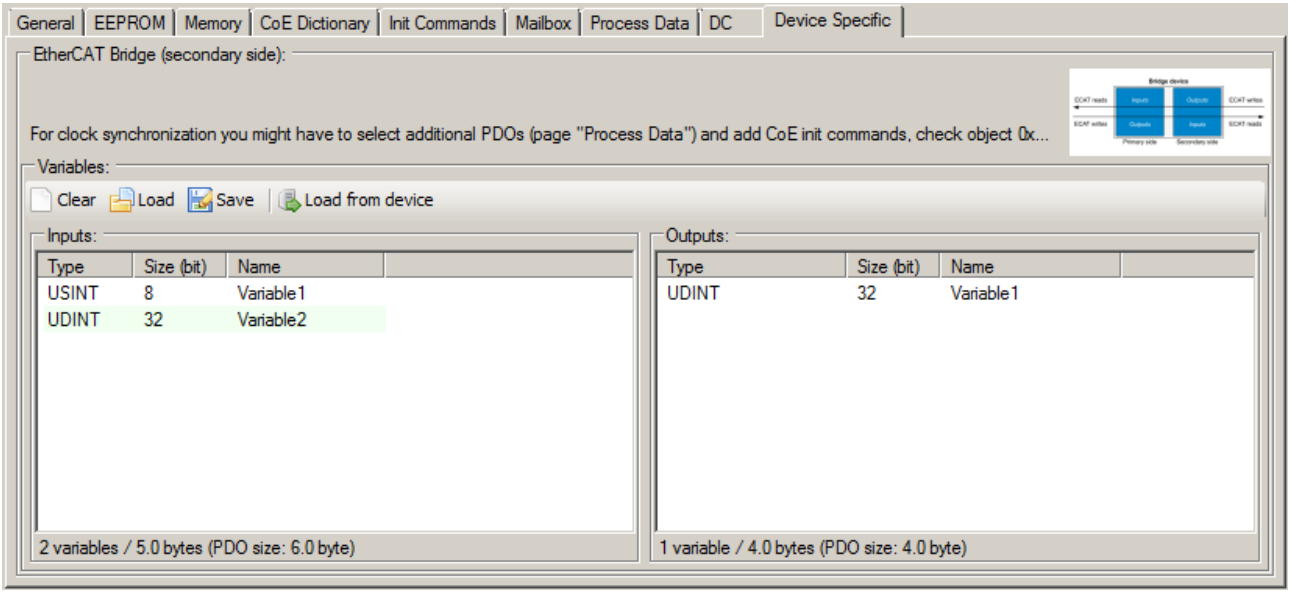



**Fig. 8:** Workbench configuration tab for the ECX-EC (Secondary)

In this example the variables are created exactly as shown in Fig. 7 and Fig. 8.

**Primary side**
Select the slave of the ECX-EC primary side: *Slave 1 (ECX-EC Pri)* (Fig. 6) and go to the corresponding Workbench configuration tab (Fig. 7).
Right-click in the list of *Inputs:* to open a context menu and select *Append item* to append a new input variable:

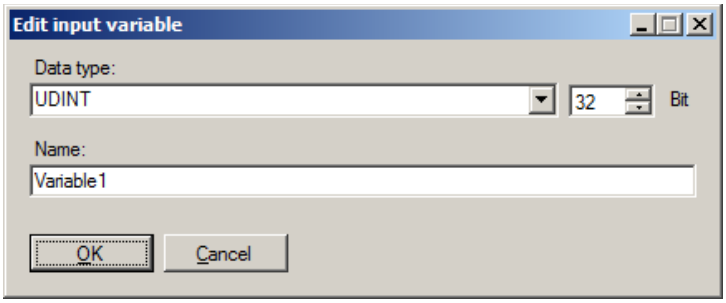



**Fig. 9:** Edit input variable, EtherCAT Workbench

Choose “UDINT”, “32 bit” and name the variable “Variable1” as shown in Fig. 9.

Now create two output variables in the list *Outputs:* :
1. “USINT”, “8 bit” and “Variable1”
2. “UDINT”, “32 bit” and “Variable2”

Make sure that the variables are the same as those described in Fig. 7.

**Secondary side**
Select the slave of the ECX-EC secondary side: *Slave 2 (ECX-EC Sec)* (Fig. 6) and go to the corresponding Workbench configuration tab (Fig. 8).

Because the input variables of one side must correspond to the output variables of the other side two input variables must be created under *Inputs:*
1. “USINT”, “8 bit” and “Variable1”
2. “UDINT” “32 bit” and “Variable2”

and one output variable under *Outputs*:
1. “UDINT”, “32 bit” and “Variable1”

Continue analogous to the creation of the variables on the primary side and make sure that the variables correspond to those shown in Fig. 8.

### 6.1.2.4 Test Data Transfer

In the online mode the data transfer can be tested.
If you are not already in this mode, change into the online mode by clicking the *Online* button.
Then click the *Free run* button to switch into *Free run* mode.
The states of the slaves and of the master should be *Operational* after a short time, as indicated by the -symbols in Fig. 10.

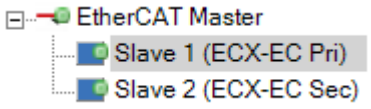



**Fig. 10:** EtherCAT Workbench with ECX-EC in *Operational* state

If the states are *Operational* go on reading at *Read and write variables*.
In case of problems please check the outputs in the EtherCAT Workbench and read the following note.

**INFORMATION**
The most probable reason of the problem that might occur here is that different configurations for primary and secondary side have been created at the configuration of the variables. In this case the ECX-EC does not switch into *Operational* or *Safe-Operational* (with error *Inconsistent Settings – AI Statuscode* 0x0026).

Make sure that the input variables of the primary side comply with the output variables of the secondary side and vice versa (the name of the variables can be disregarded).

**Read and write variables**

Select tab *Process Data/Image* of the Workbench and choose tab *Variables*.
All process variables are shown in this list. Output variables can be changed/written and input-variables can be updated/read.
To make it easier to find the generated variables it is advisable to set a filter (e.g. “Vari “ as shown in Fig. 11) to display only the variables needed.

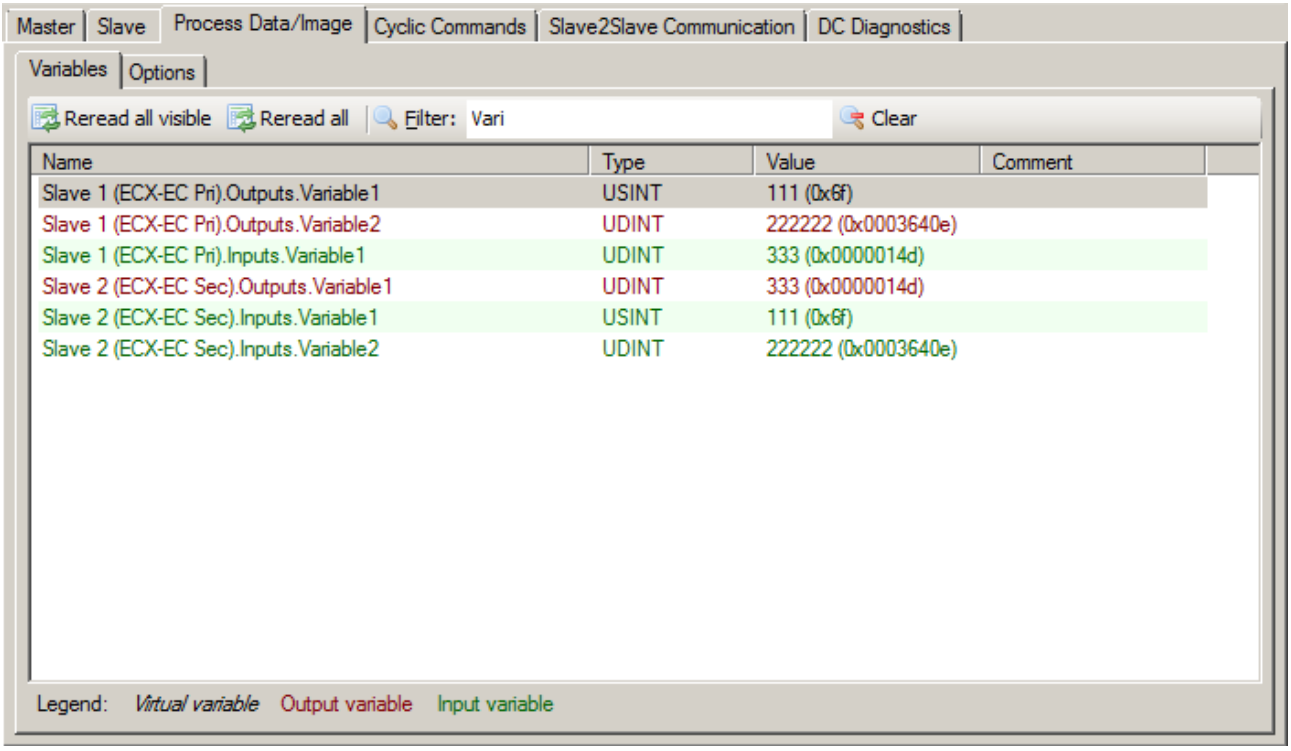



**Fig. 11:** EtherCAT Workbench process variables

Write any value into the output variable and check whether the input variables of the corresponding other side are the same. Double-click to an output variable to change its value.
Or click *Reread all visible* to update all input variables.

In the example in Fig. 11 the output variable *Slave 1 (ECX-EC Pri).Outputs.Variable1* is written with the value “111”. The corresponding input variable of the opposite side, *Slave 2 (ECX-EC Sec).Inputs.Variable1* shows this value.

The output variable *Slave 2 (ECX-EC Sec).Outputs.Variable1* of the opposite side has been written with “333” and the corresponding input variable *Slave 1 (ECX-EC Pri).Inputs.Variable1* shows the same value.
The test shows that the data transmission has been successful.

## 6.1.3 Beckhoff TwinCAT®

The procedure described herein is shown exemplary with the TwinCAT system manager version 2.11. The figures might differ for other versions, but the basic procedure is the same and can be applied to other configurators.

### 6.1.3.1 ESI Installation

Copy the file: `ESD ECX-EC.xml` into the subdirectory `Io\EtherCAT` of the TwinCAT installation. (For example `C:\Program Files\TwinCAT\Io\EtherCAT\`).
Restart TwinCAT now.

### 6.1.3.2 Scan EtherCAT Segment

Choose *I/O - Configuration* in the menu and select the entry *I/O Devices* in the tree structure.
Now you can select the entry *Append device* in the context menu and in the following dialogue *EtherCAT* the device *EtherCAT* can be selected.
Now select *Scan Boxes...* in the context menu of the EtherCAT node i.e. under *Device 1 (EtherCAT)*:

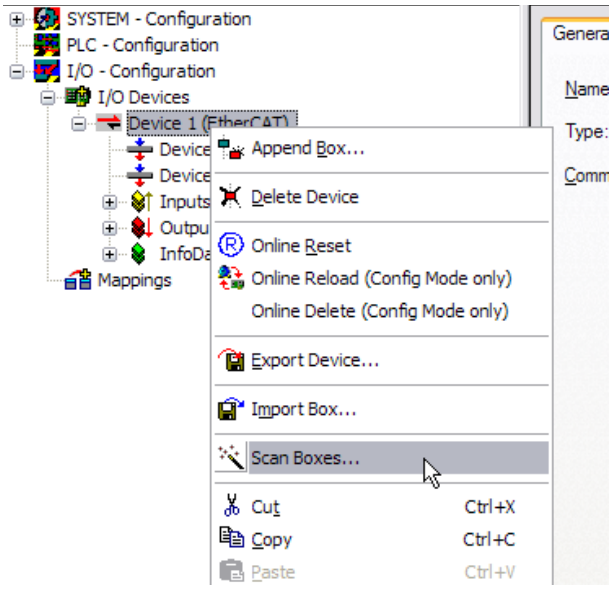



**Fig. 12:** TwinCAT EtherCAT segment scan

The detected slaves are shown in the following structure:

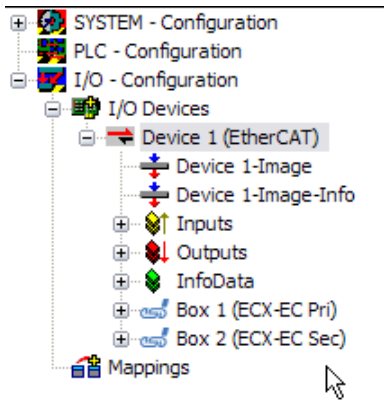



**Fig. 13:** ECX-EC in network overview of TwinCAT

### 6.1.3.3 Create variables

Because the ECX-EC can not be detected as EtherCAT bridge by TwinCAT currently, the variables have to be created via configuration of the PDOs (for details see chapter 6.2.3).

