

Documentation | EN

# EL2574

4 Channel LED Output Terminal, Pixel LED





# Table of contents

<b>1 Foreword</b>	<b>5</b>
1.1 Notes on the documentation	5
1.2 Guide through documentation	6
1.3 Safety instructions	7
1.4 Documentation issue status	8
1.5 Version identification of EtherCAT devices	9
1.5.1 General notes on marking	9
1.5.2 Version identification of EL terminals	9
1.5.3 Beckhoff Identification Code (BIC)	10
1.5.4 Electronic access to the BIC (eBIC)	12
<b>2 Product overview</b>	<b>15</b>
2.1 Introduction	15
2.2 Technical data	16
2.3 LEDs and connection	17
<b>3 Basics communication</b>	<b>19</b>
3.1 EtherCAT basics	19
3.2 EtherCAT cabling – wire-bound	19
3.3 General notes for setting the watchdog	21
3.4 EtherCAT State Machine	22
3.5 CoE Interface	24
3.6 Distributed Clock	29
<b>4 Mounting and wiring</b>	<b>30</b>
4.1 Instructions for ESD protection	30
4.2 Installation on mounting rails	31
4.3 Installation positions	34
4.4 Positioning of passive Terminals	36
4.5 Connection	37
4.5.1 Connection system	37
4.5.2 Wiring	39
4.5.3 Shielding	40
4.6 Note - power supply	40
4.7 Disposal	40
<b>5 Commissioning</b>	<b>41</b>
5.1 TwinCAT basics	41
5.1.1 TwinCAT Development Environment	41
5.1.2 TwinCAT Quick Start	81
5.2 Notes on commissioning	108
5.2.1 Connection	108
5.2.2 Supply of the connected LEDs	108
5.2.3 Adjustable parameters	109
5.2.4 Updating procedure	110
5.3 Function and parameterization	112
5.3.1 Basics of the "Modules/Slots" procedure	112

5.3.2	Modules/Slots configuration EL2574.....	115
5.3.3	Command mode.....	116
5.3.4	Extended mode.....	121
5.3.5	Status information .....	122
5.4	Process data .....	123
5.4.1	Sync-Manager (SM).....	123
5.4.2	Manual PDO Assignment.....	124
5.5	Diagnosis .....	126
5.5.1	Diagnostics in the CoE.....	126
5.5.2	Diagnostics in the Diag Messages .....	126
5.5.3	Diagnostics - basic principles of diag messages.....	126
5.6	Object description and parameterization.....	137
5.6.1	Profile-specific objects .....	137
5.6.2	Standard objects .....	144
5.7	General Commissioning Instructions for an EtherCAT Slave .....	148
<b>6</b>	<b>Sample program EL2574 .....</b>	<b>156</b>
6.1	General Information .....	157
6.2	Function block .....	159
6.3	Conversion tools .....	163
<b>7</b>	<b>Appendix.....</b>	<b>165</b>
7.1	Firmware Update EL/ES/EM/ELM/EP/EPP/ERPxxxx .....	165
7.1.1	Device description ESI file/XML .....	166
7.1.2	Firmware explanation.....	169
7.1.3	Updating controller firmware *.efw .....	170
7.1.4	FPGA firmware *.rbf .....	172
7.1.5	Simultaneous updating of several EtherCAT devices .....	176
7.2	Firmware compatibility .....	177
7.3	Restoring the delivery state.....	177
7.4	Support and Service.....	179

# 1 Foreword

## 1.1 Notes on the documentation

### Intended audience

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.

It is essential that the documentation and the following notes and explanations are followed when installing and commissioning these components.

The qualified personnel is obliged to always use the currently valid documentation.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

### Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without prior announcement.

No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

### Trademarks

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® and XTS® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.



EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

### Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

### Third-party brands

Trademarks of third parties may be used in this documentation. You can find the trademark notices here:

<https://www.beckhoff.com/trademarks>

## 1.2 Guide through documentation

### NOTICE



#### Further components of documentation

This documentation describes device-specific content. It is part of the modular documentation concept for Beckhoff I/O components. For the use and safe operation of the device / devices described in this documentation, additional cross-product descriptions are required, which can be found in the following table.

Title	Description
<b>EtherCAT System Documentation</b> ( <a href="#">PDF</a> )	<ul style="list-style-type: none"> <li>• System overview</li> <li>• EtherCAT basics</li> <li>• Cable redundancy</li> <li>• Hot Connect</li> <li>• EtherCAT devices configuration</li> </ul>
<b>Infrastructure for EtherCAT/Ethernet</b> ( <a href="#">PDF</a> )	Technical recommendations and notes for design, implementation and testing
<b>Software Declarations I/O</b> ( <a href="#">PDF</a> )	Open source software declarations for Beckhoff I/O components

The documentations can be viewed at and downloaded from the Beckhoff website ([www.beckhoff.com](http://www.beckhoff.com)) via:

- the “Documentation and Download” area of the respective product page,
- the [Download finder](#),
- the [Beckhoff Information System](#).

If you have any suggestions or proposals for our documentation, please send us an e-mail stating the documentation title and version number to: [documentation@beckhoff.com](mailto:documentation@beckhoff.com)

## 1.3 Safety instructions

### Safety regulations

Please note the following safety instructions and explanations!

Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

### Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

### Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

### Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

#### Personal injury warnings

##### **DANGER**

Hazard with high risk of death or serious injury.

##### **WARNING**

Hazard with medium risk of death or serious injury.

##### **CAUTION**

There is a low-risk hazard that could result in medium or minor injury.

#### Warning of damage to property or environment

##### **NOTICE**

The environment, equipment, or data may be damaged.

#### Information on handling the product



This information includes, for example:  
recommendations for action, assistance or further information on the product.

## 1.4 Documentation issue status

Version	Comment
1.1.0	<ul style="list-style-type: none"><li>• Sample program supplemented</li></ul>
1.0	<ul style="list-style-type: none"><li>• Addenda and corrections</li><li>• First release</li></ul>
0.1	<ul style="list-style-type: none"><li>• Provisional documentation for EL2574</li></ul>



## 1.5 Version identification of EtherCAT devices

### 1.5.1 General notes on marking

#### Designation

A Beckhoff EtherCAT device has a 14-digit designation, made up of

- family key
- type
- version
- revision

Example	Family	Type	Version	Revision
EL3314-0000-0016	EL terminal 12 mm, non-pluggable connection level	3314 4-channel thermocouple terminal	0000 basic type	0016
ES3602-0010-0017	ES terminal 12 mm, pluggable connection level	3602 2-channel voltage measurement	0010 high-precision version	0017
CU2008-0000-0000	CU device	2008 8-port fast ethernet switch	0000 basic type	0000

#### Notes

- The elements mentioned above result in the **technical designation**. EL3314-0000-0016 is used in the example below.
- EL3314-0000 is the order identifier, in the case of "-0000" usually abbreviated to EL3314. "-0016" is the EtherCAT revision.
- The **order identifier** is made up of
  - family key (EL, EP, CU, ES, KL, CX, etc.)
  - type (3314)
  - version (-0000)
- The **revision** -0016 shows the technical progress, such as the extension of features with regard to the EtherCAT communication, and is managed by Beckhoff.  
In principle, a device with a higher revision can replace a device with a lower revision, unless specified otherwise, e.g. in the documentation.  
Associated and synonymous with each revision there is usually a description (ESI, EtherCAT Slave Information) in the form of an XML file, which is available for download from the Beckhoff web site.  
From 2014/01 the revision is shown on the outside of the IP20 terminals, see Fig. "EL2872 with revision 0022 and serial number 01200815".
- The type, version and revision are read as decimal numbers, even if they are technically saved in hexadecimal.

### 1.5.2 Version identification of EL terminals

The serial number/ data code for Beckhoff IO devices is usually the 8-digit number printed on the device or on a sticker. The serial number indicates the configuration in delivery state and therefore refers to a whole production batch, without distinguishing the individual modules of a batch.

Structure of the serial number: **KK YY FF HH**

KK - week of production (CW, calendar week)  
YY - year of production  
FF - firmware version  
HH - hardware version

Example with serial number 12 06 3A 02:

12 - production week 12  
06 - production year 2006  
3A - firmware version 3A  
02 - hardware version 02



Fig. 1: EL2872 with revision 0022 and serial number 01200815

### 1.5.3 Beckhoff Identification Code (BIC)

The Beckhoff Identification Code (BIC) is increasingly being applied to Beckhoff products to uniquely identify the product. The BIC is represented as a Data Matrix Code (DMC, code scheme ECC200), the content is based on the ANSI standard MH10.8.2-2016.

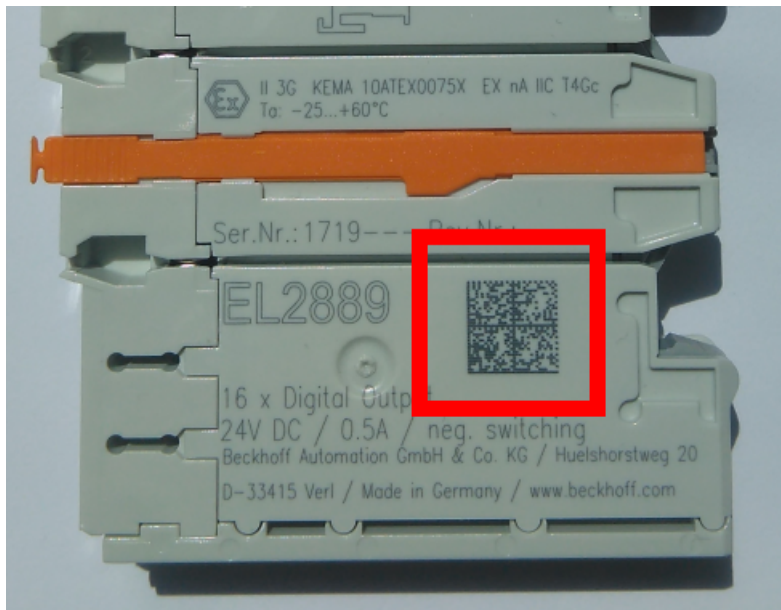


Fig. 2: BIC as data matrix code (DMC, code scheme ECC200)

The BIC will be introduced step by step across all product groups.

Depending on the product, it can be found in the following places:

- on the packaging unit
- directly on the product (if space suffices)
- on the packaging unit and the product

The BIC is machine-readable and contains information that can also be used by the customer for handling and product management.

Each piece of information can be uniquely identified using the so-called data identifier (ANSI MH10.8.2-2016). The data identifier is followed by a character string. Both together have a maximum length according to the table below. If the information is shorter, spaces are added to it.

Following information is possible, positions 1 to 4 are always present, the other according to need of production:

Position	Type of information	Explanation	Data identifier	Number of digits incl. data identifier	Example
1	Beckhoff order number	<b>Beckhoff order number</b>	1P	8	<b>1P</b> 072222
2	Beckhoff Traceability Number (BTN)	<b>Unique serial number, see note below</b>	SBTN	12	<b>SBTN</b> k4p562d7
3	Article description	<b>Beckhoff article description, e.g. EL1008</b>	1K	32	<b>1K</b> EL1809
4	Quantity	<b>Quantity in packaging unit, e.g. 1, 10, etc.</b>	Q	6	<b>Q1</b>
5	Batch number	Optional: Year and week of production	2P	14	<b>2P</b> 401503180016
6	ID/serial number	Optional: Present-day serial number system, e.g. with safety products	51S	12	<b>51S</b> 678294
7	Variant number	Optional: Product variant number on the basis of standard products	30P	12	<b>30P</b> F971, 2*K183
...					

Further types of information and data identifiers are used by Beckhoff and serve internal processes.

### Structure of the BIC

Example of composite information from positions 1 to 4 and with the above given example value on position 6. The data identifiers are highlighted in bold font:

**1P**072222**SBTN**k4p562d7**1K**EL1809 **Q1** **51S**678294

Accordingly as DMC:



Fig. 3: Example DMC **1P**072222**SBTN**k4p562d7**1K**EL1809 **Q1** **51S**678294

### BTN

An important component of the BIC is the Beckhoff Traceability Number (BTN, position 2). The BTN is a unique serial number consisting of eight characters that will replace all other serial number systems at Beckhoff in the long term (e.g. batch designations on IO components, previous serial number range for safety products, etc.). The BTN will also be introduced step by step, so it may happen that the BTN is not yet coded in the BIC.

### NOTICE

This information has been carefully prepared. However, the procedure described is constantly being further developed. We reserve the right to revise and change procedures and documentation at any time and without prior notice. No claims for changes can be made from the information, illustrations and descriptions in this documentation.

## 1.5.4 Electronic access to the BIC (eBIC)

### Electronic BIC (eBIC)

The Beckhoff Identification Code (BIC) is applied to the outside of Beckhoff products in a visible place. If possible, it should also be electronically readable.

The interface that the product can be electronically addressed by is crucial for the electronic readout.

### K-bus devices (IP20, IP67)

Currently, no electronic storage or readout is planned for these devices.

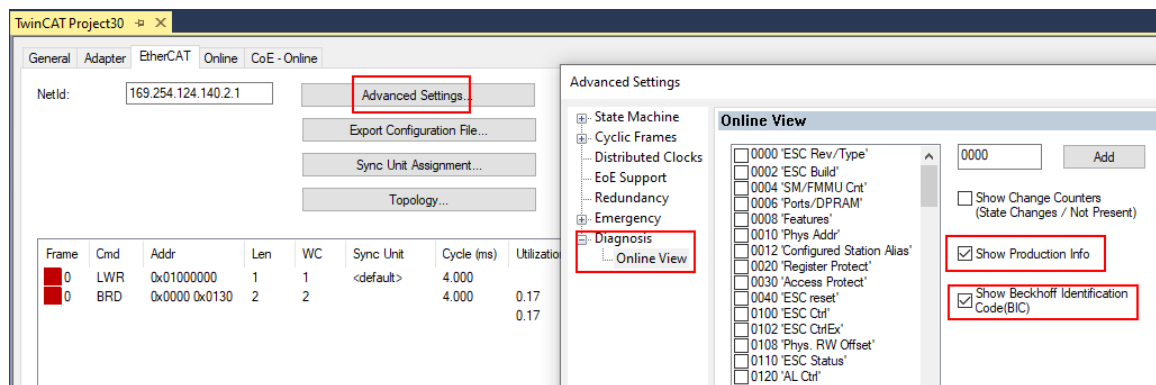
### EtherCAT devices (IP20, IP67)

All Beckhoff EtherCAT devices have an ESI-EEPROM which contains the EtherCAT identity with the revision number. The EtherCAT slave information, also colloquially known as the ESI/XML configuration file for the EtherCAT master, is stored in it. See the corresponding chapter in the EtherCAT system manual ([Link](#)) for the relationships.

Beckhoff also stores the eBIC in the ESI-EEPROM. The eBIC was introduced into Beckhoff IO production (terminals, box modules) in 2020; as of 2023, implementation is largely complete.

The user can electronically access the eBIC (if present) as follows:

- With all EtherCAT devices, the EtherCAT master (TwinCAT) can read the eBIC from the ESI-EEPROM
  - From TwinCAT 3.1 build 4024.11, the eBIC can be displayed in the online view.
  - To do this, check the "Show Beckhoff Identification Code (BIC)" checkbox under EtherCAT → Advanced Settings → Diagnostics:



- The BTN and its contents are then displayed:

No	Addr	Name	State	CRC	Fw	Hw	Production Data	ItemNo	BTN	Description	Quantity	BatchNo	SerialNo
1	1001	Term 1 (EK1100)	OP	0.0	0	0	---						
2	1002	Term 2 (EL1018)	OP	0.0	0	0	2020 KW36 Fr	072222	k4p562d7	EL1809	1		678294
3	1003	Term 3 (EL3204)	OP	0.0	7	6	2012 KW24 Sa						
4	1004	Term 4 (EL2004)	OP	0.0	0	0	---	072223	k4p562d7	EL2004	1		678295
5	1005	Term 5 (EL1008)	OP	0.0	0	0	---						
6	1006	Term 6 (EL2008)	OP	0.0	0	12	2014 KW14 Mo						
7	1007	Term 7 (EK1110)	OP	0	1	8	2012 KW25 Mo						

- Note: As shown in the figure, the production data HW version, FW version, and production date, which have been programmed since 2012, can also be displayed with "Show production info".
- Access from the PLC: From TwinCAT 3.1. build 4024.24, the functions *FB\_EcReadBIC* and *FB\_EcReadBTN* for reading into the PLC are available in the Tc2\_EtherCAT library from v3.3.19.0.
- EtherCAT devices with a CoE directory may also have the object 0x10E2:01 to display their own eBIC, which can also be easily accessed by the PLC:

- The device must be in PREOP/SAFEOP/OP for access:

Index	Name	Flags	Value
1000	Device type	RO	0x015E1389 (22942601)
1008	Device name	RO	ELM3704-0000
1009	Hardware version	RO	00
100A	Software version	RO	01
100B	Bootloader version	RO	J0.1.27.0
+ 1011:0	Restore default parameters	RO	> 1 <
+ 1018:0	Identity	RO	> 4 <
- 10E2:0	Manufacturer-specific Identification C...	RO	> 1 <
- 10E2:01	SubIndex 001	RO	1P158442SBTN0008jckp1KELM3704 Q1 2P482001000016
+ 10F0:0	Backup parameter handling	RO	> 1 <
+ 10F3:0	Diagnosis History	RO	> 21 <
- 10F8	Actual Time Stamp	RO	0x170bf6277e

- The object 0x10E2 will be preferentially introduced into stock products in the course of necessary firmware revision.
- From TwinCAT 3.1. build 4024.24, the functions *FB\_EcCoEReadBIC* and *FB\_EcCoEReadBTN* for reading into the PLC are available in the Tc2\_EtherCAT library from v3.3.19.0
- The following auxiliary functions are available for processing the BIC/BTN data in the PLC in *Tc2\_Uilities* as of TwinCAT 3.1 build 4024.24
  - *F\_SplitBIC*: The function splits the Beckhoff Identification Code (BIC) sBICValue into its components using known identifiers and returns the recognized substrings in the ST\_SplittedBIC structure as a return value
  - *BIC\_TO\_BTN*: The function extracts the BTN from the BIC and returns it as a return value
- Note: If there is further electronic processing, the BTN is to be handled as a string(8); the identifier "SBTN" is not part of the BTN.
- Technical background  
The new BIC information is written as an additional category in the ESI-EEPROM during device production. The structure of the ESI content is largely dictated by the ETG specifications, therefore the additional vendor-specific content is stored using a category in accordance with the ETG.2010. ID 03 tells all EtherCAT masters that they may not overwrite these data in the event of an update or restore the data after an ESI update.  
The structure follows the content of the BIC, see here. The EEPROM therefore requires approx. 50..200 bytes of memory.
- Special cases
  - If multiple hierarchically arranged ESCs are installed in a device, only the top-level ESC carries the eBIC information.
  - If multiple non-hierarchically arranged ESCs are installed in a device, all ESCs carry the eBIC information.
  - If the device consists of several sub-devices which each have their own identity, but only the top-level device is accessible via EtherCAT, the eBIC of the top-level device is located in the CoE object directory 0x10E2:01 and the eBICs of the sub-devices follow in 0x10E2:nn.

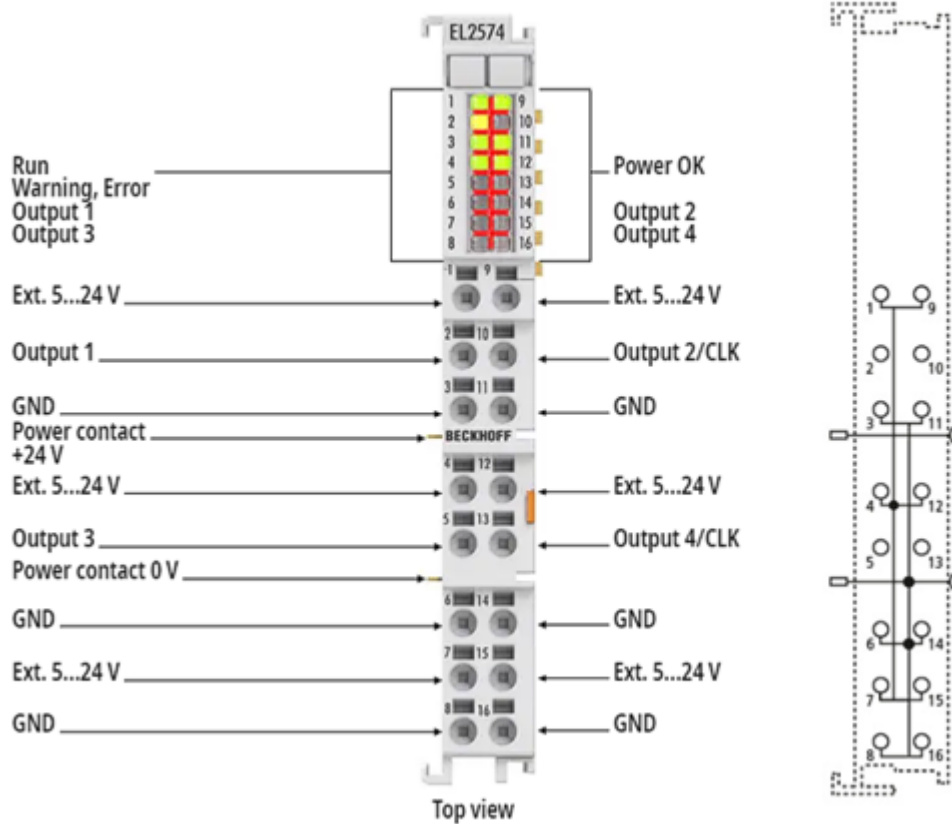
## PROFIBUS, PROFINET, and DeviceNet devices

Currently, no electronic storage or readout is planned for these devices.



## 2 Product overview

### 2.1 Introduction



#### EtherCAT Terminal, 4-channel LED output, pixel LED

The EL2574 enables the control of LEDs with an integrated chip. If these so-called pixel LEDs are connected as a strip or matrix, all LEDs can be controlled differently. This allows effects such as running lights or animations. The EL2574 supports various protocols for controlling these LEDs. Divided into four channels, up to 2048 pixels can be controlled with just one terminal, without the need for an additional controller.

Possible applications:

- Pick-by-Light
- Machine status display
- Position marking
- Stage and Show

## 2.2 Technical data

Device functions	EL2574
Application recommendation	Control of individually addressable LEDs
Number of outputs	4
Input voltage	5...24 V <sub>DC</sub>
Power supply LED	<ul style="list-style-type: none"> <li>• External</li> <li>• or via the terminal points "Ext. 5...24 V" and "GND" with the following restrictions (applies to HW &lt; 02): <ul style="list-style-type: none"> <li>◦ &gt; 50°C ambient temperature: no supply via terminal</li> <li>◦ 45...50°C ambient temperature: with 3 A supply</li> <li>◦ &lt; 45°C ambient temperature: with 6 A supply</li> <li>◦ With ZB8610 8 A: supply over the entire temperature range.</li> </ul> </li> </ul>
LED protocol	WS2801, WS2803, WS2811, WS2812, WS2812B, WS2813, WS2815, WS2818, APA-101, APA102, APA-104, APA-109, CS8812, GS8206, GS8208, INK1002, INK1003, SK6812, SK6813, SK6822, SM16703, SM16704, TM1803, TM1804, TM1809, TM1812, TM1814, UCS1903, UCS1912, UCS2903, UCS2912, UCS2904, GE8822, HD107S, SK9822
Number of pixels	2048 pixels per terminal
Supply of the internal electronics	24 V via power contacts

Communication	EL2574
Configuration	via TwinCAT System Manager
Distributed Clocks	–

General data	EL2574
Current consumption via E-bus	80 mA typ.
Weight	approx. 50 g
Dimensions	approx. 15 mm x 100 mm x 70 mm (width aligned: 12 mm)
Mounting	on 35 mm mounting rail, conforms to EN 60715
Installation position	variable

Environmental conditions	EL2574
Permissible ambient temperature range during operation	0°C ... +55°C -25°C ... +60°C (from HW02)
Permissible ambient temperature range during storage	-25°C ... +85°C
Permissible relative air humidity	95%, no condensation

Standards and approvals	EL2574
Vibration / shock resistance	conforms to EN 60068-2-6 / EN 60068-2-27
EMC immunity / emission	conforms to EN 61000-6-2 / EN 61000-6-4
Protection rating	IP20
Approvals/markings*)	CE

\*) Real applicable approvals/markings see type plate on the side (product marking).



## 2.3 LEDs and connection

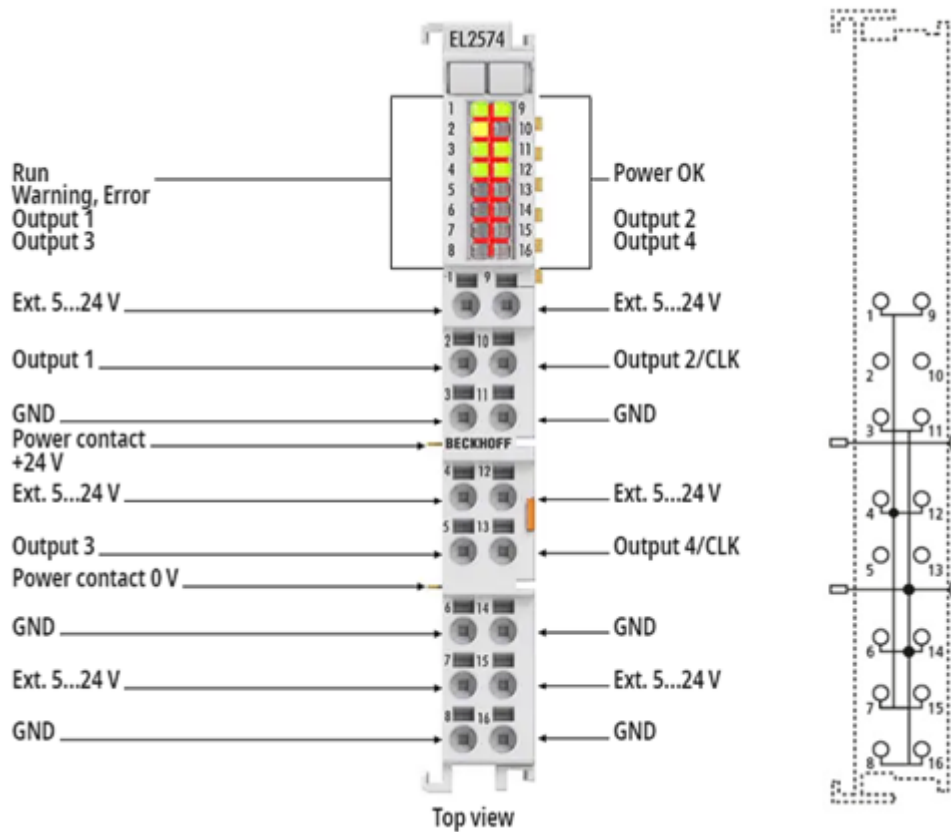


Fig. 4: EL2574 connection

EL2574 - connection		
Terminal point	No.	Comment
Ext. 5 ... 24 V	1	5 ... 24 V supply voltage for the LEDs (voltage depending on the led used; this voltage does not supply the internal electronics and is not monitored)
Output 1	2	Data output 1
GND	3	Ground of the LED supply voltage (internally connected to the 0 V power contact)
Ext. 5 ... 24 V	4	5 ... 24 V supply voltage for the LEDs (voltage depending on the led used; this voltage does not supply the internal electronics and is not monitored)
Output 3	5	Data output 3
GND	6	Ground of the LED supply voltage (internally connected to the 0 V power contact)
Ext. 5 ... 24 V	7	5 ... 24 V supply voltage for the LEDs (voltage depending on the led used; this voltage does not supply the internal electronics and is not monitored)
GND	8	Ground of the LED supply voltage (internally connected to the 0 V power contact)
Ext. 5 ... 24 V	9	5 ... 24 V supply voltage for the LEDs (voltage depending on the led used; this voltage does not supply the internal electronics and is not monitored)
Output 2 / CLK	10	Data output 2 / Clock output 1
GND	11	Ground of the LED supply voltage (internally connected to the 0 V power contact)
Ext. 5 ... 24 V	12	5 ... 24 V supply voltage for the LEDs (voltage depending on the led used; this voltage does not supply the internal electronics and is not monitored)
Output 4 / CLK	13	Data output 4 / Clock output 2
GND	14	Ground of the LED supply voltage (internally connected to the 0 V power contact)
Ext. 5 ... 24 V	15	5 ... 24 V supply voltage for the LEDs (voltage depending on the led used; this voltage does not supply the internal electronics and is not monitored)
GND	16	Ground of the LED supply voltage (internally connected to the 0 V power contact)

### NOTICE



#### Observe the connection instructions!

Observe the information and notes in chapter "[Mounting and wiring](#)" [► 30] as well as in chapter "[Connection](#)" [► 108]!

EL2574 - LEDs

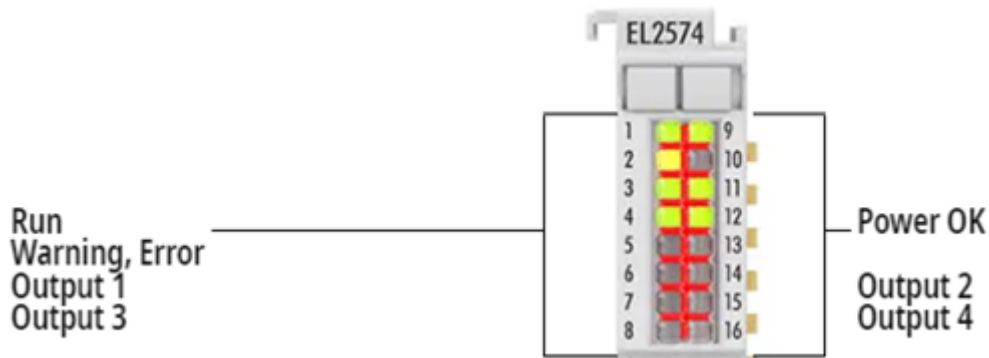


Fig. 5: EL2574 LEDs

LED	Color	Meaning
RUN	green	This LED indicates the terminal's operating state:
		offState of the EtherCAT State Machine: <b>INIT</b> = initialization of the terminal
		flashingState of the EtherCAT State Machine: <b>PREOP</b> = function for mailbox communication and different default settings set
		single flashState of the EtherCAT State Machine: <b>SAFEOP</b> = verification of the Sync Manager channels and the distributed clocks. Outputs remain in safe state
		onState of the EtherCAT State Machine: <b>OP</b> = normal operating state; mailbox and process data communication is possible
		flickeringState of the EtherCAT State Machine: <b>BOOTSTRAP</b> = function for Firmware updates of the terminal
Warning, Error	yellow	Warning
	red	Error
Power OK	green	The supply to the electronics is within the permissible range.
Output 1 ... 4	green	The LED output is active.

## 3 Basics communication

### 3.1 EtherCAT basics

Please refer to the [EtherCAT System Documentation](#) for the EtherCAT fieldbus basics.

### 3.2 EtherCAT cabling – wire-bound

The cable length between two EtherCAT devices must not exceed 100 m. This results from the FastEthernet technology, which, above all for reasons of signal attenuation over the length of the cable, allows a maximum link length of 5 + 90 + 5 m if cables with appropriate properties are used. See also the [Design recommendations for the infrastructure for EtherCAT/Ethernet](#).

#### Cables and connectors

For connecting EtherCAT devices only Ethernet connections (cables + plugs) that meet the requirements of at least category 5 (Cat5) according to EN 50173 or ISO/IEC 11801 should be used. EtherCAT uses 4 wires for signal transfer.

EtherCAT uses RJ45 plug connectors, for example. The pin assignment is compatible with the Ethernet standard (ISO/IEC 8802-3).

Pin	Color of conductor	Signal	Description
1	yellow	TD +	Transmission Data +
2	orange	TD -	Transmission Data -
3	white	RD +	Receiver Data +
6	blue	RD -	Receiver Data -

Due to automatic cable detection (auto-crossing) symmetric (1:1) or cross-over cables can be used between EtherCAT devices from Beckhoff.



#### Recommended cables

It is recommended to use the appropriate Beckhoff components e.g.

- cable sets ZK1090-9191-xxxx respectively
- RJ45 connector, field assembly ZS1090-0005
- EtherCAT cable, field assembly ZB9010, ZB9020

Suitable cables for the connection of EtherCAT devices can be found on the [Beckhoff website!](#)

#### E-Bus supply

A bus coupler can supply the EL terminals added to it with the E-bus system voltage of 5 V; a coupler is thereby loadable up to 2 A as a rule (see details in respective device documentation). Information on how much current each EL terminal requires from the E-bus supply is available online and in the catalogue. If the added terminals require more current than the coupler can supply, then power feed terminals (e.g. [EL9410](#)) must be inserted at appropriate places in the terminal strand.

The pre-calculated theoretical maximum E-Bus current is displayed in the TwinCAT System Manager. A shortfall is marked by a negative total amount and an exclamation mark; a power feed terminal is to be placed before such a position.

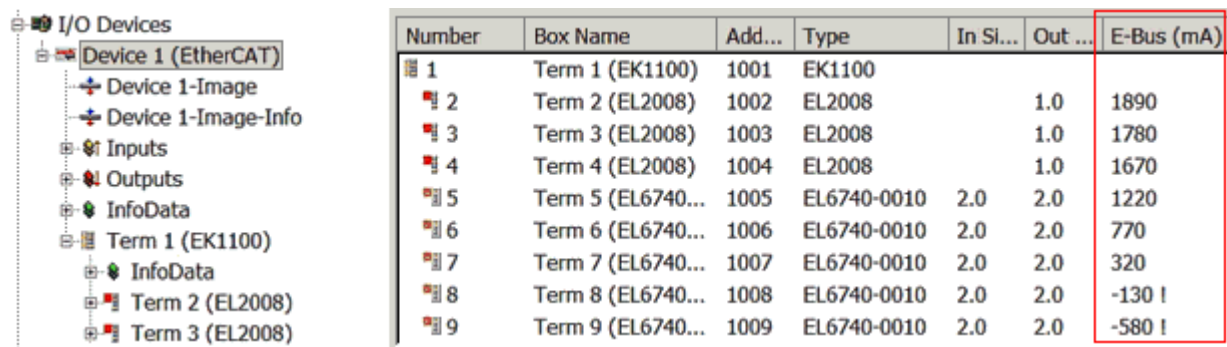


Fig. 6: System manager current calculation

**NOTICE**

**Malfunction possible!**

The same ground potential must be used for the E-Bus supply of all EtherCAT terminals in a terminal block!

### 3.3 General notes for setting the watchdog

The EtherCAT terminals are equipped with a safety device (watchdog) which, e. g. in the event of interrupted process data traffic, switches the outputs (if present) to a presettable state after a presettable time, depending on the device and setting, e. g. to FALSE (off) or an output value.