**Primary side**
Select the slave of the ECX-EC's primary side: *Slave 1 (ECX-EC Pri)* (Fig. 13, “Box 1”) and go to tab *Process Data*:

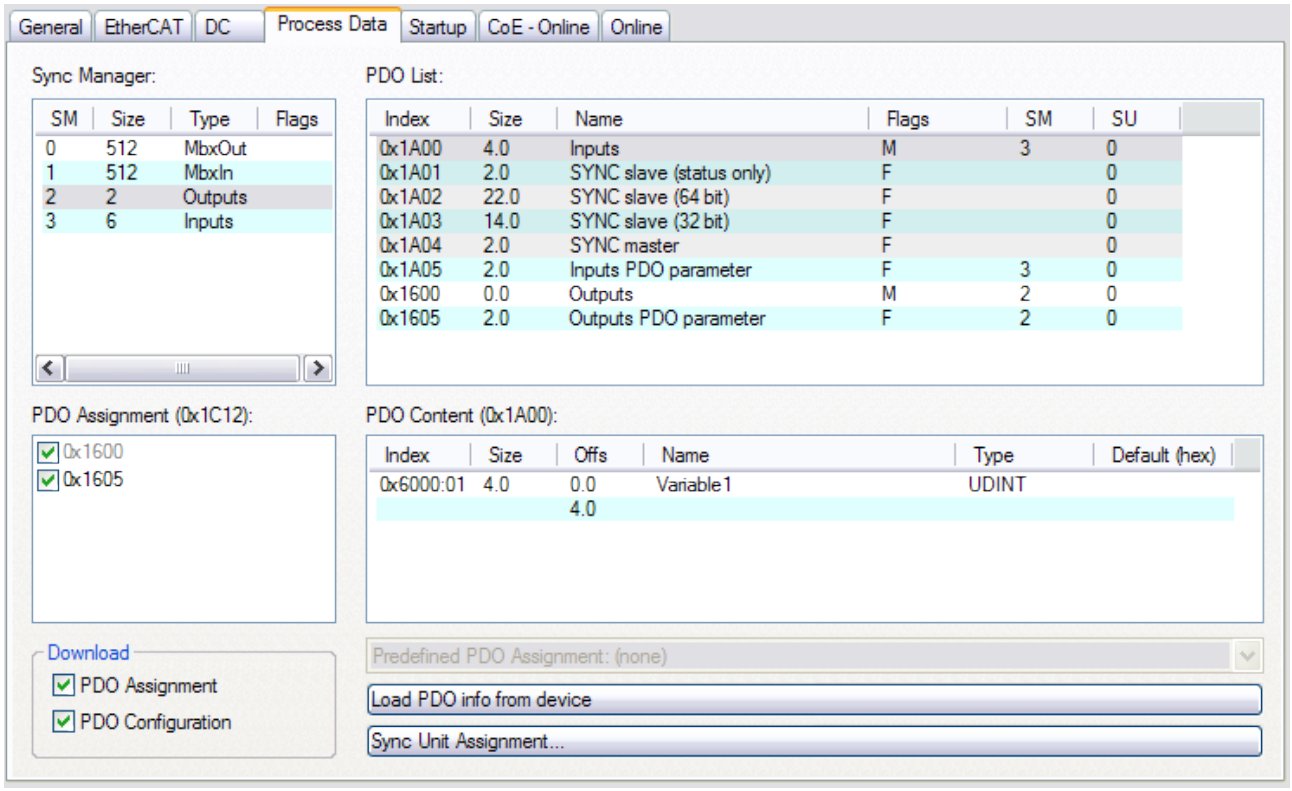



**Fig. 14:** TwinCAT tab *Process Data*

Select PDO “0x1A00” in the upper list of PDOs (*PDO List:).*
The content of the PDO is shown in the list below *PDO Content (0x1A00).*
Click *Insert* in the context menu of the list *PDO Content (0x1A00)* and then insert the object 0x6000 sub index 1 with name *Variable1* and Type “UDINT”:

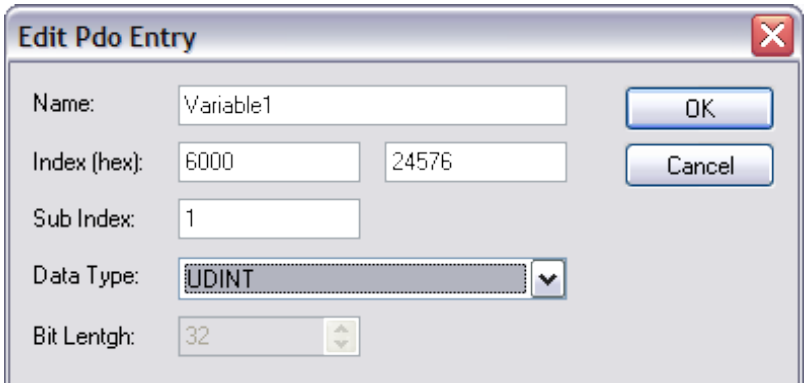



**Fig. 15:** TwinCAT *Edit PDO Entry*

In the upper *PDO List* select PDO “0x1600” for the outputs and insert:

1. Object 0x7000 Sub Index 1 with name *Variable1* and data type “USINT”
2. Object 0x7000 Sub Index 2 with name *Variable2* and data type “UDINT”

**Secondary side**

Select the slave of the ECX-EC's secondary side: *Slave 2 (ECX-EC Sec)* (Fig. 13, “Box 2”) and switch to tab *Process data* (Fig. 14).

Because the input variables of one side must correspond to the output variables of the other side two input variables:

1. “USINT”, “8 bit” and “Variable1”
2. “UDINT”, “32 bit” and “Variable2”

and one output variable:

1. “UDINT”, “32 bit” and “Variable1”

have to be created here.

Continue analogous to the creation of the variables on the primary side and make sure that the variables correspond to those shown in Fig. 16:

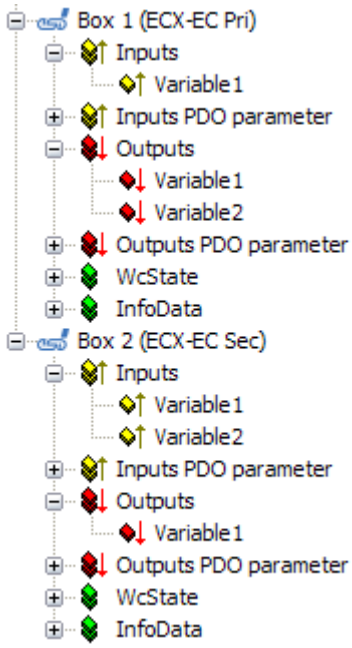



**Fig. 16:** TwinCAT network structure with variables shown in expanded subdirectories

### 6.1.3.4 Test Data Transfer

Click the *Reload I/O Devices* button to switch into *Free Run* state (see upper menu bar):

**Fig. 17:** TwinCAT *Reload I/O Devices* button

The dialogue *Activate Free Run* opens and has to be confirmed. Make sure that all slaves are in “OP” state:

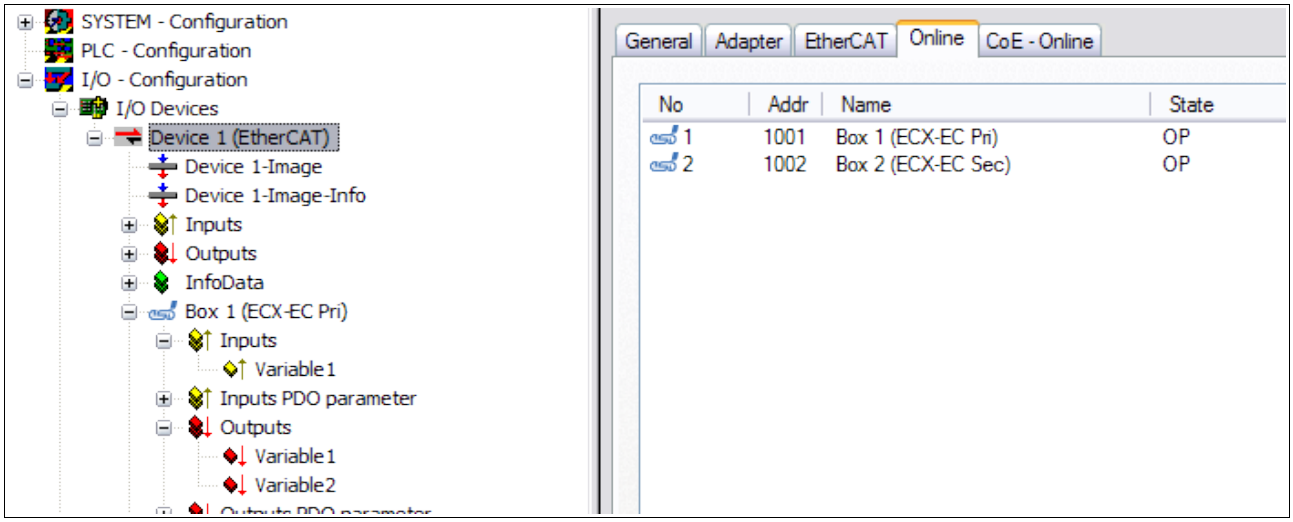



**Fig. 18:** TwinCAT tab *Online*

If the states are *Operational* read *Read and write variables* for further information.
In case of problems please check the outputs in the EtherCAT Workbench and read the following note.

**INFORMATION**
The most probable reason of the problem that might occur here is that different configurations for primary and secondary side have been created at the configuration of the variables. In this case the ECX-EC does not switch into *Operational* or *Safe-Operational* (with error *Inconsistent Settings – AI Statuscode* 0x0026).

Make sure that the input variables of the primary side comply with the output variables of the secondary side and vice versa (the name of the variables can be disregarded).

### Read and write variables

Select the first output variable of the primary side and click *Online Write..* in the context menu:

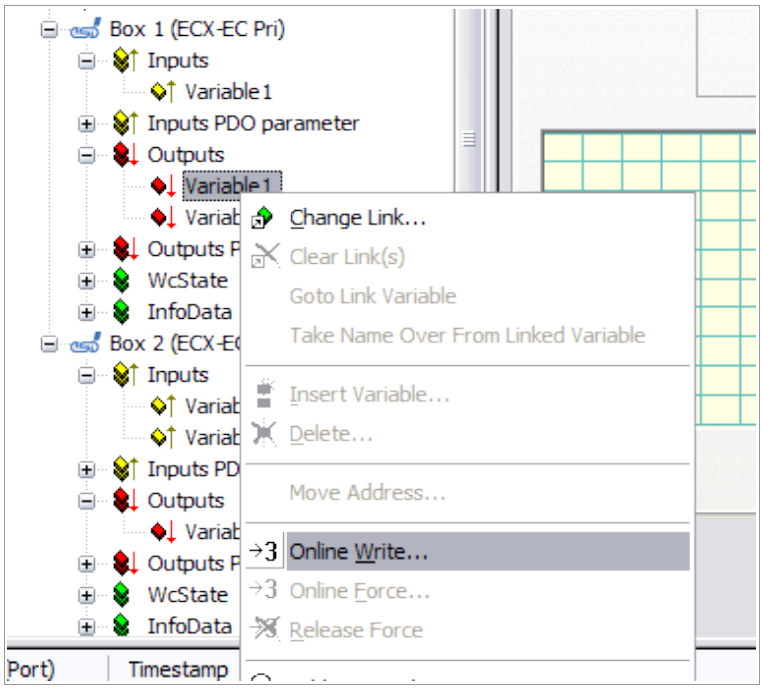



**Fig. 19:** TwinCAT context menu output variable

In the following dialogue the value of the variable can be entered. In this example “123” is written.

Click on the corresponding input variable of the other side and choose the tab *Online*:

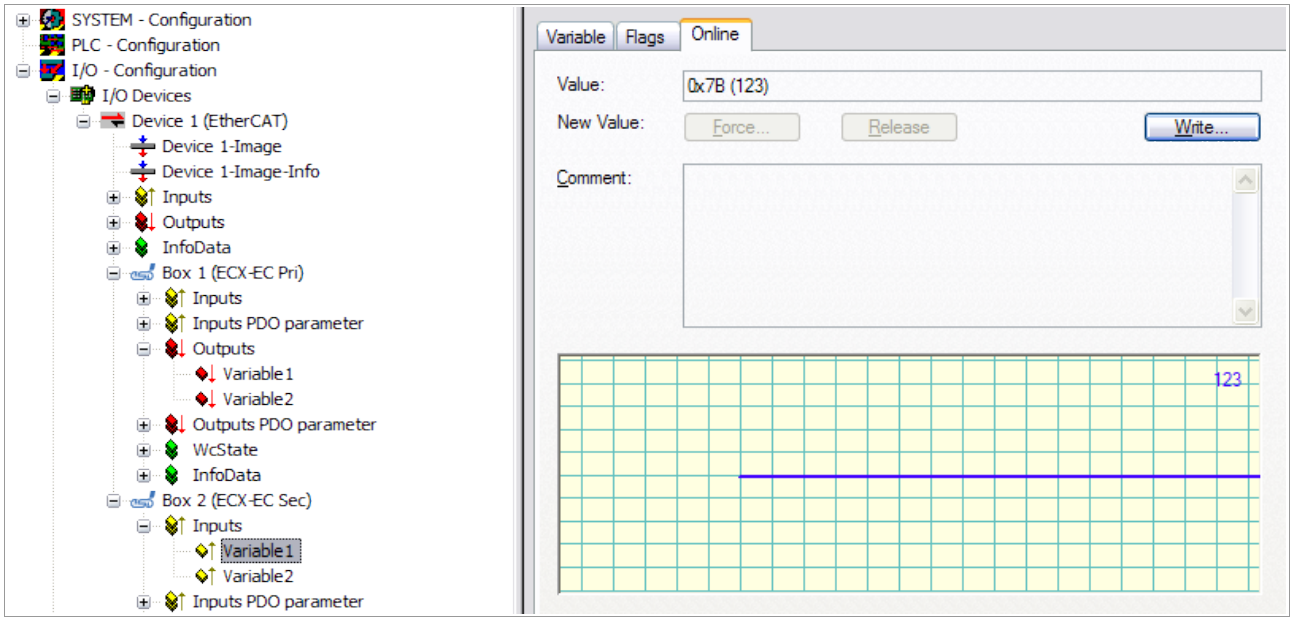



**Fig. 20:** TwinCAT input variable, tab *Online*

The value “0x7B (123)” is the entered value – the data transfer has been successful.

## 6.2 Start-Up

**NOTICE**
The enclosed ECX-EC-CD contains the EtherCAT Workbench as network configuration and diagnostic tool for EtherCAT networks (see chapter “esd EtherCAT Workbench”, on page 22).
For quick configuration of the ECX-EC the esd software **UDDC** (“Universal Device Description Composer”) can be used to generate a complete EtherCAT ESI file that already includes the variables.
The tool is included on the enclosed ECX-EC-CD. For further information read the UDDC Software Manual on the ECX-EC-CD.

### 6.2.1 Functionality of the ECX-EC

As already mentioned in the Overview, the ECX-EC connects two EtherCAT Slaves (primary and secondary side). Each of them can be seen as I/O module that receives its input data as output data of the respective other I/O module:

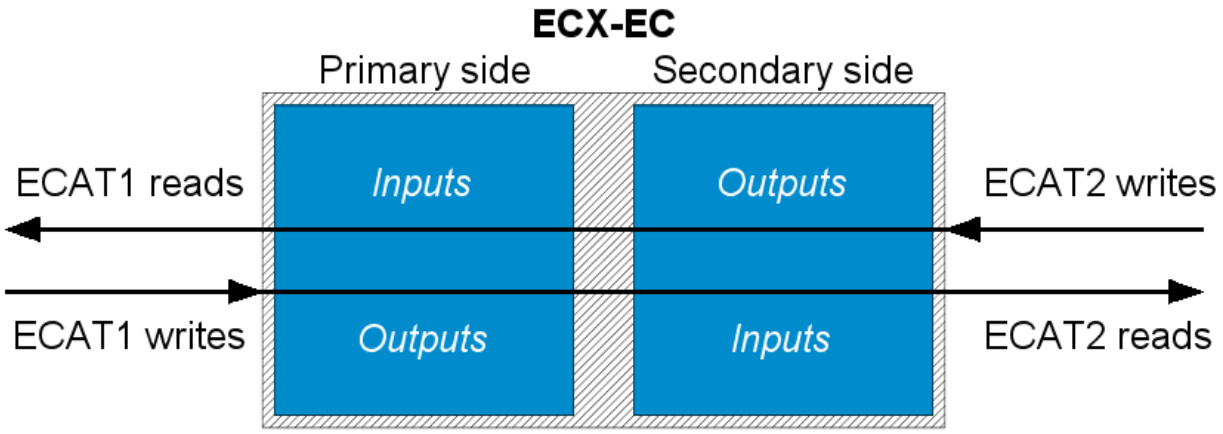



**Fig. 21:** Functionality of the ECX-EC

“ECAT1” and “ECAT2” are two independent EtherCAT segments between which data are exchanged.
For this data exchange in principle only the size of the in- and output data has to be specified. The ECX-EC only copies the data to the corresponding other side. The size is not entered directly but results from the definition of the in- and output variables on both sides.
This is exemplified in chapter 6.1. For further details see chapter 6.2.3.