The EtherCAT slave controller features two watchdogs:

- Sync Manager (SM) watchdog (default: 100 ms)
- Process Data (PDI) watchdog (default: 100 ms)

Their times are individually parameterized in TwinCAT as follows:

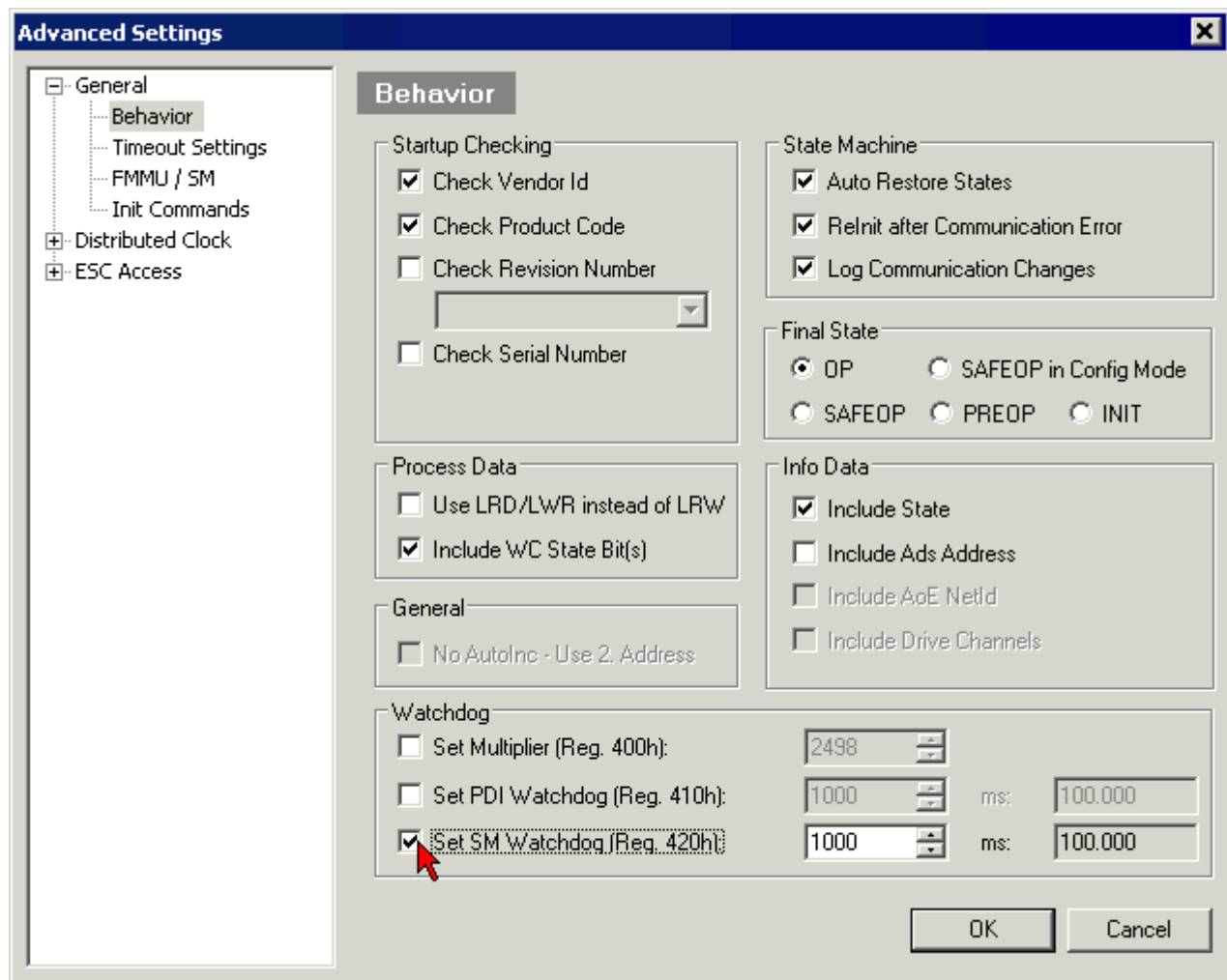


Fig. 7: eEtherCAT tab -> Advanced Settings -> Behavior -> Watchdog

Notes:

- the Multiplier Register 400h (hexadecimal, i. e. 0x0400) is valid for both watchdogs.
- each watchdog has its own timer setting 410h or 420h, which together with the Multiplier results in a resulting time.
- important: the Multiplier/Timer setting is only loaded into the slave at EtherCAT startup if the checkbox in front of it is activated.
- if it is not checked, nothing is downloaded and the setting located in the ESC remains unchanged.
- the downloaded values can be seen in the ESC registers 400h, 410h and 420h: ESC Access -> Memory

**SM watchdog (SyncManager Watchdog)**

The SyncManager watchdog is reset with each successful EtherCAT process data communication with the terminal. If, for example, no EtherCAT process data communication with the terminal takes place for longer than the set and activated SM watchdog time due to a line interruption, the watchdog is triggered. The status of the terminal (usually OP) remains unaffected. The watchdog is only reset again by a successful EtherCAT process data access.

The SyncManager watchdog is therefore a monitoring for correct and timely process data communication with the ESC from the EtherCAT side.

The maximum possible watchdog time depends on the device. For example, for "simple" EtherCAT slaves (without firmware) with watchdog execution in the ESC it is usually up to 170 seconds. For complex EtherCAT slaves (with firmware) the SM watchdog function is usually parameterized via register 400h/420h but executed by the microcontroller (µC) and can be significantly lower. In addition, the execution may then be subject to a certain time uncertainty. Since the TwinCAT dialog may allow inputs up to 65535, a test of the desired watchdog time is recommended.

**PDI watchdog (Process Data Watchdog)**

If there is no PDI communication with the ESC for longer than the set and activated Process Data Interface (PDI) watchdog time, this watchdog is triggered.

The PDI is the internal interface of the ESC, e.g. to local processors in the EtherCAT slave. With the PDI watchdog this communication can be monitored for failure.

The PDI watchdog is therefore a monitoring for correct and timely process data communication with the ESC, but viewed from the application side.

**Calculation**

Watchdog time =  $[1/25 \text{ MHz} * (\text{Watchdog multiplier} + 2)] * \text{SM/PDI watchdog}$

Example: default setting Multiplier = 2498, SM watchdog = 1000 => 100 ms

The value in "Watchdog multiplier + 2" in the formula above corresponds to the number of 40ns base ticks representing one watchdog tick.

**⚠ CAUTION****Undefined state possible!**

The function for switching off the SM watchdog via SM watchdog = 0 is only implemented in terminals from revision -0016. In previous versions this operating mode should not be used.

**⚠ CAUTION****Damage of devices and undefined state possible!**

If the SM watchdog is activated and a value of 0 is entered the watchdog switches off completely. This is the deactivation of the watchdog! Set outputs are NOT set in a safe state if the communication is interrupted.

**3.4 EtherCAT State Machine**

The state of the EtherCAT slave is controlled via the EtherCAT State Machine (ESM). Depending upon the state, different functions are accessible or executable in the EtherCAT slave. Specific commands must be sent by the EtherCAT master to the device in each state, particularly during the bootup of the slave.

A distinction is made between the following states:

- Init
- Pre-Operational
- Safe-Operational
- Operational

- Bootstrap

The regular state of each EtherCAT slave after bootup is the OP state.

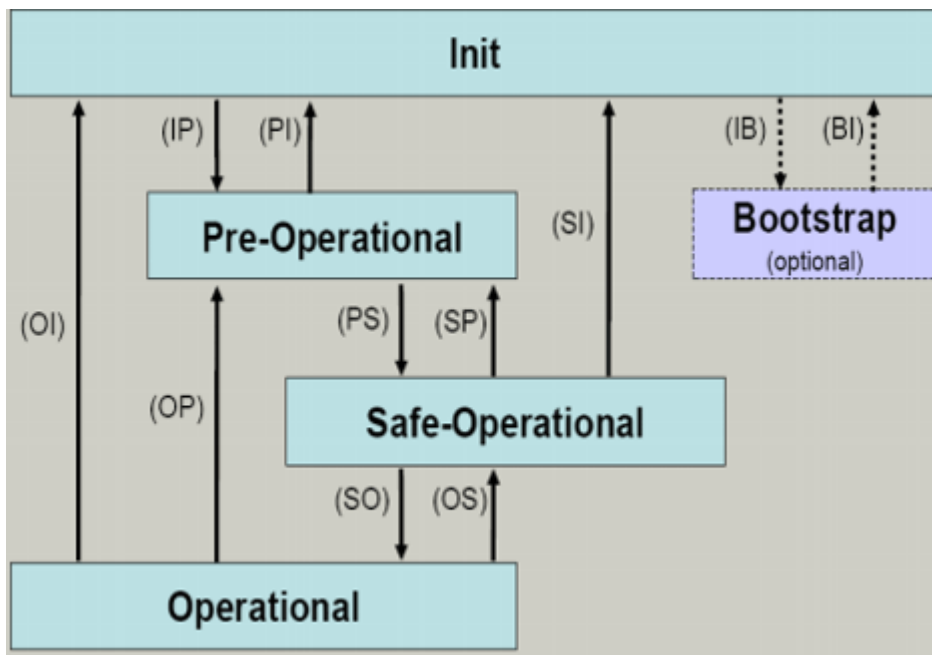


Fig. 8: States of the EtherCAT State Machine

## Init

After switch-on the EtherCAT slave in the *Init* state. No mailbox or process data communication is possible. The EtherCAT master initializes sync manager channels 0 and 1 for mailbox communication.

## Pre-Operational (Pre-Op)

During the transition between *Init* and *Pre-Op* the EtherCAT slave checks whether the mailbox was initialized correctly.

In *Pre-Op* state mailbox communication is possible, but not process data communication. The EtherCAT master initializes the sync manager channels for process data (from sync manager channel 2), the Fieldbus Memory Management Unit (FMMU) channels and, if the slave supports configurable mapping, PDO mapping or the sync manager PDO assignment. In this state the settings for the process data transfer and perhaps terminal-specific parameters that may differ from the default settings are also transferred.

## Safe-Operational (Safe-Op)

During transition between *Pre-Op* and *Safe-Op* the EtherCAT slave checks whether the sync manager channels for process data communication and, if required, the Distributed Clocks settings are correct. Before it acknowledges the change of state, the EtherCAT slave copies current input data into the associated Dual Port (DP)-RAM areas of the ESC.

In *Safe-Op* state mailbox and process data communication is possible, although the slave keeps its outputs in a safe state, while the input data are updated cyclically.

### ● Outputs in SAFEOP state

**i**

The default set watchdog monitoring sets the outputs of the ESC module in a safe state - depending on the settings in SAFEOP and OP - e.g. in OFF state. If this is prevented by deactivation of the monitoring in the module, the outputs can be switched or set also in the SAFEOP state.

## Operational (Op)

Before the EtherCAT master switches the EtherCAT slave from *Safe-Op* to *Op* it must transfer valid output data.

In the *Op* state the slave copies the output data of the masters to its outputs. Process data and mailbox communication is possible.

### Boot

In the *Boot* state the slave firmware can be updated. The *Boot* state can only be reached via the *Init* state.

In the *Boot* state mailbox communication via the file access over EtherCAT (FoE) protocol is possible, but no other mailbox communication and no process data communication.

## 3.5 CoE Interface

### General description

The CoE interface (CAN application protocol over EtherCAT interface) is used for parameter management of EtherCAT devices. EtherCAT slaves or the EtherCAT master manage fixed (read only) or variable parameters which they require for operation, diagnostics or commissioning.

CoE parameters are arranged in a table hierarchy. In principle, the user has access via the fieldbus. The EtherCAT master (TwinCAT System Manager) can access the local CoE lists of the slaves via EtherCAT in read or write mode, depending on the attributes.

Different CoE data types are possible, including string (text), integer numbers, Boolean values or larger byte fields. They can be used to describe a wide range of features. Examples of such parameters include manufacturer ID, serial number, process data settings, device name, calibration values for analog measurement or passwords.

The order is specified in two levels via hexadecimal numbering: (main)index, followed by subindex.

The value ranges are

- Index: 0x0000 ... 0xFFFF (0...65535<sub>dec</sub>)
- Subindex: 0x00...0xFF (0...255<sub>dec</sub>)

A parameter localized in this way is normally written as 0x8010:07, with preceding "0x" to identify the hexadecimal numerical range and a colon between index and subindex.

The relevant ranges for EtherCAT fieldbus users are:

- 0x1000: This is where fixed identity information for the device is stored, including name, manufacturer, serial number etc., plus information about the current and available process data configurations.
- 0x8000: This is where the operational and functional parameters for all channels are stored, such as filter settings or output frequency.

Other important ranges are:

- 0x4000: here are the channel parameters for some EtherCAT devices. Historically, this was the first parameter area before the 0x8000 area was introduced. EtherCAT devices that were previously equipped with parameters in 0x4000 and changed to 0x8000 support both ranges for compatibility reasons and mirror internally.
- 0x6000: Input PDOs ("inputs" from the perspective of the EtherCAT master)
- 0x7000: Output PDOs ("outputs" from the perspective of the EtherCAT master)

### ● Availability



Not every EtherCAT device must have a CoE list. Simple I/O modules without dedicated processor usually have no variable parameters and therefore no CoE list.

If a device has a CoE list, it is shown in the TwinCAT System Manager as a separate tab with a listing of the elements:



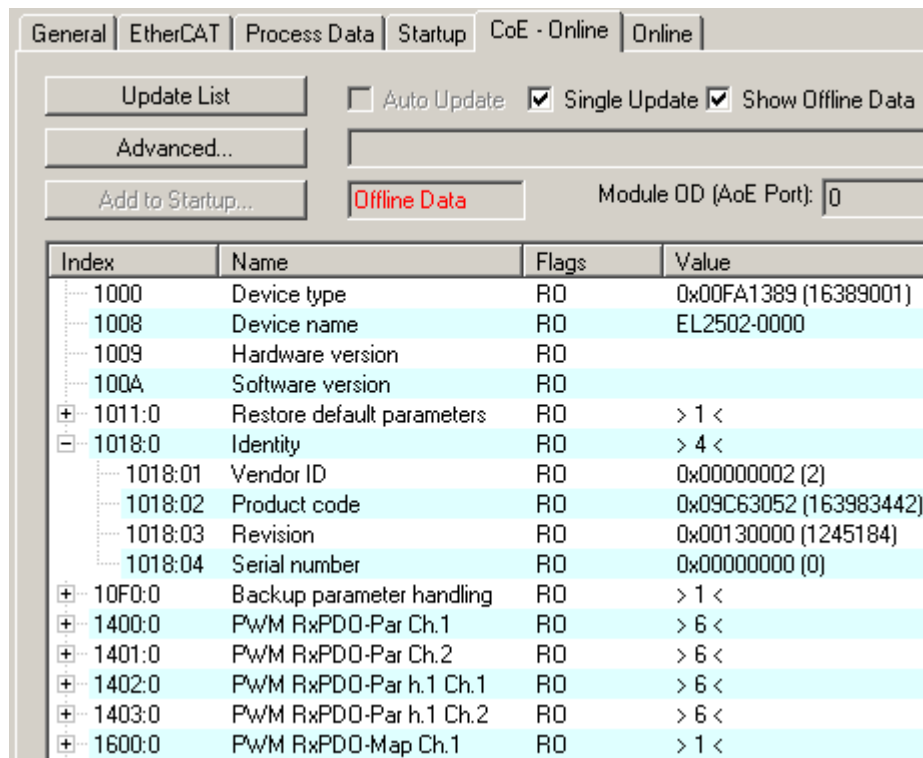


Fig. 9: "CoE Online" tab

The figure "CoE Online" tab shows the CoE objects available in device "EL2502", ranging from 0x1000 to 0x1600. The subindices for 0x1018 are expanded.

### NOTICE

#### Changes in the CoE directory (CAN over EtherCAT directory), program access

When using/manipulating the CoE parameters observe the general CoE notes in chapter "[CoE interface](#)" of the EtherCAT system documentation:

- Keep a startup list if components have to be replaced,
- Distinction between online/offline dictionary,
- Existence of current XML description (download from the [Beckhoff website](#)),
- "CoE-Reload" for resetting the changes
- Program access during operation via PLC (see [TwinCAT 3 | PLC Library: "Tc2\\_EtherCAT"](#) and [Example program R/W CoE](#))

#### Data management and function "NoCoeStorage"

Some parameters, particularly the setting parameters of the slave, are configurable and writeable,

- via the System Manager (Fig. "CoE Online" tab) by clicking.  
This is useful for commissioning of the system or slaves. Click on the row of the index to be parameterized and enter a value in the "SetValue" dialog.
- from the control system or PLC via ADS, e.g. through blocks from the TcEtherCAT.lib library.  
This is recommended for modifications while the system is running or if no System Manager or operating staff are available.

## **i** Data management

If slave CoE parameters are modified online, Beckhoff devices store any changes in a fail-safe manner in the EEPROM, i.e. the modified CoE parameters are still available after a restart. The situation may be different with other manufacturers.

An EEPROM is subject to a limited lifetime with respect to write operations. From typically 100,000 write operations onwards it can no longer be guaranteed that new (changed) data are reliably saved or are still readable. This is irrelevant for normal commissioning. However, if CoE parameters are continuously changed via ADS at machine runtime, it is quite possible for the lifetime limit to be reached. Support for the NoCoeStorage function, which suppresses the saving of changed CoE values, depends on the firmware version.

Please refer to the technical data in this documentation as to whether this applies to the respective device.

- If the function is supported: the function is activated by entering the code word 0x12345678 once in CoE index 0xF008 and remains active as long as the code word is not changed. After switching the device on it is then inactive. Changed CoE values are not saved in the EEPROM and can thus be changed any number of times.
- If the function is not supported: continuous changing of CoE values is not permissible in view of the lifetime limit.

## **i** Startup list

Changes in the local CoE list of the terminal are lost if the terminal is replaced. If a terminal is replaced with a new Beckhoff terminal, it will have the default settings. It is therefore advisable to link all changes in the CoE list of an EtherCAT slave with the Startup list of the slave, which is processed whenever the EtherCAT fieldbus is started. In this way a replacement EtherCAT slave can automatically be parameterized with the specifications of the user.

If EtherCAT slaves are used which are unable to store local CoE values permanently, the Startup list must be used.

### Recommended approach for manual modification of CoE parameters

- Make the required change in the System Manager (the values are stored locally in the EtherCAT slave).
- If the value is to be stored permanently, enter it in the Startup list.  
The order of the Startup entries is usually irrelevant.

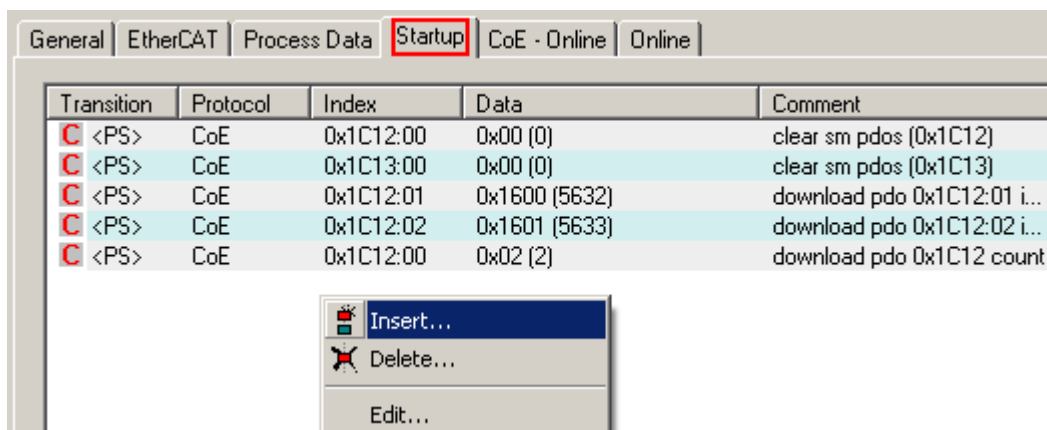


Fig. 10: Startup list in the TwinCAT System Manager

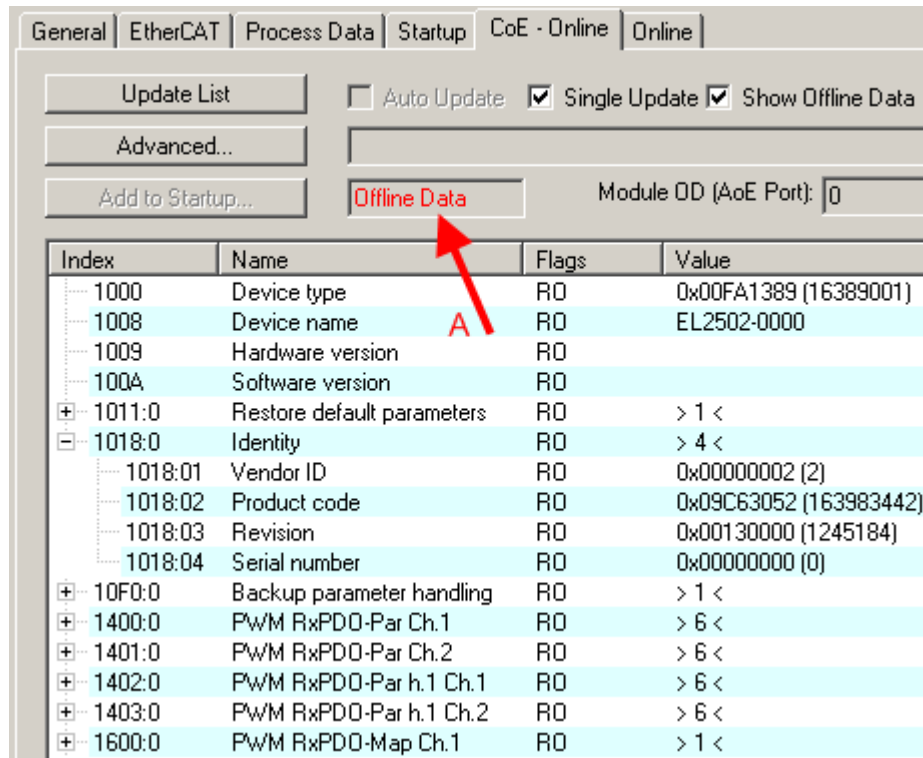
The Startup list may already contain values that were configured by the System Manager based on the ESI specifications. Additional application-specific entries can also be created.

### Online / offline list

When working with the TwinCAT System Manager, a distinction must be made as to whether the EtherCAT device is currently "available", i.e. switched on and connected via EtherCAT - i.e. **online** - or whether a configuration is created **offline** without slaves being connected.

In both cases a CoE list as shown in Fig. "CoE online tab" is displayed. The connectivity is shown as offline/online.

- If the slave is offline:
  - The offline list from the ESI file is displayed. In this case modifications are not meaningful or possible.
  - The configured status is shown under Identity.
  - No firmware or hardware version is displayed since these are features of the physical device.
  - **Offline Data** is shown in red.



The screenshot shows the 'CoE - Online' tab in a software interface. The 'Offline Data' label is highlighted in red in the table header. A red arrow points to this label. The table lists various device parameters and their values.

Index	Name	Flags	Value
1000	Device type	RO	0x00FA1389 (16389001)
1008	Device name	RO	EL2502-0000
1009	Hardware version	RO	
100A	Software version	RO	
1011:0	Restore default parameters	RO	> 1 <
1018:0	Identity	RO	> 4 <
1018:01	Vendor ID	RO	0x00000002 (2)
1018:02	Product code	RO	0x09C63052 (163983442)
1018:03	Revision	RO	0x00130000 (1245184)
1018:04	Serial number	RO	0x00000000 (0)
10F0:0	Backup parameter handling	RO	> 1 <
1400:0	PWM RxD0-Par Ch.1	RO	> 6 <
1401:0	PWM RxD0-Par Ch.2	RO	> 6 <
1402:0	PWM RxD0-Par h.1 Ch.1	RO	> 6 <
1403:0	PWM RxD0-Par h.1 Ch.2	RO	> 6 <
1600:0	PWM RxD0-Map Ch.1	RO	> 1 <

Fig. 11: Offline list

- If the slave is online:
  - The actual current slave list is read. This may take several seconds, depending on the size and cycle time.
  - The actual identity is displayed.
  - The firmware and hardware status of the device is displayed in the CoE.
  - **Online Data** is shown in green.

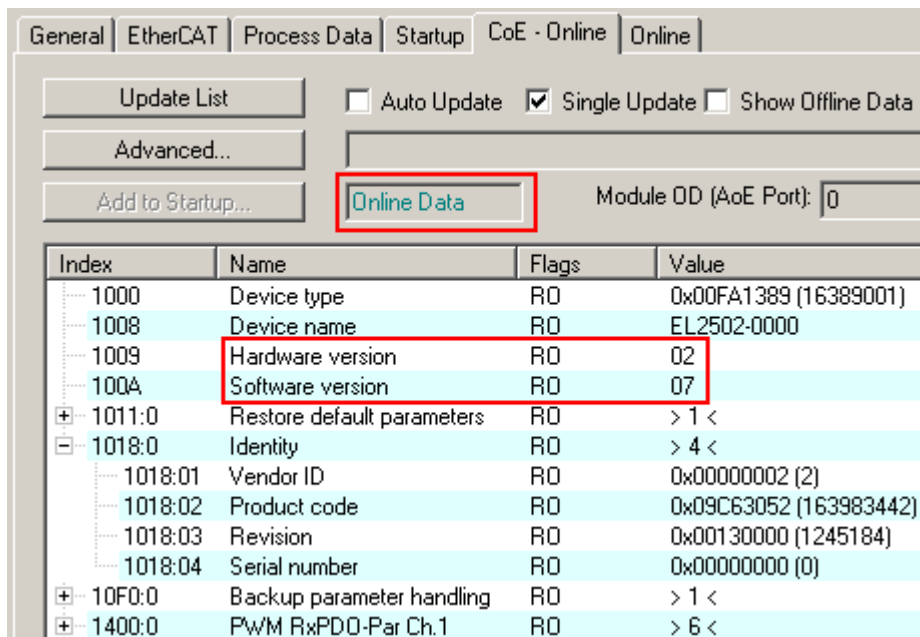


Fig. 12: Online list

### Channel-based order

The CoE list is available in EtherCAT devices that usually feature several functionally equivalent channels, for example, a 4-channel analog input terminal also has four logical channels and therefore four identical sets of parameter data for the channels. In order to avoid having to list each channel in the documentation, the placeholder “n” tends to be used for the individual channel numbers.

In the CoE system 16 indices, each with 255 subindices, are generally sufficient for representing all channel parameters. The channel-based order is therefore arranged in  $16_{\text{dec}}$  or  $10_{\text{hex}}$  steps. The parameter range 0x8000 exemplifies this:

- Channel 0: parameter range 0x8000:00 ... 0x800F:255
- Channel 1: parameter range 0x8010:00 ... 0x801F:255
- Channel 2: parameter range 0x8020:00 ... 0x802F:255
- ...

This is generally written as 0x80n0.

Detailed information on the CoE interface can be found in the [EtherCAT system documentation](#) on the Beckhoff website.

## 3.6 Distributed Clock

The distributed clock represents a local clock in the EtherCAT slave controller (ESC) with the following characteristics:

- Unit *1 ns*
- Zero point *1.1.2000 00:00*
- Size *64 bit* (sufficient for the next 584 years; however, some EtherCAT slaves only offer 32-bit support, i.e. the variable overflows after approx. 4.2 seconds)
- The EtherCAT master automatically synchronizes the local clock with the master clock in the EtherCAT bus with a precision of  $< 100$  ns.

For detailed information please refer to the [EtherCAT system description](#).

## 4 Mounting and wiring

### 4.1 Instructions for ESD protection

#### NOTICE

**Destruction of the devices by electrostatic discharge possible!**

The devices contain components at risk from electrostatic discharge caused by improper handling.

- When handling the components, ensure that there is no electrostatic discharge; also avoid touching the spring contacts directly (see illustration).
- Contact with highly insulating materials (synthetic fibers, plastic films, etc.) should be avoided when handling components at the same time.
- When handling the components, ensure that the environment (workplace, packaging and persons) is properly earthed.
- Each bus station must be terminated on the right-hand side with the [EL9011](#) or [EL9012](#) end cap to ensure the degree of protection and ESD protection.

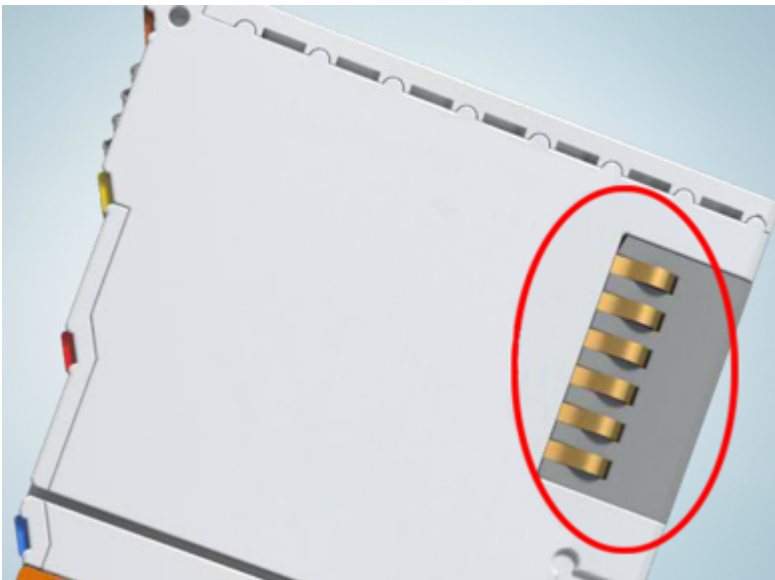


Fig. 13: Spring contacts of the Beckhoff I/O components

## 4.2 Installation on mounting rails

### ⚠ WARNING

#### Risk of electric shock and damage of device!

Bring the bus terminal system into a safe, powered down state before starting installation, disassembly or wiring of the bus terminals!

The Bus Terminal system and is designed for mounting in a control cabinet or terminal box.

#### Assembly

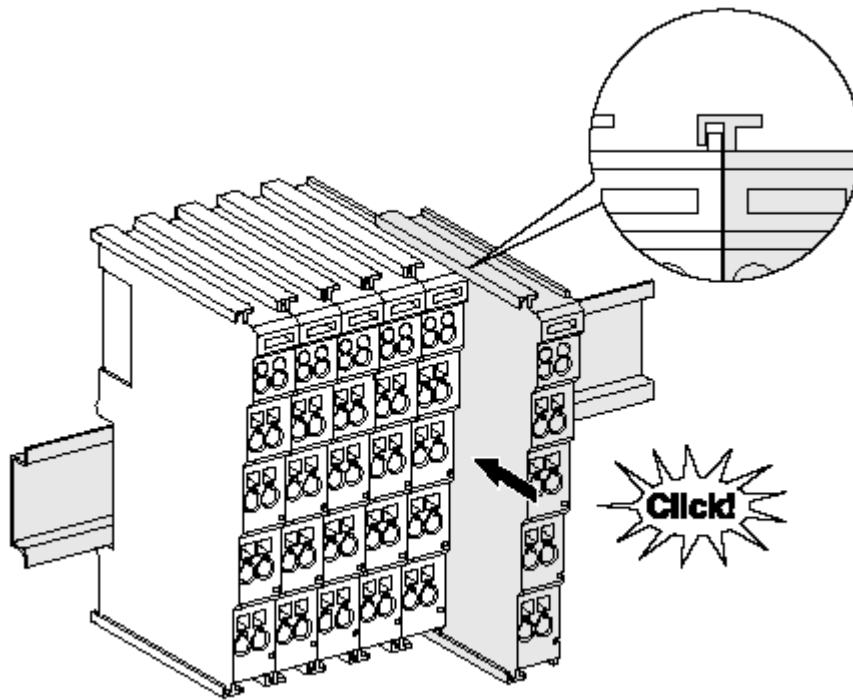


Fig. 14: Attaching on mounting rail

The bus coupler and bus terminals are attached to commercially available 35 mm mounting rails (DIN rails according to EN 60715) by applying slight pressure:

1. First attach the fieldbus coupler to the mounting rail.
2. The bus terminals are now attached on the right-hand side of the fieldbus coupler. Join the components with tongue and groove and push the terminals against the mounting rail, until the lock clicks onto the mounting rail.

If the terminals are clipped onto the mounting rail first and then pushed together without tongue and groove, the connection will not be operational! When correctly assembled, no significant gap should be visible between the housings.

#### ● Fixing of mounting rails

**i** The locking mechanism of the terminals and couplers extends to the profile of the mounting rail. At the installation, the locking mechanism of the components must not come into conflict with the fixing bolts of the mounting rail. To mount the mounting rails with a height of 7.5 mm under the terminals and couplers, you should use flat mounting connections (e.g. countersunk screws or blind rivets).

## Disassembly

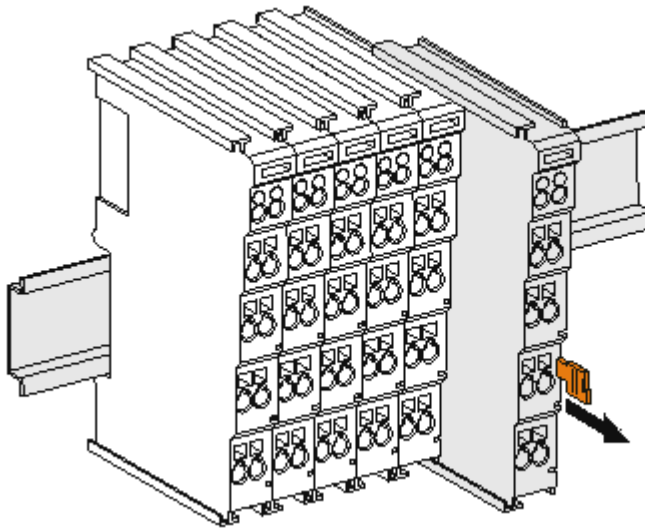


Fig. 15: Disassembling of terminal

Each terminal is secured by a lock on the mounting rail, which must be released for disassembly:

1. Pull the terminal by its orange-colored lugs approximately 1 cm away from the mounting rail. In doing so for this terminal the mounting rail lock is released automatically and you can pull the terminal out of the bus terminal block easily without excessive force.
2. Grasp the released terminal with thumb and index finger simultaneous at the upper and lower grooved housing surfaces and pull the terminal out of the bus terminal block.

## Connections within a bus terminal block

The electric connections between the Bus Coupler and the Bus Terminals are automatically realized by joining the components:

- The six spring contacts of the K-Bus/E-Bus deal with the transfer of the data and the supply of the Bus Terminal electronics.
- The power contacts deal with the supply for the field electronics and thus represent a supply rail within the bus terminal block. The power contacts are supplied via terminals points on the Bus Coupler (up to 24 V) or for higher voltages via power feed terminals.

### ● Power Contacts

**i** During the design of a bus terminal block, the pin assignment of the individual Bus Terminals must be taken account of, since some types (e.g. analog Bus Terminals or digital 4-channel Bus Terminals) do not or not fully loop through the power contacts. Power Feed Terminals (KL91xx, KL92xx or EL91xx, EL92xx) interrupt the power contacts and thus represent the start of a new supply rail.