### 6.2.2 Performance

The duration of the copy operation within the ECX-EC mainly depends on the amount of data: usually it can be expected to be around 1 µs per byte.

With 100 bytes in- and output data on both sides for example, after the ECX-EC received an EtherCAT frame with new output data it will have made that data available as input data on the other side after approx. 100 µs. (See also chapter 6.3.2.3, CoE object 0x2010.09 “Provision time”)

Note that both sides have equal rights, i.e. one side’s copy operation blocks the other side. Therefore the worst case for the sample above is 200 µs.

### 6.2.3 Definition of the Bridge Variables

The PDO 0x1A00 is specified for input data and PDO 0x1600 for output data.
The definition of the corresponding variables, and thereby the size of the process data, result from the objects that are mapped into these PDOs.
As input variables the entries of object 0x6000 and as output variables the entries of object 0x7000 are used.
The length of the in- and output variables may be at maximum 1024 byte in sum.

**Example** (corresponds to the example for the primary side in chapter 6.1 / Fig. 7)

- one input variable, 32 bit
- two output variables, 8 and 32 bit

The following PDO content results:

| PDO | Sub index | Data | Description |
|---|---|---|---|
| 0x1A00 | 0x00 | 0x01 | Max. sub index / number of inputs |
| 0x1A00 | 0x01 | 0x60000120 | Object 0x6000 sub index 1, 32 bit |
| 0x1600 | 0x00 | 0x02 | Max. sub index / number of entries |
| 0x1600 | 0x01 | 0x70000108 | Object 0x7000 sub index 1, 8 bit |
| 0x1600 | 0x02 | 0x70000220 | Object 0x7000 sub index 2, 32 bit |

The sub indices of the variables used (0x6000/0x7000) are incremented for each variable. If another input variable would be mapped, it would be 0x6000 sub index 2, the following output variable 0x7000 sub index 3, an so on.
(For alignment purposes the object 0x0000 sub index 0x00 can be additionally mapped in any of this two PDOs. The ECX-EC does not need this.)

**INFORMATION**
The input variables of one side must exactly match the output variables of the other side, otherwise the ECX-EC Slave will not allow the change into the “SafeOp/Op”-state.

For the example the content of the PDO of the other side is:

| PDO | Sub index | Data | Description |
|---|---|---|---|
| 0x1A00 | 0x00 | 0x02 | Max. sub index / number of entries |
| 0x1A00 | 0x01 | 0x60000108 | Object 0x6000 sub index 1, 8-bit |
| 0x1A00 | 0x02 | 0x60000220 | Object 0x6000 sub index 2, 32-bit |
| 0x1600 | 0x00 | 0x01 | Max. sub index / number of entries |
| 0x1600 | 0x01 | 0x70000120 | Object 0x7000 sub index 1, 32-bit |

**INFORMATION**
Ideally the configuration of the PDOs shown above is simplified by an EtherCAT configuration tool. An example for the configuration of the variables with the EtherCAT Workbench is shown in chapter 6.1.2.

## 6.2.4 Distributed Clocks

Basically each single side of the ECX-EC acts as an ordinary DC slave – its time can be distributed to the following slaves as a reference or if another reference is present, it can be simply adjusted. (The data transfer to/from the other side is thereby independent of the SYNC signal)
The specific characteristic of the ECX-EC is, that the difference of the DC times on both of its sides can be determined with high-precision. For this the “Latch Units” of both EtherCAT slave controllers are used: At regular intervals (object 0x10F5.11, see chapter 6.3.1.7) time stamps of both sides are recorded and provided in the process data (object 0x10F4, see chapter 6.3.1.6).

By means of the difference of these two time stamps a control loop can be fed. For example in Fig. 24 the second master adjusts its own time to the time of the first EtherCAT segment and distributes it to the slaves of the second segment, while Fig. 22 shows the automatic synchronisation.

### 6.2.4.1 Automatic Synchronisation

In “DC auto sync” mode the ECX-EC controls the DC times itself and regularly adapts the time on primary side to the time on secondary side.
It is activated in object 0x2000.05, see 6.3.2.1. The adaptation occurs in the interval set by object 0x10F5.11 (see 6.3.1.7), what should always be set to 1 ms in this mode.
By object 0x2000.06 in addition it is possible to set a target difference, for the case a constant offset occurs or is needed.

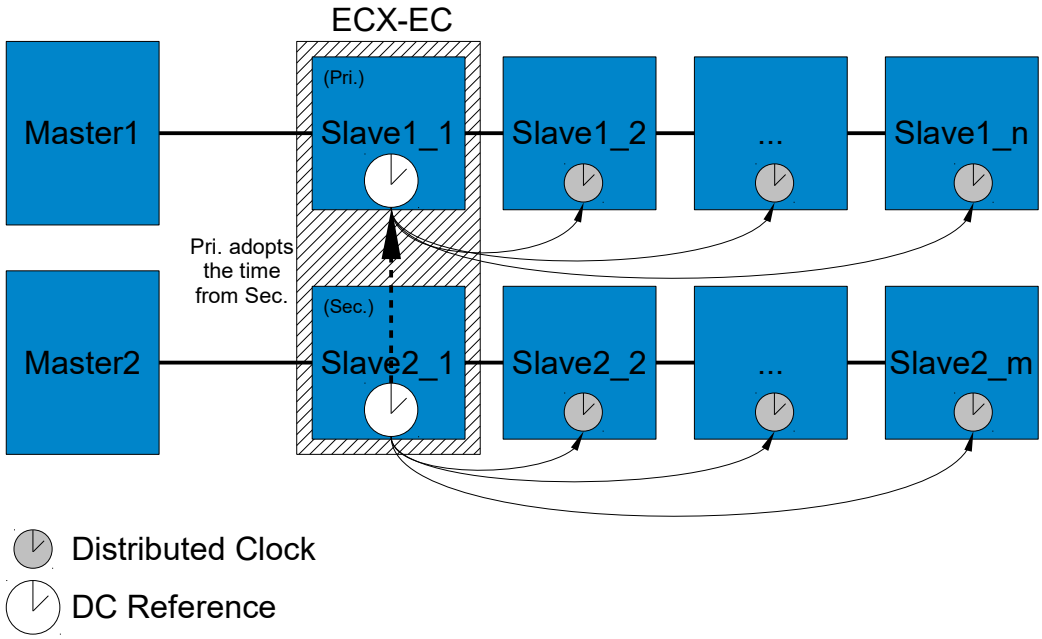



**Fig. 22:** Automatic DC synchronisation of two EtherCAT segments

Used in a redundant EtherCAT network this mode is enabled only when the redundancy case actually occurs. (After the Master also updated the DC Delays (register 0x0928) of the slaves in the redundant segment, etc.)

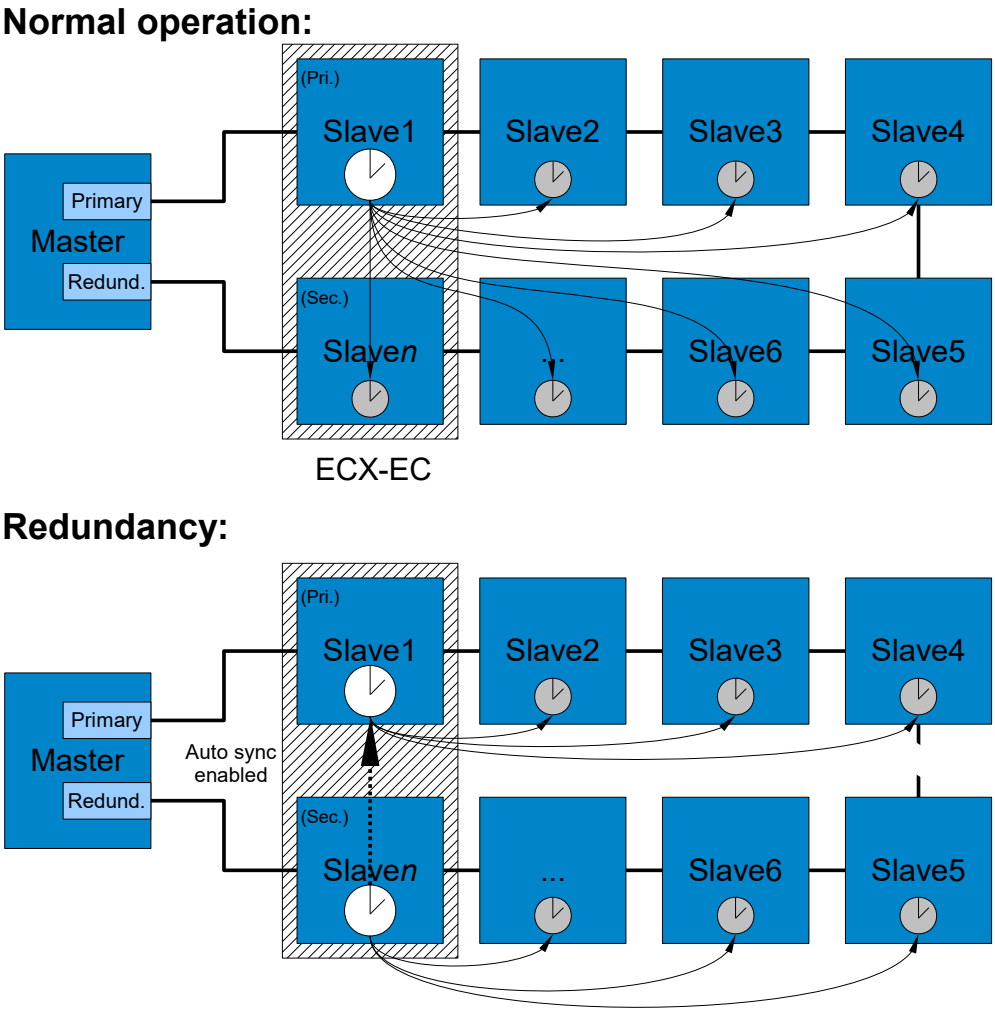



**Fig. 23:** Automatic DC synchronisation in redundancy case

**Important notes about automatic DC synchronisation:**

- Primary and secondary side can't be exchanged, it's always the primary side that adapts its time to the secondary side's time
- The primary side has to be the DC reference slave of its EtherCAT segment
- The EtherCAT master must accept the time from the primary side as it is: Usually each slave's time is adapted with the DC offset register (0x0920) by the master – automatic synchronisation is no longer possible then.

  If the master is not configurable accordingly the application must handle the automatic synchronisation and e.g. enable it only after the master has written the DC offset registers (which usually implies restarting the slave's DC sync units as well, etc.)
- The secondary side's time should not be changed uncontrolled – the time stamps and their "invalid" bit in the process data (object 0x10F4, see 6.3.1.6) should be used to react on the loss of synchronisation
- Typically an accuracy of about 100 ns can be reached

### 6.2.4.2 Standard Synchronisation

In the standard DC synchronisation the ECX-EC only serves to measure the difference between the DC times of both EtherCAT segments, i.e. the adaptation is done from an EtherCAT master.

#### 6.2.4.2.1 PDOs

The required time stamps are contained in Object 0x10F4 – “External Synchronization Status” (see chapter 6.3.1.6), for that PDO 0x1A02 – *SYNC slave (64 bit)* (6.2.5.5) is mapped.
If only 32-bit time stamps are used, PDO 0x1A03 is appropriate (see chapter 6.2.5.6). The other side can optionally use PDO 0x1A04 – *SYNC master* to get status information (see chapter 6.2.5.7).

#### 6.2.4.2.2 Synchronisation of two EtherCAT Segments

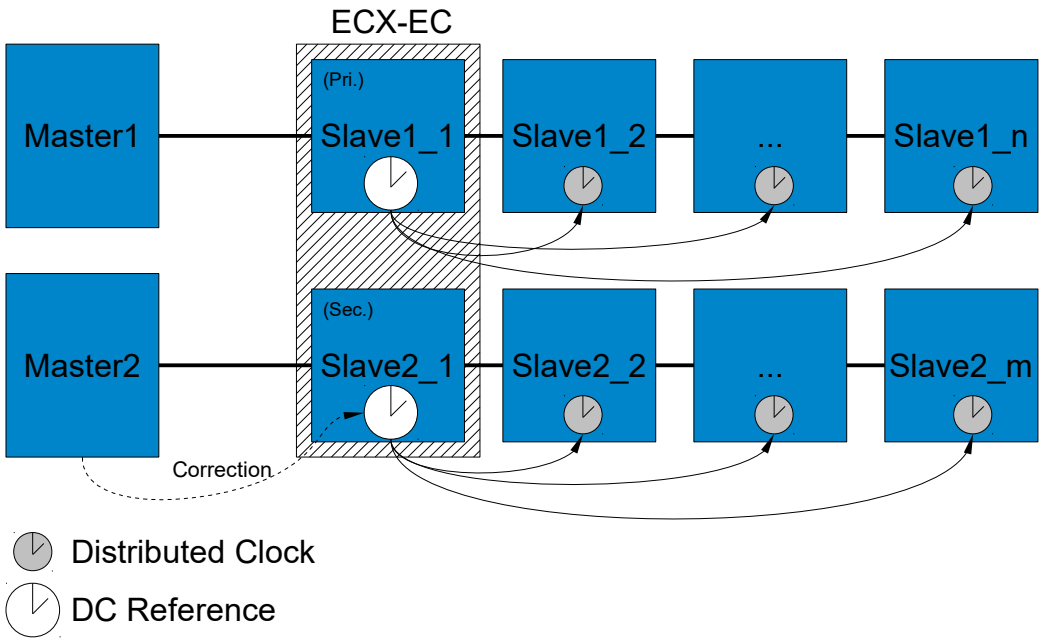



**Fig. 24:** DC synchronisation of two EtherCAT segments

Usually the time distribution in an EtherCAT segment is done by means of one slave whose time is distributed to all following slaves.
At the synchronisation of two segments this procedure is still being used.

Additionally the master of the ECX-EC secondary side adapts the time that is spread by this slave, so it matches the other side's time ("Correction" in Fig. 24).

This correction is done by writing the slave's *DC System Time* (register 0x0910), which provides for the fact that the clock frequency is adapted accordingly, when the timestamps show the clock runs too slow, a higher value is written, and vice versa.

**INFORMATION**
The described correction can be performed basically with every EtherCAT slave, the ECX-EC does not have to be the DC Reference slave and/or the first slave as shown.

#### 6.2.4.2.3 DC in Redundancy Case

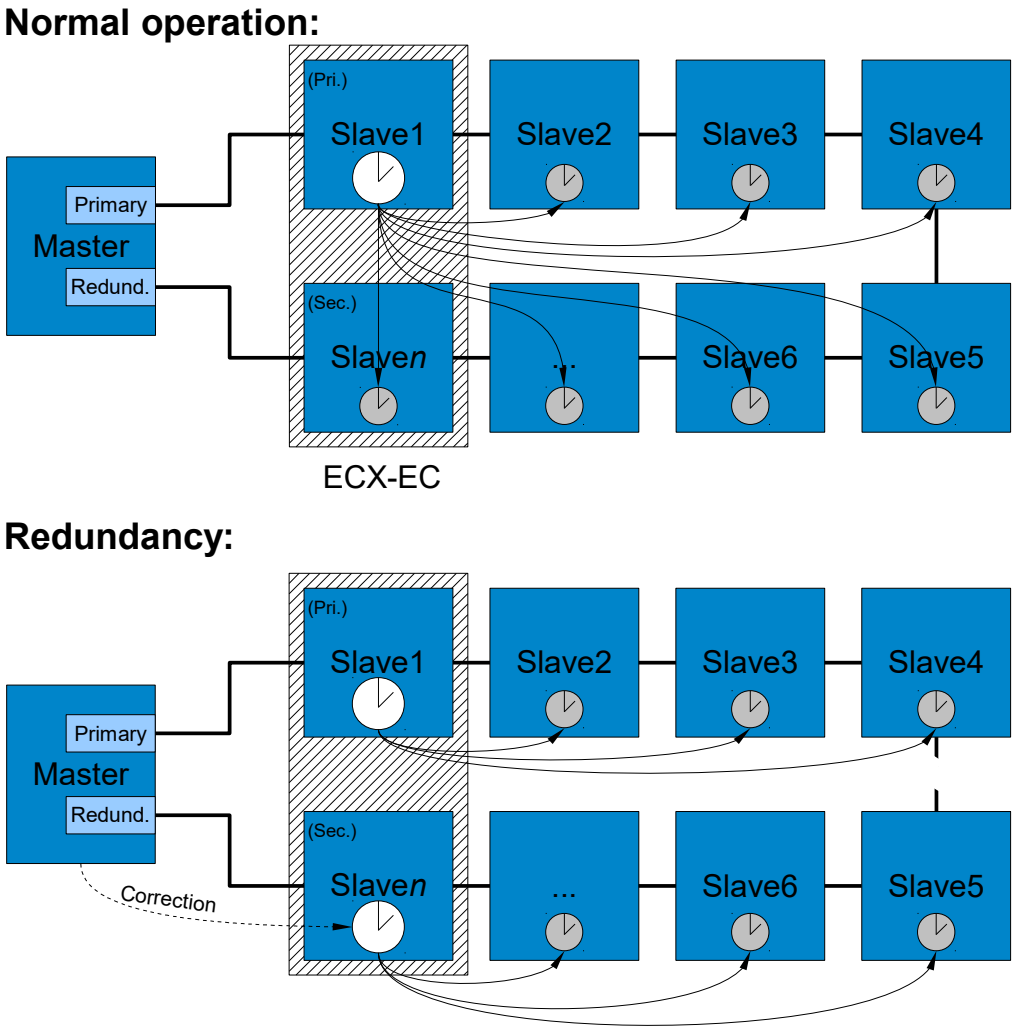



**Fig. 25:** DC synchronisation in redundancy case

In normal operation in this example the time of the first slave is distributed to all following slaves, as this is would be possible with every other DC slave too.
In case of redundancy there are only two segments and the last slave of the primary segment becomes the first slave of the new segment – its time can now be distributed to the other slaves.
The principle remains the same as in 6.2.4.2.2, but in this case the master additionally has to recalculate the *DC Delays* (register 0x0928) of the slaves in the redundant segment to avoid loss of accuracy.

#### 6.2.4.2.4 Sample (synchronization of two EtherCAT segments)

Base for this sample shall be a structure as shown in Fig. 24.

- ECX-EC Primary side
  - This side is the “Sync Master”, i.e. this time is the given standard: therefore no special steps are necessary and this EtherCAT segment can be configured as usual

  → With an EtherCAT configuration tool the DC mode is selected for the slaves and the ENI is exported for the master
- ECX-EC secondary side
  - This side is the “Sync Slave”, i.e. in this segment the time is adapted:
  - Create CoE init. commands for 0x10F5.01 and 0x10F5.11 (0 → ”Sync Slave” and “1” → to simplify control loop algorithm, see 6.3.1.7)
  - Activate the input PDO 0x1A02 (this contains the needed timestamps, see 6.2.5.5)

  → Select DC Modes for the slaves and export the ENI for the master
  - The master application can now use these process variables from the PDO 0x1A02:
    - “Time stamp update toggle”, “Time stamps invalid”, “Internal time stamp” und “External time stamp”
  - The “**invalid”** bit is observed at all times, when set, no loop control is possible and it must be assumed that the segments are **not synchronized**
  - At first a target difference is determined, this is the difference between “Internal time stamp” and “External time stamp” that shall be the goal of our loop control: called *TargetDiff* in this sample. This can be e.g. the difference when the timestamps became valid for the first time (the higher this difference, the longer takes the compensation – therefore “0” as target difference is unsuitable as both sides could be started with very different times)
  - Then, with every change of the toggle bit, the current difference is calculated: *CurrentDiff* = *ExternalTS - InternalTS*.
    This follows for the setpoint: $InternalTS_{SP}$ = *ExternalTS - TargetDiff*.
    - When the internal time stamp *InternalTS* is above its setpoint $InternalTS_{SP}$, the frequency must be reduced and vice versa.
      The change in clock frequency is part of each EtherCAT Slave Controller that supports DC: when the *DC System Time* register (0x0910) is written with a value above its current time, the frequency is increased, and if below, it’s reduced accordingly, see [4]
    - Now only the appropriate time has to be determined and written to the slave: on one hand classical loop control techniques could be used, but on the other hand its usually sufficient to write *InternalTS* to reduce the frequency and $InternalTS_{SP}$ + a few milliseconds to increase the frequency
  - If now the deviation between the actual internal time stamp value and its setpoint is within a specified tolerance limit for some time, the two EtherCAT segments are considered to be synchronized. (while these values are naturally defined according to your requirements)

## 6.2.5 PDOs

### 6.2.5.1 0x1600 – “Outputs”

Output variables. This PDO must be assigned (SM2) and is writeable, see 6.2.3 for details.
Content:

| Index | Sub index | Length (Type) | Description |
|---|---|---|---|
| 0x7000 | 1 | variable | Process variable, see 6.2.3 |
| 0x7000 | 2 | | |
| 0x7000 | ⋮ | | |
| 0x7000 | *n* | | |

### 6.2.5.2 0x1605 – “Outputs PDO Parameter”

PDO parameter (see also [2]) for PDO 0x1600. This PDO is optional.
Content:

| Index | Sub index | Length (Type) | Description |
|---|---|---|---|
| 0x1400 | 0x09 | 1 (BOOL) | *RxPDO Toggle*, see 6.3.1.9 |
| 0x0000 | 0x00 | 7 | Alignment |
| 0x1400 | 0x08 | 1 (BOOL) | *RxPDO Control*, see 6.3.1.9 |
| 0x0000 | 0x00 | 7 | Alignment |

### 6.2.5.3 0x1A00 – “Inputs”

Input variables. This PDO must be assigned (SM3) and is writeable, see 6.2.3 for details.
Content:

| Index | Sub index | Length (Type) | Description |
|---|---|---|---|
| 0x6000 | 1 | variable | Process variable, see 6.2.3 |
| 0x6000 | 2 | | |
| 0x6000 | ⋮ | | |
| 0x6000 | *n* | | |

#### 6.2.5.4 0x1A01 – “SYNC Slave (status only)”

This PDO is optional, a synchronous usage of PDO 0x1A02/0x1A03/0x1A04 is excluded.
Used for DC synchronisation, see 6.2.4.2.2.
Content:

| Index | Sub index | Length (Type) | Description |
|---|---|---|---|
| 0x10F4 | 0x01 | 2 (BIT2) | *SYNC mode*, see 6.3.1.6 |
| 0x0000 | 0x00 | 6 | Alignment |
| 0x0000 | 0x00 | 5 | Alignment |
| 0x10F4 | 0x0E | 1 (BOOL) | *Control value update toggle*, see 6.3.1.6 |
| 0x10F4 | 0x0F | 1 (BOOL) | *Time stamp update toggle*, see 6.3.1.6 |
| 0x10F4 | 0x10 | 1 (BOOL) | *Time stamps invalid*, see 6.3.1.6 |

#### 6.2.5.5 0x1A02 – “SYNC Slave (64 bit)”

This PDO is optional, a synchronous usage of PDO 0x1A01/0x1A03/0x1A04 is excluded.
Used for DC synchronisation, see 6.2.4.2.2.
Content:

| Index | Sub index | Length (Type) | Description |
|---|---|---|---|
| 0x10F4 | 0x01 | 2 (BIT2) | *SYNC mode*, see 6.3.1.6 |
| 0x0000 | 0x00 | 6 | Alignment |
| 0x0000 | 0x00 | 5 | Alignment |
| 0x10F4 | 0x0E | 1 (BOOL) | *Control value update toggle*, see 6.3.1.6 |
| 0x10F4 | 0x0F | 1 (BOOL) | *Time stamp update toggle*, see 6.3.1.6 |
| 0x10F4 | 0x10 | 1 (BOOL) | *Time stamps invalid*, see 6.3.1.6 |
| 0x10F4 | 0x11 | 64 (ULINT) | *Internal time stamp*, see 6.3.1.6 |
| 0x10F4 | 0x12 | 64 (ULINT) | *External time stamp*, see 6.3.1.6 |
| 0x10F4 | 0x13 | 32 (DINT) | *Time control value*, see 6.3.1.6 |

### 6.2.5.6 0x1A03 – “SYNC Slave (32 bit)”

As object 0x1A02, but with 32-bit time stamps. This PDO is optional, a synchronous usage of PDO 0x1A01/0x1A02/0x1A04 is excluded. Used for DC synchronisation, see 6.2.4.2.2.

Content:

| Index | Sub index | Length (Type) | Description |
|---|---|---|---|
| 0x10F4 | 0x01 | 2 (BIT2) | *SYNC mode*, see 6.3.1.6 |
| 0x0000 | 0x00 | 6 | Alignment |
| 0x0000 | 0x00 | 5 | Alignment |
| 0x10F4 | 0x0E | 1 (BOOL) | *Control value update toggle*, see 6.3.1.6 |
| 0x10F4 | 0x0F | 1 (BOOL) | *Time stamp update toggle*, see 6.3.1.6 |
| 0x10F4 | 0x10 | 1 (BOOL) | *Time stamps invalid*, see 6.3.1.6 |
| 0x10F4 | 0x11 | 32 (UDINT) | *Internal time stamp*, see 6.3.1.6 |
| 0x10F4 | 0x12 | 32 (UDINT) | *External time stamp*, see 6.3.1.6 |
| 0x10F4 | 0x13 | 32 (DINT) | *Time control value*, see 6.3.1.6 |

### 6.2.5.7 0x1A04 – “SYNC Master”

This PDO is optional, a synchronous usage of PDO 0x1A01/0x1A02/0x1A03 is excluded.
Used for DC synchronisation, see 6.2.4.2.2.

Content:

| Index | Sub index | Length (Type) | Description |
|---|---|---|---|
| 0x10F4 | 0x01 | 2 (BIT2) | *SYNC mode*, see 6.3.1.6 |
| 0x0000 | 0x00 | 6 | Alignment |
| 0x0000 | 0x00 | 7 | Alignment |
| 0x10F4 | 0x10 | 1 (BOOL) | *Time stamps invalid*, see 6.3.1.6 |

### 6.2.5.8 0x1A05 – “Inputs PDO Parameter”

PDO parameter (see also [2]) for PDO 0x1A00. This PDO is optional and its content is constant.

Content:

| Index | Sub index | Length (Type) | Description |
|---|---|---|---|
| 0x1800 | 0x09 | 1 (BOOL) | *TxPDO Toggle*, see 6.3.1.12 |
| 0x0000 | 0x00 | 7 | Alignment |
| 0x1800 | 0x07 | 1 (BOOL) | *TxPDO State*, see 6.3.1.12 |
| 0x1400 | 0x07 | 1 (BOOL) | *RxPDO State*, see 6.3.1.9 |
| 0x0000 | 0x00 | 6 | Alignment |

## 6.2.6 UDP Gateway

This is a special mode of the ECX-EC in which the EtherCAT secondary (EC-B) is left unused. Instead of that the UDP remote station (ETH) acts as a primary, i.e. the input data of the primary (EC-A) are received by the ECX-EC by UDP and the output data are accordingly transmitted by UDP.
The definition of the variables and the process image is executed as described in chapter 6.2.3, i.e. the variables are defined by the PDOs written from the EtherCAT side.

### 6.2.6.1 Configuration

Basically it is sufficient to activate the UDP gateway mode in CoE-Objekt 0x2030.01: The ECX-EC immediately starts to transmit, respectively receives the process data using the default settings.
EoE and all PDOs except 0x1600 - “Outputs” and 0x1A00 - “Inputs” are not supported in UDP gateway mode.
At least the cycle time for process data transmission has to be defined. In the default setting the transmission happens less than 10 ms each (Object 0x2030.0b). Additionally many objects for diagnostic purposed are available.
The configuration object 0x2030 “UDP gateway” and its default settings are described in chapter 6.3.2.5 (see page 60).

**IMPORTANT**

**Important Notes on UDP Gateway Mode**

Some restrictions appear on the described basic ECX-EC functions, because in UDP gateway mode there are not two EtherCAT segments be connected.

- Only the PDOs 0x1600 (outputs) and 0x1a00 (inputs) are supported.
- All functions regarding EtherCAT distributed clocks (inclusive all entries in Object 0x2000 – “Other Settings”) are no longer supported.
- EoE is not supported
- The automatic check of variable identity at both sides is not supported. This check is in the user's responsibility.