## PE power contact

The power contact labeled PE can be used as a protective earth. For safety reasons this contact mates first when plugging together, and can ground short-circuit currents of up to 125 A.



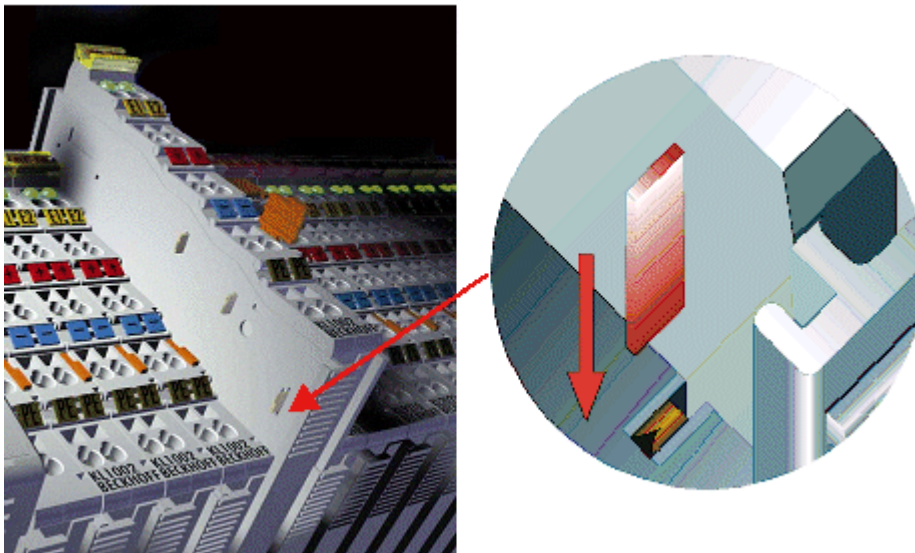


Fig. 16: Power contact on left side

### NOTICE

#### Possible damage of the device

Note that, for reasons of electromagnetic compatibility, the PE contacts are capacitatively coupled to the mounting rail. This may lead to incorrect results during insulation testing or to damage on the terminal (e.g. disruptive discharge to the PE line during insulation testing of a consumer with a nominal voltage of 230 V). For insulation testing, disconnect the PE supply line at the Bus Coupler or the Power Feed Terminal! In order to decouple further feed points for testing, these Power Feed Terminals can be released and pulled at least 10 mm from the group of terminals.

### ⚠ WARNING

#### Risk of electric shock!

The PE power contact must not be used for other potentials!

## 4.3 Installation positions

### NOTICE

#### Constraints regarding installation position and operating temperature range

Please refer to the technical data for a terminal to ascertain whether any restrictions regarding the installation position and/or the operating temperature range have been specified. When installing high power dissipation terminals ensure that an adequate spacing is maintained between other components above and below the terminal in order to guarantee adequate ventilation!

#### Optimum installation position (standard)

The optimum installation position requires the mounting rail to be installed horizontally and the connection surfaces of the EL- / KL terminals to face forward (see Fig. "Recommended distances for standard installation position"). The terminals are ventilated from below, which enables optimum cooling of the electronics through convection. "From below" is relative to the acceleration of gravity.

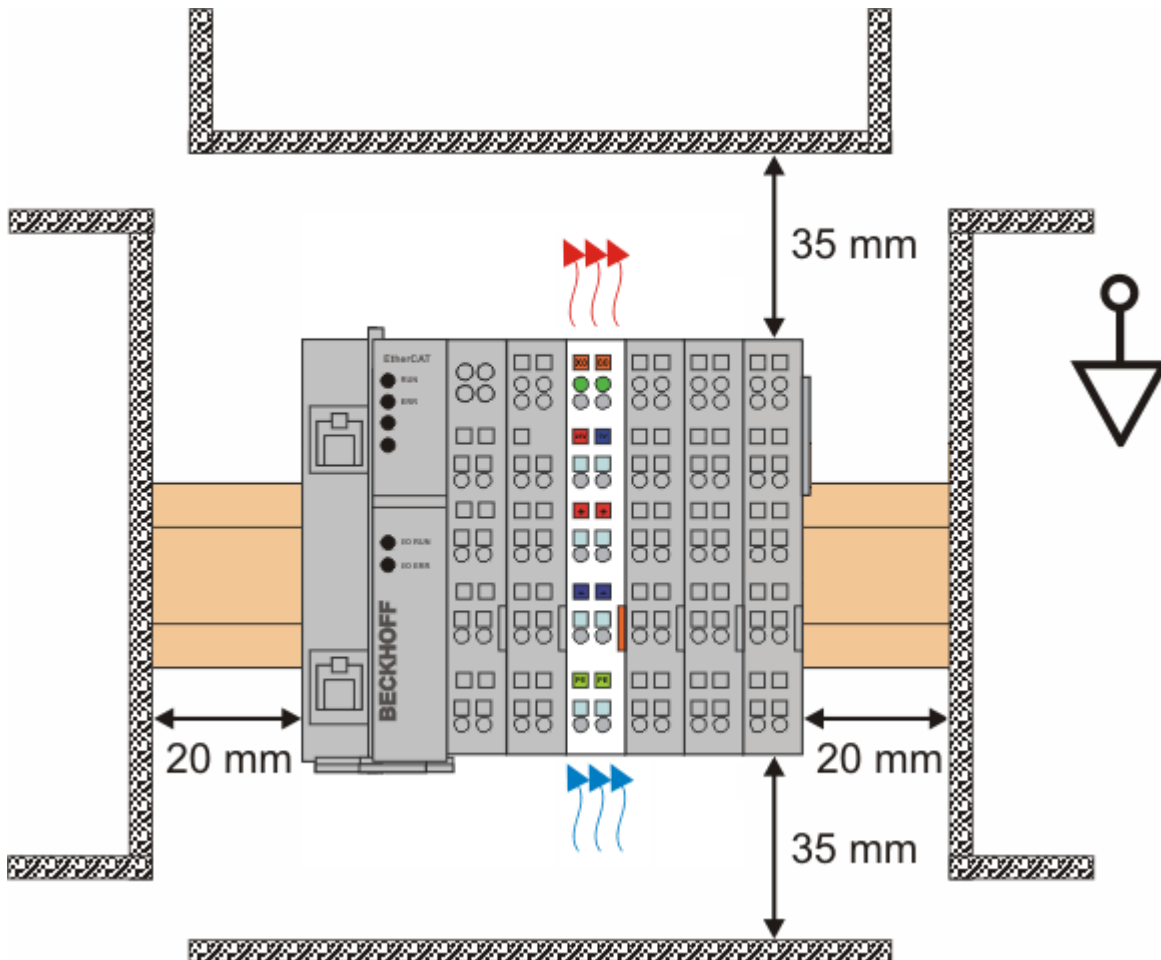


Fig. 17: Recommended distances for standard installation position

Compliance with the distances shown in Fig. "Recommended distances for standard installation position" is recommended.

#### Other installation positions

All other installation positions are characterized by different spatial arrangement of the mounting rail - see Fig "Other installation positions".

The minimum distances to ambient specified above also apply to these installation positions.

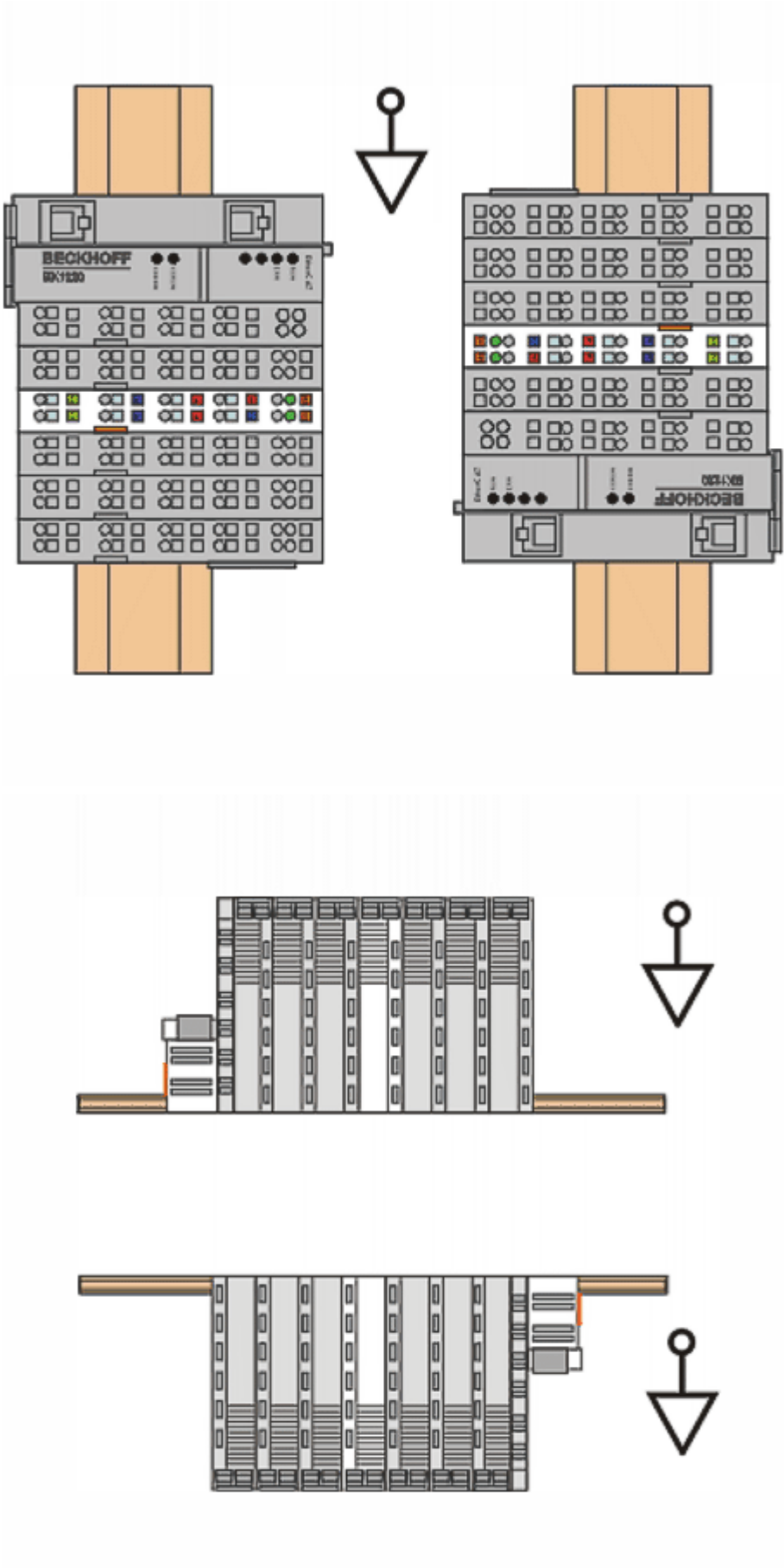


Fig. 18: Other installation positions

## 4.4 Positioning of passive Terminals

### **i** Hint for positioning of passive terminals in the bus terminal block

EtherCAT Terminals (ELxxxx / ESxxxx), which do not take an active part in data transfer within the bus terminal block are so called passive terminals. The passive terminals have no current consumption out of the E-Bus.

To ensure an optimal data transfer, you must not directly string together more than two passive terminals!

#### Examples for positioning of passive terminals (highlighted)

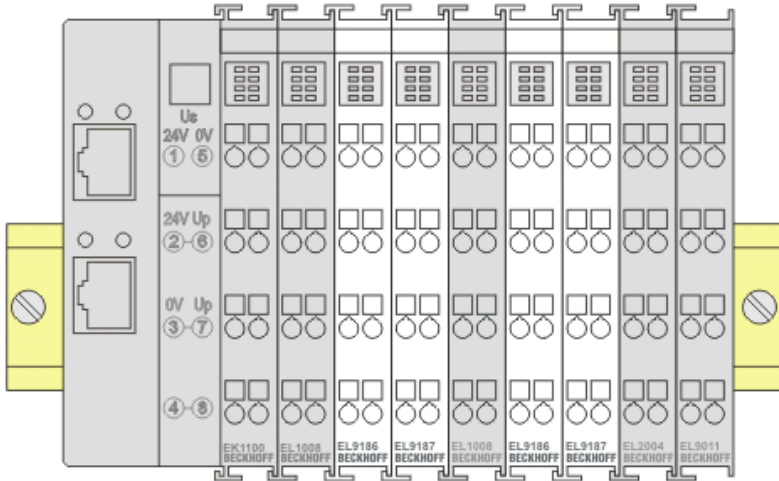


Fig. 19: Correct positioning

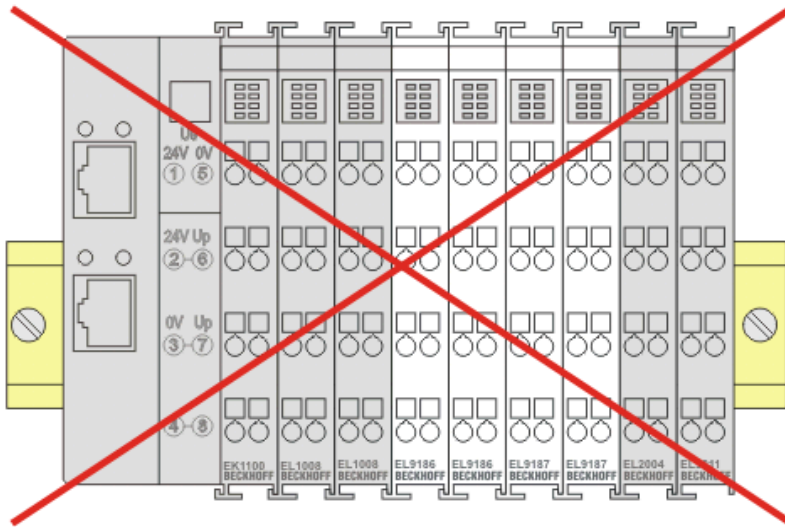


Fig. 20: Incorrect positioning

## 4.5 Connection

### 4.5.1 Connection system

#### ⚠ WARNING

##### **Risk of electric shock and damage of device!**

Bring the bus terminal system into a safe, powered down state before starting installation, disassembly or wiring of the bus terminals!

#### Overview

The bus terminal system offers different connection options for optimum adaptation to the respective application:

- The terminals of ELxxxx and KLxxxx series with standard wiring include electronics and connection level in a single enclosure.
- The terminals of ESxxxx and KSxxxx series feature a pluggable connection level and enable steady wiring while replacing.
- The High Density Terminals (HD Terminals) include electronics and connection level in a single enclosure and have advanced packaging density.

#### Standard wiring (ELxxxx / KLxxxx)



Fig. 21: Standard wiring

The terminals of the ELxxxx and KLxxxx series integrate screwless spring-cage technology for quick and easy wiring.

#### Pluggable wiring (ESxxxx / KSxxxx)



Fig. 22: Pluggable wiring

The terminals of ESxxxx and KSxxxx series feature a pluggable connection level. The assembly and wiring procedure is the same as for the ELxxxx and KLxxxx series. The pluggable connection level enables the complete wiring to be removed as a plug connector from the top of the housing for servicing. The lower section can be removed from the terminal block by pulling the unlocking tab. Insert the new component and plug in the connector with the wiring. This reduces the installation time and eliminates the risk of wires being mixed up.

The familiar dimensions of the terminal only had to be changed slightly. The new connector adds about 3 mm. The maximum height of the terminal remains unchanged.

A tab for strain relief of the cable simplifies assembly in many applications and prevents tangling of individual connection wires when the connector is removed.

Conductor cross sections between 0.08 mm<sup>2</sup> and 2.5 mm<sup>2</sup> can continue to be used with the proven spring force technology.

The overview and nomenclature of the product names for ESxxxx and KSxxxx series has been retained as known from ELxxxx and KLxxxx series.

### High Density Terminals (HD Terminals)



Fig. 23: High Density Terminals

The terminals from these series with 16/32 terminal points are distinguished by a particularly compact design, as the packaging density is twice as large as that of the standard 12 mm bus terminals. Massive conductors and conductors with a wire end sleeve can be inserted directly into the spring loaded terminal point without tools.



#### Wiring HD Terminals

The High Density Terminals of the ELx8xx and KLx8xx series doesn't support pluggable wiring.

### Ultrasonically compacted (ultrasonically welded) strands



#### Ultrasonically compacted (ultrasonically welded) strands

Ultrasonically compacted (ultrasonically welded) strands can also be connected to the standard and high-density terminals. In this case, please note the tables concerning the wire-size width [► 40]!

## 4.5.2 Wiring

### ⚠ WARNING

#### Risk of electric shock and damage of device!

Bring the bus terminal system into a safe, powered down state before starting installation, disassembly or wiring of the bus terminals!

Terminals for standard wiring ELxxxx/KLxxxx and for pluggable wiring ESxxxx/KSxxxx

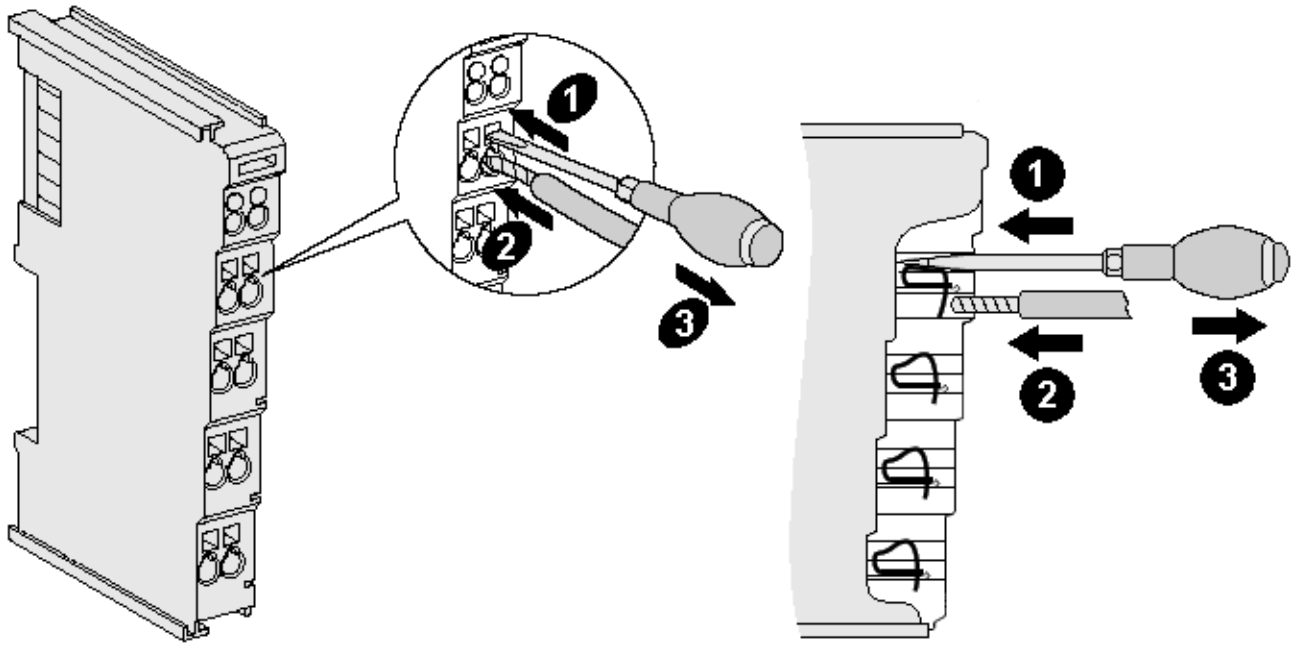


Fig. 24: Connecting a cable on a terminal point

Up to eight terminal points enable the connection of solid or finely stranded cables to the bus terminal. The terminal points are implemented in spring force technology. Connect the cables as follows (see fig. "Connecting a cable on a terminal point"):

1. Open a terminal point by pushing a screwdriver straight against the stop into the square opening above the terminal point. Do not turn the screwdriver or move it alternately (don't toggle).
2. The wire can now be inserted into the round terminal opening without any force.
3. When the screwdriver is removed, the terminal point closes automatically and holds the wire securely and permanently in place

See the following table for the suitable wire size width:

Terminal housing	ELxxxx, KLxxxx	ESxxxx, KSxxxx
Wire size width (single core wires)	0.08 ... 2.5 mm <sup>2</sup>	0.08 ... 2.5 mm <sup>2</sup>
Wire size width (fine-wire conductors)	0.08 ... 2.5 mm <sup>2</sup>	0.08 ... 2.5 mm <sup>2</sup>
Wire size width (conductors with a wire end sleeve)	0.14 ... 1.5 mm <sup>2</sup>	0.14 ... 1.5 mm <sup>2</sup>
Wire stripping length	8 ... 9 mm	9 ... 10 mm



### High Density Terminals (**HD Terminals** [► 38]) with 16/32 terminal points

The conductors of the HD Terminals are connected without tools for single-wire conductors using the direct plug-in technique, i.e. after stripping the wire is simply plugged into the terminal point. The cables are released, as usual, using the contact release with the aid of a screwdriver. See the following table for the suitable wire size width.

Terminal housing	High Density Housing
Wire size width (single core wires)	0.08 ... 1.5 mm <sup>2</sup>
Wire size width (fine-wire conductors)	0.25 ... 1.5 mm <sup>2</sup>
Wire size width (conductors with a wire end sleeve)	0.14 ... 0.75 mm <sup>2</sup>
Wire size width (ultrasonically compacted [ultrasonically welded] strands)	only 1.5 mm <sup>2</sup> (see <a href="#">notice</a> [► 38])
Wire stripping length	8 ... 9 mm

## 4.5.3 Shielding



### Shielding

Encoder, analog sensors and actuators should always be connected with shielded, twisted paired wires.

## 4.6 Note - power supply

### ⚠ WARNING

#### Power supply from SELV / PELV power supply unit!

SELV / PELV circuits (safety extra-low voltage / protective extra-low voltage) according to IEC 61010-2-201 must be used to supply this device.

Notes:

- SELV / PELV circuits may give rise to further requirements from standards such as IEC 60204-1 et al, for example with regard to cable spacing and insulation.
- A SELV supply provides safe electrical isolation and limitation of the voltage without a connection to the protective conductor, a PELV supply also requires a safe connection to the protective conductor.

## 4.7 Disposal



Products marked with a crossed-out wheeled bin shall not be discarded with the normal waste stream. The device is considered as waste electrical and electronic equipment. The national regulations for the disposal of waste electrical and electronic equipment must be observed.



## 5 Commissioning

### 5.1 TwinCAT basics

#### 5.1.1 TwinCAT Development Environment

The Software for automation TwinCAT (The Windows Control and Automation Technology) will be distinguished into:

- TwinCAT 2: System Manager (Configuration) & PLC Control (Programming)
- TwinCAT 3: Enhancement of TwinCAT 2 (Programming and Configuration takes place via a common Development Environment)

**Details:**

- **TwinCAT 2:**
  - Connects I/O devices to tasks in a variable-oriented manner
  - Connects tasks to tasks in a variable-oriented manner
  - Supports units at the bit level
  - Supports synchronous or asynchronous relationships
  - Exchange of consistent data areas and process images
  - Datalink on NT - Programs by open Microsoft Standards (OLE, OCX, ActiveX, DCOM+, etc.)
  - Integration of IEC 61131-3-Software-SPS, Software- NC and Software-CNC within Windows NT/ 2000/XP/Vista, Windows 7, NT/XP Embedded, CE
  - Interconnection to all common fieldbusses
  - [More...](#)

**Additional features:**

- **TwinCAT 3 (eXtended Automation):**
  - Visual Studio® integration
  - Choice of the programming language
  - Supports object orientated extension of IEC 61131-3
  - Usage of C/C++ as programming language for real time applications
  - Connection to MATLAB®/Simulink®
  - Open interface for expandability
  - Flexible run-time environment
  - Active support of multi-core- and 64 bit operating system
  - Automatic code generation and project creation with the TwinCAT Automation Interface
  - [More...](#)

Within the following sections commissioning of the TwinCAT Development Environment on a PC System for the control and also the basically functions of unique control elements will be explained.

Please see further information to TwinCAT 2 and TwinCAT 3 at <http://infosys.beckhoff.com>.

##### 5.1.1.1 Installation of the TwinCAT real-time driver

In order to assign real-time capability to a standard Ethernet port of an IPC controller, the Beckhoff real-time driver has to be installed on this port under Windows.

This can be done in several ways.

### A: Via the TwinCAT Adapter dialog

In the System Manager call up the TwinCAT overview of the local network interfaces via Options → Show Real Time Ethernet Compatible Devices.

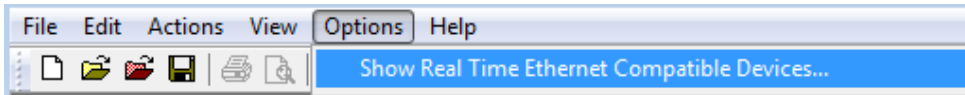


Fig. 25: System Manager “Options” (TwinCAT 2)

This has to be called up by the menu “TwinCAT” within the TwinCAT 3 environment:

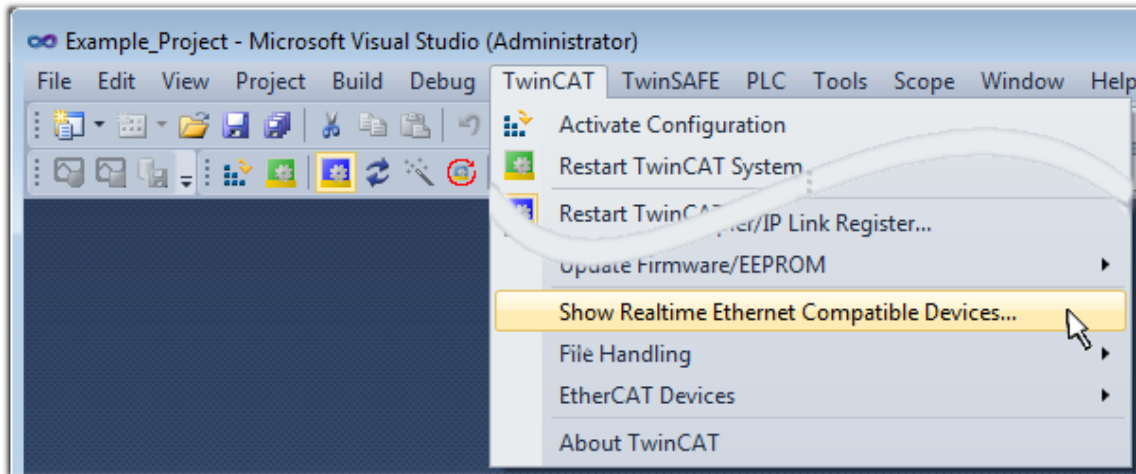


Fig. 26: Call up under VS Shell (TwinCAT 3)

### B: Via TcRteInstall.exe in the TwinCAT directory

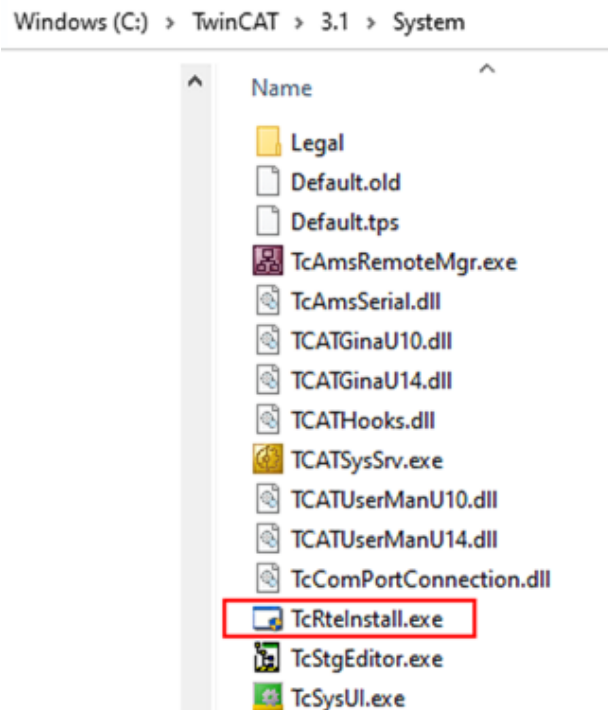


Fig. 27: TcRteInstall in the TwinCAT directory

In both cases, the following dialog appears:

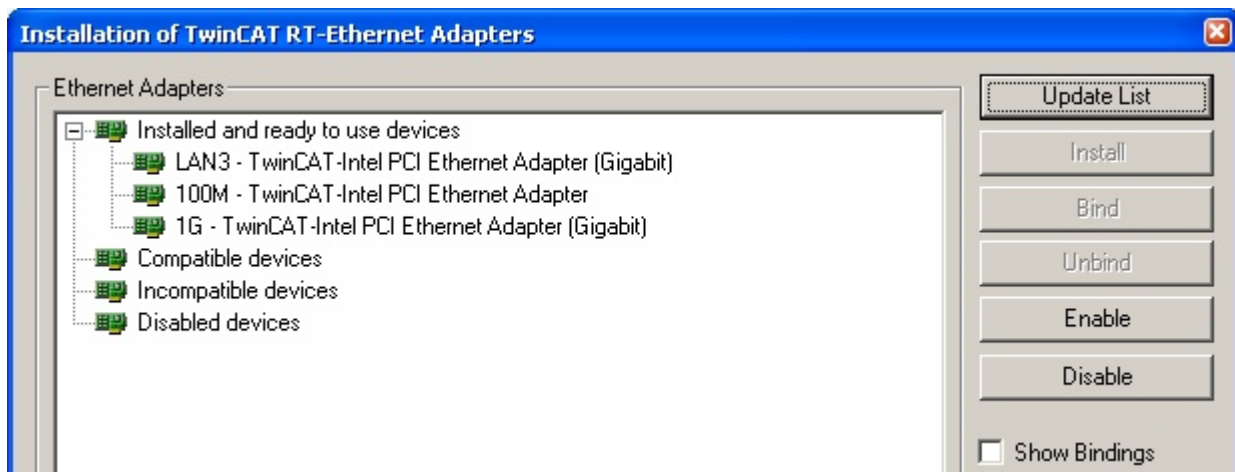


Fig. 28: Overview of network interfaces

Interfaces listed under “Compatible devices” can be assigned a driver via the “Install” button. A driver should only be installed on compatible devices.

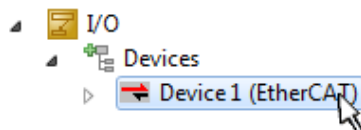
A Windows warning regarding the unsigned driver can be ignored.

**Alternatively** an EtherCAT-device can be inserted first of all as described in chapter [Offline configuration creation](#), section “Creating the EtherCAT device” [► 52] in order to view the compatible ethernet ports via its EtherCAT properties (tab “Adapter”, button “Compatible Devices...”):



Fig. 29: EtherCAT device properties (TwinCAT 2): click on “Compatible Devices...” of tab “Adapter”

TwinCAT 3: the properties of the EtherCAT device can be opened by double click on “Device .. (EtherCAT)” within the Solution Explorer under “I/O”:



After the installation the driver appears activated in the Windows overview for the network interface (Windows Start → System Properties → Network)

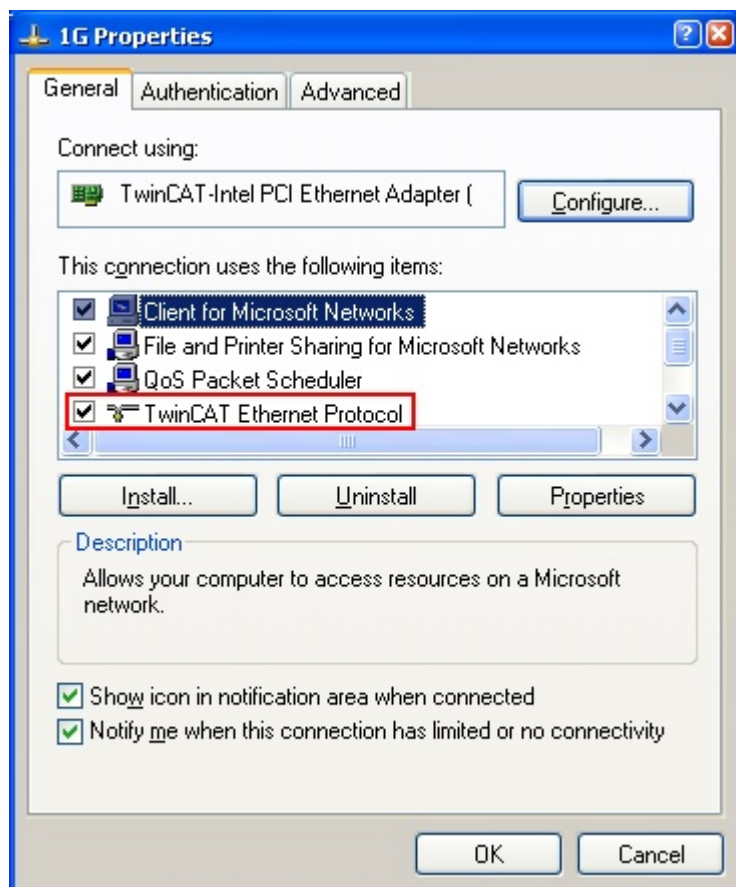


Fig. 30: Windows properties of the network interface

A correct setting of the driver could be:

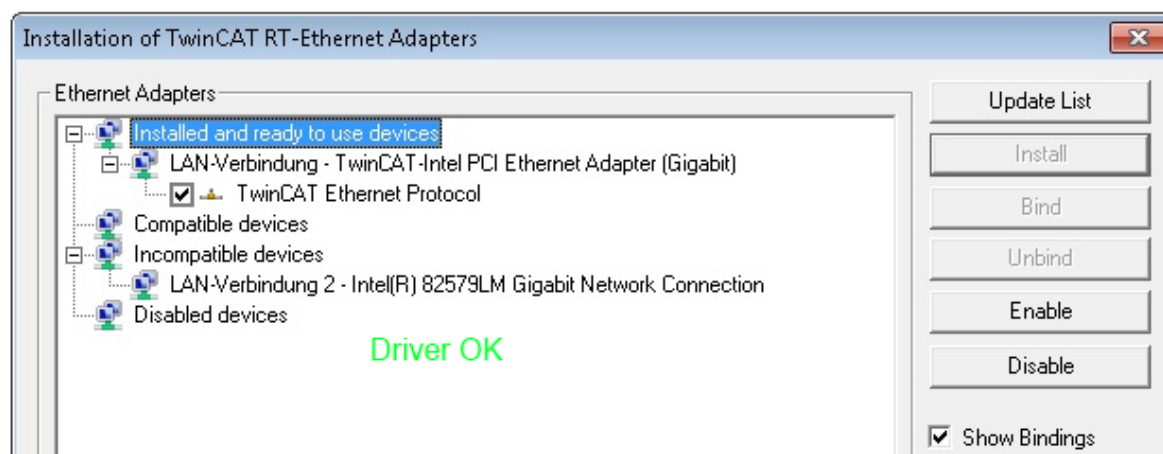


Fig. 31: Exemplary correct driver setting for the Ethernet port

Other possible settings have to be avoided:

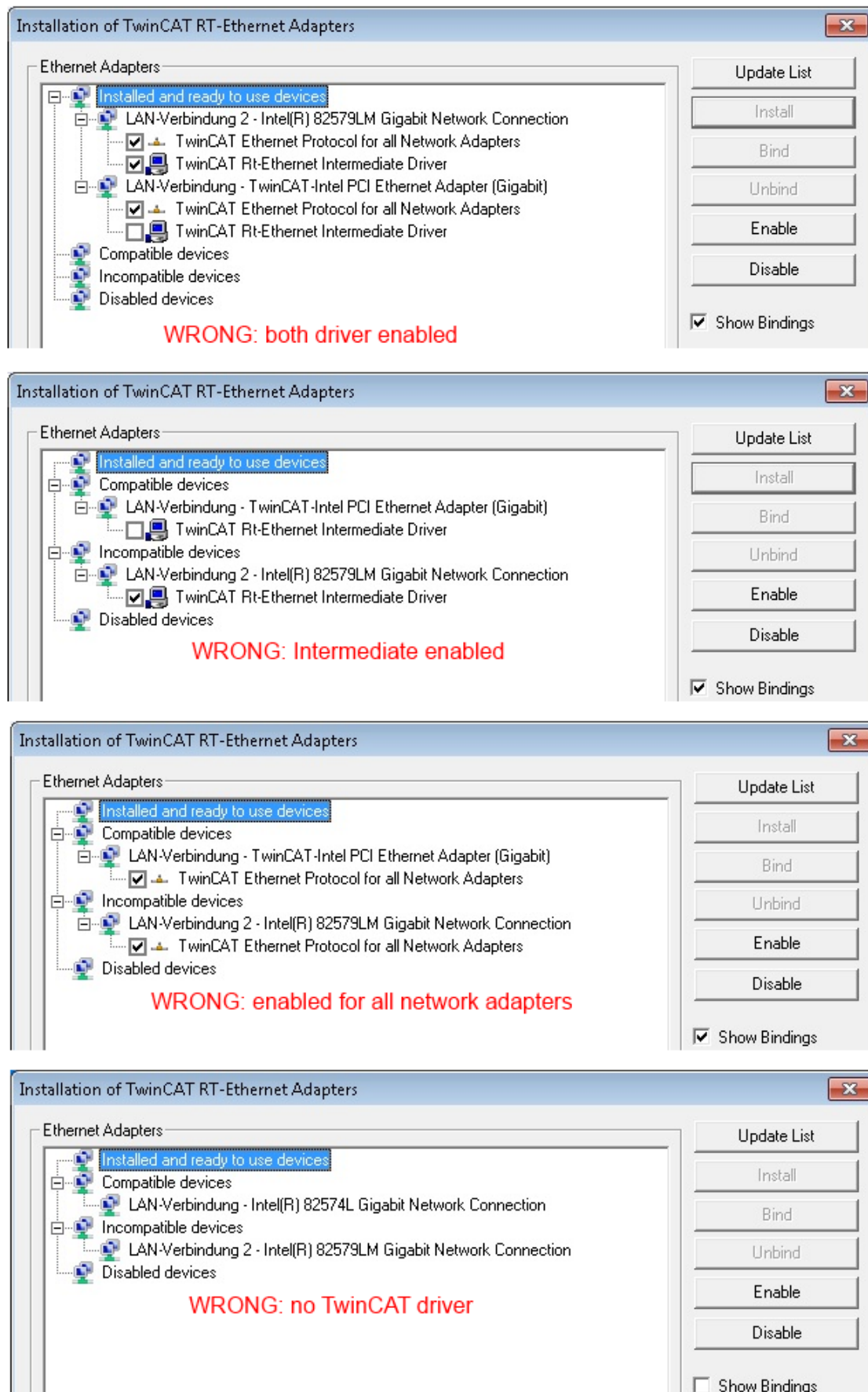


Fig. 32: Incorrect driver settings for the Ethernet port

## IP address of the port used



### IP address/DHCP

In most cases an Ethernet port that is configured as an EtherCAT device will not transport general IP packets. For this reason and in cases where an EL6601 or similar devices are used it is useful to specify a fixed IP address for this port via the "Internet Protocol TCP/IP" driver setting and to disable DHCP. In this way the delay associated with the DHCP client for the Ethernet port assigning itself a default IP address in the absence of a DHCP server is avoided. A suitable address space is 192.168.x.x, for example.