Additionally the meaning of the LED B display (see 2.2) is changed:

| | | | |
|---|---|---|---|
| RUN | green | on | UDP remote station is connected (see 6.3.2.5, object 0x2030.0d) |
| | | off | UDP remote station is not connected |
| | | blinking | process data received from UDP remote station |
| ERR | red | on | UDP remote station is not connected |
| | | off | UDP remote station is connected |

**INFORMATION**
The quick start of the UDP-Gateway with the UDDC (Universal Device Description Composer) is described in the following chapter.

### 6.2.6.2 Configuration with UDDC - Universal Device Description Composer

For quick configuration the esd software **UDDC** (“Universal Device Description Composer”) can be used to generate a complete EtherCAT ESI file that already includes the variables. UDDC additionally generates a C header file that might be helpful for the configuration of the UDP remote station.
This minimizes the configuration effort at the EtherCAT side for the file import into the EtherCAT configuration tool and, if applicable generating the CoE initializing commands for the CoE object 0x2030 – “UDP Gateway”.

**INFORMATION**
The esd software tool **UDDC** (“Universal Device Description Composer”) is included on the enclosed ECX-EC-CD.
For further information read the UDDC Software Manual on the ECX-EC-CD.

**UDP Gateway Quickstart with UDDC:**

1. Start new UDDC project for “EtherCAT UDP Gateway”.
2. Create the respectively required **output** variables for both sides.
   (Outputs are the variables that can be written.)
3. Copy the respective output variables to the respective input side
   (e.g. by click to arrow between the sides).
4. Export device description.

   → Now a matching “.xml”-ESI file and a “.h” file are available
5. Import the .xml ESI file into the EtherCAT Master/-Configurator and write it in the ECX-EC as EEPROM ESI (primary site).
6. In order to bring the ECX-EC to “operational” state, start the EtherCAT Master with the ECX-EC.

   → Now the UDP-Gateway is active, transmits UDP frames and is ready to process received UDP frames.
7. The “.h” file is ready for use with the Windows example UDP remote station – for other systems a special UDP remote station has to be developed according to the protocol description.

### 6.2.6.3 Protocol Overview

In both directions the communication is done via so called ”commands”, packed in UDP frames with a defined header. For more details see chapter 6.2.6.4.
Each time new output data are received for the EtherCAT ECX-EC primary site a command with these output data is transmitted via UDP (if this does not contradict the configured minimum interval time). With each receipt command with process data the data will be immediately copied into the process image of the EtherCAT slave.
If the ECX-EC is not in the EtherCAT state operational, i.e. it does not receive data from the EtherCAT site, it will return a command with information about the EtherCAT state.
As typical for UDP, lost frames are not repeated. Additionally there is no authentication, i.e. practically each UDP frame at the according port is evaluated. However a 4-byte “magic number” is helping to aviod possible inadvertent evaluation of other UDP protocols. A counter in the frame header is used to recognize lost frames.
Additionally UDP broadcasts are usually used. (In principle this can be configured, see notes to the the according objects in chapter 6.3.2.5).

### 6.2.6.4 UDP Protocol Details

**UDP Data Structure**

| 8 | 16 | 24 | 32 Bit |
|---|---|---|---|
| Magic | | | |
| Index | | Last received index | |
| Reserved | | Id | Last received Id |
| *1..n commands* | | | |

- Magic:
  - Fix value – frames with deviating magic have to be ignored (value is 0x50445545 in default setting, see 6.3.2.5)
- Index:
  - Index has to be incremented after each transmitted frame.
- Last received index:
  - Value of the field index of the last received frame.
- Reserved:
  - currently 0x0000 - Frames where this value is unequal 0 have to be ignored.
- Id:
  - Arbitrary value, which has to be stored by the receiver if it is unequal 0.
- Last received Id:
  - Stored Id value, i.e. Id of the last received frame with Id unequal 0.
- Commands:
  - At least one command must be included, structure as follows:

The frame index is used to detect missing frames or frames received in wrong order. Additionally the field “Id” can be evaluated for reception acknowledge (i.e. a frame with Id unequal 0 will be repeated until a frame with this value is received in the field “Last received Id”).

### Command Structure

| 8 | 16 | 24 | 32 Bit |
|---|---|---|---|
| Type | | Length | |
| *0..n command data bytes* | | | |

- Type
  - See following table “Commands”.
    Unknown commands have to be ignored.
- Length
  - Data length of the commands in byte (without header, i.e. starting with 0).
    If the number of bytes is lower than specified, the command has to be skipped. If the number of bytes is higher than specified only the exceeding bytes have to be ignored.

The maximum number/length of the commands is defined by the maximum length of an unfragmented UDP frame (Fragmented UDP frames are not supported by the ECX-EC. Usually this are 1400 bytes in a LAN.)

**INFORMATION**

**All data are transmitted in “Little Endian” format.**

### Commands

| Name (*UDPGW_CMD_...*) | Value | Data | Description |
|---|---|---|---|
| *PROCESSDATA* | 0x4450 | 8 byte timestamp +<br>0..n byte process data | Cyclically transmitted as described in chapter 6.2.6.3.<br><br>All timestamps in nano seconds<br>(But without defined reference, i.e. only relative time measurement.<br>The ECX-EC does not evaluate received timestamps.) |
| *ECATSTATEINFO* | 0x4953 | 1 byte EtherCAT state +<br>2 byte EtherCAT error code | Cyclically transmitted as described in chapter 6.2.6.3, if ECX-EC is not in “operational” state.<br><br>States/error codes according to [4] (ESC register 0x0130 respectively 0x0134). |
| *ECATSTATECHANGE* | 0x4353 | 1 byte EtherCAT state +<br>2 byte EtherCAT error code | Instructs the ECX-EC to change its EtherCAT state because of an error. Only a change to a lower state is possible (usually 2 for “PreOp” or 4 for “SafeOp”).<br><br>States/error codes according to [4] (ESC register 0x0130 respectively 0x0134). |
| *PING* | 0x4950 | 8 byte timestamp | Instructs the ECX-EC to send a *PONG* command. |

| Name (*UDPGW_CMD_...*) | Value | Data | Description |
|---|---|---|---|
| *PONG* | 0x4f50 | 8 byte timestamp +<br>8 byte timestamp of according *PING* | Transmitted as answer to a *PING* – can be used to estimate the latency |
| *SENDLOSTCNT* | 0x4c53 | none | Instructs the ECX-EC to send a *LOSTCNT* command |
| *LOSTCNT* | 0x434c | 4 byte lost counter +<br>4 byte number of lost counter changes<br>4 byte number OoO frames | Transmitted as answer to a *SENDLOSTCNT* – can be used to determine how many/how often UDP frames get lost<br><br>OoO: See “RX out of order“ in 6.3.2.5 |
| *RESETCOUNTER* | 0x4352 | none | Instructs the ECX-EC to reset the counter in objekt 0x2030 (SubIdx 0x10..0x30).<br><br>(This also includes the index for the next frame to be transmitted – which must taken into account at the determination of the lost frames. The own Tx counter not necessarily has to be reset: the first received frame after a reset is treated separately by the ECX-EC – this not necessarily causes to an “Rx Loast Event”.) |

## 6.3 CoE Object Directory

All subindices not listed do not exist, an access to them leads to the according error.

### 6.3.1 Standard Objects (0x1000..0x1FFF)

#### 6.3.1.1 Object 0x1000 – "Device Type"

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *Device Type* | UDINT | Device type of the EtherCAT slave<br>The LoWord contains the CoE profile used (5000: Modular device). The HiWord contains the module profile (0: Manufacturer specific) |

#### 6.3.1.2 Object 0x1008 – "Device Name"

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *Device Name* | STRING | Device name ("ECX-EC Pri" or "ECX-EC Sec") |

#### 6.3.1.3 Object 0x1009 – "Hardware Version"

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *Hardware Version* | STRING | Version number of the used hardware (not available in the first version of the firmware) |

#### 6.3.1.4 Object 0x100A – "Software Version"

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *Software Version* | STRING | Version number of the firmware (e.g. "1.0.0") |

#### 6.3.1.5 Object 0x1018 – "Identity"

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices |
| 0x01 | *Vendor Id* | UDINT | Manufacturer ID for identification of the slave (23 / 0x17) |
| 0x02 | *Product Code* | UDINT | Product number for identification of the slave (7) |
| 0x03 | *Revision Number* | UDINT | Revision number |
| 0x04 | *Serial Number* | UDINT | Serial number |

### 6.3.1.6 Object 0x10F4 – “External Synchronization Status”

Used at DC synchronisation, see also 6.2.4.2.2.

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x13) |
| 0x01 | *Sync mode* | BIT2 | 0: no synchronisation,<br>1: slave is SYNC master,<br>2: slave is SYNC slave<br>defined via 0x10F5.01, see 6.3.1.7 |
| 0x0E | *Control value update toggle* | BOOL | Not used, always 0 |
| 0x0F | *Time stamp update toggle* | BOOL | Will be toggled with every change of sub index 0x11/0x12 |
| 0x10 | *Time stamps invalid* | BOOL | 1: values in sub index 0x11/0x12 are invalid |
| 0x11 | *Internal time stamp* | ULINT | Own time stamp |
| 0x12 | *External time stamp* | ULINT | Time stamp of the other ECX-EC side |
| 0x13 | *Time control value* | DINT | Not used, always 0 |

### 6.3.1.7 Object 0x10F5 – “Synchronization Settings”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x12) |
| 0x01 | *Sync master* | BOOL | 1: slave is SYNC master,<br>0: slave is SYNC slave<br>Standard: 1, i.e. every side is master / no synchronisation.<br>(This setting is not needed by ECX-EC itself, but if both sides signalize to be SYNC Slave, the change into “SafeOp/Op” is denied) |
| 0x02 | *32 bit time stamps* | BOOL | 1: only 32 bits of the time stamp are used (The ECX-EC does not need/check this setting) |
| 0x11 | *Time control interval (ms)* | UINT | Defines how often the time stamps in 0x10F4 are updated.<br>Can be written from both sides of the ECX-EC, but refers to the same value.<br>Limited to [1...1000] ms. |
| 0x12 | *System time additional offset* | ULINT | If the Sync slave’s target difference between both clocks is not 0 it may store that value here as info for the other side. (The ECX-EC itself does not use this value) |

### 6.3.1.8 Object 0x10FF – “Bridge Variables Configuration”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices |
| 0x00..0xFF | *Bridge Variable Entry* | OCTET_STRING | See below |

This object can be written on the primary side to transmit further information to the created variables. As described in chapter 6.2.3 the definition of the process data is done only via the entries in the PDOs: only the length of a variable is defined, no name or type etc.

If the configuration of the variables shall e.g. be done only on one side and be transferred to the other side, this object can be used to define a detailed configuration. (E.g. the esd EtherCAT Workbench uses this object on the primary side via *Download variable Configuration* and for the variables on the secondary side via *Load from device* – see tab *Device specific* in the Workbench)

The ECX-EC does not currently use this object itself.

Every sub index describes exactly one variable. The OCTET_STRING is structured as follows:

| Byte | Description |
|---|---|
| 0 | Reserved (1 write) |
| 1 | Reserved (0 write) |
| [2, 3] | Index. 0x6000 for inputs (Pri.) and 0x7000 for outputs |
| 4 | Sub index |
| 5 | Reserved (0 write) |
| [6, 7] | Data type (see [1]) |
| [8, 9] | Length in bit |
| [10, 11] | Reserved (0 write) |
| [12, n] | Name (max. 256 characters, final 0 not necessary) |

#### 6.3.1.9 Object 0x1400 – "RxPDO 0x1600 Parameter"

See also Object 0x1800 – "TxPDO 0x1A00 Parameter".

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x09) |
| 0x07 | *RxPDO State* | BOOL | Is set by the slave if the output variables could not be copied to the opposite side. (Because this opposite side is not in "SafeOp/Op") |
| 0x08 | *RxPDO Control* | BOOL | Can be set by the master if the output data is not valid.<br>By default the ECX-EC ignores this value.<br>In object 0x2000 it can be defined to transfer this value to the counterpart *Input data invalid* (TxPDO state) of the opposite side, see chapter 6.3.2.1 |
| 0x09 | *RxPDO Toggle* | BOOL | Can be toggled by the master if the output data have changed.<br>By default the ECX-EC ignores this value.<br>In object 0x2000 it can be defined to use this value to optimise the copy process to the other side or to transfer this value to the counterpart *Input data changed* (TxPDO Toggle) of the opposite side, see chapter 6.3.2.1 |

#### 6.3.1.10 Object 0x1600 – "Outputs"

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices |
| 0x01..0xFF | *PDO Mapping* | PDO_MAPPING | Output variables, see chapter "PDOs" (6.2.5.1) |

#### 6.3.1.11 Object 0x1605 – "Outputs PDO Parameters"

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x04) |
| 0x01 | *PDO Mapping* | PDO_MAPPING | See chapter "PDOs" (6.2.5.2) |
| 0x02 | *PDO Mapping* | PDO_MAPPING | |
| 0x03 | *PDO Mapping* | PDO_MAPPING | |
| 0x04 | *PDO Mapping* | PDO_MAPPING | |

### 6.3.1.12 Object 0x1800 – “TxPDO 0x1A00 Parameter”

See also Object 0x1400 – “RxPDO 0x1600 Parameter”.

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x09) |
| 0x07 | *TxPDO State* | BOOL | Is set by the slave if the input data are invalid, i.e. the other side is not in “Operational”.<br>Additionally the value *RxPDO State* of the opposite side can be used, adjustable in object 0x2000, see 6.3.2.1 |
| 0x09 | *TxPDO Toggle* | BOOL | Toggled by the slave if the input image is rewritten, i.e. after new output data from EtherCAT have been received on the other side.<br>Alternatively the *RxPDO Toggle* value of the opposite side can be used, adjustable in object 0x2000, see 6.3.2.1 |

### 6.3.1.13 Object 0x1801 – “TxPDO 0x1A01 Parameter”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x06) |
| 0x06 | *Exclude TxPDOs* | OCTET_STRING | PDOs, that must not be simultaneously assigned with this:<br>0x1A02/0x1A03/0x1A04 |

### 6.3.1.14 Object 0x1802 – “TxPDO 0x1A02 Parameter”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x06) |
| 0x06 | *Exclude TxPDOs* | OCTET_STRING | PDOs, that must not be simultaneously assigned with this:<br>0x1A01/0x1A03/0x1A04 |

### 6.3.1.15 Object 0x1803 – “TxPDO 0x1A03 Parameter”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x06) |
| 0x06 | *Exclude TxPDOs* | OCTET_STRING | PDOs, that must not be simultaneously assigned with this:<br>0x1A01/0x1A02/0x1A04 |

#### 6.3.1.16 Object 0x1804 – “TxPDO 0x1A04 Parameter”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x06) |
| 0x06 | *Exclude TxPDOs* | OCTET_STRING | PDOs, that must not be simultaneously assigned with this: 0x1A01/0x1A02/0x1A03 |

#### 6.3.1.17 Object 0x1A00 – “Inputs”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices |
| 0x01..0xFF | *PDO Mapping* | PDO_MAPPING | Input variables, see chapter. “PDOs“ (6.2.5.3) |

#### 6.3.1.18 Object 0x1A01 – “SYNC Slave (Status Only)”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x06) |
| 0x01 | *PDO Mapping* | PDO_MAPPING | See chapter “PDOs” (6.2.5.4) |
| 0x02 | *PDO Mapping* | PDO_MAPPING | |
| 0x03 | *PDO Mapping* | PDO_MAPPING | |
| 0x04 | *PDO Mapping* | PDO_MAPPING | |
| 0x05 | *PDO Mapping* | PDO_MAPPING | |
| 0x06 | *PDO Mapping* | PDO_MAPPING | |

### 6.3.1.19 Object 0x1A02 – “SYNC Slave (64 bit)”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x09) |
| 0x01 | *PDO Mapping* | PDO_MAPPING | See chapter “PDOs” (6.2.5.5) |
| 0x02 | *PDO Mapping* | PDO_MAPPING | |
| 0x03 | *PDO Mapping* | PDO_MAPPING | |
| 0x04 | *PDO Mapping* | PDO_MAPPING | |
| 0x05 | *PDO Mapping* | PDO_MAPPING | |
| 0x06 | *PDO Mapping* | PDO_MAPPING | |
| 0x07 | *PDO Mapping* | PDO_MAPPING | |
| 0x08 | *PDO Mapping* | PDO_MAPPING | |
| 0x09 | *PDO Mapping* | PDO_MAPPING | |

### 6.3.1.20 Object 0x1A03 – “SYNC Slave (32 bit)”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x09) |
| 0x01 | *PDO Mapping* | PDO_MAPPING | See chapter “PDOs” (6.2.5.6) |
| 0x02 | *PDO Mapping* | PDO_MAPPING | |
| 0x03 | *PDO Mapping* | PDO_MAPPING | |
| 0x04 | *PDO Mapping* | PDO_MAPPING | |
| 0x05 | *PDO Mapping* | PDO_MAPPING | |
| 0x06 | *PDO Mapping* | PDO_MAPPING | |
| 0x07 | *PDO Mapping* | PDO_MAPPING | |
| 0x08 | *PDO Mapping* | PDO_MAPPING | |
| 0x09 | *PDO Mapping* | PDO_MAPPING | |

#### 6.3.1.21 Object 0x1A04 – “SYNC Master”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x04) |
| 0x01 | *PDO Mapping* | PDO_MAPPING | See chapter “PDOs” (6.2.5.7) |
| 0x02 | *PDO Mapping* | PDO_MAPPING | |
| 0x03 | *PDO Mapping* | PDO_MAPPING | |
| 0x04 | *PDO Mapping* | PDO_MAPPING | |

#### 6.3.1.22 Object 0x1A05 – “Inputs PDO Parameter”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x05) |
| 0x01 | *PDO Mapping* | PDO_MAPPING | See chapter “PDOs” (6.2.5.8) |
| 0x02 | *PDO Mapping* | PDO_MAPPING | |
| 0x03 | *PDO Mapping* | PDO_MAPPING | |
| 0x04 | *PDO Mapping* | PDO_MAPPING | |
| 0x05 | *PDO Mapping* | PDO_MAPPING | |

#### 6.3.1.23 Object 0x1C00 – “SM Types”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x04) |
| 0x01 | *SubIndex001* | ENUM | Type of SM0 (1/MBoxOut) |
| 0x02 | *SubIndex002* | ENUM | Type of SM1 (2/MBoxIn) |
| 0x03 | *SubIndex003* | ENUM | Type of SM2 (3/Outputs) |
| 0x04 | *SubIndex004* | ENUM | Type of SM3 (4/Inputs) |

### 6.3.1.24 Object 0x1C12 – “PDO Assignment”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices |
| 0x01..0xFF | *PDO Assignment* | UINT | SM2 assigned PDOs, see also 6.2.4.2.1 |

### 6.3.1.25 Object 0x1C13 – “PDO Assignment”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices |
| 0x01..0xFF | *PDO Assignment* | UINT | SM3 assigned PDOs, see also 6.2.4.2.1 |

## 6.3.2 Manufacturer-specific Objects (0x2000..0x5FFF)

### 6.3.2.1 Object 0x2000 – "Other Settings"

All this settings affect both sides, no matter from which side accessed, and remain valid until the ECX-EC is rebooted or other values are written.

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x07) |
| 0x01 | *Forward master PDO toggle bit* | BOOL | 1: The *RxPDO Toggle* bit (see 6.3.1.9) shall be copied to the *Tx PDO Toggle* bit (see 6.3.1.12) of the other side |
| 0x02 | *Optimize copy by master PDO toggle bit* | BOOL | 1: The copy of the output data to the input data of the other side shall be performed only if the *RxPDO Toggle* bit (see 6.3.1.9) signalizes a change of the output data.<br>If the masters on both sides can serve the toggle bit accordingly, a performance gain can be reached, otherwise this object must not be set |
| 0x03 | *Forward master PDO invalid bit* | BOOL | 1: The *RxPDO Control* bit (see 6.3.1.9) shall be copied into the *TxPDO State* bit (see 6.3.1.12) of the other side |
| 0x04 | *Ignore variable mismatch* | BOOL | 1: Different variable configuration/length on each side is allowed (All variables are internally a single contiguous block of memory: if of different length the last part just remains unused) |
| 0x05 | *DCTimes auto sync* | UDINT | Bit0: 1: "DC Auto Sync" activated, see 6.2.4.1.<br>All other Bits have to be written to 0 |
| 0x06 | *DCTimes auto sync target* | LINT | Target delta for DC Auto Sync: 0 means both clocks shall be synchronous, 100 for example means 100 ns difference between Sec. and Pri. clock.<br>Current difference can be read from 0x2001.02 or calculated by process data's internal/external timestamp values |
| 0x07 | *Zero inputs on other's Op fall back* | BOOL | All input variables are set to 0 when other side leaves Operational state |

### 6.3.2.2 Object 0x2001 – "Other Infos"

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x02) |
| 0x01 | *Other side's next SYNC0 (0x0990)* | ULINT | Value of the other side's ESC register 0x0990 (as little helper if both SYNC Units shall also be synchronized) |
| 0x02 | *DC times delta (Pri. minus Sec.)* | LINT | 0x10f4.11 minus 0x10f4.12<br>(Respectively 0x10f4.12 minus 0x10f4.11 for Sec. side) |

### 6.3.2.3 Object 0x2010 – “Statistics”

This object contains several statistics, usually it is only used for troubleshooting.

| Sub-idx | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x16) |
| 0x01 | *Reset* | UDINT | Every write access resets the statistics (as far as possible). Via a read access the time (see sub index 0x02) of the last reset is returned |
| 0x02 | *Time* | UDINT | Time in milliseconds since the ECX-EC has been booted |
| 0x03 | *Output events (Pri.)* | UDINT | Counts how often new output data were transferred from EtherCAT to the primary side |
| 0x04 | *Output events (Sec.)* | UDINT | Counts how often new output data were transferred from EtherCAT to the secondary side |
| 0x05 | *Output interval (Pri.)* | UDINT | Measures in which interval (µs) new output data are transferred from EtherCAT to the primary side (average, calculated after 1000 output events) |
| 0x06 | *Output interval (Sec.)* | UDINT | Measures in which interval (µs) new output data are transferred from EtherCAT to the secondary side (average, calculated after 1000 output events) |
| 0x07 | *Output jitter (Pri.)* | UDINT | Average deviation (µs) for *Output interval (Pri.)* |
| 0x08 | *Output jitter (Sec.)* | UDINT | Average deviation (µs) for *Output interval (Sec.)* |
| 0x09 | *Provision time* | UDINT | Measures the time (µs) in which the last new output data were transferred from the primary side to the secondary side.<br>Ideally this is approximately the time from which new output data are provided by the EtherCAT (Outputs SM Interrupt) and these are completely transferred to the input data (Inputs SM) of the other side – Normally for this procedure approximately the double *Provision time* can be assumed. (This assumption does no longer apply if the copy-operation is determined by the settings in object 0x2000 (see 6.3.2.1)) |
| 0x10 | *EoE Frames EtherCAT Rx* | UDINT | No. of Ethernet frames received from EtherCAT |
| 0x11 | *EoE Frames EtherCAT Tx* | UDINT | No. of Ethernet frames sent to EtherCAT |
| 0x12 | *EoE Frames EtherCAT Tx Error* | UDINT | No. of Ethernet frames that could not be sent to EtherCAT due to an error |
| 0x13 | *EoE Frames EtherCAT Tx Overrun* | UDINT | No. of Ethernet frames that could not be sent to EtherCAT due to Tx buffer overrun<br>(This kind of frame loss is likely to happen – higher level protocols on Ethernet side, such as TCP/IP, will handle this) |
| 0x14 | *EoE Frames Local Rx* | UDINT | No. of frames received on Ethernet interface |
| 0x15 | *EoE Frames Local Tx* | UDINT | No. of frames sent to Ethernet interface |

| Sub-idx | Name | Type | Description |
|---|---|---|---|
| 0x16 | *EoE Frames Local Tx Overrun* | UDINT | No. of frames that could not be sent to Ethernet interface due to Tx buffer overrun |

#### 6.3.2.4 Object 0x2020 – “Ethernet”

This object contains some data about the Ethernet interface, usually it is only used for troubleshooting.

| Sub-idx | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices (0x05) |
| 0x01 | *Link* | STRING | Link status, e.g. “100 Mbps, Full Duplex” or “None” |
| 0x02 | *Link lost count* | UDINT | Counts how often “Link” fell back to “None” |
| 0x03 | *Defective RX frames* | UDINT | No. of erroneous frames, e.g. because of CRC error |
| 0x04 | *Possible RX overruns* | UDINT | Counts how often new data arrived while the receive buffer was full |
| 0x05 | *TX frames dropped* | UDINT | No. of frames not sent |

### 6.3.2.5 Object 0x2030 – “UDP Gateway”

This objekt contains the data of the UDP gateway mode as described in chapter 6.2.6. It exists at the primary site only.

| Sub-idx | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. used subindex (0x30) |
| 0x01 | *Enabled* | UINT | 1: UDP gateway mode (see 6.2.6) active<br>Standard: 0 / off |
| 0x02 | *Destination MAC address* | OCTET_ STRING | Destination address of the UDP frames transmitted by the ECX-EC. Default is Ethernet broadcast, “ff ff ff ff ff ff” |
| 0x03 | *Destination IP address* | UDINT | Destination address of the UDP frames transmitted by the ECX-EC. Default is IP broadcast, “0x ff ff ff” (255.255.255.255).<br>If another destination is set the destination MAC address must match (no ARP). |
| 0x04 | *Destination port* | UINT | Destination port of the UDP frames transmitted by the ECX-EC. Standard is 0x4543 |
| 0x06 | *Own MAC address* | OCTET_ STRING | Own MAC address for the transmission of the UDP frames (can not be modified). |
| 0x07 | *Own IP address* | UDINT | Own IP address for the reception and the transmission of the UDP frames. Default: 0.<br>If set to 0 only the own MAC address is checked at reception, i.e. all IP addresses are accepted. (Broadcasts are always accepted in any cases.)<br>If set, 0 is used for transmission, too!<br>ARP is not supported, i.e. if an IP address is set, at the remote station a corresponding entry has to be written in the ARP table. |
| 0x08 | *Subnet mask* | UDINT | If the UDP remote station wants to send to the subnet broadcast address this command sets the subnet mask (requires the setting of “Own IP address”).<br>Default value: 0 / none |
| 0x09 | *Own port* | UINT | Own UDP port to which the UDP gateway frames have to be addresses.<br>Default value: 0x4543 |
| 0x0a | *Packet magic* | UDINT | Value for “Magic” field in the header for transmission and reception – received frames with different “Magic” will be rejected (see 6.2.6.4).<br>Default value: 0x50445545 |
| 0x0b | *UDP data min tx interval* | DINT | > 0: min. interval in µs, i.e. if new data are received from EtherCAT earlier, they will be ignored/no UDP command is generated.<br>< 0: interval in EtherCAT cyles, i.e. with “-1” for example an UDP command is generated with *every* EthertCAT cycle. With “-3” an UDP command is only generated with *every 3rd* cycle and so on.<br>(0: no UDP process data commands)<br>Default value: 10000 |

| Sub-idx | Name | Type | Description |
|---|---|---|---|
| 0x0c | *UDP not ready interval* | UDINT | Transmission interval in µs for *ECATSTATEINFO* commands, if the ECX-EC is not in EtherCAT state “Op”. (values limited to ≥ 1000, default value: 100000) |
| 0x0d | *Client timeout* | UINT | If no valid frame is received from the UDP remote station within this time, the connection is assumed to be interrupted.<br>0: no timeout; remote station will never be detected as interrupted (Attention! In this case it must be continued with the according Tx frame index after an interruption.)<br>(Default value: 3000. Result of interruption: LED signal) |
| 0x10 | *RX frames* | UDINT | Number of determined UDP frames (i.e. frames with correct Magic etc.) |
| 0x11 | *Last RX index* | UINT | Value of “Index” field of header of last received frame. |
| 0x12 | *Last RX id* | USINT | Value of “Id” field of header of last received frame. |
| 0x13 | *RX lost* | UDINT | Number of lost frames determined from “Index” field of last received frames. |
| 0x14 | *RX lost events* | UDINT | Number of changes of “RX lost”, i.e. how often an unexpected index was received. |
| 0x15 | *RX out of order* | UDINT | Number of “Out of Order” frames - frames that have been ignored because, with respect of the index, they have already been received.<br>(see example UDP remote station for calculation) |
| 0x16 | *RX dropped* | UDINT | Number of ignored frames (“Magic” or “Reserved” fields unvalid). |
| 0x20 | *TX frames* | UDINT | Number of transmitted frames. |
| 0x21 | *Next TX index* | UINT | “Index” field for the next frame. |
| 0x22 | *TX failed* | UDINT | Number of frames for which an error was detected at transmission. |
| 0x30 | *UDP data rx processed* | UDINT | Number of executed *PROCESSDATA* commands.<br>(A command is not executed if the ECX-EC is not in state “SafeOp”/”Op” or if the number of received data is too small.) |

## 6.3.3 Profile-specific Objects (0x6000..0xFFFF)

### 6.3.3.1 Object 0x6000 – “Inputs”

The entries of this object correspond to the created input variables, i.e. they are only generated if the PDO 0x1A00 configuration is downloaded into the ECX-EC.