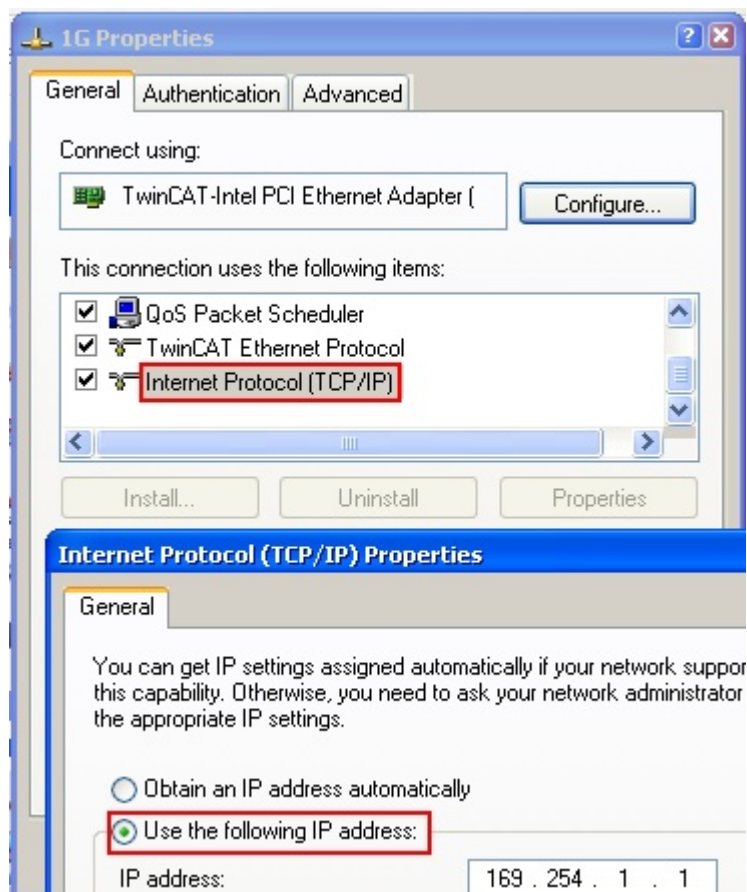


Fig. 33: TCP/IP setting for the Ethernet port



### 5.1.1.2 Notes regarding ESI device description

#### Installation of the latest ESI device description

The TwinCAT EtherCAT master/System Manager needs the device description files for the devices to be used in order to generate the configuration in online or offline mode. The device descriptions are contained in the so-called ESI files (EtherCAT Slave Information) in XML format. These files can be requested from the respective manufacturer and are made available for download. An \*.xml file may contain several device descriptions.

The ESI files for Beckhoff EtherCAT devices are available on the [Beckhoff website](#).

The ESI files should be stored in the TwinCAT installation directory.

Default settings:

- **TwinCAT 2:** C:\TwinCAT\IO\EtherCAT
- **TwinCAT 3:** C:\TwinCAT\3.1\Config\Io\EtherCAT

The files are read (once) when a new System Manager window is opened, if they have changed since the last time the System Manager window was opened.

A TwinCAT installation includes the set of Beckhoff ESI files that was current at the time when the TwinCAT build was created.

For TwinCAT 2.11/TwinCAT 3 and higher, the ESI directory can be updated from the System Manager, if the programming PC is connected to the Internet; by

- **TwinCAT 2:** Option → “Update EtherCAT Device Descriptions”
- **TwinCAT 3:** TwinCAT → EtherCAT Devices → “Update Device Descriptions (via ETG Website)...”

The [TwinCAT ESI Updater](#) [► 51] is available for this purpose.



#### ESI

The \*.xml files are associated with \*.xsd files, which describe the structure of the ESI XML files. To update the ESI device descriptions, both file types should therefore be updated.

#### Device differentiation

EtherCAT devices/slaves are distinguished by four properties, which determine the full device identifier. For example, the device identifier EL2521-0025-1018 consists of:

- family key “EL”
- name “2521”
- type “0025”
- and revision “1018”

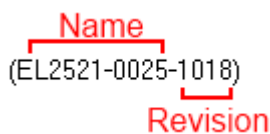


Fig. 34: Identifier structure

The order identifier consisting of name + type (here: EL2521-0025) describes the device function. The revision indicates the technical progress and is managed by Beckhoff. In principle, a device with a higher revision can replace a device with a lower revision, unless specified otherwise, e.g. in the documentation. Each revision has its own ESI description. See [further notes](#) [► 9].

## Online description

If the EtherCAT configuration is created online through scanning of real devices (see section Online setup) and no ESI descriptions are available for a slave (specified by name and revision) that was found, the System Manager asks whether the description stored in the device should be used. In any case, the System Manager needs this information for setting up the cyclic and acyclic communication with the slave correctly.

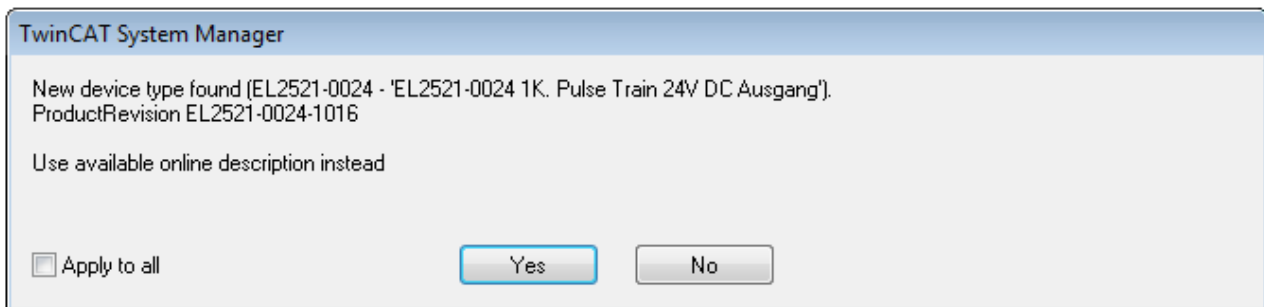


Fig. 35: OnlineDescription information window (TwinCAT 2)

In TwinCAT 3 a similar window appears, which also offers the Web update:

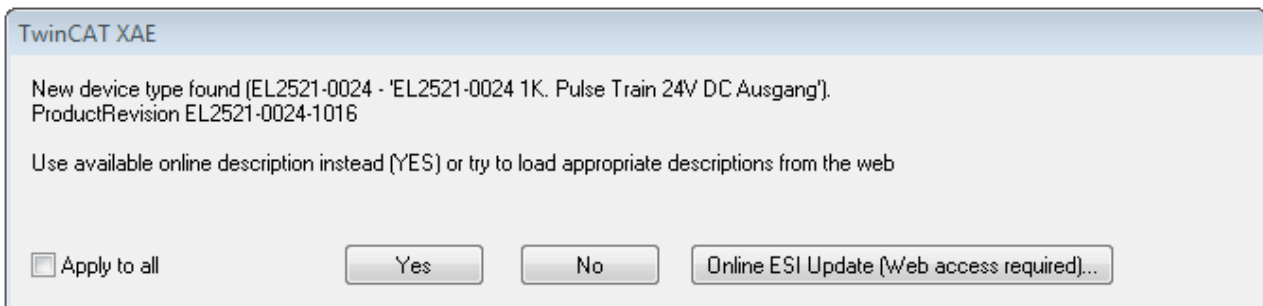


Fig. 36: Information window OnlineDescription (TwinCAT 3)

If possible, the Yes is to be rejected and the required ESI is to be requested from the device manufacturer. After installation of the XML/XSD file the configuration process should be repeated.

### NOTICE

#### Changing the “usual” configuration through a scan

- ✓ If a scan discovers a device that is not yet known to TwinCAT, distinction has to be made between two cases. Taking the example here of the EL2521-0000 in the revision 1019
  - a) no ESI is present for the EL2521-0000 device at all, either for the revision 1019 or for an older revision. The ESI must then be requested from the manufacturer (in this case Beckhoff).
  - b) an ESI is present for the EL2521-0000 device, but only in an older revision, e.g. 1018 or 1017. In this case an in-house check should first be performed to determine whether the spare parts stock allows the integration of the increased revision into the configuration at all. A new/higher revision usually also brings along new features. If these are not to be used, work can continue without reservations with the previous revision 1018 in the configuration. This is also stated by the Beckhoff compatibility rule.

Refer in particular to the chapter “General notes on the use of Beckhoff EtherCAT IO components” and for manual configuration to the chapter “Offline configuration creation [► 52]”.

If the OnlineDescription is used regardless, the System Manager reads a copy of the device description from the EEPROM in the EtherCAT slave. In complex slaves the size of the EEPROM may not be sufficient for the complete ESI, in which case the ESI would be *incomplete* in the configurator. Therefore it's recommended using an offline ESI file with priority in such a case.

The System Manager creates for online recorded device descriptions a new file “OnlineDescription0000...xml” in its ESI directory, which contains all ESI descriptions that were read online.



OnlineDescriptionCache000000002.xml

Fig. 37: File OnlineDescription.xml created by the System Manager

If a slave is desired to be added manually to the configuration at a later stage, online created slaves are indicated by a prepended symbol ">" in the selection list (see Figure *Indication of an online recorded ESI of EL2521 as an example*).

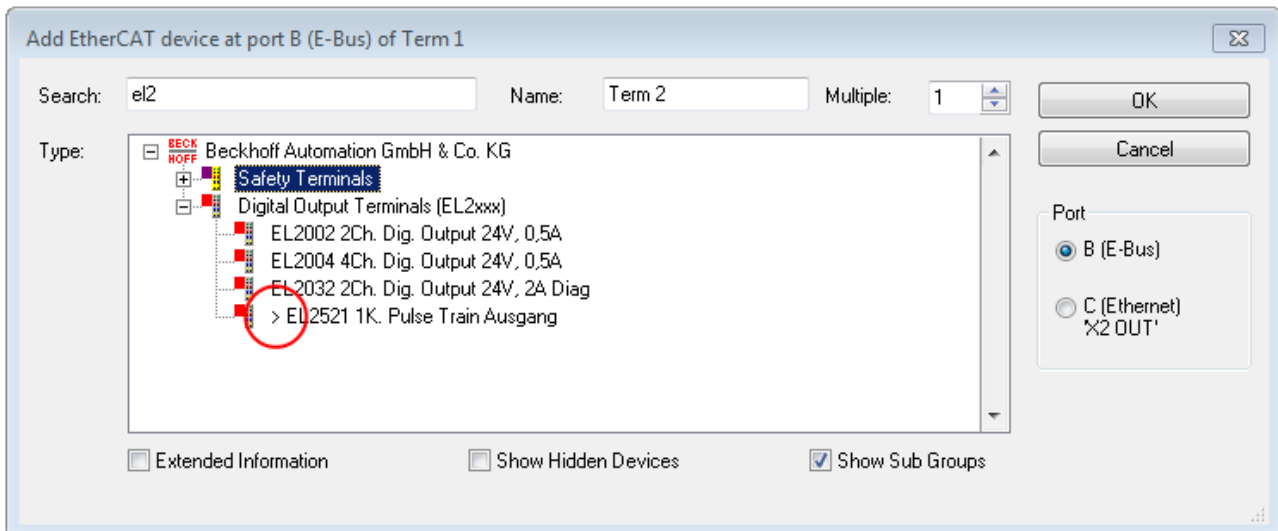


Fig. 38: Indication of an online recorded ESI of EL2521 as an example

If such ESI files are used and the manufacturer's files become available later, the file OnlineDescription.xml should be deleted as follows:

- close all System Manager windows
- restart TwinCAT in Config mode
- delete "OnlineDescription0000...xml"
- restart TwinCAT System Manager

This file should not be visible after this procedure, if necessary press <F5> to update

### **OnlineDescription for TwinCAT 3.x**

In addition to the file described above "OnlineDescription0000...xml", a so called EtherCAT cache with new discovered devices is created by TwinCAT 3.x, e.g. under Windows 7:

`C:\User\[USERNAME]\AppData\Roaming\Beckhoff\TwinCAT3\Components\Base\EtherCATCache.xml`

(Please note the language settings of the OS!)

You have to delete this file, too.

### **Faulty ESI file**

If an ESI file is faulty and the System Manager is unable to read it, the System Manager brings up an information window.

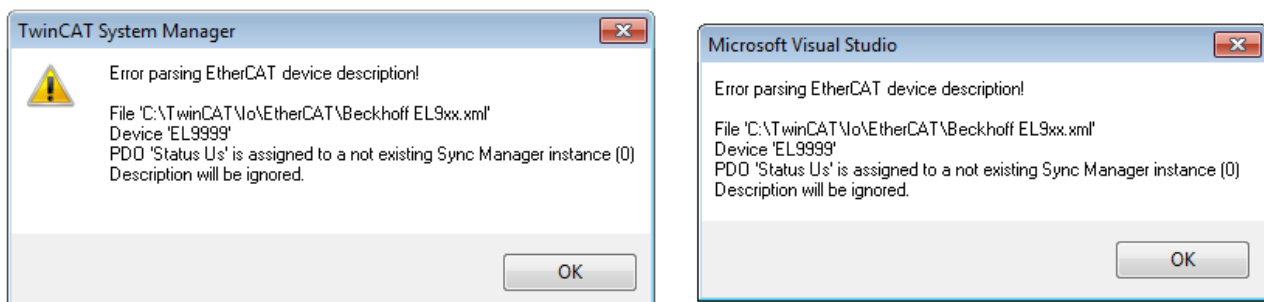


Fig. 39: Information window for faulty ESI file (left: TwinCAT 2; right: TwinCAT 3)

Reasons may include:

- Structure of the \*.xml does not correspond to the associated \*.xsd file → check your schematics
- Contents cannot be translated into a device description → contact the file manufacturer

### 5.1.1.3 TwinCAT ESI Updater

For TwinCAT 2.11 and higher, the System Manager can search for current Beckhoff ESI files automatically, if an online connection is available:

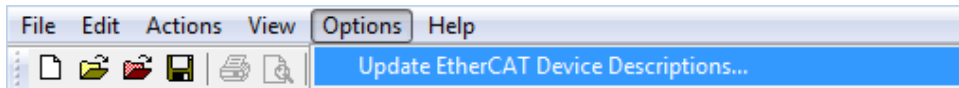


Fig. 40: Using the ESI Updater (>= TwinCAT 2.11)

The call up takes place under:

“Options” → “Update EtherCAT Device Descriptions”

Selection under TwinCAT 3:

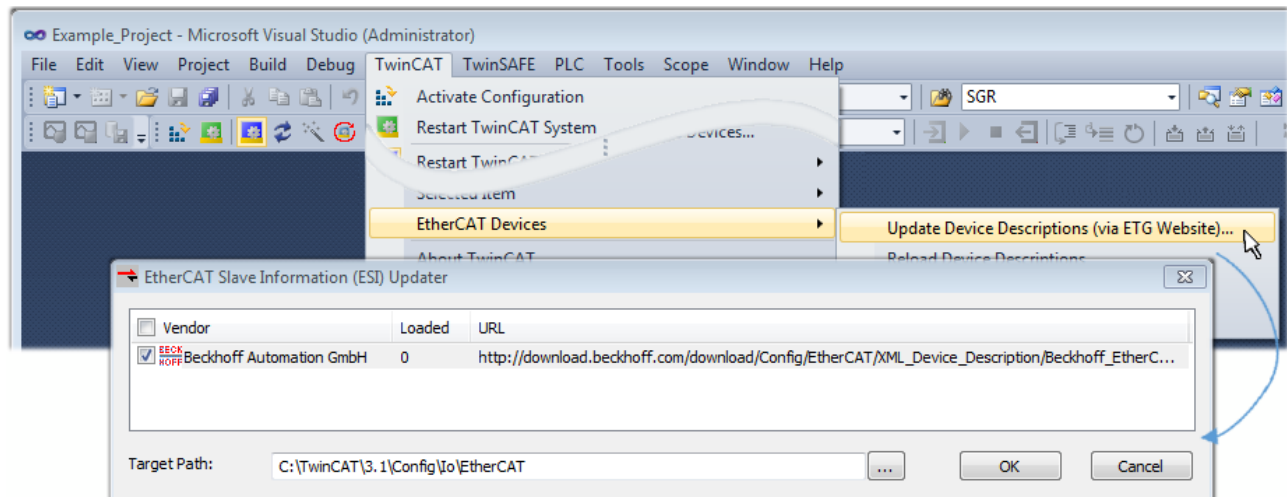


Fig. 41: Using the ESI Updater (TwinCAT 3)

The ESI Updater (TwinCAT 3) is a convenient option for automatic downloading of ESI data provided by EtherCAT manufacturers via the Internet into the TwinCAT directory (ESI = EtherCAT slave information). TwinCAT accesses the central ESI ULR directory list stored at ETG; the entries can then be viewed in the Updater dialog, although they cannot be changed there.

The call up takes place under:

“TwinCAT” → “EtherCAT Devices” → “Update Device Description (via ETG Website)...”.

### 5.1.1.4 Distinction between Online and Offline

The distinction between online and offline refers to the presence of the actual I/O environment (drives, terminals, EJ-modules). If the configuration is to be prepared in advance of the system configuration as a programming system, e.g. on a laptop, this is only possible in “Offline configuration” mode. In this case all components have to be entered manually in the configuration, e.g. based on the electrical design.

If the designed control system is already connected to the EtherCAT system and all components are energised and the infrastructure is ready for operation, the TwinCAT configuration can simply be generated through “scanning” from the runtime system. This is referred to as online configuration.

In any case, during each startup the EtherCAT master checks whether the slaves it finds match the configuration. This test can be parameterised in the extended slave settings. Refer to note “Installation of the latest ESI-XML device description” [► 47].

#### For preparation of a configuration:

- the real EtherCAT hardware (devices, couplers, drives) must be present and installed
- the devices/modules must be connected via EtherCAT cables or in the terminal/ module strand in the same way as they are intended to be used later
- the devices/modules be connected to the power supply and ready for communication

- TwinCAT must be in CONFIG mode on the target system.

#### The online scan process consists of:

- detecting the EtherCAT device [► 57] (Ethernet port at the IPC)
- detecting the connected EtherCAT devices [► 58]. This step can be carried out independent of the preceding step
- troubleshooting [► 61]

The scan with existing configuration [► 62] can also be carried out for comparison.

### 5.1.1.5 OFFLINE configuration creation

#### Creating the EtherCAT device

Create an EtherCAT device in an empty System Manager window.

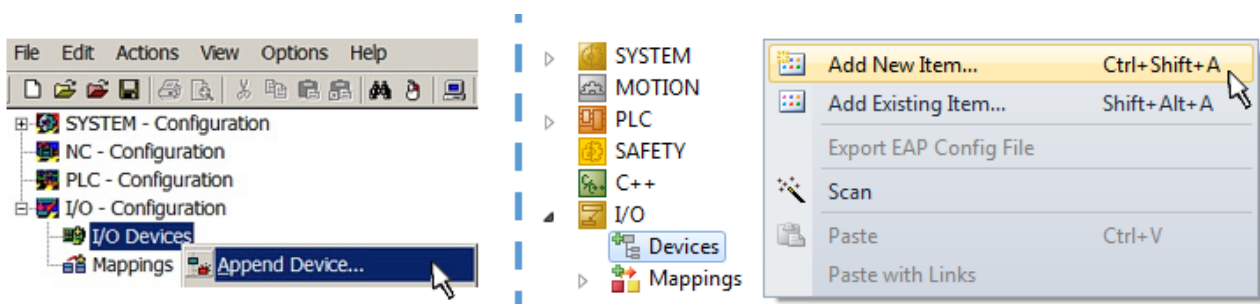


Fig. 42: Append EtherCAT device (left: TwinCAT 2; right: TwinCAT 3)

Select type “EtherCAT” for an EtherCAT I/O application with EtherCAT slaves. For the present publisher/subscriber service in combination with an EL6601/EL6614 terminal select “EtherCAT Automation Protocol via EL6601”.

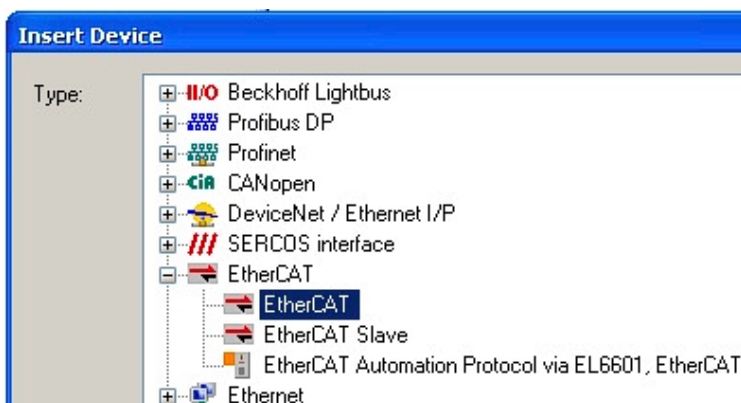


Fig. 43: Selecting the EtherCAT connection (TwinCAT 2.11, TwinCAT 3)

Then assign a real Ethernet port to this virtual device in the runtime system.

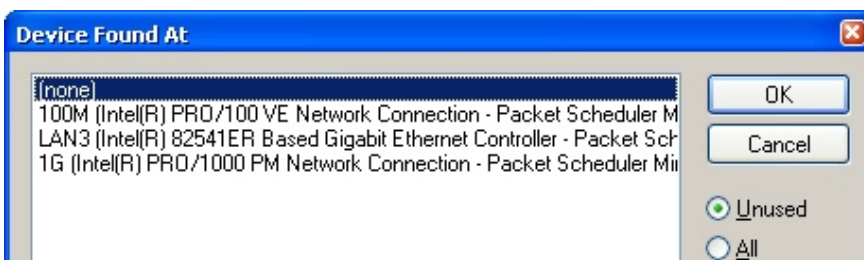


Fig. 44: Selecting the Ethernet port

This query may appear automatically when the EtherCAT device is created, or the assignment can be set/modified later in the properties dialog; see Fig. “EtherCAT device properties (TwinCAT 2)”.

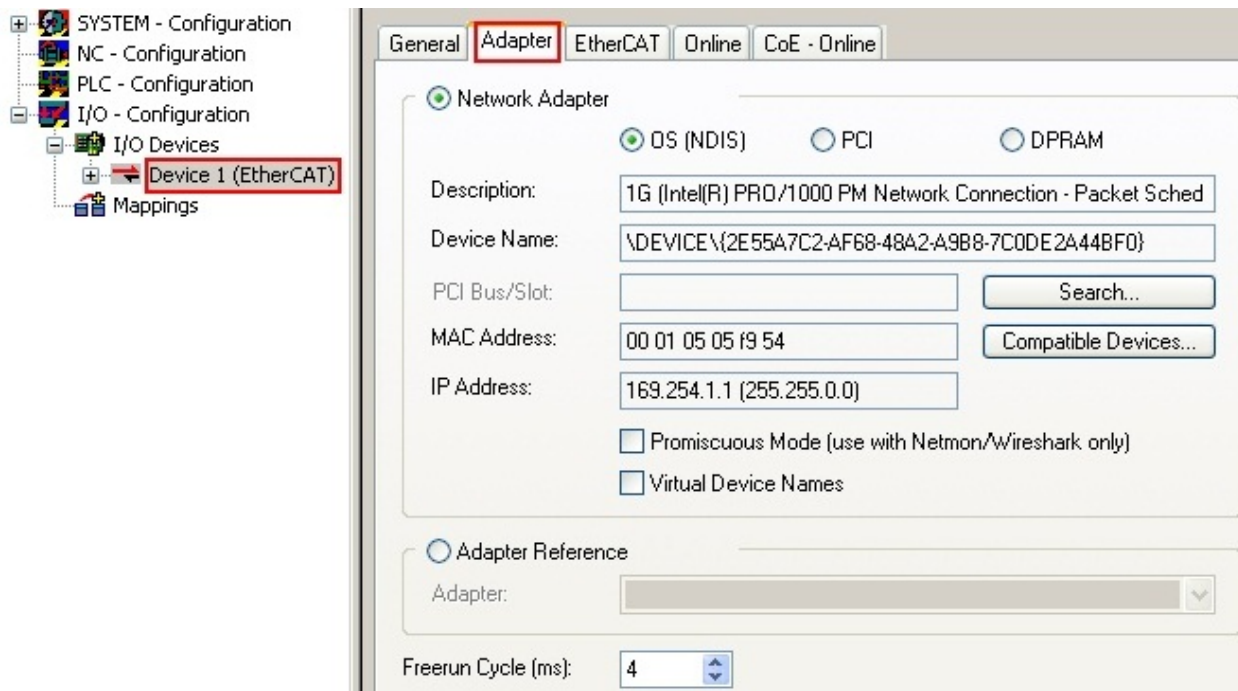
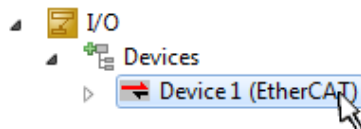


Fig. 45: EtherCAT device properties (TwinCAT 2)

TwinCAT 3: the properties of the EtherCAT device can be opened by double click on “Device .. (EtherCAT)” within the Solution Explorer under “I/O”:



## ● Selecting the Ethernet port

**i** Ethernet ports can only be selected for EtherCAT devices for which the TwinCAT real-time driver is installed. This has to be done separately for each port. Please refer to the respective [installation page \[► 41\]](#).

## Defining EtherCAT slaves

Further devices can be appended by right-clicking on a device in the configuration tree.

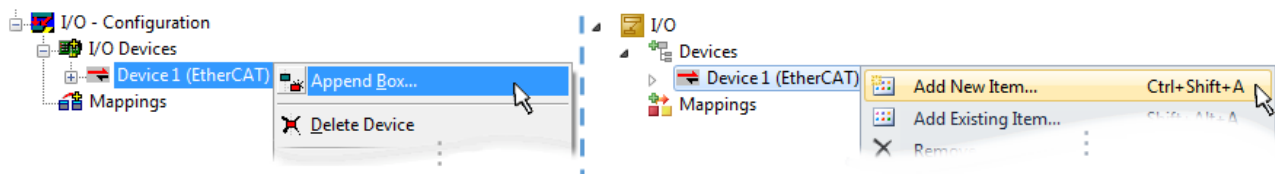


Fig. 46: Appending EtherCAT devices (left: TwinCAT 2; right: TwinCAT 3)

The dialog for selecting a new device opens. Only devices for which ESI files are available are displayed.

Only devices are offered for selection that can be appended to the previously selected device. Therefore, the physical layer available for this port is also displayed (Fig. “Selection dialog for new EtherCAT device”, A). In the case of cable-based Fast-Ethernet physical layer with PHY transfer, then also only cable-based devices are available, as shown in Fig. “Selection dialog for new EtherCAT device”. If the preceding device has several free ports (e.g. EK1122 or EK1100), the required port can be selected on the right-hand side (A).

## Overview of physical layer

- “Ethernet”: cable-based 100BASE-TX: couplers, box modules, devices with RJ45/M8/M12 connector

- “E-Bus”: LVDS “terminal bus”, EtherCAT plug-in modules (EJ), EtherCAT terminals (EL/ES), various modular modules

The search field facilitates finding specific devices (since TwinCAT 2.11 or TwinCAT 3).

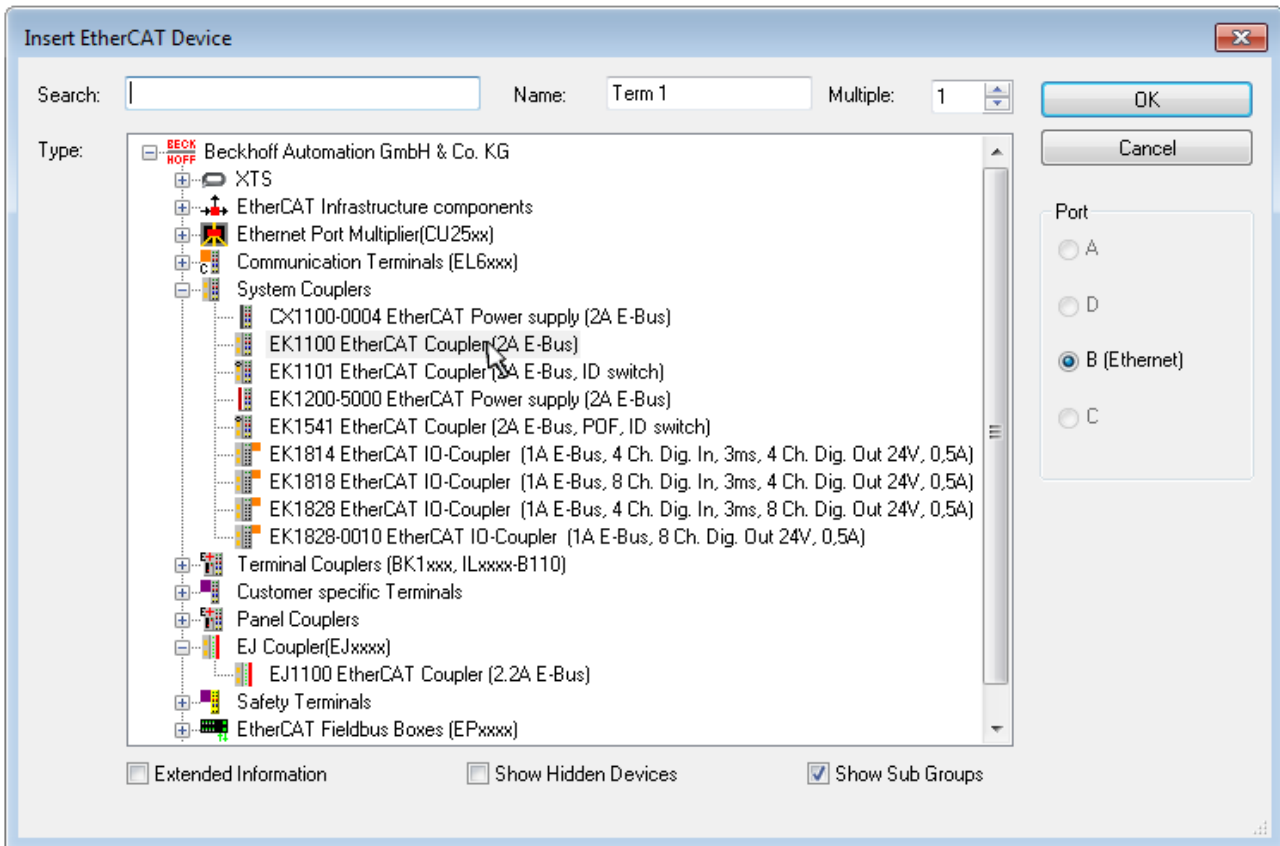


Fig. 47: Selection dialog for new EtherCAT device

By default, only the name/device type is used as selection criterion. For selecting a specific revision of the device, the revision can be displayed as “Extended Information”.

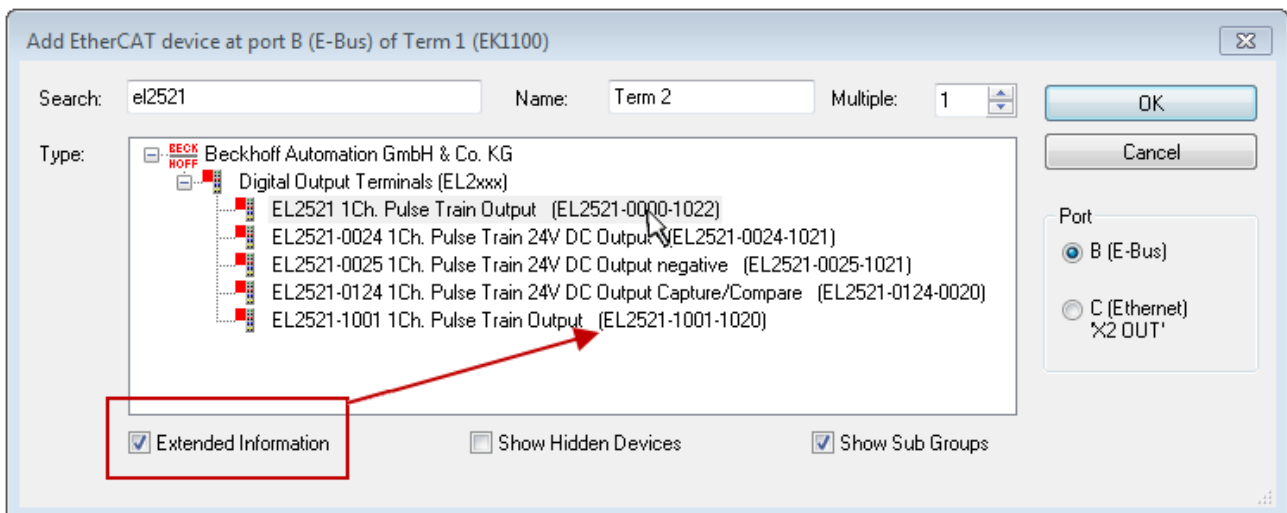


Fig. 48: Display of device revision

In many cases several device revisions were created for historic or functional reasons, e.g. through technological advancement. For simplification purposes (see Fig. “Selection dialog for new EtherCAT device”) only the last (i.e. highest) revision and therefore the latest state of production is displayed in the selection dialog for Beckhoff devices. To show all device revisions available in the system as ESI descriptions tick the “Show Hidden Devices” check box, see Fig. “Display of previous revisions”.

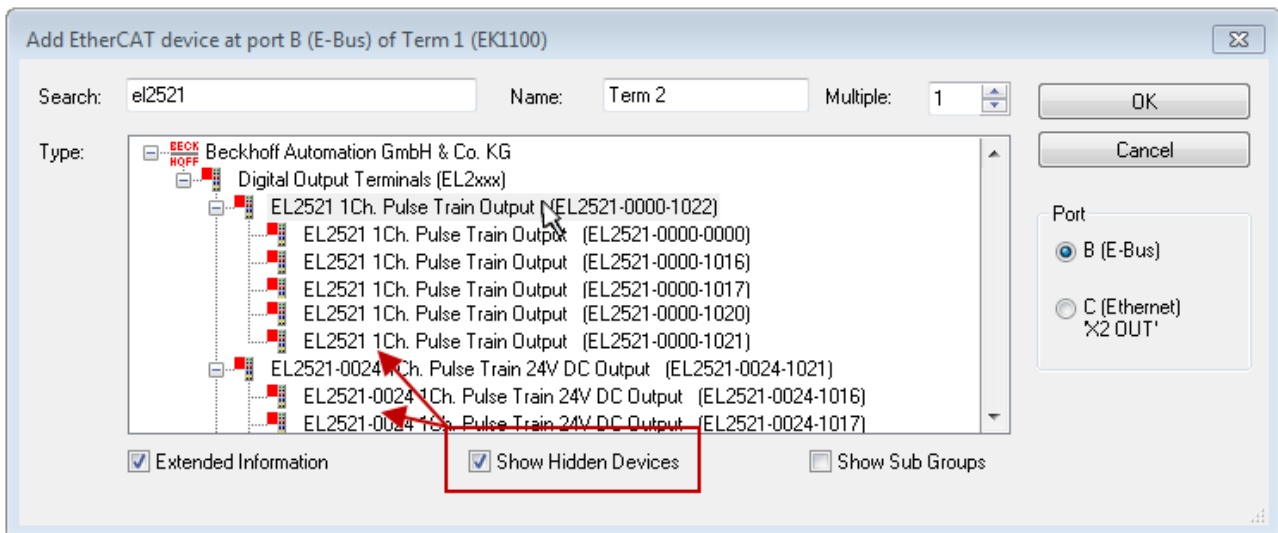


Fig. 49: Display of previous revisions

### ● Device selection based on revision, compatibility

**i** The ESI description also defines the process image, the communication type between master and slave/device and the device functions, if applicable. The physical device (firmware, if available) has to support the communication queries/settings of the master. This is backward compatible, i.e. newer devices (higher revision) should be supported if the EtherCAT master addresses them as an older revision. The following compatibility rule of thumb is to be assumed for Beckhoff EtherCAT Terminals/ Boxes/ EJ-modules:

#### **device revision in the system $\geq$ device revision in the configuration**

This also enables subsequent replacement of devices without changing the configuration (different specifications are possible for drives).

### Example

If an EL2521-0025-**1018** is specified in the configuration, an EL2521-0025-**1018** or higher (**-1019**, **-1020**) can be used in practice.

**Name**  
(EL2521-0025-1018)  
**Revision**

Fig. 50: Name/revision of the terminal

If current ESI descriptions are available in the TwinCAT system, the last revision offered in the selection dialog matches the Beckhoff state of production. It is recommended to use the last device revision when creating a new configuration, if current Beckhoff devices are used in the real application. Older revisions should only be used if older devices from stock are to be used in the application.

In this case the process image of the device is shown in the configuration tree and can be parameterized as follows: linking with the task, CoE/DC settings, plug-in definition, startup settings, ...



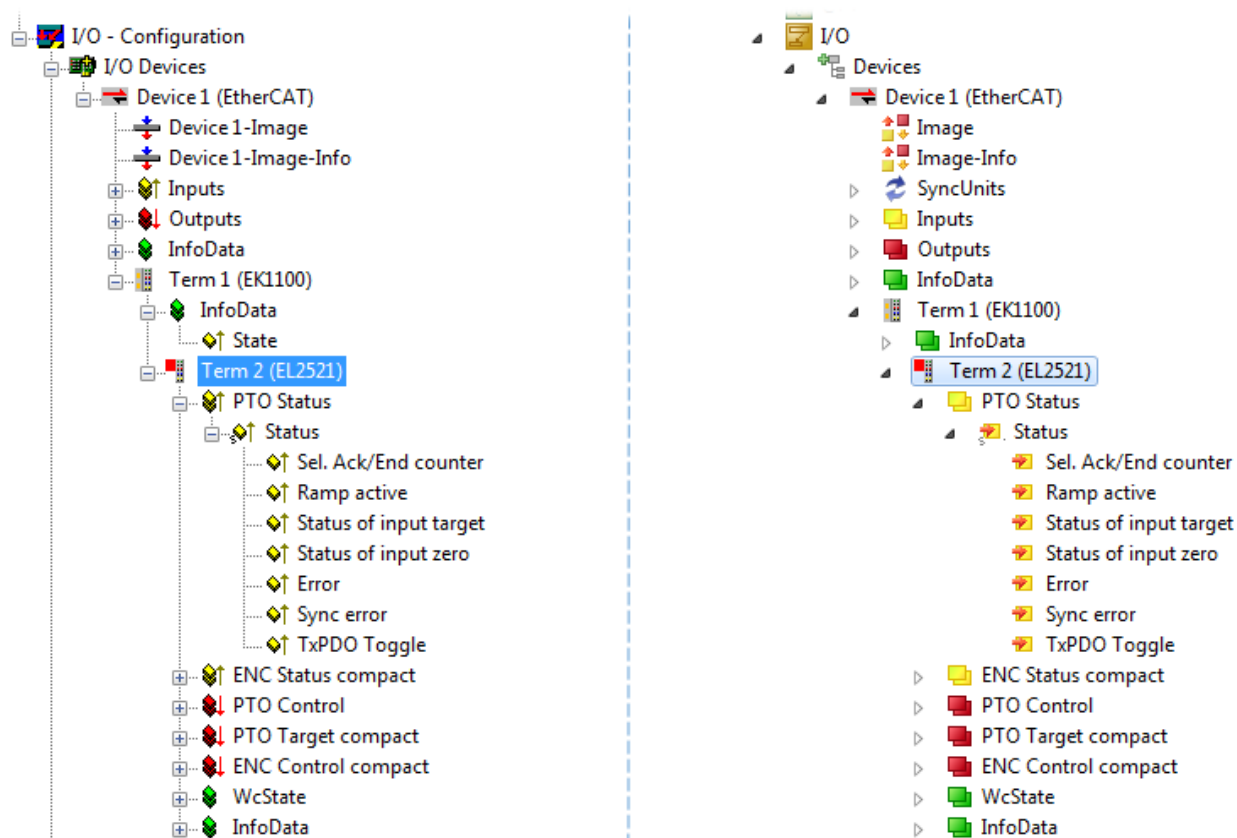




Fig. 51: EtherCAT terminal in the TwinCAT tree (left: TwinCAT 2; right: TwinCAT 3)





### 5.1.1.6 ONLINE configuration creation

#### Detecting/scanning of the EtherCAT device

The online device search can be used if the TwinCAT system is in CONFIG mode. This can be indicated by a symbol right below in the information bar:



- on TwinCAT 2 by a blue display “Config Mode” within the System Manager window:  .
- on TwinCAT 3 within the user interface of the development environment by a symbol  .

TwinCAT can be set into this mode:

- TwinCAT 2: by selection of  in the Menubar or by “Actions” → “Set/Reset TwinCAT to Config Mode...”
- TwinCAT 3: by selection of  in the Menubar or by “TwinCAT” → “Restart TwinCAT (Config Mode)”

#### **i** Online scanning in Config mode

The online search is not available in RUN mode (production operation). Note the differentiation between TwinCAT programming system and TwinCAT target system.

The TwinCAT 2 icon (  ) or TwinCAT 3 icon (  ) within the Windows-Taskbar always shows the TwinCAT mode of the local IPC. Compared to that, the System Manager window of TwinCAT 2 or the user interface of TwinCAT 3 indicates the state of the target system.

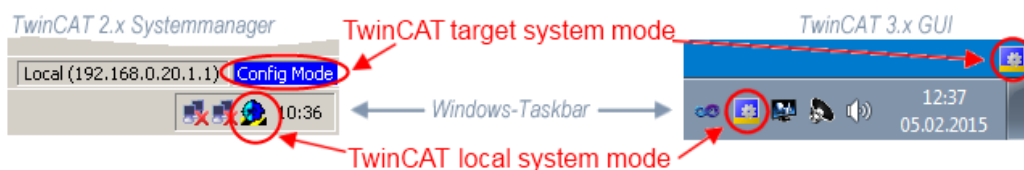


Fig. 52: Differentiation local/target system (left: TwinCAT 2; right: TwinCAT 3)

Right-clicking on “I/O Devices” in the configuration tree opens the search dialog.

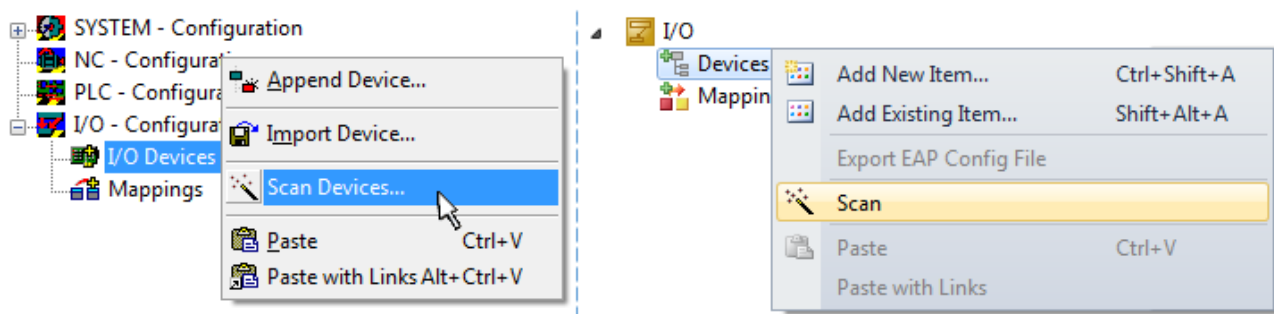


Fig. 53: Scan Devices (left: TwinCAT 2; right: TwinCAT 3)

This scan mode attempts to find not only EtherCAT devices (or Ethernet ports that are usable as such), but also NOVRA, fieldbus cards, SMB etc. However, not all devices can be found automatically.

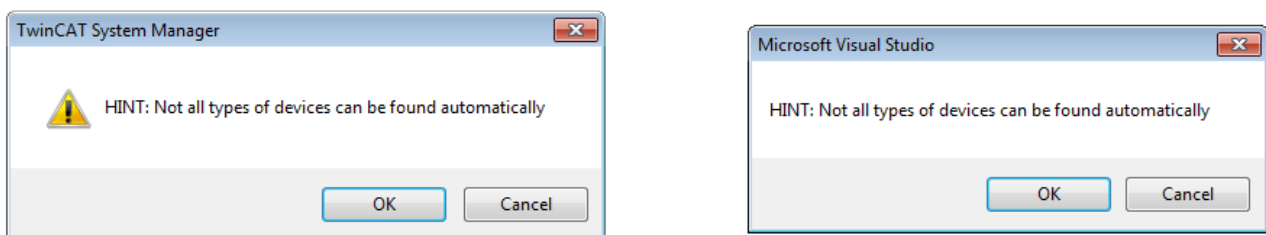


Fig. 54: Note for automatic device scan (left: TwinCAT 2; right: TwinCAT 3)

Ethernet ports with installed TwinCAT real-time driver are shown as “RT Ethernet” devices. An EtherCAT frame is sent to these ports for testing purposes. If the scan agent detects from the response that an EtherCAT slave is connected, the port is immediately shown as an “EtherCAT Device”.

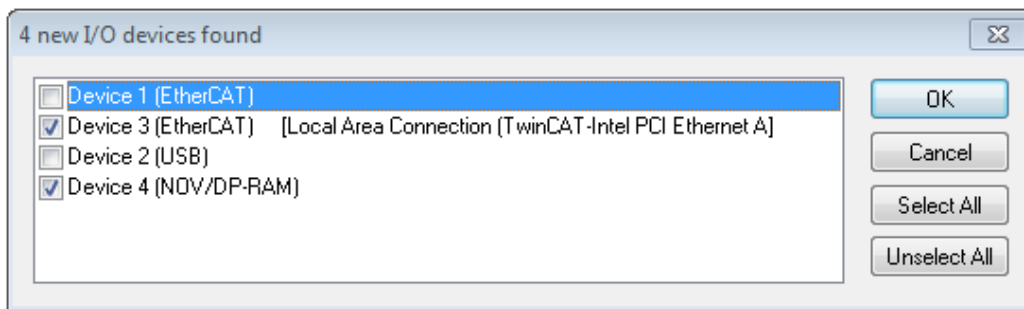


Fig. 55: Detected Ethernet devices

Via respective checkboxes devices can be selected (as illustrated in Fig. “Detected Ethernet devices” e.g. Device 3 and Device 4 were chosen). After confirmation with “OK” a device scan is suggested for all selected devices, see Fig.: “Scan query after automatic creation of an EtherCAT device”.

### ● Selecting the Ethernet port



Ethernet ports can only be selected for EtherCAT devices for which the TwinCAT real-time driver is installed. This has to be done separately for each port. Please refer to the respective [installation page](#) [► 41].

## Detecting/Scanning the EtherCAT devices

### ● Online scan functionality



During a scan the master queries the identity information of the EtherCAT slaves from the slave EEPROM. The name and revision are used for determining the type. The respective devices are located in the stored ESI data and integrated in the configuration tree in the default state defined there.

**Name**  
(EL2521-0025-1018)  
**Revision**

Fig. 56: Example default state

## NOTICE

### Slave scanning in practice in series machine production

The scanning function should be used with care. It is a practical and fast tool for creating an initial configuration as a basis for commissioning. In series machine production or reproduction of the plant, however, the function should no longer be used for the creation of the configuration, but if necessary for [comparison](#) [► 62] with the defined initial configuration. Background: since Beckhoff occasionally increases the revision version of the delivered products for product maintenance reasons, a configuration can be created by such a scan which (with an identical machine construction) is identical according to the device list; however, the respective device revision may differ from the initial configuration.

### Example:

Company A builds the prototype of a machine B, which is to be produced in series later on. To do this the prototype is built, a scan of the IO devices is performed in TwinCAT and the initial configuration “B.tsm” is created. The EL2521-0025 EtherCAT terminal with the revision 1018 is located somewhere. It is thus built into the TwinCAT configuration in this way:

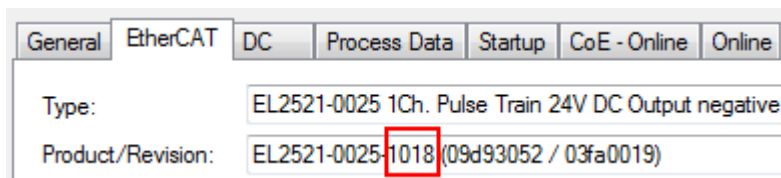


Fig. 57: Installing EtherCAT terminal with revision -1018

Likewise, during the prototype test phase, the functions and properties of this terminal are tested by the programmers/commissioning engineers and used if necessary, i.e. addressed from the PLC “B.pro” or the NC. (the same applies correspondingly to the TwinCAT 3 solution files).

The prototype development is now completed and series production of machine B starts, for which Beckhoff continues to supply the EL2521-0025-0018. If the commissioning engineers of the series machine production department always carry out a scan, a B configuration with the identical contents results again for each machine. Likewise, A might create spare parts stores worldwide for the coming series-produced machines with EL2521-0025-1018 terminals.

After some time Beckhoff extends the EL2521-0025 by a new feature C. Therefore the FW is changed, outwardly recognizable by a higher FW version and a **new revision -1019**. Nevertheless the new device naturally supports functions and interfaces of the predecessor version(s); an adaptation of “B.tsm” or even “B.pro” is therefore unnecessary. The series-produced machines can continue to be built with “B.tsm” and “B.pro”; it makes sense to perform a comparative scan [► 62] against the initial configuration “B.tsm” in order to check the built machine.

However, if the series machine production department now doesn't use “B.tsm”, but instead carries out a scan to create the productive configuration, the revision **-1019** is automatically detected and built into the configuration:

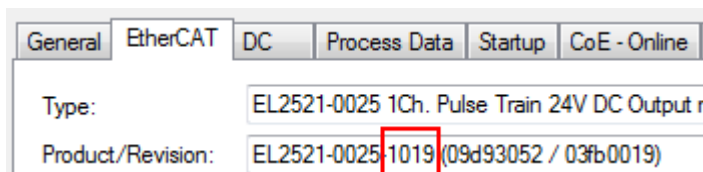


Fig. 58: Detection of EtherCAT terminal with revision -1019

This is usually not noticed by the commissioning engineers. TwinCAT cannot signal anything either, since a new configuration is essentially created. According to the compatibility rule, however, this means that no EL2521-0025-**1018** should be built into this machine as a spare part (even if this nevertheless works in the vast majority of cases).

In addition, it could be the case that, due to the development accompanying production in company A, the new feature C of the EL2521-0025-1019 (for example, an improved analog filter or an additional process data for the diagnosis) is discovered and used without in-house consultation. The previous stock of spare part devices are then no longer to be used for the new configuration “B2.tsm” created in this way. If series machine production is established, the scan should only be performed for informative purposes for comparison with a defined initial configuration. Changes are to be made with care!

If an EtherCAT device was created in the configuration (manually or through a scan), the I/O field can be scanned for devices/slaves.

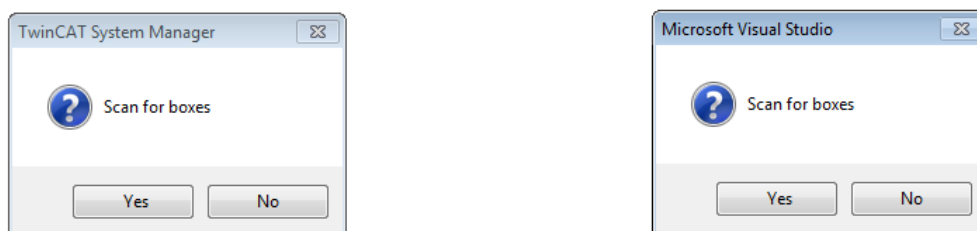


Fig. 59: Scan query after automatic creation of an EtherCAT device (left: TwinCAT 2; right: TwinCAT 3)

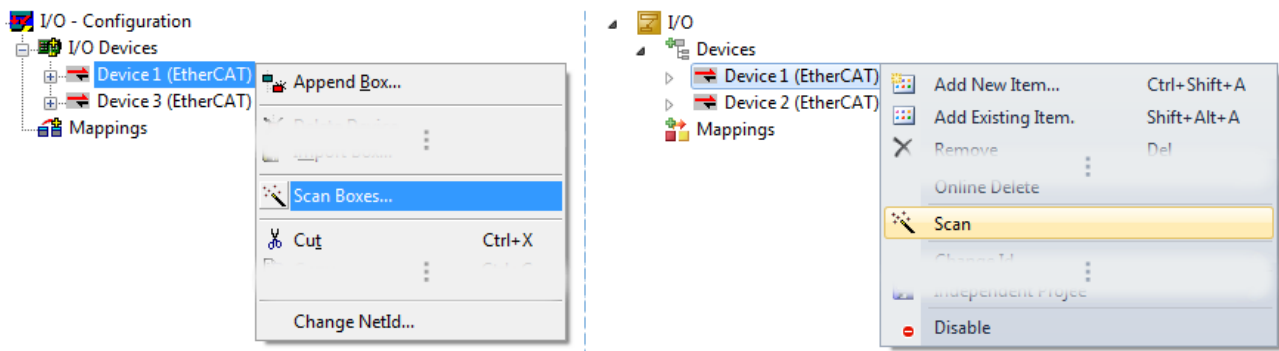


Fig. 60: Manual scanning for devices on a specified EtherCAT device (left: TwinCAT 2; right: TwinCAT 3)

In the System Manager (TwinCAT 2) or the User Interface (TwinCAT 3) the scan process can be monitored via the progress bar at the bottom in the status bar.



Fig. 61: Scan progress example by TwinCAT 2

The configuration is established and can then be switched to online state (OPERATIONAL).



Fig. 62: Config/FreeRun query (left: TwinCAT 2; right: TwinCAT 3)

In Config/FreeRun mode the System Manager display alternates between blue and red, and the EtherCAT device continues to operate with the idling cycle time of 4 ms (default setting), even without active task (NC, PLC).



Fig. 63: Displaying of "Free Run" and "Config Mode" toggling right below in the status bar



Fig. 64: TwinCAT can also be switched to this state by using a button (left: TwinCAT 2; right: TwinCAT 3)

The EtherCAT system should then be in a functional cyclic state, as shown in Fig. *Online display example*.

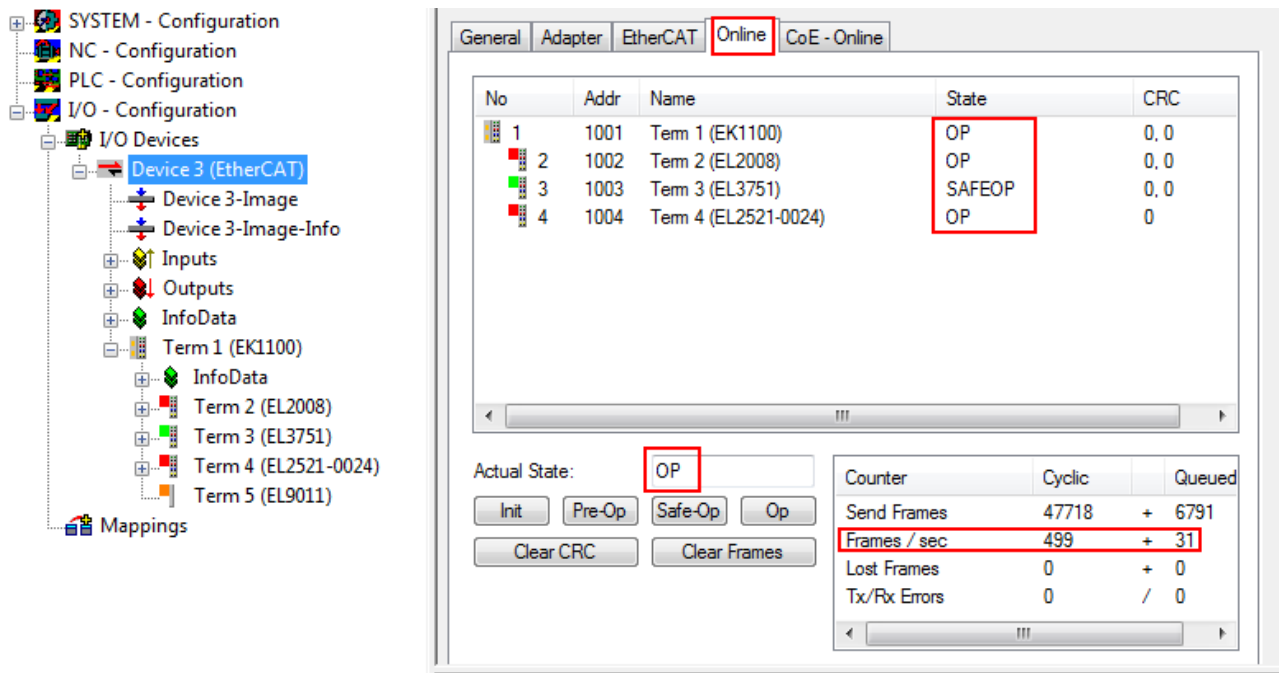


Fig. 65: Online display example

Please note:

- all slaves should be in OP state
- the EtherCAT master should be in “Actual State” OP
- “frames/sec” should match the cycle time taking into account the sent number of frames
- no excessive “LostFrames” or CRC errors should occur

The configuration is now complete. It can be modified as described under [manual procedure](#) [► 52].

## Troubleshooting

Various effects may occur during scanning.

- An **unknown device** is detected, i.e. an EtherCAT slave for which no ESI XML description is available. In this case the System Manager offers to read any ESI that may be stored in the device. This case is described in the chapter “Notes regarding ESI device description”.
- **Device are not detected properly**  
Possible reasons include:
  - faulty data links, resulting in data loss during the scan
  - slave has invalid device description

The connections and devices should be checked in a targeted manner, e.g. via the emergency scan. Then re-run the scan.

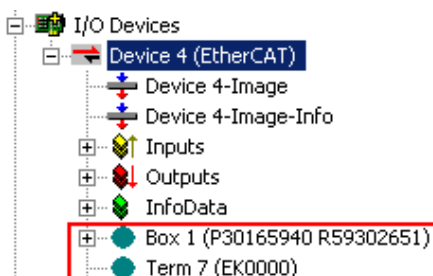


Fig. 66: Faulty identification

In the System Manager such devices may be set up as EK0000 or unknown devices. Operation is not possible or meaningful.

## Scan over existing Configuration

**NOTICE****Change of the configuration after comparison**

With this scan (TwinCAT 2.11 or 3.1) only the device properties vendor (manufacturer), device name and revision are compared at present! A “ChangeTo” or “Copy” should only be carried out with care, taking into consideration the Beckhoff IO compatibility rule (see above). The device configuration is then replaced by the revision found; this can affect the supported process data and functions.

If a scan is initiated for an existing configuration, the actual I/O environment may match the configuration exactly or it may differ. This enables the configuration to be compared.



Fig. 67: Identical configuration (left: TwinCAT 2; right: TwinCAT 3)

If differences are detected, they are shown in the correction dialog, so that the user can modify the configuration as required.

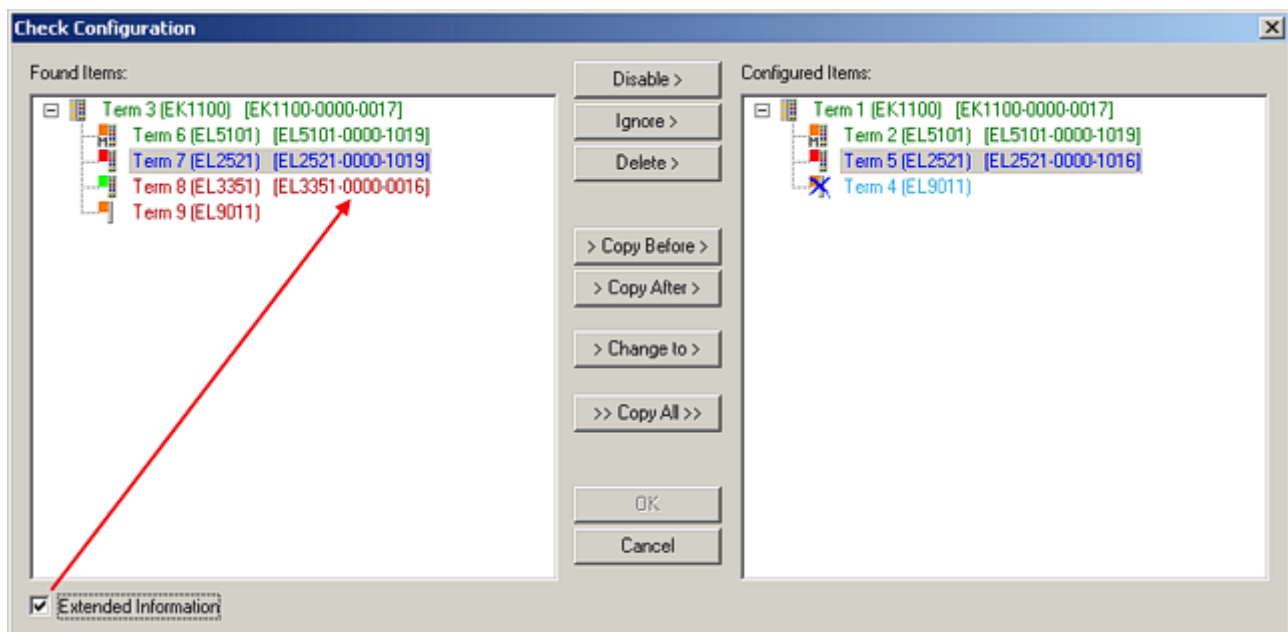


Fig. 68: Correction dialog

It is advisable to tick the “Extended Information” check box to reveal differences in the revision.

Color	Explanation
green	This EtherCAT slave matches the entry on the other side. Both type and revision match.
blue	<p>This EtherCAT slave is present on the other side, but in a different revision. This other revision can have other default values for the process data as well as other/additional functions.</p> <p>If the found revision is higher than the configured revision, the slave may be used provided compatibility issues are taken into account.</p> <p>If the found revision is lower than the configured revision, it is likely that the slave cannot be used. The found device may not support all functions that the master expects based on the higher revision number.</p>
light blue	This EtherCAT slave is ignored ("Ignore" button)
red	<ul style="list-style-type: none"> <li>This EtherCAT slave is not present on the other side.</li> <li>It is present, but in a different revision, which also differs in its properties from the one specified. The compatibility principle then also applies here: if the found revision is higher than the configured revision, use is possible provided compatibility issues are taken into account, since the successor devices should support the functions of the predecessor devices. If the found revision is lower than the configured revision, it is likely that the slave cannot be used. The found device may not support all functions that the master expects based on the higher revision number.</li> </ul>

### **i** Device selection based on revision, compatibility

The ESI description also defines the process image, the communication type between master and slave/device and the device functions, if applicable. The physical device (firmware, if available) has to support the communication queries/settings of the master. This is backward compatible, i.e. newer devices (higher revision) should be supported if the EtherCAT master addresses them as an older revision. The following compatibility rule of thumb is to be assumed for Beckhoff EtherCAT Terminals/ Boxes/ EJ-modules:

#### **device revision in the system $\geq$ device revision in the configuration**

This also enables subsequent replacement of devices without changing the configuration (different specifications are possible for drives).

### Example

If an EL2521-0025-**1018** is specified in the configuration, an EL2521-0025-**1018** or higher (**-1019**, **-1020**) can be used in practice.

(EL2521-0025-1018)

Fig. 69: Name/revision of the terminal

If current ESI descriptions are available in the TwinCAT system, the last revision offered in the selection dialog matches the Beckhoff state of production. It is recommended to use the last device revision when creating a new configuration, if current Beckhoff devices are used in the real application. Older revisions should only be used if older devices from stock are to be used in the application.

In this case the process image of the device is shown in the configuration tree and can be parameterized as follows: linking with the task, CoE/DC settings, plug-in definition, startup settings, ...



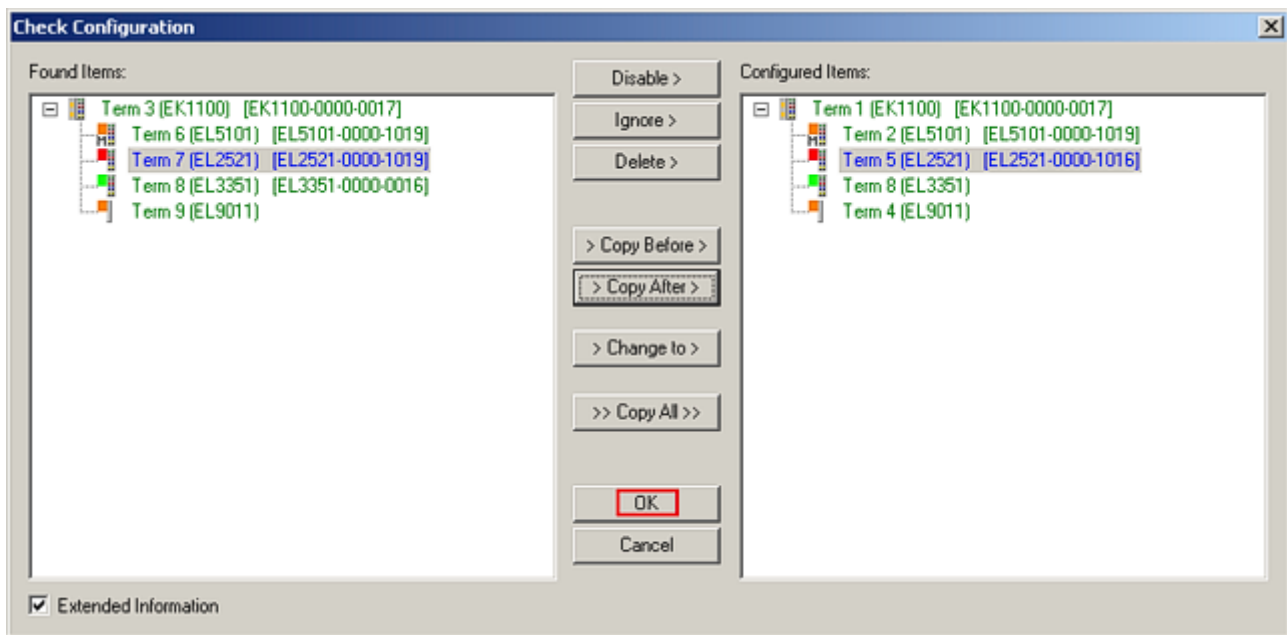


Fig. 70: Correction dialog with modifications

Once all modifications have been saved or accepted, click “OK” to transfer them to the real \*.tsm configuration.