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices |
| 0x01..0xFF | *Bridge Variable* | variable | Bridge variable |

Variables smaller than 8 bit are generated as bit-type, i.e. “BIT1” .. “BIT7”, all others as “OCTET_STRING” with their corresponding length.

### 6.3.3.2 Object 0x7000 – “Outputs”

The entries of this object correspond to the created output variables, i.e. they are only generated if the PDO 0x1600 configuration is downloaded into the ECX-EC.

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices |
| 0x01..0xFF | *Bridge Variable* | variable | Bridge variable |

See previous note on *Inputs*. Output variables can still be written in state “PreOp”. (These values will be overwritten by EtherCAT when switching into “SafeOp/Op”)

### 6.3.3.3 Object 0xF000 – “Modular Device Profile”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices |
| 0x01 | *Module index distance* | UINT | For compatibility with the Modular Device Profile |
| 0x02 | *Maximum number of modules* | UINT | |

### 6.3.3.4 Object 0xF008 – “Code Word”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *Code word* | UDINT | not used |

### 6.3.3.5 Object 0xF010 – “Module List”

| Sub index | Name | Type | Description |
|---|---|---|---|
| 0x00 | *SubIndex000* | USINT | Max. number of sub indices |
| 0x01 | *SubIndex001* | UDINT | For compatibility with the Modular Device Profile only |

## 6.4 FoE (Firmware-Update)

The firmware of the ECX-EC can be updated via FoE. This is only possible on the primary side, and this side has to be in the EtherCAT state “BootStrap”. The BootStrap state in turn is only possible if the secondary side is not in “SafeOp/Operational”.
After the update the ECX-EC has to be restarted (Disconnect the power supply voltage)

To update just the firmware file has to be transferred to the ECX-EC via FoE-Download. The FoE file name has to begin with “`firmwareUpdate`” and the FoE password is “0”.

**NOTICE**
During firmware update the power supply of the ECX-EC must not be cut off, because the module could get into an inoperable state.

**INFORMATION**
Make sure the latest/correct EEPROM ESI is also installed and updated if necessary.

### 6.4.1 Firmware-Update with esd EtherCAT Workbench

**INFORMATION**
The EtherCAT Workbench is contained on the enclosed ECX-EC-CD.

- Connect the primary side of the ECX-EC with the Workbench system and choose the button *Online* in the Online toolbar.
- Select the slave and then → tab *Mailbox* → and tab *FoE*

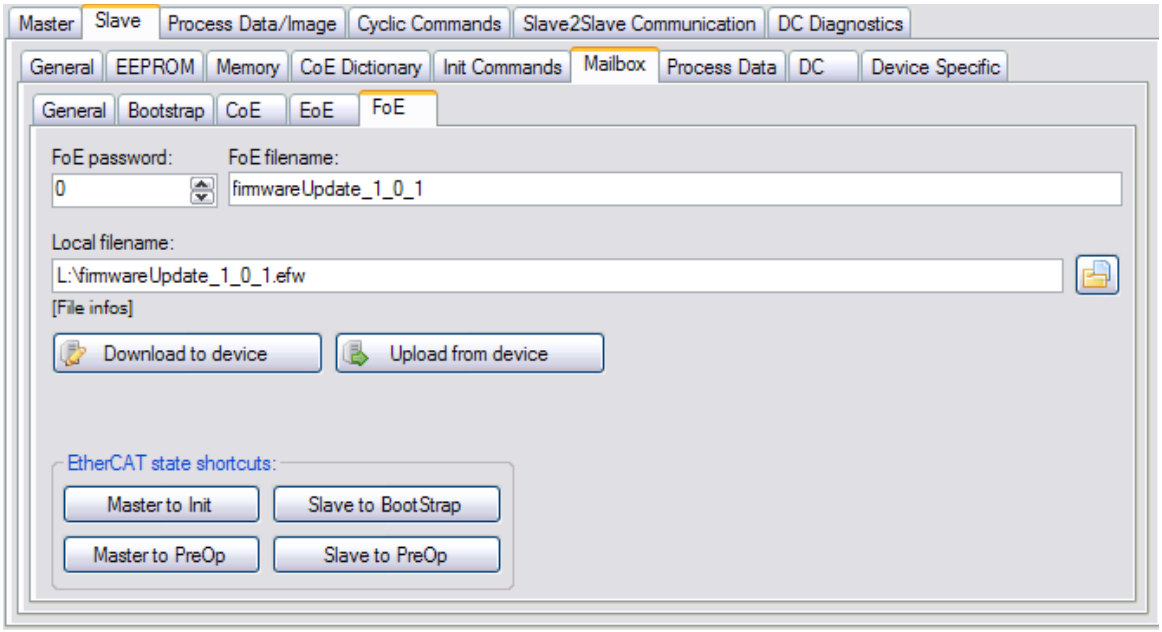



**Fig. 26:** Choose BootStrap-state

- Click the button *Slave to BootStrap* to switch into BootStrap-state.
  - Recognizable by the blue symbol at the slave. If the slave remains in “Init” (red symbol) either the secondary side is in “SafeOp/Op” state or the selected slave is the secondary side itself.

- Enter the firmware file-name (per -button or per manual input)
  - Do not change the *FoE password* (0). The *FoE filename* will be automatically generated according to the file name (this has to start with `firmwareUpdate` as mentioned above)

- Activate the button *Download to device* to start the transmission

- Wait until the file is transferred (see progress bar)

- The “Flash” LED (yellow) is on while the update of the ECX-EC is in progress.

- End of the update
  - The LED “Status” (green) flickers:
    - Update was successful. Disconnect the power supply voltage for approximately 5 seconds to restart the ECX-EC
  - The LED “EC-A Error” (red) flickers:
    - Update failed. Read the error code from the “AL Statuscode” register and contact our support.
  - In all other cases the transfer of the file to the ECX-EC has failed and the update has not been started.
    - Try to transfer the file again. If the transfer of the file fails again, the file might be defective. Contact the support (support@esd.eu)

## 6.4.2 Firmware-Update with Beckhoff TwinCAT/Configurator

- Connect the primary side of the ECX-EC with the TwinCAT system and choose *Scan Boxes* (see Fig. 12).

- Select the slave and click the button *Boostrap* on the tab *Online* in the field *State Machine.*

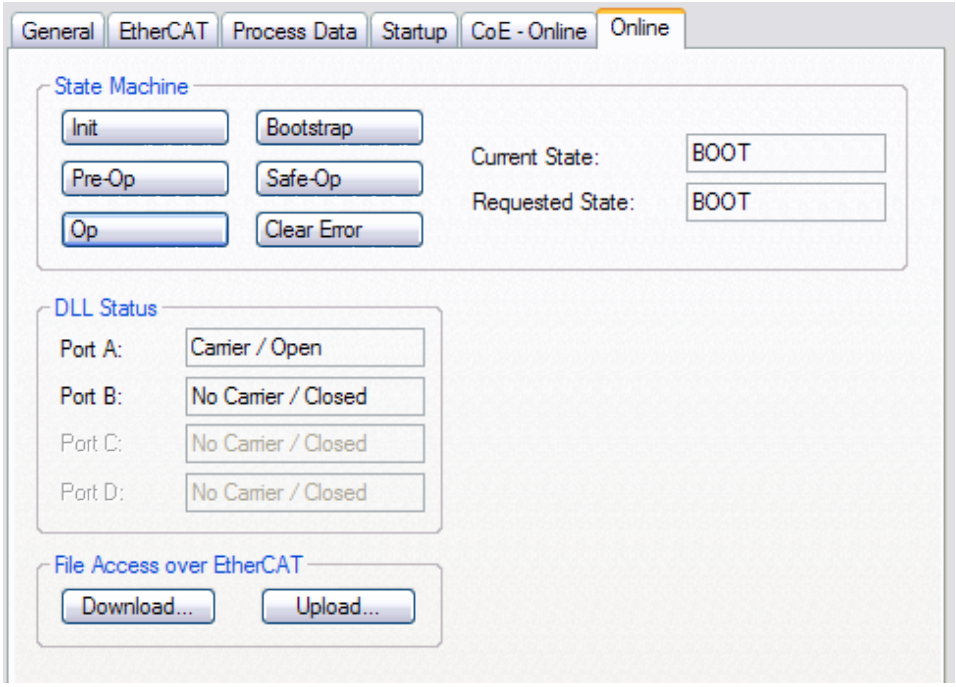



**Fig. 27:** Choose Bootstrap state via TwinCAT

  - (If the slave remains in “Init” state either the secondary side is in “SafeOp/Op” or the selected slave is the secondary side itself)

- Activate the button *Download* in the range *File access over EtherCAT*
- Select the file, e.g. `firmwareUpdate_1_0_1.efw`

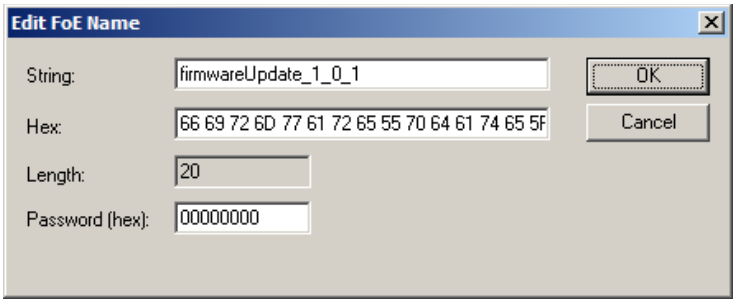



**Fig. 28:** Select firmware

- Confirm your selection with the *OK* button
- Wait until the file is transferred (see progress bar)
- The “Flash” LED (yellow) is on while the update of the ECX-EC is in progress.
- End of the update
  - LED “Status” (green) flickers:
    - Update was successful. Disconnect the power supply voltage for several seconds to restart the ECX-EC
  - LED “EC-A Error” (red) flickers:
    - Update failed. Read the error code from the “AL Statuscode” register and contact our support.
  - In all other cases the transfer of the file to the ECX-EC has failed and the update has not been started.

## 6.5 EoE (Switch Port)

The ECX-EC supports the EoE Mode “Switch Port”. In this mode all frames received on the Ethernet interface are forwarded to the EtherCAT master and vice versa.

For the Switch Port mode no configuration is necessary, it’s **enabled by default** (in “PreOp”/“SafeOp”/“Op” state) and requires only the according EtherCAT Master support.

For the EoE activities some statistics are recorded, see 6.3.2.3 (CoE object 0x2010).

**INFORMATION**
The EoE support is bound to the **primary side** slave (EC-A). Therefore the secondary side does not support EoE.

The EoE support may be deactivated: The ECX-EC reads the EoE enabled bit from its primary-side EEPROM (Category “General”, see [3]). If that is set to 0 (evaluated at first state change to “PreOp”) all EoE activities are stopped.

Fig. 29 shows this exemplary for the EtherCAT Workbench:

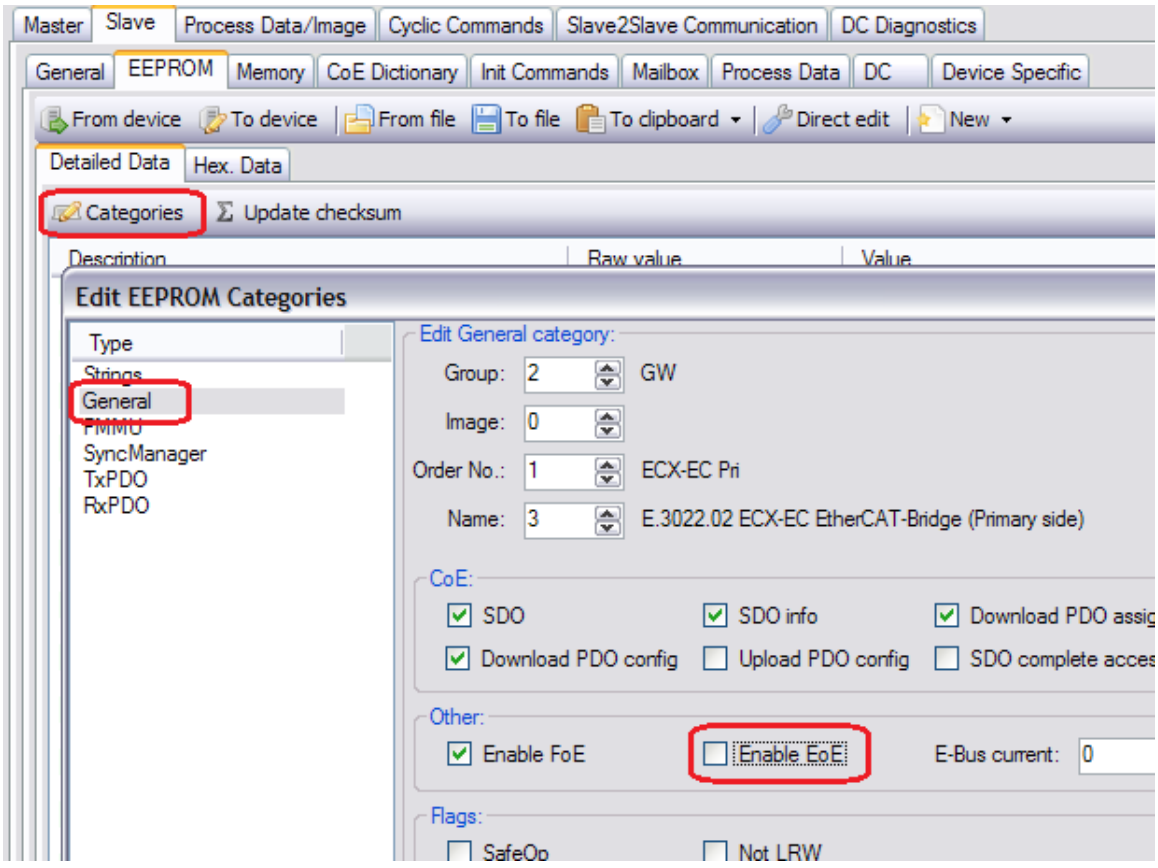


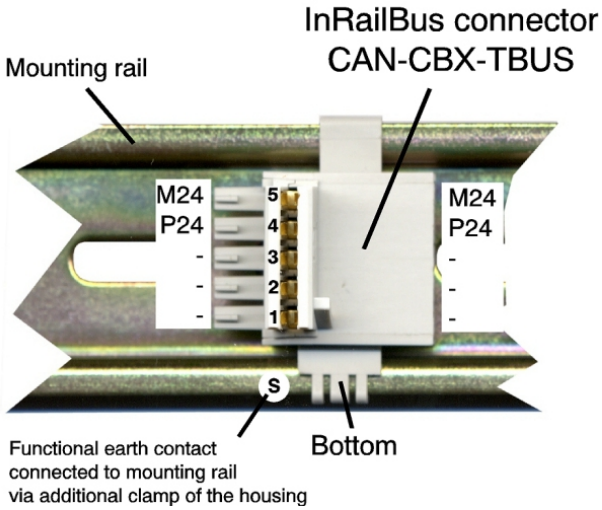

**Fig. 29:** EoE deactivation with the esd EtherCAT Workbench

# 7. Appendix InRailBus (Option)

## 7.1 Connector Assignment 24V via InRailBus (Option)

Connector type: Mounting-rail bus connector CAN-CBX-TBUS
Phoenix Contact ME 22,5 TBUS 1,5/5-ST-3,81 KMGY

**Connector View:**




**Pin Assignment:**

| Pin | Signal |
|---|---|
| 5 | M24 (GND) |
| 4 | P24 (+24 V) |
| 3 | Do not connect! |
| 2 | Do not connect! |
| 1 | Do not connect! |
| | |
| S | FE (PE_GND) |

**Signal Description:**

P24... power supply voltage +24 V
M24... reference potential
FE... functional earth contact (EMC) (connected to mounting rail potential)

## 7.2 Using InRailBus Connector

### 7.2.1 Connection of the Power Supply Voltage

The power supply voltage can be connected via the +24V connector for the power supply voltage or via InRailBus connector.