### Change to Compatible Type

TwinCAT offers a function *Change to Compatible Type...* for the exchange of a device whilst retaining the links in the task.

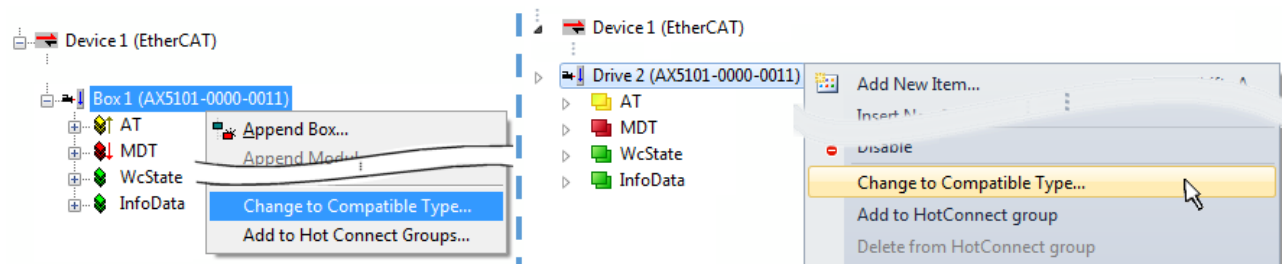


Fig. 71: Dialog “Change to Compatible Type...” (left: TwinCAT 2; right: TwinCAT 3)

The following elements in the ESI of an EtherCAT device are compared by TwinCAT and assumed to be the same in order to decide whether a device is indicated as “compatible”:

- Physics (e.g. RJ45, Ebus...)
- FMMU (additional ones are allowed)
- SyncManager (SM, additional ones are allowed)
- EoE (attributes MAC, IP)
- CoE (attributes SdoInfo, PdoAssign, PdoConfig, PdoUpload, CompleteAccess)
- FoE
- PDO (process data: Sequence, SyncUnit SU, SyncManager SM, EntryCount, Entry.Datatype)

This function is preferably to be used on AX5000 devices.

### Change to Alternative Type

The TwinCAT System Manager offers a function for the exchange of a device: Change to Alternative Type



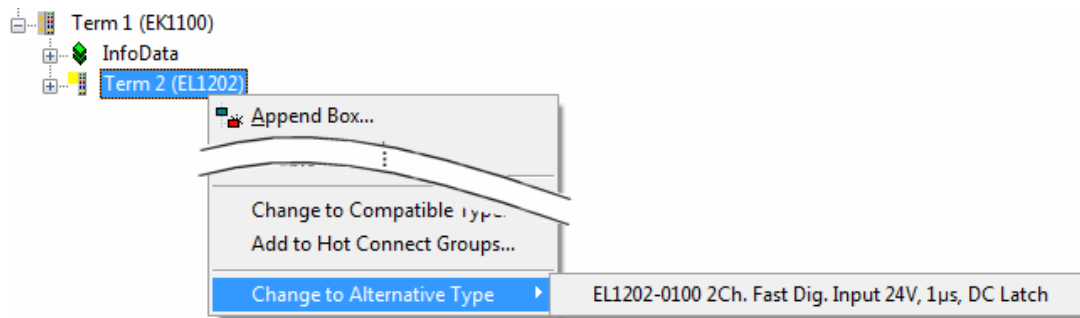


Fig. 72: TwinCAT 2 Dialog Change to Alternative Type

If called, the System Manager searches in the procured device ESI (in this example: EL1202-0000) for details of compatible devices contained there. The configuration is changed and the ESI-EEPROM is overwritten at the same time – therefore this process is possible only in the online state (ConfigMode).

### 5.1.1.7 EtherCAT subscriber configuration

In the left-hand window of the TwinCAT 2 System Manager or the Solution Explorer of the TwinCAT 3 Development Environment respectively, click on the element of the terminal within the tree you wish to configure (in the example: EL3751 Terminal 3).

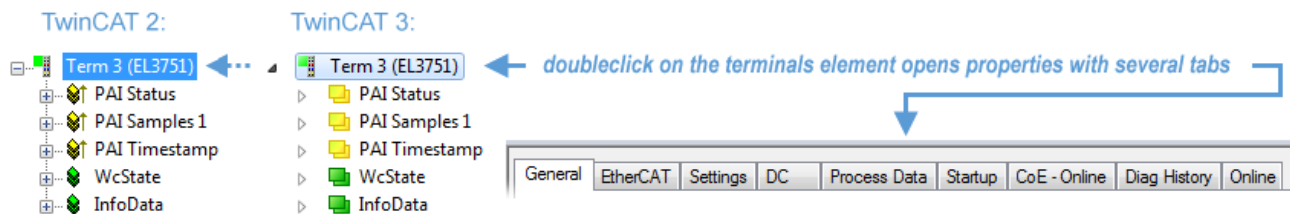


Fig. 73: Branch element as terminal EL3751

In the right-hand window of the TwinCAT System Manager (TwinCAT 2) or the Development Environment (TwinCAT 3), various tabs are now available for configuring the terminal. And yet the dimension of complexity of a subscriber determines which tabs are provided. Thus as illustrated in the example above the terminal EL3751 provides many setup options and also a respective number of tabs are available. On the contrary by the terminal EL1004 for example the tabs “General”, “EtherCAT”, “Process Data” and “Online” are available only. Several terminals, as for instance the EL6695 provide special functions by a tab with its own terminal name, so “EL6695” in this case. A specific tab “Settings” by terminals with a wide range of setup options will be provided also (e.g. EL3751).

#### “General” tab

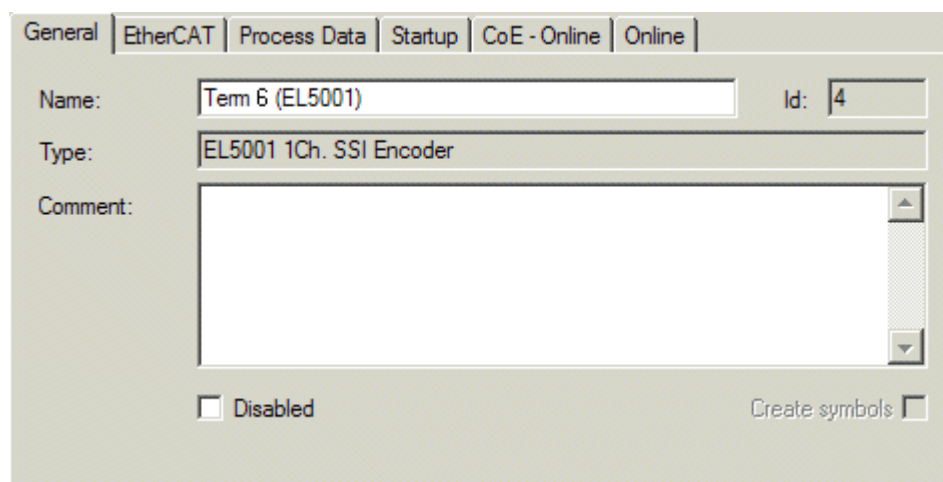


Fig. 74: “General” tab

<b>Name</b>	Name of the EtherCAT device
<b>Id</b>	Number of the EtherCAT device
<b>Type</b>	EtherCAT device type
<b>Comment</b>	Here you can add a comment (e.g. regarding the system).
<b>Disabled</b>	Here you can deactivate the EtherCAT device.
<b>Create symbols</b>	Access to this EtherCAT slave via ADS is only available if this control box is activated.

#### “EtherCAT” tab

The screenshot shows the 'EtherCAT' configuration tab. It includes the following fields and controls:

- Type:** Text field containing 'EL5001 1Ch. SSI Encoder'.
- Product/Revision:** Text field containing 'EL5001-0000-0000'.
- Auto Inc Addr:** Text field containing 'FFFD'.
- EtherCAT Addr:** A checkbox is unchecked, followed by a text field containing '1004' and a small increment/decrement control.
- Advanced Settings...** A button to the right of the EtherCAT Addr field.
- Previous Port:** A dropdown menu showing 'Term 5 (EL6021) - B'.
- URL:** A link at the bottom: <https://www.beckhoff.com/EL5001>.

Fig. 75: “EtherCAT” tab

<b>Type</b>	EtherCAT device type
<b>Product/Revision</b>	Product and revision number of the EtherCAT device
<b>Auto Inc Addr.</b>	Auto increment address of the EtherCAT device. The auto increment address can be used for addressing each EtherCAT device in the communication ring through its physical position. Auto increment addressing is used during the start-up phase when the EtherCAT master allocates addresses to the EtherCAT devices. With auto increment addressing the first EtherCAT slave in the ring has the address 0000 <sub>hex</sub> . For each further slave the address is decremented by 1 (FFFF <sub>hex</sub> , FFFE <sub>hex</sub> etc.).
<b>EtherCAT Addr.</b>	Fixed address of an EtherCAT slave. This address is allocated by the EtherCAT master during the start-up phase. Tick the control box to the left of the input field in order to modify the default value.
<b>Previous Port</b>	Name and port of the EtherCAT device to which this device is connected. If it is possible to connect this device with another one without changing the order of the EtherCAT devices in the communication ring, then this combination field is activated and the EtherCAT device to which this device is to be connected can be selected.
<b>Advanced Settings</b>	This button opens the dialogs for advanced settings.

The link at the bottom of the tab points to the product page for this EtherCAT device on the web.

#### “Process Data” tab

Indicates the configuration of the process data. The input and output data of the EtherCAT slave are represented as CANopen process data objects (**Process Data Objects**, PDOs). The user can select a PDO via PDO assignment and modify the content of the individual PDO via this dialog, if the EtherCAT slave supports this function.

General | **EtherCAT** | Process Data | Startup | CoE - Online | Online

Sync Manager:

SM	Size	Type	Flags
0	246	MbxOut	
1	246	MbxIn	
2	0	Outputs	
3	5	Inputs	

PDO List:

Index	Size	Name	Flags	SM	SU
0x1A00	5.0	Channel 1	F	3	0

PDO Assignment (0x1C13):

☒ 0x1A00

Download

☒ PDO Assignment

☒ PDO Configuration

PDO Content (0x1A00):

Index	Size	Offs	Name	Type	Default (hex)
0x3101:01	1.0	0.0	Status	BYTE	
0x3101:02	4.0	1.0	Value	UDINT	
		5.0			

Load PDO info from device

Sync Unit Assignment...

Fig. 76: "Process Data" tab

The process data (PDOs) transferred by an EtherCAT slave during each cycle are user data which the application expects to be updated cyclically or which are sent to the slave. To this end the EtherCAT master (Beckhoff TwinCAT) parameterizes each EtherCAT slave during the start-up phase to define which process data (size in bits/bytes, source location, transmission type) it wants to transfer to or from this slave. Incorrect configuration can prevent successful start-up of the slave.

For Beckhoff EtherCAT EL, ES, EM, EJ and EP slaves the following applies in general:

- The input/output process data supported by the device are defined by the manufacturer in the ESI/XML description. The TwinCAT EtherCAT Master uses the ESI description to configure the slave correctly.
- The process data can be modified in the System Manager. See the device documentation. Examples of modifications include: mask out a channel, displaying additional cyclic information, 16-bit display instead of 8-bit data size, etc.
- In so-called "intelligent" EtherCAT devices the process data information is also stored in the CoE directory. Any changes in the CoE directory that lead to different PDO settings prevent successful startup of the slave. It is not advisable to deviate from the designated process data, because the device firmware (if available) is adapted to these PDO combinations.

If the device documentation allows modification of process data, proceed as follows (see Figure *Configuring the process data*).

- A: select the device to configure
- B: in the "Process Data" tab select Input or Output under SyncManager (C)
- D: the PDOs can be selected or deselected
- H: the new process data are visible as linkable variables in the System Manager  
The new process data are active once the configuration has been activated and TwinCAT has been restarted (or the EtherCAT master has been restarted)
- E: if a slave supports this, Input and Output PDO can be modified simultaneously by selecting a so-called PDO record ("predefined PDO settings").

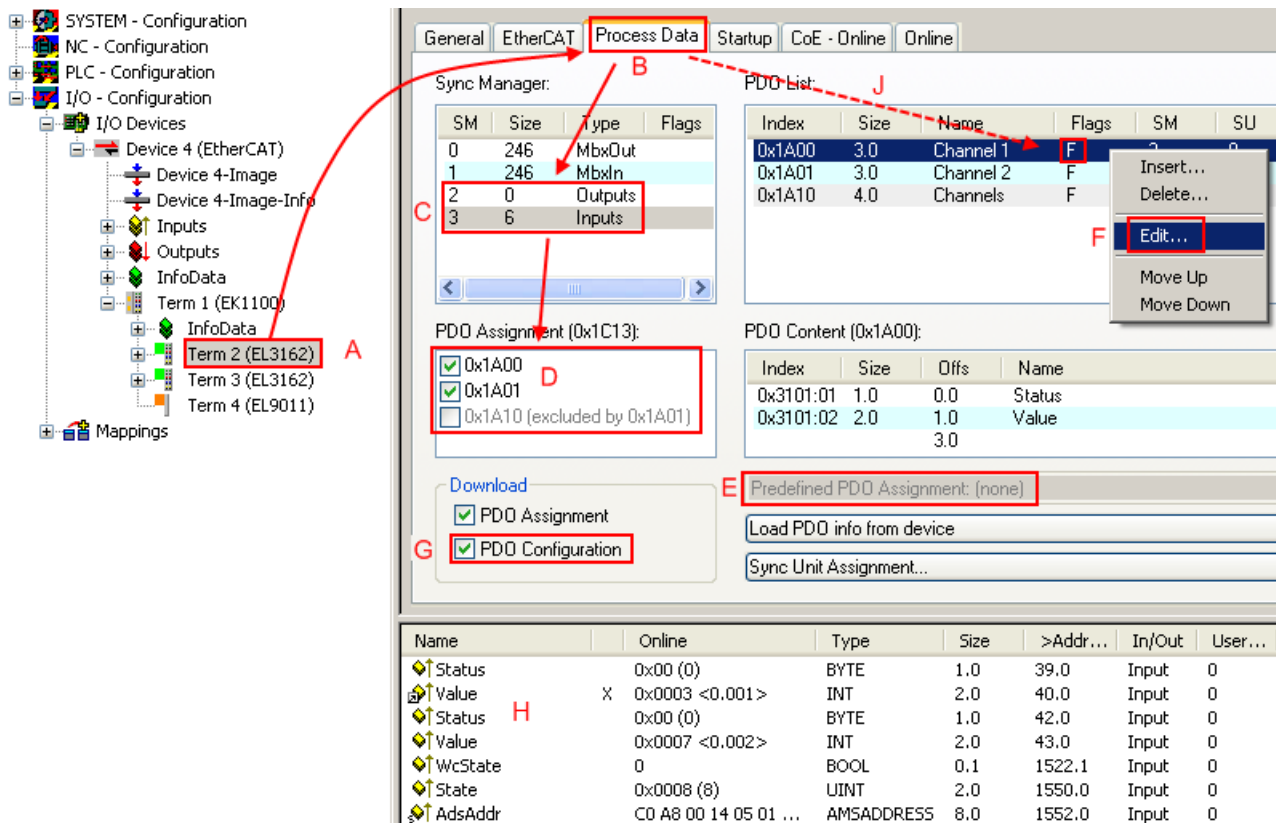


Fig. 77: Configuring the process data

## Manual modification of the process data

According to the ESI description, a PDO can be identified as “fixed” with the flag “F” in the PDO overview (Fig. *Configuring the process data*, J). The configuration of such PDOs cannot be changed, even if TwinCAT offers the associated dialog (“Edit”). In particular, CoE content cannot be displayed as cyclic process data. This generally also applies in cases where a device supports download of the PDO configuration, “G”. In case of incorrect configuration the EtherCAT slave usually refuses to start and change to OP state. The System Manager displays an “invalid SM cfg” logger message: This error message (“invalid SM IN cfg” or “invalid SM OUT cfg”) also indicates the reason for the failed start.

A detailed description [► 73] can be found at the end of this section.

### “Startup” tab

The *Startup* tab is displayed if the EtherCAT slave has a mailbox and supports the *CANopen over EtherCAT* (CoE) or *Servo drive over EtherCAT* protocol. This tab indicates which download requests are sent to the mailbox during startup. It is also possible to add new mailbox requests to the list display. The download requests are sent to the slave in the same order as they are shown in the list.

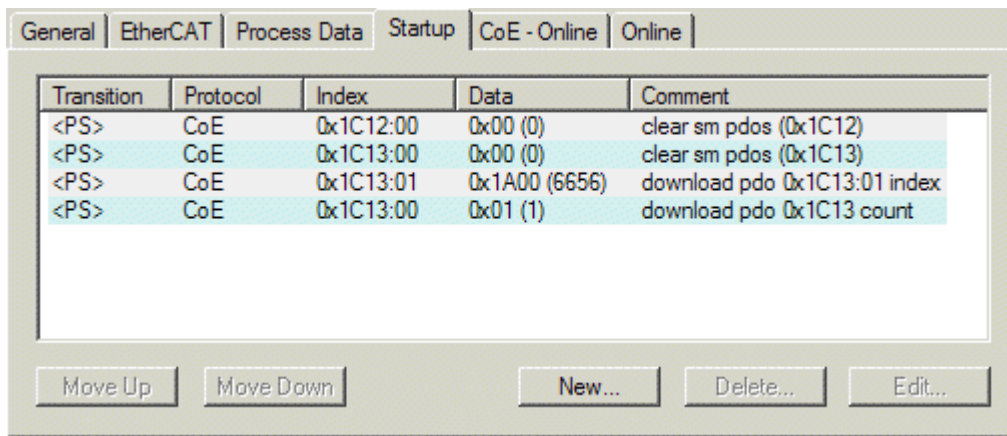


Fig. 78: "Startup" tab

Column	Description
Transition	Transition to which the request is sent. This can either be <ul style="list-style-type: none"> <li>the transition from pre-operational to safe-operational (PS), or</li> <li>the transition from safe-operational to operational (SO).</li> </ul> If the transition is enclosed in "<>" (e.g. <PS>), the mailbox request is fixed and cannot be modified or deleted by the user.
Protocol	Type of mailbox protocol
Index	Index of the object
Data	Date on which this object is to be downloaded.
Comment	Description of the request to be sent to the mailbox

<b>Move Up</b>	This button moves the selected request up by one position in the list.
<b>Move Down</b>	This button moves the selected request down by one position in the list.
<b>New</b>	This button adds a new mailbox download request to be sent during startup.
<b>Delete</b>	This button deletes the selected entry.
<b>Edit</b>	This button edits an existing request.

#### "CoE - Online" tab

The additional *CoE - Online* tab is displayed if the EtherCAT slave supports the *CANopen over EtherCAT* (CoE) protocol. This dialog lists the content of the object list of the slave (SDO upload) and enables the user to modify the content of an object from this list. Details for the objects of the individual EtherCAT devices can be found in the device-specific object descriptions.

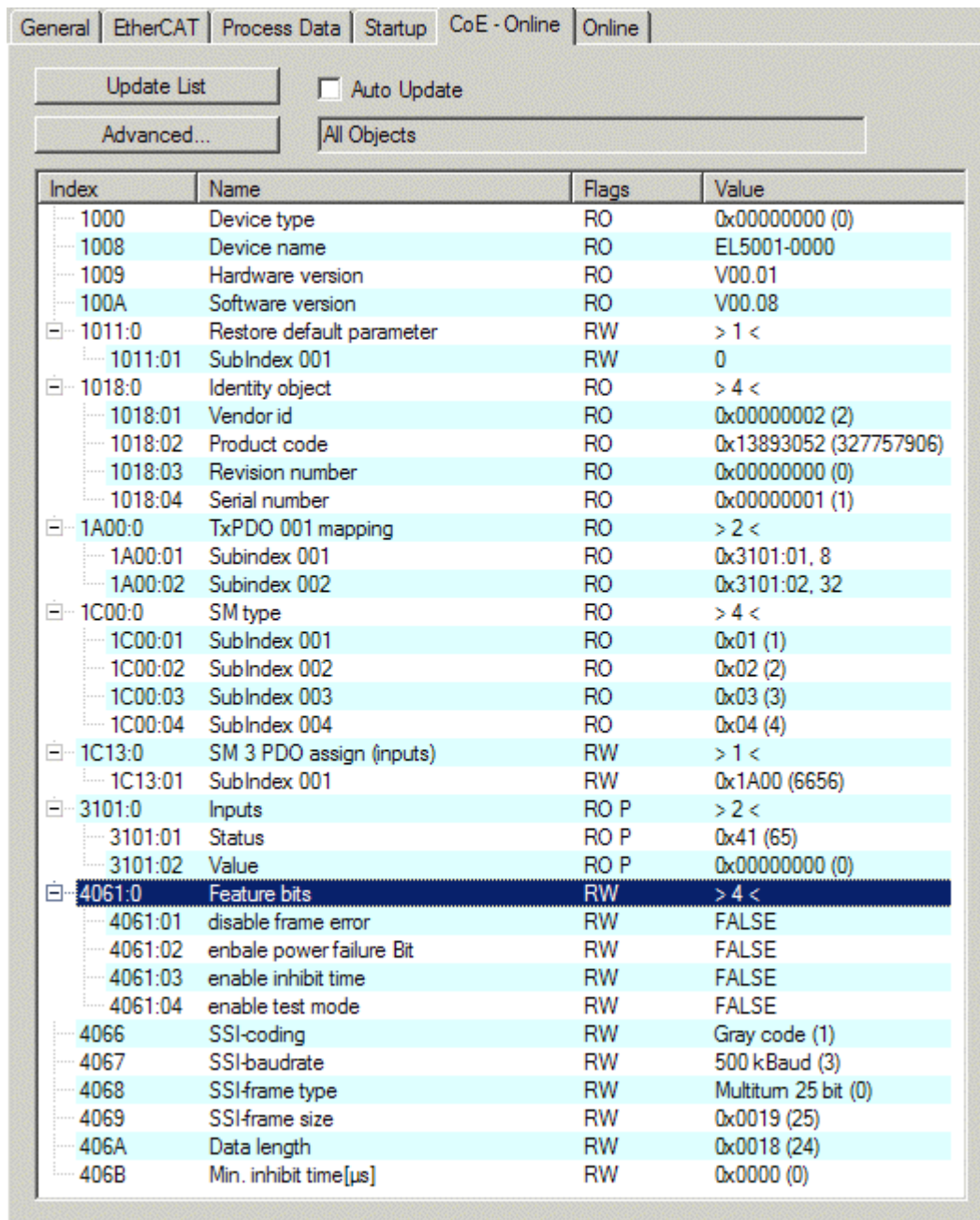


Fig. 79: "CoE - Online" tab

### Object list display

Column	Description
Index	Index and sub-index of the object
Name	Name of the object
Flags	RW The object can be read, and data can be written to the object (read/write)
	RO The object can be read, but no data can be written to the object (read only)
	P An additional P identifies the object as a process data object.
Value	Value of the object

**Update List** The *Update list* button updates all objects in the displayed list

**Auto Update** If this check box is selected, the content of the objects is updated automatically.

**Advanced** The *Advanced* button opens the *Advanced Settings* dialog. Here you can specify which objects are displayed in the list.



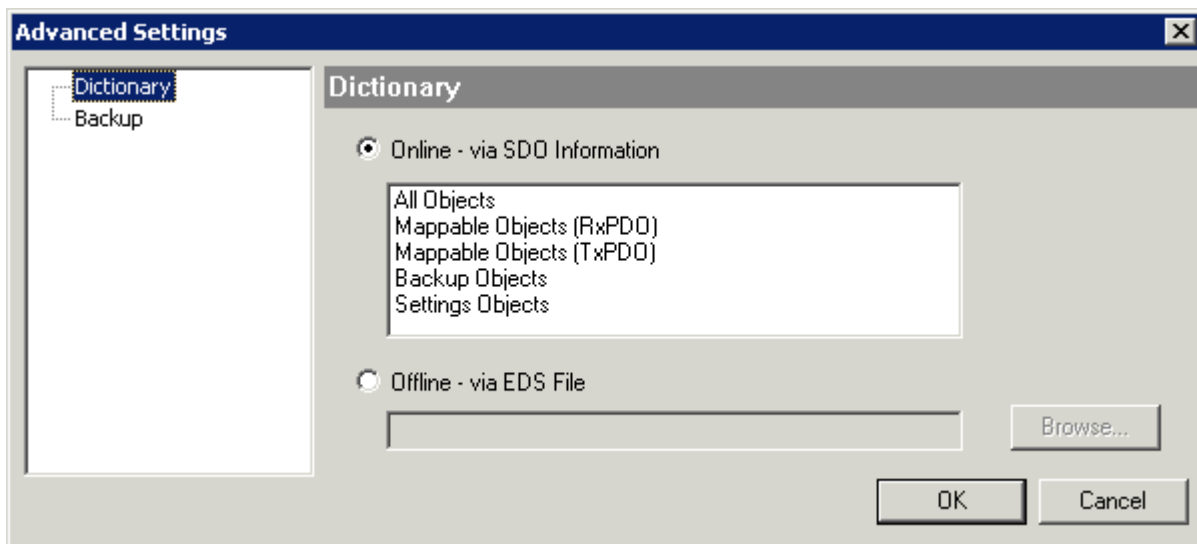


Fig. 80: Dialog “Advanced settings”

**Online - via SDO Information** If this option button is selected, the list of the objects included in the object list of the slave is uploaded from the slave via SDO information. The list below can be used to specify which object types are to be uploaded.

**Offline - via EDS File** If this option button is selected, the list of the objects included in the object list is read from an EDS file provided by the user.

#### “Online” tab

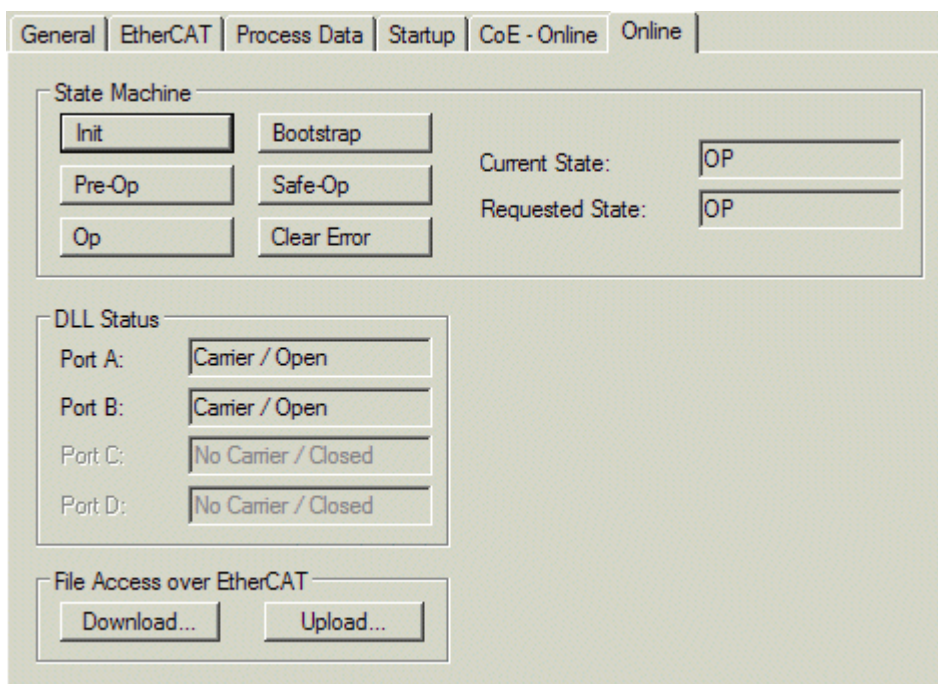


Fig. 81: “Online” tab

## State Machine

<b>Init</b>	This button attempts to set the EtherCAT device to the <i>Init</i> state.
<b>Pre-Op</b>	This button attempts to set the EtherCAT device to the <i>pre-operational</i> state.
<b>Op</b>	This button attempts to set the EtherCAT device to the <i>operational</i> state.
<b>Bootstrap</b>	This button attempts to set the EtherCAT device to the <i>Bootstrap</i> state.
<b>Safe-Op</b>	This button attempts to set the EtherCAT device to the <i>safe-operational</i> state.
<b>Clear Error</b>	This button attempts to delete the fault display. If an EtherCAT slave fails during change of state it sets an error flag.  Example: An EtherCAT slave is in PREOP state (pre-operational). The master now requests the SAFEOP state (safe-operational). If the slave fails during change of state it sets the error flag. The current state is now displayed as ERR PREOP. When the <i>Clear Error</i> button is pressed the error flag is cleared, and the current state is displayed as PREOP again.
<b>Current State</b>	Indicates the current state of the EtherCAT device.
<b>Requested State</b>	Indicates the state requested for the EtherCAT device.

## DLL Status

Indicates the DLL status (data link layer status) of the individual ports of the EtherCAT slave. The DLL status can have four different states:

Status	Description
No Carrier / Open	No carrier signal is available at the port, but the port is open.
No Carrier / Closed	No carrier signal is available at the port, and the port is closed.
Carrier / Open	A carrier signal is available at the port, and the port is open.
Carrier / Closed	A carrier signal is available at the port, but the port is closed.

## File Access over EtherCAT

<b>Download</b>	With this button a file can be written to the EtherCAT device.
<b>Upload</b>	With this button a file can be read from the EtherCAT device.

## “DC” tab (Distributed Clocks)

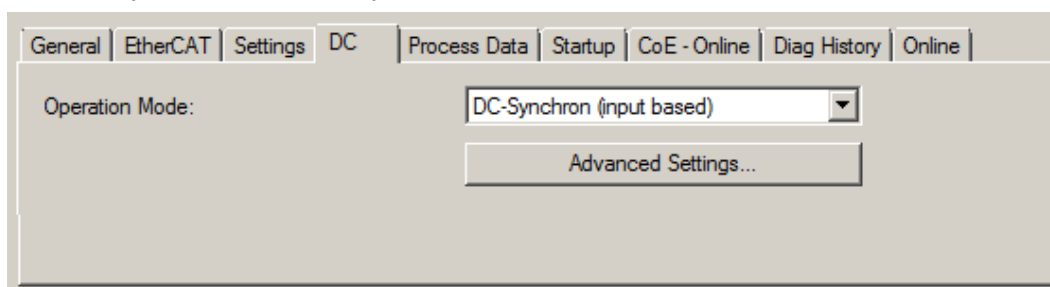


Fig. 82: “DC” tab (Distributed Clocks)

<b>Operation Mode</b>	Options (optional): <ul style="list-style-type: none"> <li>• FreeRun</li> <li>• SM-Synchron</li> <li>• DC-Synchron (Input based)</li> <li>• DC-Synchron</li> </ul>
<b>Advanced Settings...</b>	Advanced settings for readjustment of the real time determinant TwinCAT-clock

Detailed information to Distributed Clocks is specified on <http://infosys.beckhoff.com>:

**Fieldbus Components** → EtherCAT Terminals → EtherCAT System documentation → EtherCAT basics → Distributed Clocks



### 5.1.1.7.1 Detailed description of Process Data tab

#### Sync Manager

Lists the configuration of the Sync Manager (SM).

If the EtherCAT device has a mailbox, SM0 is used for the mailbox output (MbxOut) and SM1 for the mailbox input (MbxIn).

SM2 is used for the output process data (outputs) and SM3 (inputs) for the input process data.

If an input is selected, the corresponding PDO assignment is displayed in the *PDO Assignment* list below.

#### PDO Assignment

PDO assignment of the selected Sync Manager. All PDOs defined for this Sync Manager type are listed here:

- If the output Sync Manager (outputs) is selected in the Sync Manager list, all RxPDOs are displayed.
- If the input Sync Manager (inputs) is selected in the Sync Manager list, all TxPDOs are displayed.

The selected entries are the PDOs involved in the process data transfer. In the tree diagram of the System Manager these PDOs are displayed as variables of the EtherCAT device. The name of the variable is identical to the *Name* parameter of the PDO, as displayed in the PDO list. If an entry in the PDO assignment list is deactivated (not selected and greyed out), this indicates that the input is excluded from the PDO assignment. In order to be able to select a greyed out PDO, the currently selected PDO has to be deselected first.

#### **i** Activation of PDO assignment

- ✓ If you have changed the PDO assignment, in order to activate the new PDO assignment,
  - a) the EtherCAT slave has to run through the PS status transition cycle (from pre-operational to safe-operational) once (see [Online tab \[► 71\]](#)),
  - b) and the System Manager has to reload the EtherCAT slaves



( button for TwinCAT 2 or



button for TwinCAT 3)

#### PDO list

List of all PDOs supported by this EtherCAT device. The content of the selected PDOs is displayed in the *PDO Content* list. The PDO configuration can be modified by double-clicking on an entry.

Column	Description	
Index	PDO index.	
Size	Size of the PDO in bytes.	
Name	Name of the PDO. If this PDO is assigned to a Sync Manager, it appears as a variable of the slave with this parameter as the name.	
Flags	F	Fixed content: The content of this PDO is fixed and cannot be changed by the System Manager.
	M	Mandatory PDO. This PDO is mandatory and must therefore be assigned to a Sync Manager! Consequently, this PDO cannot be deleted from the <i>PDO Assignment</i> list
SM	Sync Manager to which this PDO is assigned. If this entry is empty, this PDO does not take part in the process data traffic.	
SU	Sync unit to which this PDO is assigned.	

#### PDO Content

Indicates the content of the PDO. If flag F (fixed content) of the PDO is not set the content can be modified.

**Download**

If the device is intelligent and has a mailbox, the configuration of the PDO and the PDO assignments can be downloaded to the device. This is an optional feature that is not supported by all EtherCAT slaves.

**PDO Assignment**

If this check box is selected, the PDO assignment that is configured in the PDO Assignment list is downloaded to the device on startup. The required commands to be sent to the device can be viewed in the [Startup \[► 68\]](#) tab.

**PDO Configuration**

If this check box is selected, the configuration of the respective PDOs (as shown in the PDO list and the PDO Content display) is downloaded to the EtherCAT slave.

### 5.1.1.8 Import/Export of EtherCAT devices with SCI and XTI

#### SCI and XTI Export/Import – Handling of user-defined modified EtherCAT slaves

##### 5.1.1.8.1 Basic principles

An EtherCAT slave is basically parameterized through the following elements:

- Cyclic process data (PDO)
- Synchronization (Distributed Clocks, FreeRun, SM-Synchron)
- CoE parameters (acyclic object dictionary)

Note: Not all three elements may be present, depending on the slave.