**NOTICE**
Please note the safety instructions to requirements on the supply current circuit (see page 4).

**NOTICE**
The connections for the 24V power supply are internally connected and must not be supplied by two independent current sources at once!

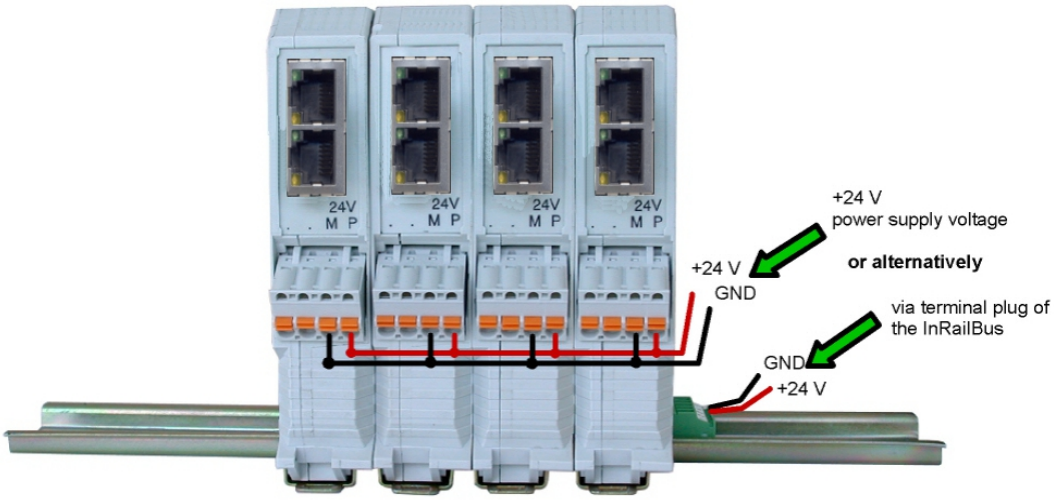



**Fig. 30:** ECX-EC with connection of the power supply voltage

#### Earthing the mounting rail

**NOTICE**
Connect the mounting rail to functional earth potential. Please note that the impedance of the connecting cable has to be kept low.

Functional earth is a current path of low impedance between current circuits and earth, which is used to increase the interference immunity. It is not intended as protective measure and does not protect against accidental contact.

## 7.2.2 Installation of the Module Using the InRailBus Connector

If the power supply voltage shall be fed via the InRailBus, please proceed as follows:

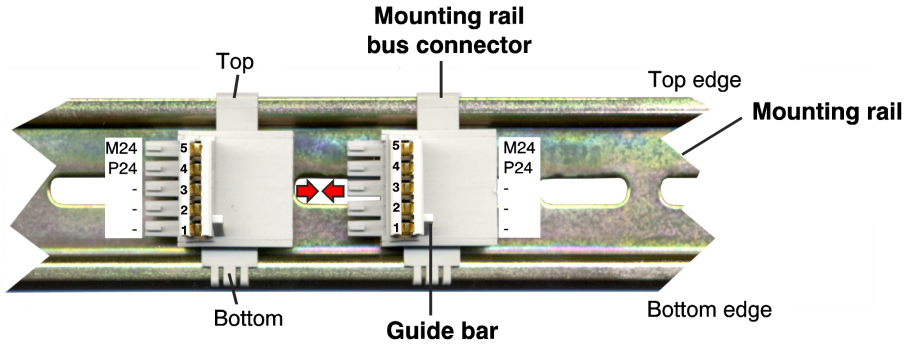



**Fig. 31:** Mounting rail with bus connector

**NOTICE**
The pins 1,2 and 3 of the mounting rail bus connector are reserved and must not be connected!

1. Position the mounting rail bus connector of the InRailBus (CAN-CBX-TBUS, order no.: C.3000.01) on the mounting rail and snap it onto the mounting rail using slight pressure. Plug the bus connectors together to contact the communication and power signals (in parallel with one). The bus connectors can be plugged together before or after mounting the CAN-CBX modules. One mounting rail bus connector is included in delivery.
   Functional earth must be connected to the mounting rail (see page 68).

2. Place the CAN-CBX module with the DIN rail guideway on the top edge of the mounting rail.

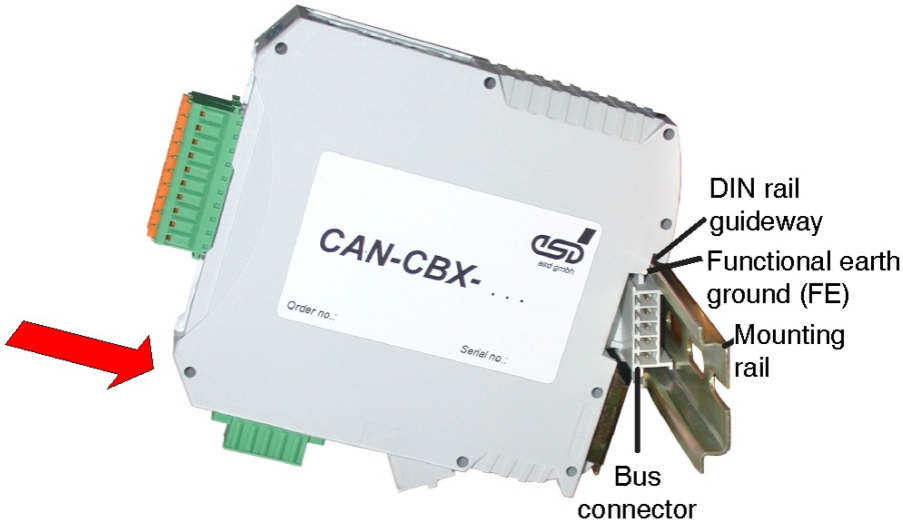



**Fig. 32:** Mounting CAN-CBX modules

3. Swivel the CAN-CBX module onto the mounting rail in pressing the module downwards according to the arrow as shown in Fig. 31. The housing is mechanically guided by the DIN rail bus connector.

4. When mounting the CAN-CBX module the metal foot catch snaps on the bottom edge of the mounting rail. Now the module is mounted on the mounting rail and connected to the InRailBus via the bus connector. Connect the bus connectors and the InRailBus, if not already done.

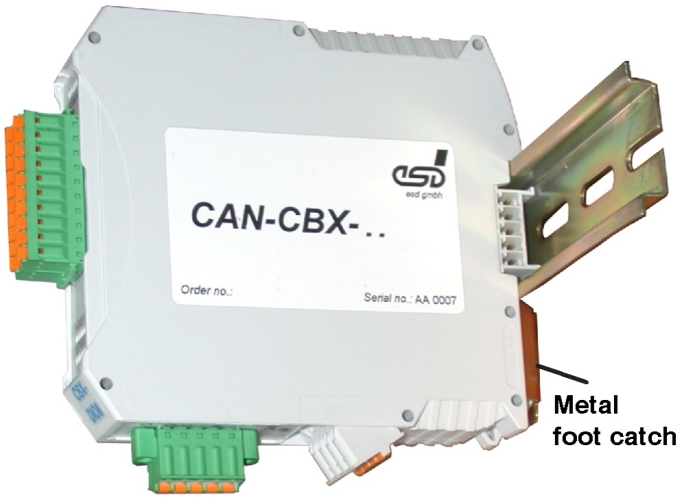



**Fig. 33:** Mounted CAN-CBX module

### 7.2.2.1 Connecting Power Supply via CBX-InRailBus

To connect the power supply and the CAN-signals via the InRailBus, a terminal plug (CAN-CBX-TBUS Connector, order no.: C.3000.02) is needed. The terminal plug is not included in delivery and must be ordered separately.

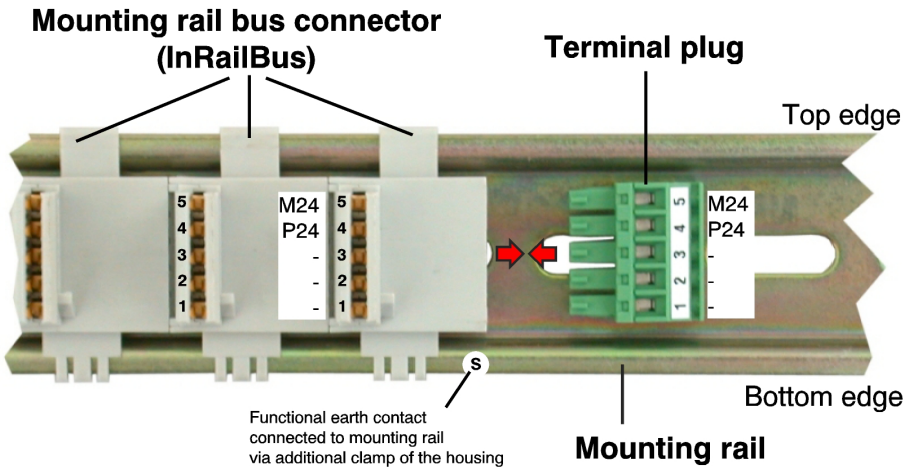



**Fig. 34:** Mounting rail with InRailBus and terminal plug

**NOTICE**
The pins 1,2 and 3 of the mounting rail bus connector are reserved. Do not connect!

Plug the terminal plug into the socket on the right of the mounting-rail bus connector of the InRailBus, as described in Fig. 30. Then connect the CAN interface and the power supply voltage via the terminal plug.
Please pay attention to the notes on the connection of the power supply voltage on page 68.

## 7.2.3 Remove the ECX-EC Module from InRailBus

If the ECX-EC module is connected to the InRailBus please proceed as follows:

Release the module from the mounting rail in moving the foot catch (see Fig. 8) downwards (e.g. with a screwdriver). Now the module is detached from the bottom edge of the mounting rail and can be removed.

**INFORMATION**
It is possible to remove individual devices from the whole without interrupting the InRailBus connection, because the contact chain will not be interrupted.

# 8. References

[1] EtherCAT Technology Group, EtherCAT Specification – Part 5 (ETG.1000.5), Version 1.0.2
[2] EtherCAT Technology Group, EtherCAT Protocol Enhancements (ETG.1020), Version 1.0.0
[3] EtherCAT Technology Group, EtherCAT Specification – Part 6 (ETG.1000.6), Version 1.0.2
[4] Beckhoff Automation GmbH, ET1100 Hardware Data Sheet, Version 1.8

# 9. Declaration of Conformity

## EU-KONFORMITÄTSERKLÄRUNG
## *EU DECLARATION OF CONFORMITY*

Adresse / *Address*
**esd electronics gmbh**
**Vahrenwalder Str. 207**
**30165 Hannover**
**Germany**

| | |
|---|---|
| esd erklärt, dass das Produkt<br>*esd declares, that the product* | Typ, Modell, Artikel-Nr.<br>*Type, Model, Article No.* |
| **ECX-EC** | **E.3022.02** |
| die Anforderungen der Normen<br>*fulfills the requirements of the standards* | **EN 61000-6-2:2005,**<br>**EN 61000-6-3:2007/A1:2011** |
| gemäß folgendem Prüfbericht erfüllt.<br>*according to test certificate.* | **H-K00-0504-13** |
| Das Produkt entspricht damit der EU-Richtlinie „EMV"<br>*Therefore the product conforms to the EU Directive 'EMC'* | **2014/30/EU** |
| Das Produkt entspricht den EU-Richtlinien „RoHS"<br>*The product conforms to the EU Directives 'RoHS'* | **2011/65/EU, 2015/863/EU** |

Diese Erklärung verliert ihre Gültigkeit, wenn das Produkt nicht den Herstellerunterlagen entsprechend eingesetzt und betrieben wird, oder das Produkt abweichend modifiziert wird.
*This declaration loses its validity if the product is not used or run according to the manufacturer's documentation or if non-compliant modifications are made.*

Name / *Name* T. Bielert
Funktion / *Title* QM-Beauftragter / *QM Representative*
Datum / *Date* Hannover, 2019-06-27

Rechtsgültige Unterschrift / *authorized signature*

I:\Texte\Doku\MANUALS\ECX\ECX-EC\Konformitaetserklaerung\ECX-EC_Konformitaetserklaerung_2019-06-27.odt

# 10. Order Information

| Type | Properties | Order No. |
|---|---|---|
| ECX-EC | EtherCAT Bridge | E.3022.02 |
| **Accessories for usage with InRailBus** | | |
|  **CAN-CBX-TBUS** | Mounting-rail bus connector of the CBX-InRailBus for CAN-CBX modules | C.3000.01 |
|  **CAN-CBX-TBUS-Connector** | Terminal plug of the CBX-InRailBus for the connection of the +24V power supply voltage and the CAN interface, Female type | C.3000.02 |
|  **CAN-CBX-TBUS-Connection adapter** | Terminal plug of the CBX-InRailBus for the connection of the +24V power supply voltage and the CAN-Interface, Male type | C.3000.03 |

**Table 10:** Order information

**PDF Manuals**
Manuals are available in English and usually in German as well. For availability of English manuals see table below.
Please download the manuals as PDF documents from our esd website www.esd.eu for free.

| Manuals | | Order No. |
|---|---|---|
| ECX-EC-ME | Hardware manual in English | E.3022.21 |

**Table 11:** Available manuals

**Printed Manuals**
If you need a printout of the manual additionally, please contact our sales team: sales@esd.eu for a quotation. Printed manuals may be ordered for a fee.