For a better understanding of the export/import function, let's consider the usual procedure for IO configuration:

- The user/programmer processes the IO configuration in the TwinCAT system environment. This involves all input/output devices such as drives that are connected to the fieldbuses used.  
Note: In the following sections, only EtherCAT configurations in the TwinCAT system environment are considered.
- For example, the user manually adds devices to a configuration or performs a scan on the online system.
- This results in the IO system configuration.
- On insertion, the slave appears in the system configuration in the default configuration provided by the vendor, consisting of default PDO, default synchronization method and CoE StartUp parameter as defined in the ESI (XML device description).
- If necessary, elements of the slave configuration can be changed, e.g. the PDO configuration or the synchronization method, based on the respective device documentation.

It may become necessary to reuse the modified slave in other projects in this way, without having to make equivalent configuration changes to the slave again. To accomplish this, proceed as follows:

- Export the slave configuration from the project,
- Store and transport as a file,
- Import into another EtherCAT project.

TwinCAT offers two methods for this purpose:

- within the TwinCAT environment: Export/Import as **x**ti file or
- outside, i.e. beyond the TwinCAT limits: Export/Import as **s**ci file.

An example is provided below for illustration purposes: an EL3702 terminal with standard setting is switched to 2-fold oversampling (blue) and the optional PDO "StartTimeNextLatch" is added (red):

The screenshot shows the TwinCAT Project34 interface. The left pane displays the project tree with 'Term 2 (EL3702)' selected. The right pane shows the 'Process Data' tab with a table of PDO assignments. Red arrows indicate the flow from the project tree to the 'Process Data' tab, then to the 'PDO List' table, and finally to the 'PDO Assignment' table.

**Sync Manager:**

SM	Size	Type	Flags
0	6	Inputs	
1	6	Inputs	
2	4	Inputs	

**PDO List:**

Index	Size	Name
0x1B00	2.0	Ch1 CycleCount
0x1A00	2.0	Ch1 Sample 0
0x1A01	2.0	Ch1 Sample 1
0x1A02	2.0	Ch1 Sample 2
0x1A03	2.0	Ch1 Sample 3
0x1A04	2.0	Ch1 Sample 4
0x1A05	2.0	Ch1 Sample 5

**PDO Assignment (0x1C12):**

Index	Size	Name
0x1AE0	2.0	Ch1 CycleCount
0x1AE1	2.0	Ch1 Sample 0
0x1AE2	2.0	Ch1 Sample 1
0x1AE3	2.0	Ch1 Sample 2
0x1B10	2.0	Ch1 CycleCount

**PDO Content (0x1B00):**

Index	Size	Offs	Name
0x6800:01	2.0	0.0	Ch1 CycleCount
		2.0	

**Download:**

☐ PDO Assignment

☐ PDO Configuration

**Predefined PDO Assignment:** (none)

**Load PDO info from device**

**Sync Unit Assignment...**

**Term 2 (EL3702) Properties:**

Name	Online	Type	Size	>Addr...
Ch1 CycleCount		UINT	2.0	58.0
Ch1 Value		INT	2.0	60.0
Ch1 Value		INT	2.0	62.0
Ch2 CycleCount		UINT	2.0	64.0
Ch2 Value		INT	2.0	66.0
Ch2 Value		INT	2.0	68.0
StartTimeNextLa...		UDINT	4.0	70.0
WcState		BIT	0.1	1522.2

The two methods for exporting and importing the modified terminal referred to above are demonstrated below.

### 5.1.1.8.2 Procedure within TwinCAT with xti files

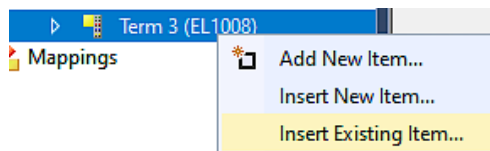
Each IO device can be exported/saved individually:

The screenshot shows the context menu for 'Term 2 (EL3702)' in the TwinCAT Project34 interface. The menu options are: Add New Item..., Insert New Item..., Insert Existing Item..., Remove, and Save Term 2 (EL3702) As...

The xti file can be stored:

The screenshot shows the file save dialog with the filename 'Term 2 (EL3702).xti' and the file type 'TwinCAT Export File (\*.xti)'.

and imported again in another TwinCAT system via "Insert Existing item":



### 5.1.1.8.3 Procedure within and outside TwinCAT with sci file

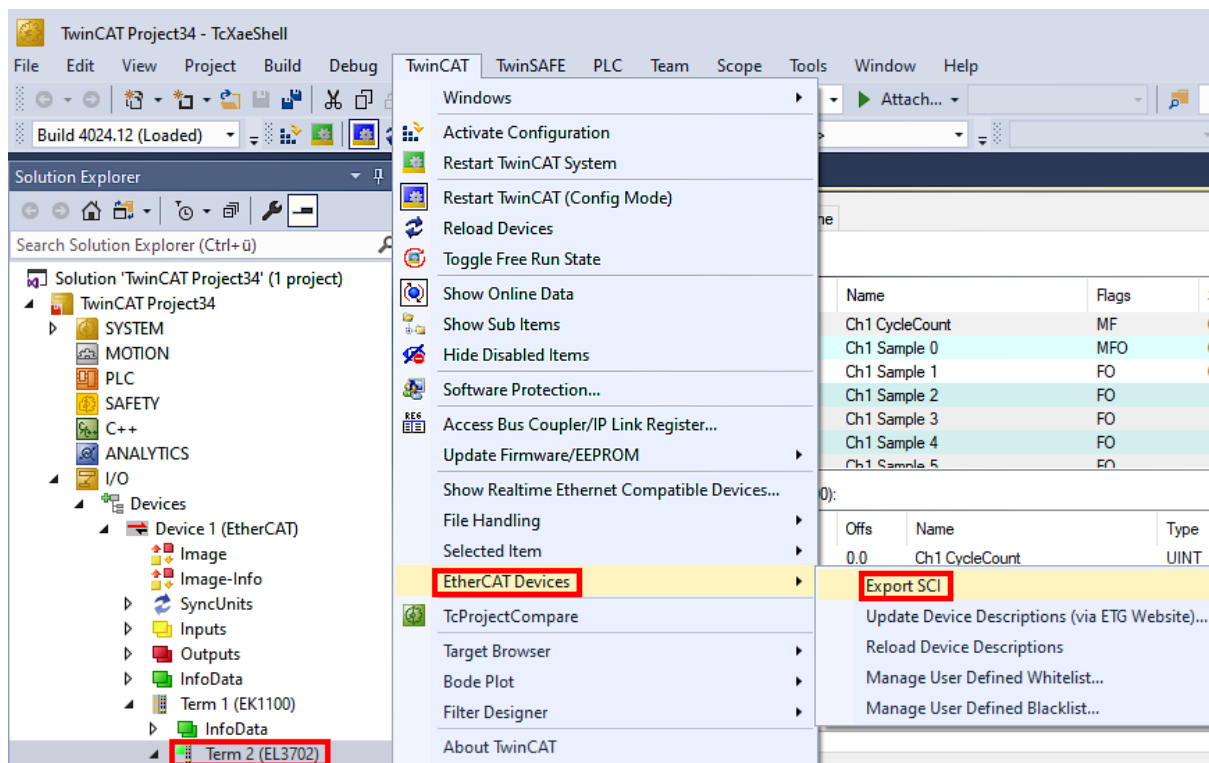
*Note regarding availability (2021/01)*

*The SCI method is available from TwinCAT 3.1 build 4024.14.*

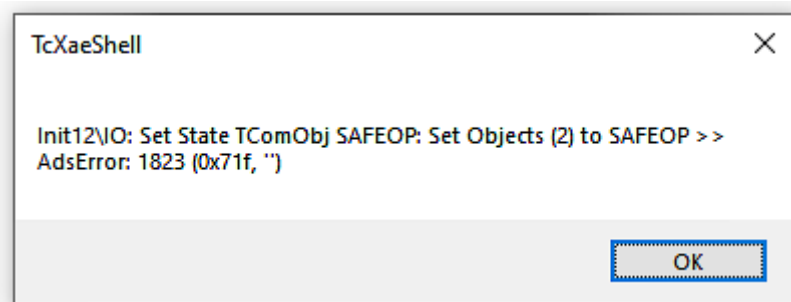
The Slave Configuration Information (SCI) describes a specific complete configuration for an EtherCAT slave (terminal, box, drive...) based on the setting options of the device description file (ESI, EtherCAT Slave Information). That is, it includes PDO, CoE, synchronization.

#### Export:

- select a single device via the menu (multiple selection is also possible):  
TwinCAT → EtherCAT Devices → Export SCI.



- If TwinCAT is offline (i.e. if there is no connection to an actual running controller) a warning message may appear, because after executing the function the system attempts to reload the EtherCAT segment. However, in this case this is not relevant for the result and can be acknowledged by clicking OK:



- A description may also be provided:

- Explanation of the dialog box:

Name	Name of the SCI, assigned by the user.	
Description	Description of the slave configuration for the use case, assigned by the user.	
Options	Keep modules	If a slave supports modules/slots, the user can decide whether these are to be exported or whether the module and device data are to be combined during export.
	AoE   Set AmsNetId	The configured AmsNetId is exported. Usually this is network-dependent and cannot always be determined in advance.
	EoE   Set MAC and IP	The configured virtual MAC and IP addresses are stored in the SCI. Usually these are network-dependent and cannot always be determined in advance.
	CoE   Set cycle time(0x1C3x.2)	The configured cycle time is exported. Usually this is network-dependent and cannot always be determined in advance.
ESI	Reference to the original ESI file.	
Export	Save SCI file.	

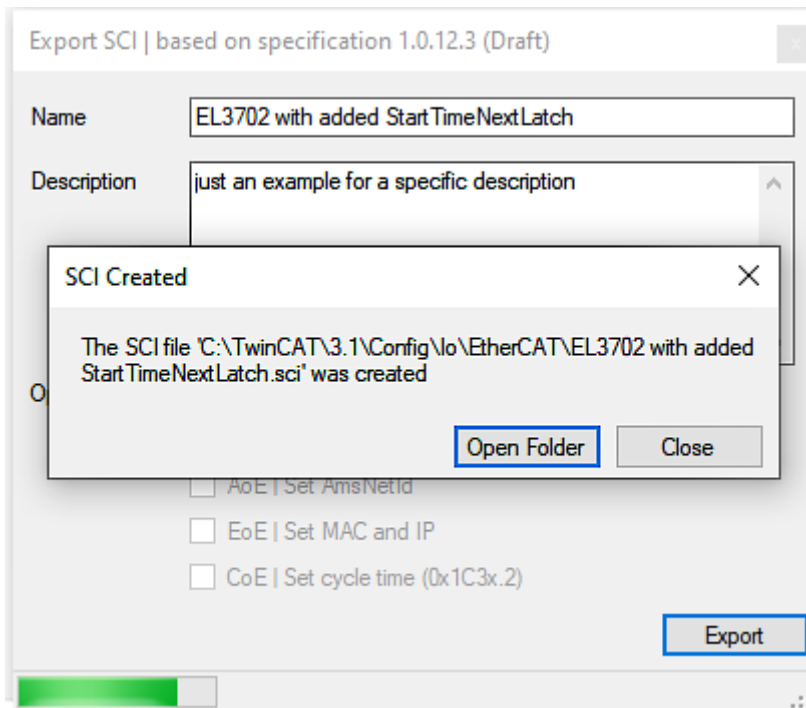
- A list view is available for multiple selections (*Export multiple SCI files*):

- Selection of the slaves to be exported:
  - All:  
All slaves are selected for export.

- None:  
All slaves are deselected.
- The sci file can be saved locally:

Dateiname:	EL3702 with added StartTimeNextLatch.sci
Dateityp:	SCI file (*.sci)

- The export takes place:

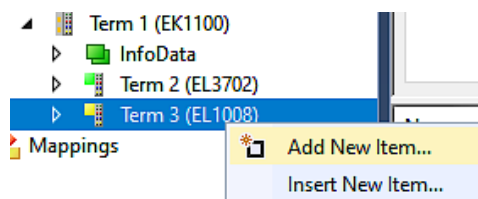


## Import

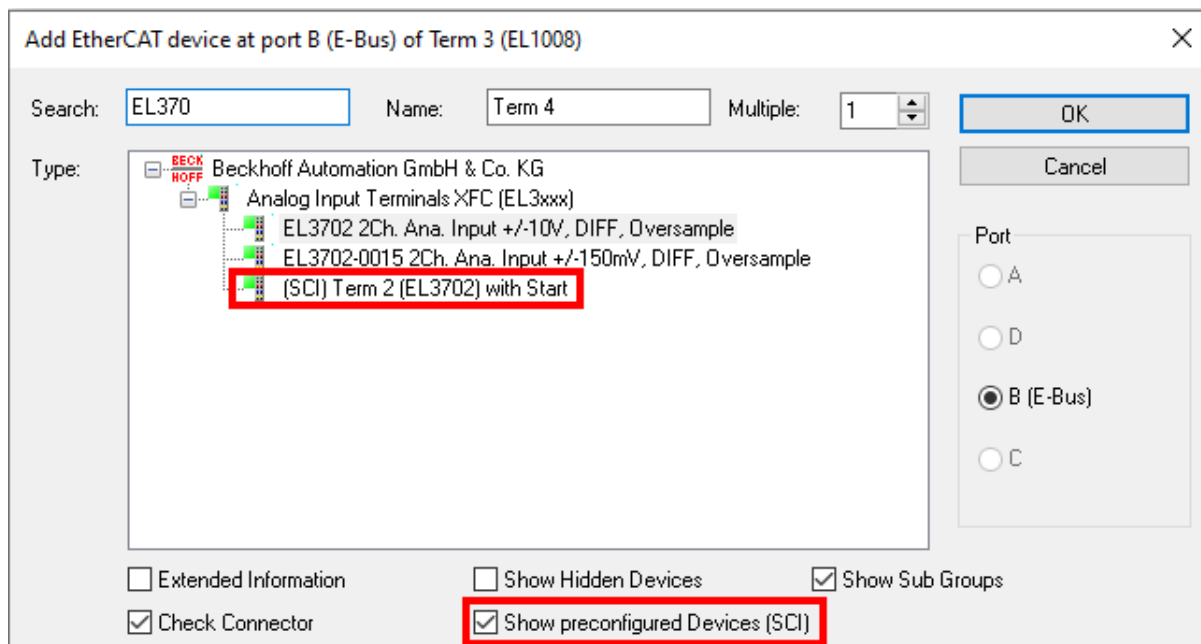
- An sci description can be inserted manually into the TwinCAT configuration like any normal Beckhoff device description.
- The sci file must be located in the TwinCAT ESI path, usually under:  
C:\TwinCAT\3.1\Config\Io\EtherCAT

	EL3702 with added StartTimeNextLatch.sci	11.01.2021 13:29	SCI-Datei	6 KB
---	--	------------------	-----------	------

- Open the selection dialog:

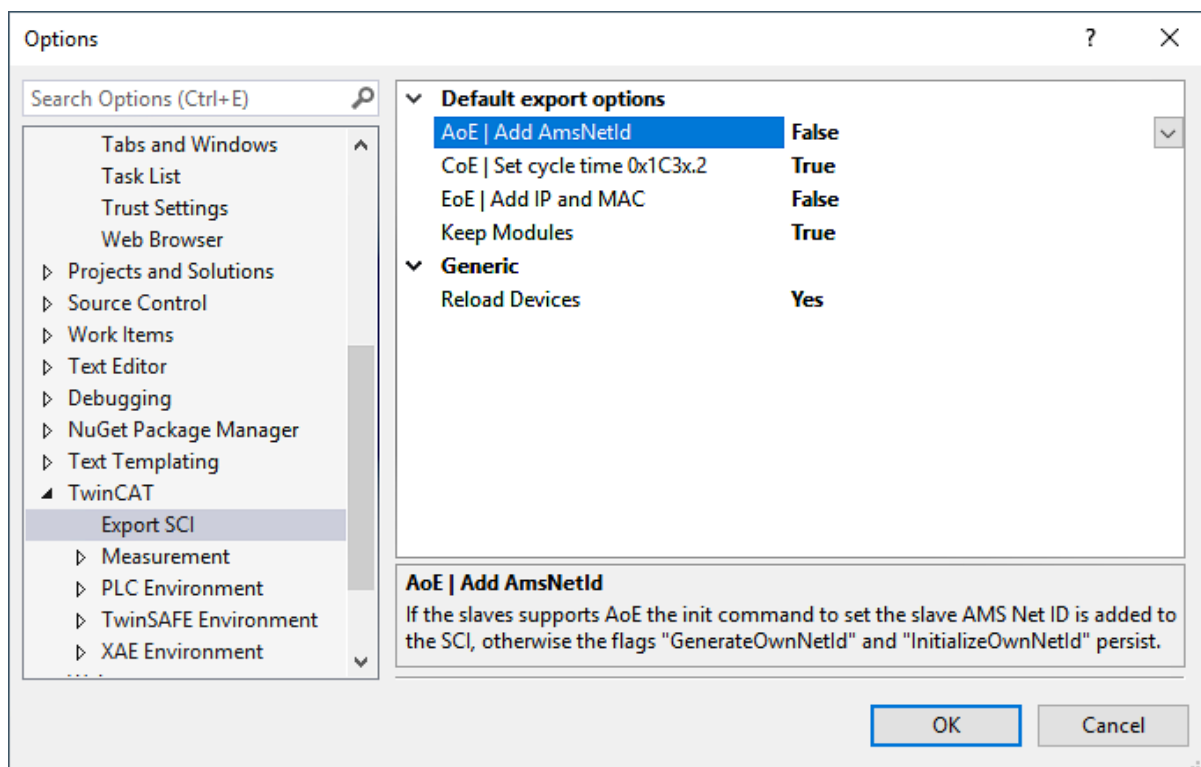


- Display SCI devices and select and insert the desired device:



### Additional Notes

- Settings for the SCI function can be made via the general Options dialog (Tools → Options → TwinCAT → Export SCI):

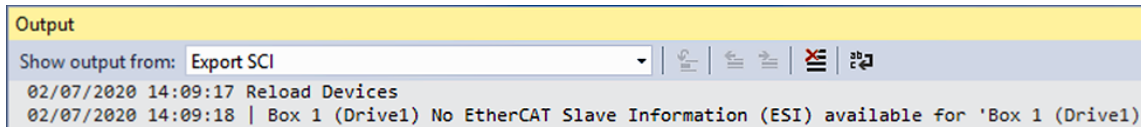


Explanation of the settings:

Default export options	AoE   Set AmsNetId	Default setting whether the configured AmsNetId is exported.
	CoE   Set cycle time(0x1C3x.2)	Default setting whether the configured cycle time is exported.
	EoE   Set MAC and IP	Default setting whether the configured MAC and IP addresses are exported.
	Keep modules	Default setting whether the modules persist.
Generic	Reload Devices	Setting whether the Reload Devices command is executed before the SCI export. This is strongly recommended to ensure a consistent slave configuration.



SCI error messages are displayed in the TwinCAT logger output window if required:



## 5.1.2 TwinCAT Quick Start

TwinCAT is a development environment for real-time control including a multi PLC system, NC axis control, programming and operation. The whole system is mapped through this environment and enables access to a programming environment (including compilation) for the controller. Individual digital or analog inputs or outputs can also be read or written directly, in order to verify their functionality, for example.

For further information, please refer to <http://infosys.beckhoff.com>:

- **EtherCAT System Manual:**  
Fieldbus Components → EtherCAT Terminals → EtherCAT System Documentation → Setup in the TwinCAT System Manager
- **TwinCAT 2** → TwinCAT System Manager → I/O Configuration
- In particular, for TwinCAT – driver installation:  
**Fieldbus components** → Fieldbus Cards and Switches → FC900x – PCI Cards for Ethernet → Installation

Devices contain the relevant terminals for the actual configuration. All configuration data can be entered directly via editor functions (offline) or via the `scan function (online):

- **“offline”**: The configuration can be customized by adding and positioning individual components. These can be selected from a directory and configured.
  - The procedure for the offline mode can be found under <http://infosys.beckhoff.com>:  
**TwinCAT 2** → TwinCAT System Manager → IO Configuration → Add an I/O device
- **“online”**: The existing hardware configuration is read
  - See also <http://infosys.beckhoff.com>:  
**Fieldbus components** → Fieldbus Cards and Switches → FC900x – PCI Cards for Ethernet → Installation → Searching for devices

The following relationship is envisaged between the user PC and individual control elements:

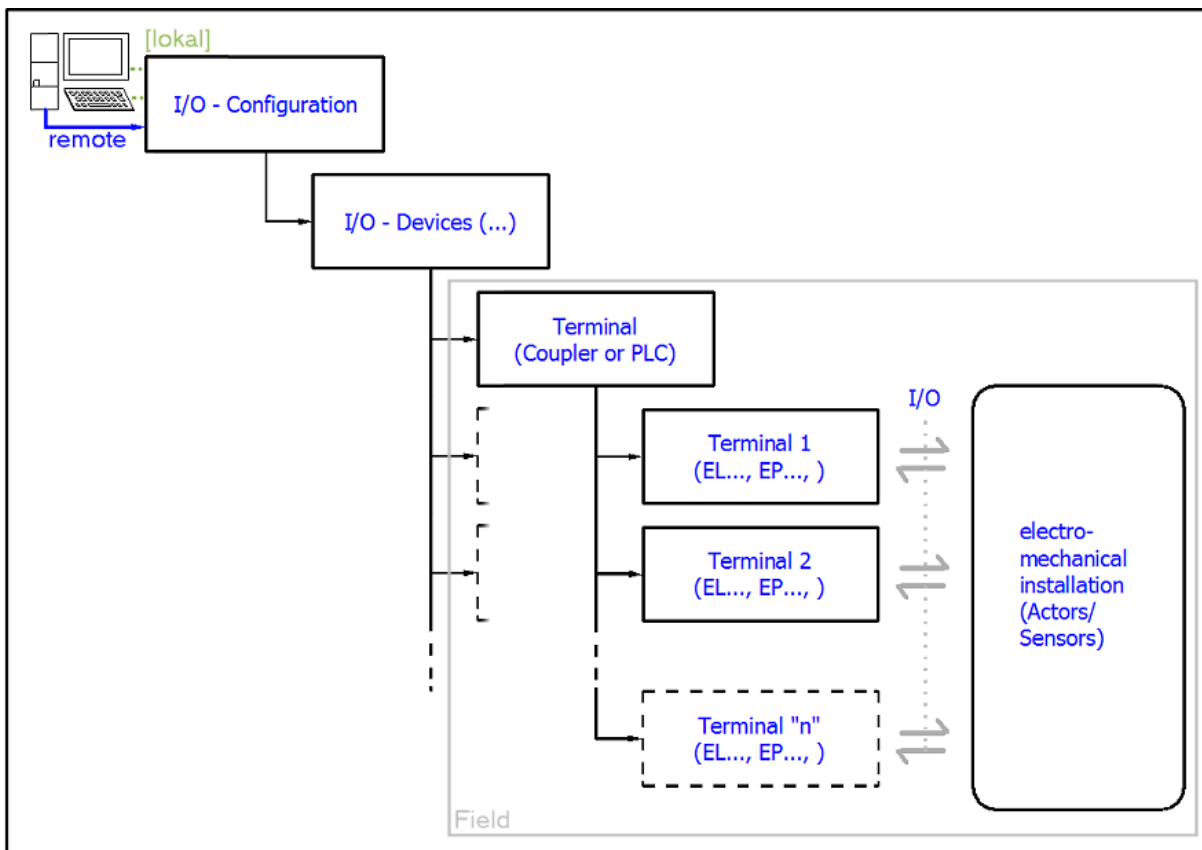


Fig. 83: Relationship between user side (commissioning) and installation

Insertion of certain components (I/O device, terminal, box...) by users functions the same way as in TwinCAT 2 and TwinCAT 3. The descriptions below relate solely to the online procedure.

### Example configuration (actual configuration)

Based on the following example configuration, the subsequent subsections describe the procedure for TwinCAT 2 and TwinCAT 3:

- **CX2040** control system (PLC) including **CX2100-0004** power supply unit
- Connected to CX2040 on the right (E-bus):  
**EL1004** (4-channel digital input terminal 24 V<sub>DC</sub>)
- Linked via the X001 port (RJ-45): **EK1100** EtherCAT Coupler
- Connected to the EK1100 EtherCAT Coupler on the right (E-bus):  
**EL2008** (8-channel digital output terminal 24 V<sub>DC</sub>; 0.5 A)
- (Optional via X000: a link to an external PC for the user interface)

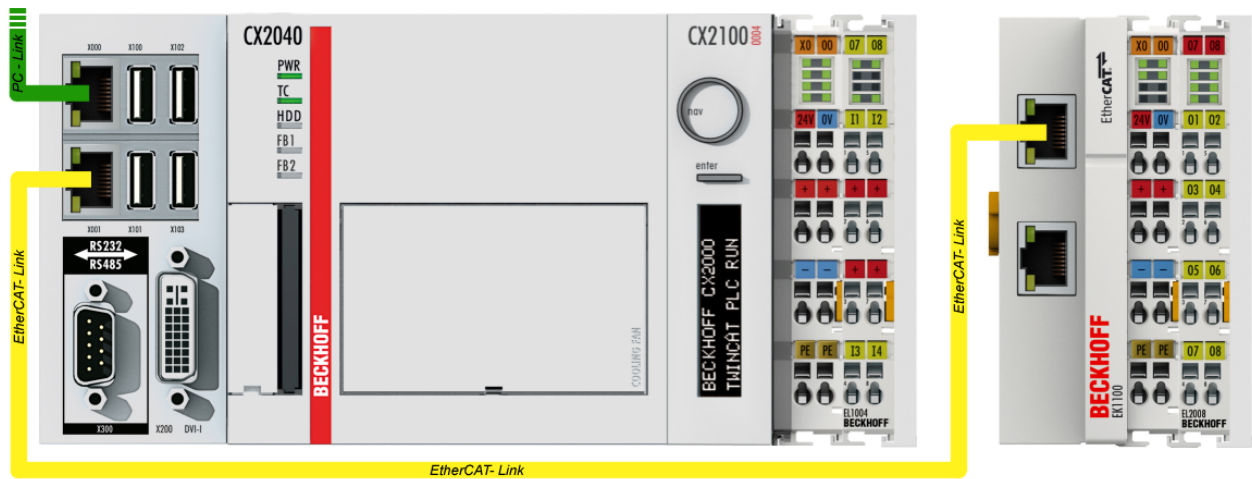


Fig. 84: Control configuration with Embedded PC, input (EL1004) and output (EL2008)

Note that all combinations of a configuration are possible; for example, the EL1004 terminal could also be connected after the coupler, or the EL2008 terminal could additionally be connected to the CX2040 on the right, in which case the EK1100 coupler wouldn't be necessary.

### 5.1.2.1 TwinCAT 2

#### Startup

TwinCAT 2 basically uses two user interfaces: the TwinCAT System Manager for communication with the electromechanical components and TwinCAT PLC Control for the development and compilation of a controller. The starting point is the TwinCAT System Manager.

After successful installation of the TwinCAT system on the PC to be used for development, the TwinCAT 2 System Manager displays the following user interface after startup:

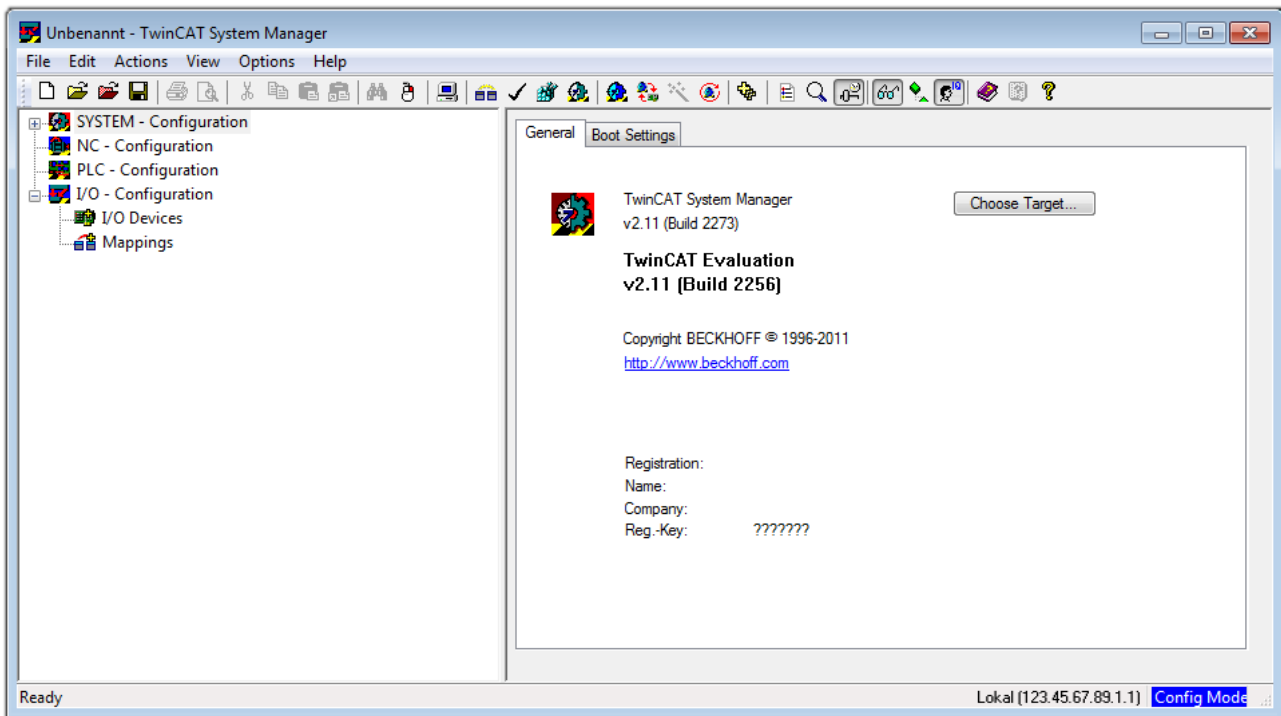



Fig. 85: Initial TwinCAT 2 user interface

Generally, TwinCAT can be used in local or remote mode. Once the TwinCAT system, including the user interface (standard) is installed on the respective PLC, TwinCAT can be used in local mode and thus the next step is “Insert Device [► 86]”.

If the intention is to address the TwinCAT runtime environment installed on a PLC remotely from another system used as a development environment, the target system must be made known first. In the menu under

“Actions” → “Choose Target System...”, the following window is opened for this via the symbol “” or the “F8” key:

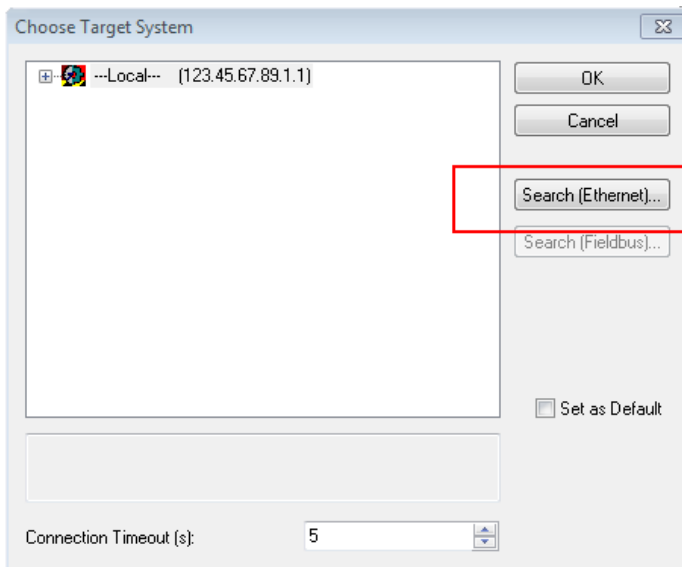


Fig. 86: Selection of the target system

Use “Search (Ethernet)...” to enter the target system. Thus another dialog opens to either:

- enter the known computer name after “Enter Host Name / IP:” (as shown in red)
- perform a “Broadcast Search” (if the exact computer name is not known)
- enter the known computer – IP or AmsNetID

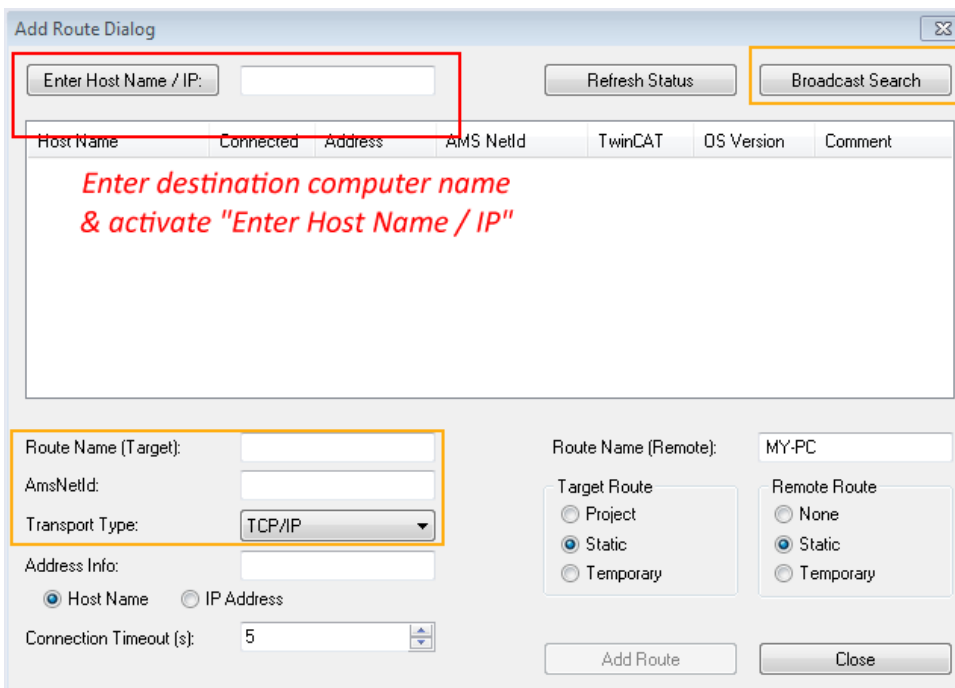
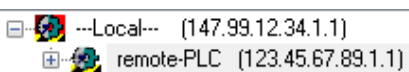


Fig. 87: specify the PLC for access by the TwinCAT System Manager: selection of the target system



Once the target system has been entered, it is available for selection as follows (a correct password may have to be entered before this):



After confirmation with “OK”, the target system can be accessed via the System Manager.

## Adding devices

In the configuration tree of the TwinCAT 2 System Manager user interface on the left, select “I/O Devices” and then right-click to open a context menu and select “Scan Devices...”, or start the action in the menu bar

via . The TwinCAT System Manager may first have to be set to “Config Mode” via  or via the menu “Actions” → “Set/Reset TwinCAT to Config Mode...” (Shift + F4).

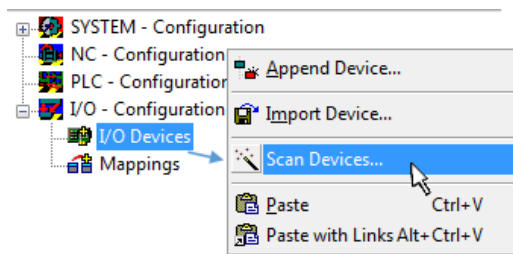


Fig. 88: Select “Scan Devices...”

Confirm the warning message, which follows, and select the “EtherCAT” devices in the dialog:

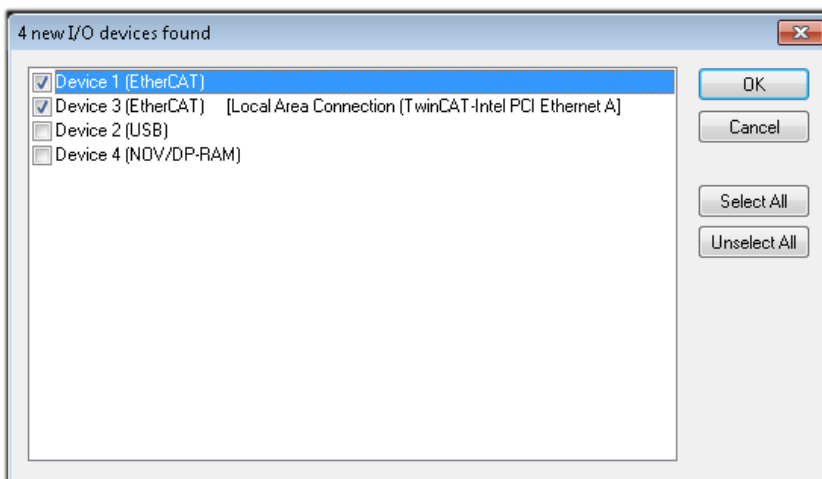


Fig. 89: Automatic detection of I/O devices: selection of the devices to be integrated

Confirm the message “Find new boxes”, in order to determine the terminals connected to the devices. “Free Run” enables manipulation of input and output values in “Config Mode” and should also be acknowledged.

Based on the [example configuration](#) [► 82] described at the beginning of this section, the result is as follows:

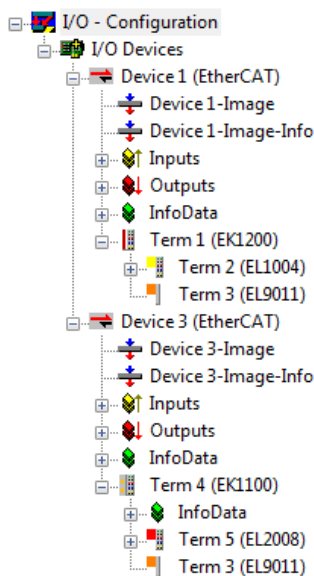


Fig. 90: Mapping of the configuration in the TwinCAT 2 System Manager

The whole process consists of two stages, which can also be performed separately (first determine the devices, then determine the connected elements such as boxes, terminals, etc.). A scan (search function) can also be initiated by selecting “Device ...” from the context menu, which then only reads the elements below which are present in the configuration:

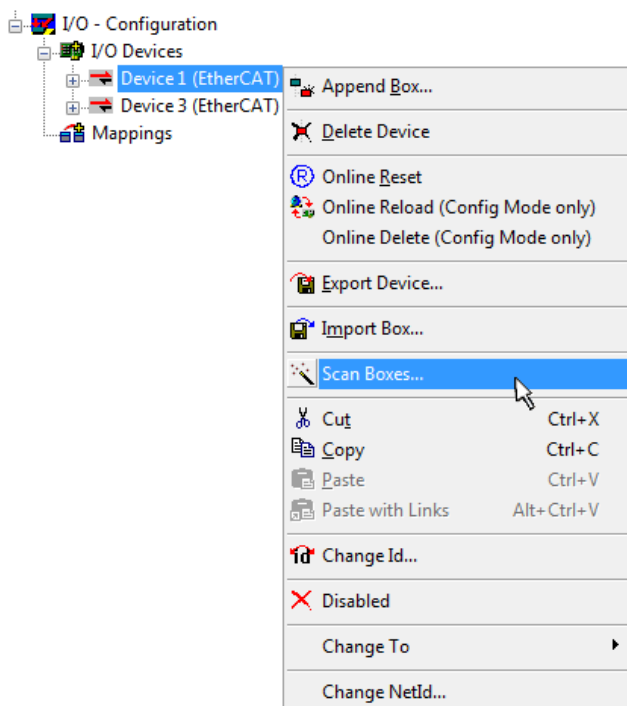


Fig. 91: Reading of individual terminals connected to a device

This functionality is useful if the actual configuration is modified at short notice.

## Programming and integrating the PLC

TwinCAT PLC Control is the development environment for generating the controller in different program environments: TwinCAT PLC Control supports all languages described in IEC 61131-3. There are two text-based languages and three graphical languages.

- **Text-based languages**
  - Instruction List (IL)
  - Structured Text (ST)

- **Graphical languages**
  - Function Block Diagram (FBD)
  - Ladder Diagram (LD)
  - The Continuous Function Chart Editor (CFC)
  - Sequential Function Chart (SFC)

The following section refers solely to Structured Text (ST).

After starting TwinCAT PLC Control, the following user interface is shown for an initial project:

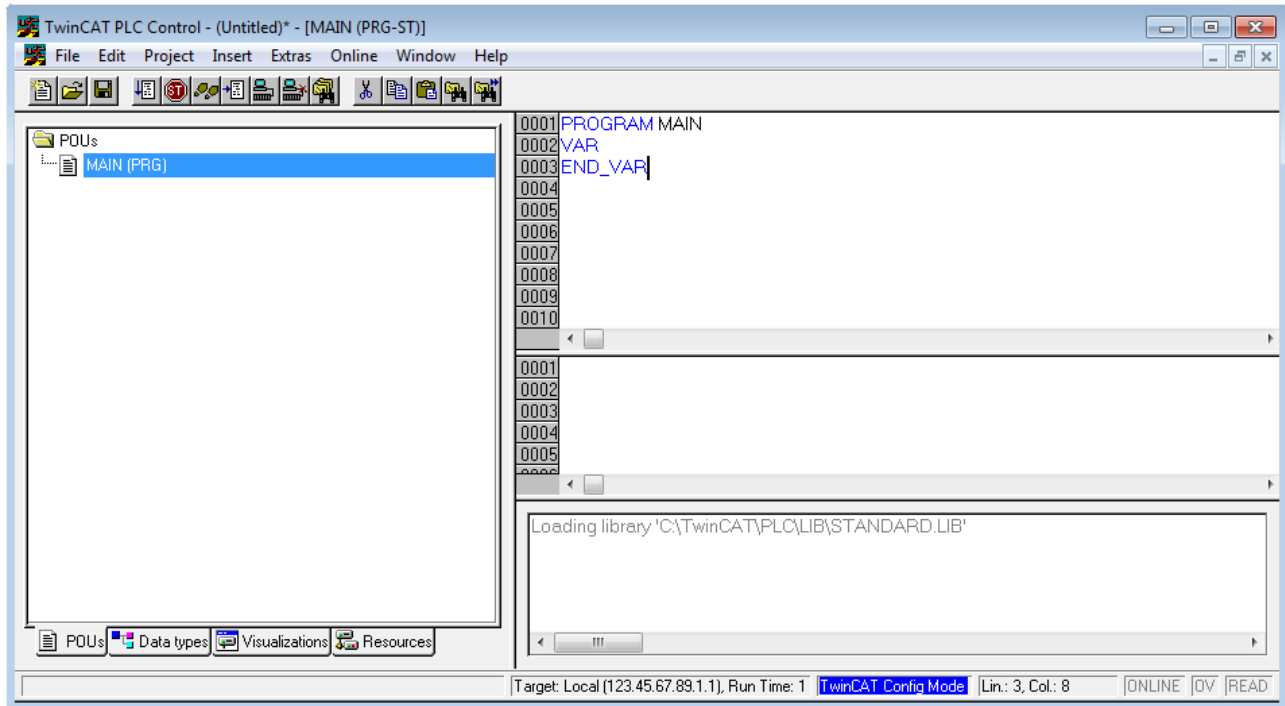


Fig. 92: TwinCAT PLC Control after startup

Example variables and an example program have been created and stored under the name "PLC\_example.pro":



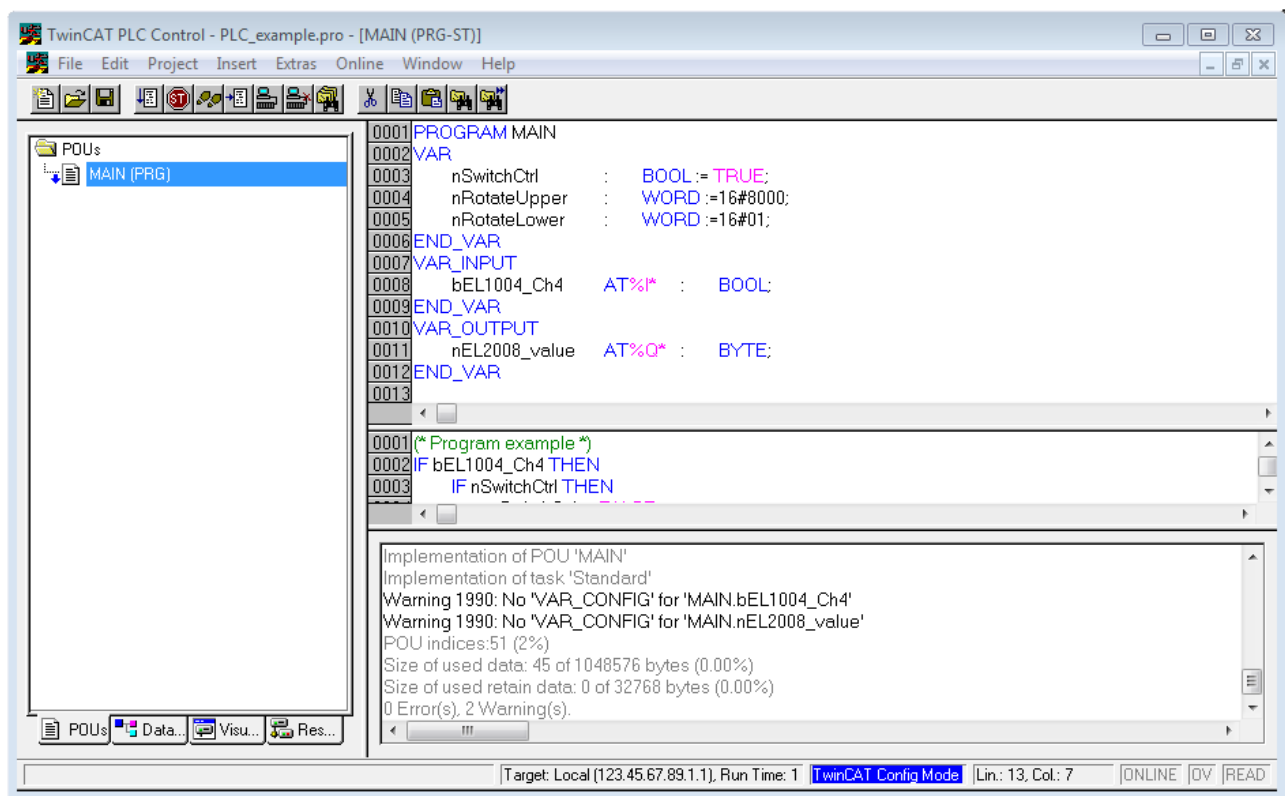


Fig. 93: Example program with variables after a compile process (without variable integration)

Warning 1990 (missing “VAR\_CONFIG”) after a compile process indicates that the variables defined as external (with the ID “AT%I\*” or “AT%Q\*”) have not been assigned. After successful compilation, TwinCAT PLC Control creates a “\*.tpy” file in the directory in which the project was stored. This file (“\*.tpy”) contains variable assignments and is not known to the System Manager, hence the warning. Once the System Manager has been notified, the warning no longer appears.

First, integrate the TwinCAT PLC Control project in the **System Manager**. This is performed via the context menu of the PLC configuration (right-click) and selecting “Append PLC Project...”:

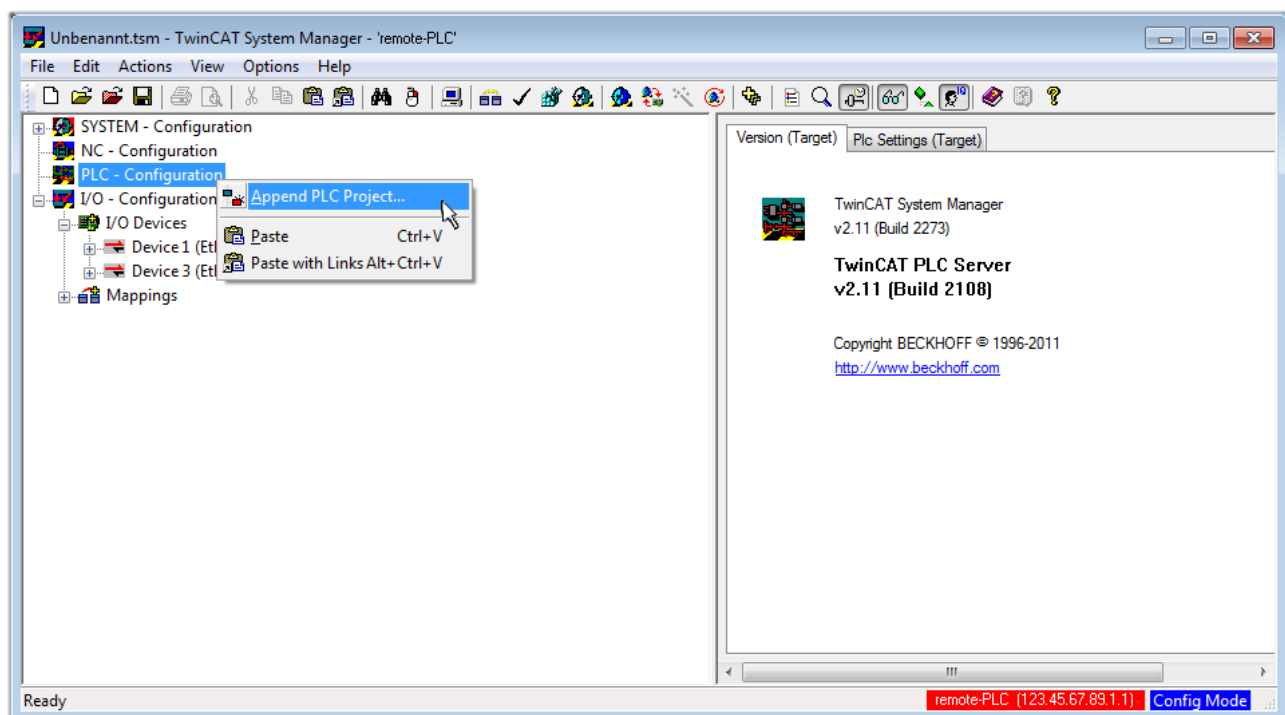


Fig. 94: Appending the TwinCAT PLC Control project

Select the PLC configuration “PLC\_example.tpy” in the browser window that opens. The project including the two variables identified with “AT” are then integrated in the configuration tree of the System Manager:

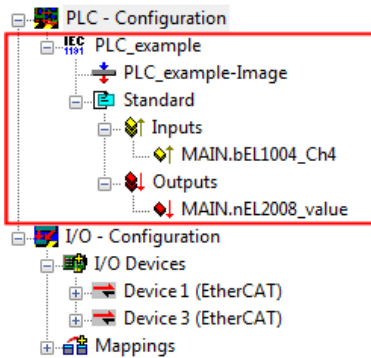


Fig. 95: PLC project integrated in the PLC configuration of the System Manager

The two variables “bEL1004\_Ch4” and “nEL2008\_value” can now be assigned to certain process objects of the I/O configuration.

### Assigning variables

Open a window for selecting a suitable process object (PDO) via the context menu of a variable of the integrated project “PLC\_example” and via “Modify Link...” “Standard”:

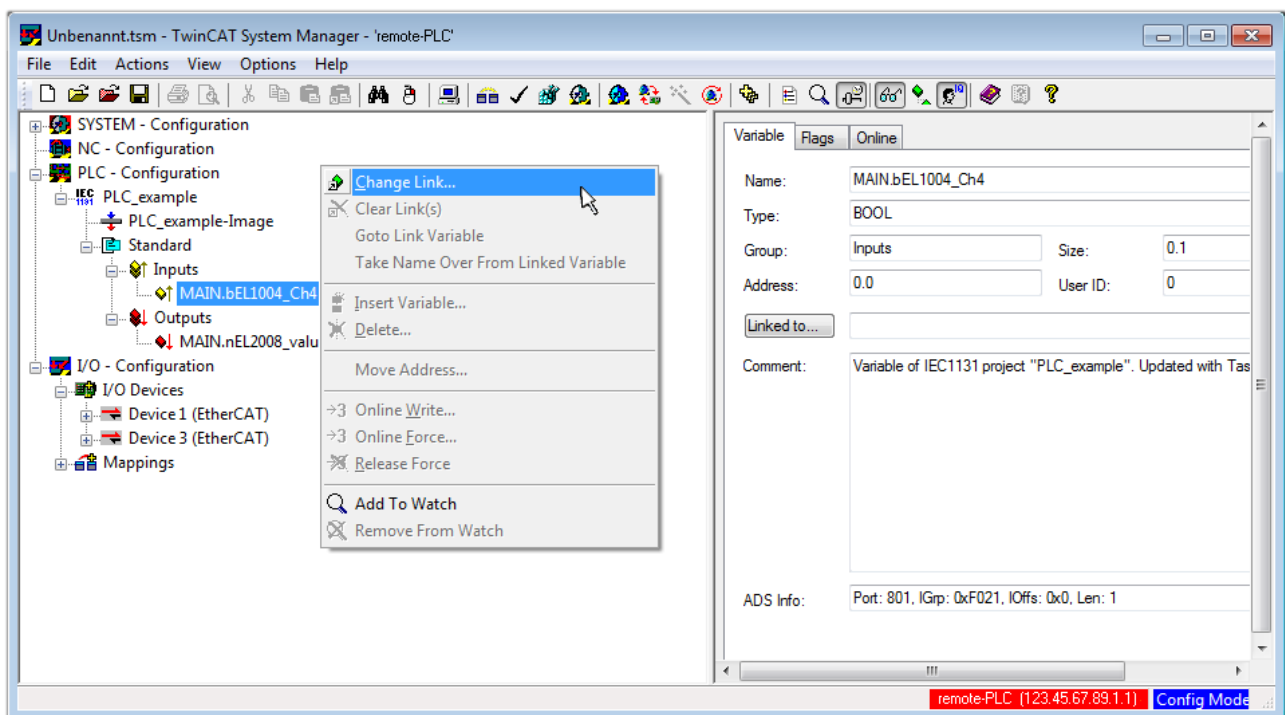


Fig. 96: Creating the links between PLC variables and process objects

In the window that opens, the process object for the “bEL1004\_Ch4” BOOL-type variable can be selected from the PLC configuration tree:

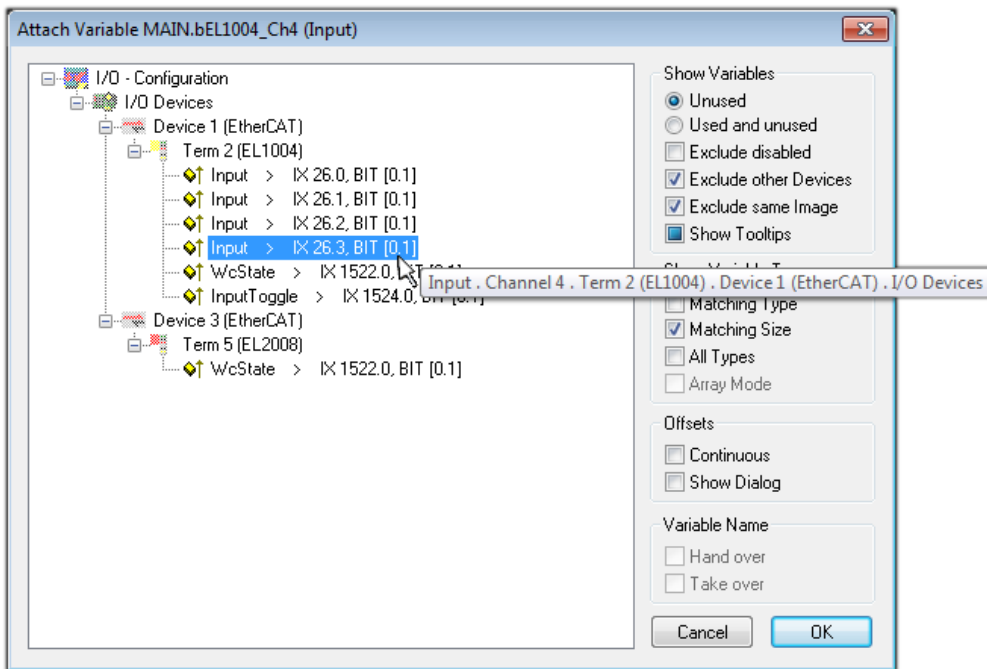


Fig. 97: Selecting BOOL-type PDO

According to the default setting, only certain PDO objects are now available for selection. In this example, the input of channel 4 of the EL1004 terminal is selected for linking. In contrast, the checkbox “All types” must be ticked to create the link for the output variables, in order to allocate a set of eight separate output bits to a byte variable in this case. The following diagram shows the whole process:

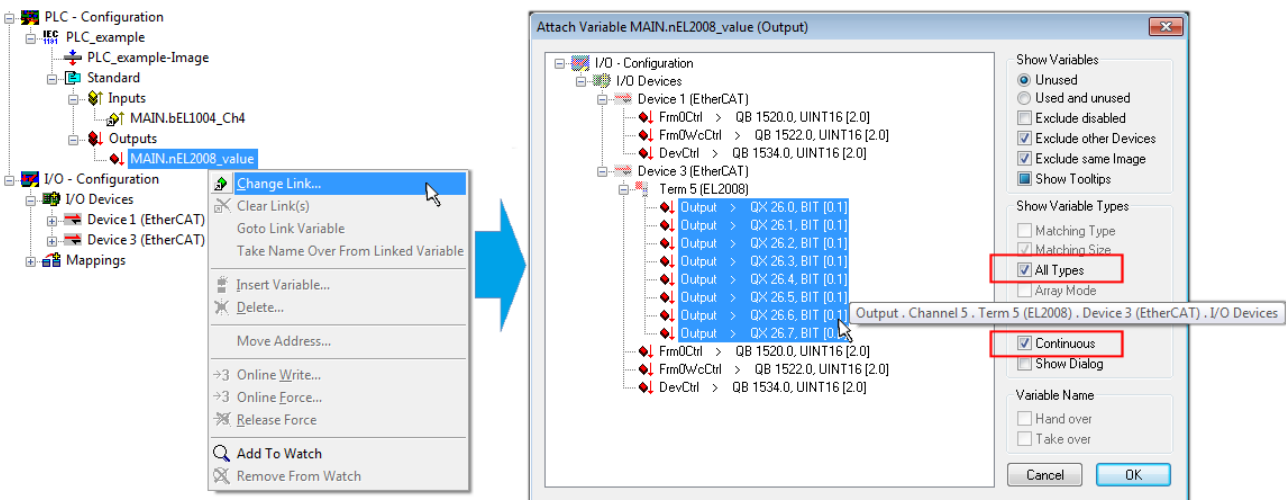



Fig. 98: Selecting several PDOs simultaneously: activate “Continuous” and “All types”

Note that the “Continuous” checkbox was also activated. This is designed to allocate the bits contained in the byte of the “nEL2008\_value” variable sequentially to all eight selected output bits of the EL2008 Terminal. It is thus possible to subsequently address all eight outputs of the terminal in the program with a byte corresponding to bit 0 for channel 1 to bit 7 for channel 8 of the PLC. A special symbol (  ) on the yellow or red object of the variable indicates that a link exists. The links can also be checked by selecting “Goto Link Variable” from the context menu of a variable. The opposite linked object, in this case the PDO, is automatically selected:

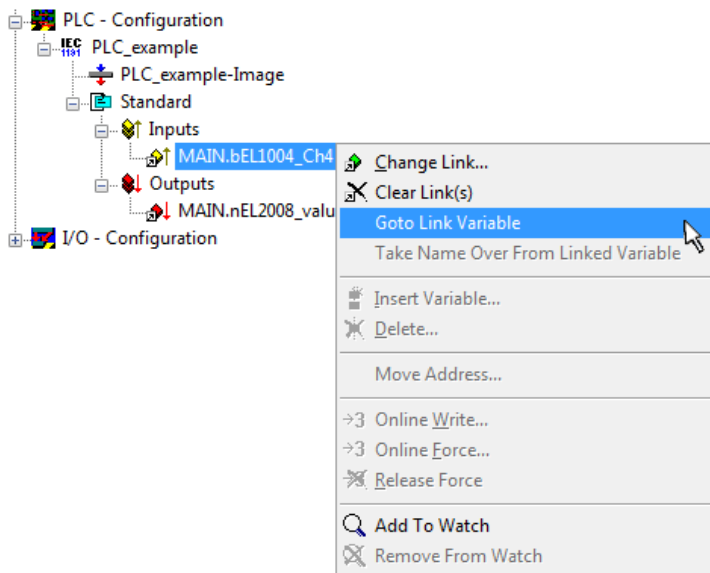

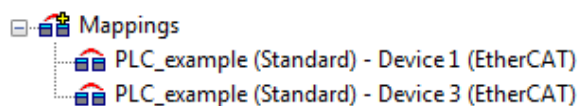


Fig. 99: Application of a “Goto Link Variable”, using “MAIN.bEL1004\_Ch4” as an example

The process of assigning variables to the PDO is completed via the menu option “Actions” → “Create

assignment”, or via .


This can be visualized in the configuration:




The process of creating links can also be performed in the opposite direction, i.e. starting with individual PDOs to a variable. However, in this example, it would not be possible to select all output bits for the EL2008, since the terminal only makes individual digital outputs available. If a terminal has a byte, word, integer or similar PDO, it is also possible to allocate this to a set of bit-standardized variables. Here, too, a “Goto Link Variable” can be executed in the other direction, so that the respective PLC instance can then be selected.

### Activation of the configuration

The allocation of PDO to PLC variables has now established the connection from the controller to the inputs and outputs of the terminals. The configuration can now be activated. First, the configuration can be verified

via  (or via “Actions” → “Check Configuration”). If no error is present, the configuration can be

activated via  (or via “Actions” → “Activate Configuration...”) to transfer the System Manager settings to the runtime system. Confirm the messages “Old configurations will be overwritten!” and “Restart TwinCAT system in Run mode” with “OK”.

A few seconds later, the real-time status **RTime 0%** is displayed at the bottom right in the System Manager. The PLC system can then be started as described below.

### Starting the controller

Starting from a remote system, the PLC control has to be linked with the embedded PC over the Ethernet via “Online” → “Choose Runtime System...”:

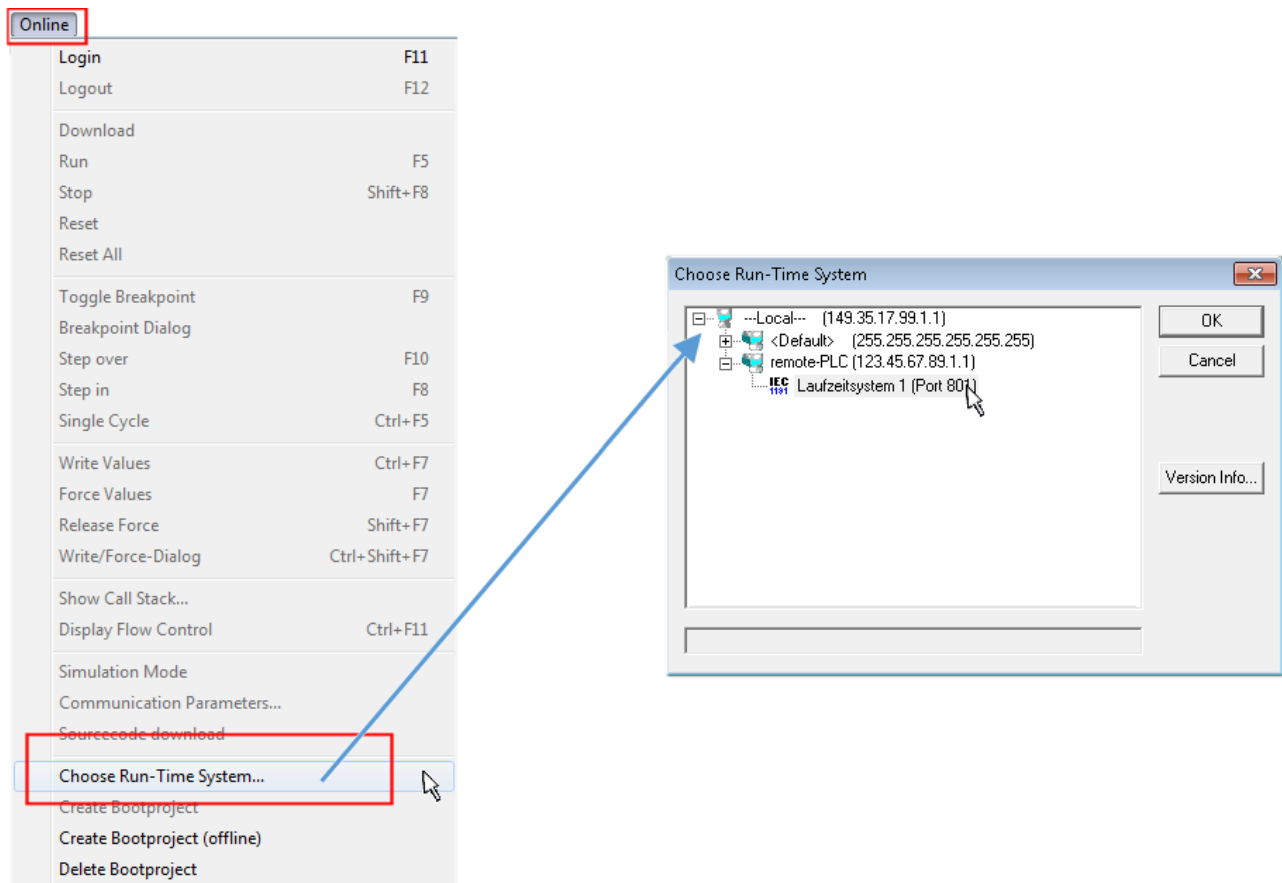



Fig. 100: Choose target system (remote)

In this example, "Runtime system 1 (port 801)" is selected and confirmed. Link the PLC with the real-time

system via the menu option "Online" → "Login", the F11 key or by clicking on the symbol . The control program can then be loaded for execution. This results in the message "No program on the controller! Should the new program be loaded?", which should be confirmed with "Yes". The runtime environment is ready for the program start:

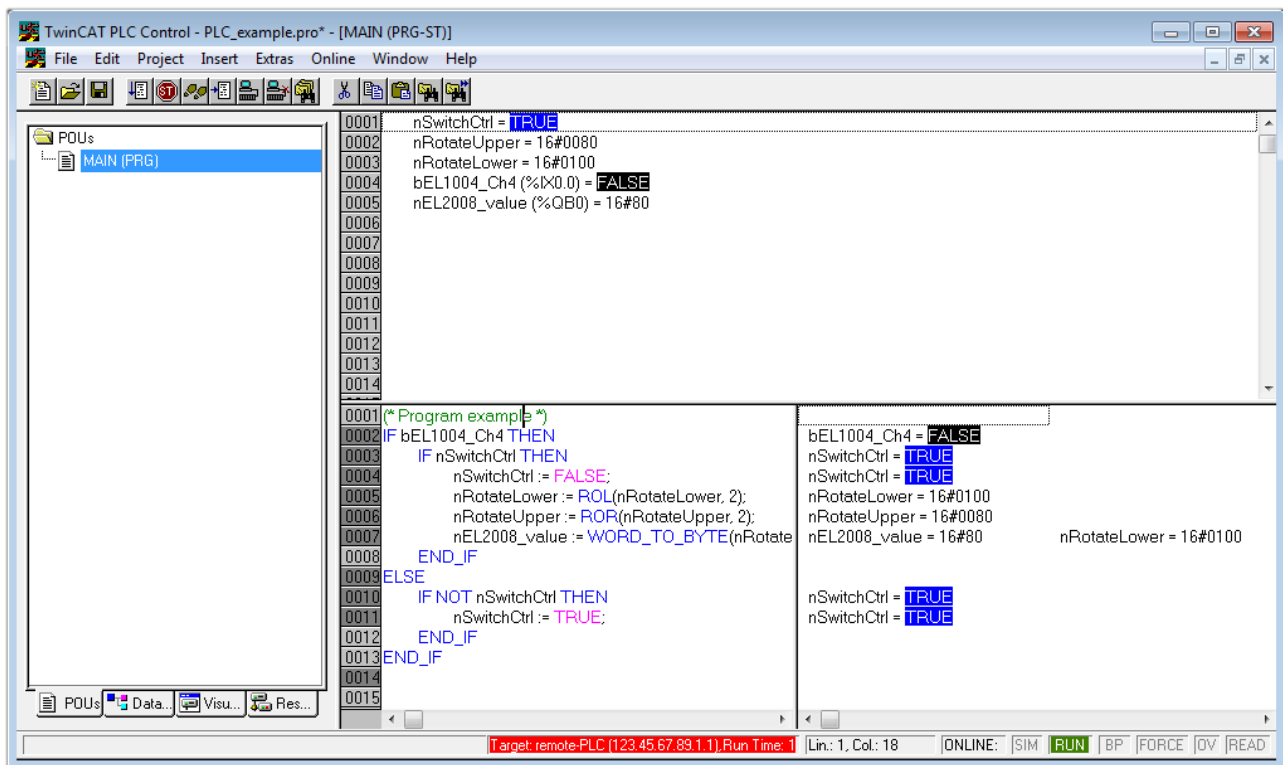


Fig. 101: PLC Control logged in, ready for program startup

The PLC can now be started via “Online” → “Run”, F5 key or .

### 5.1.2.2 TwinCAT 3


#### Startup

TwinCAT 3 makes the development environment areas available all together, with Microsoft Visual Studio: after startup, the project folder explorer appears on the left in the general window area (see “TwinCAT System Manager” of TwinCAT 2) for communication with the electromechanical components.

After successful installation of the TwinCAT system on the PC to be used for development, TwinCAT 3 (shell) displays the following user interface after startup:



Fig. 102: Initial TwinCAT 3 user interface

First create a new project via  **New TwinCAT Project...** (or under “File”→“New”→“Project...”). In the following dialog, make the corresponding entries as required (as shown in the diagram):

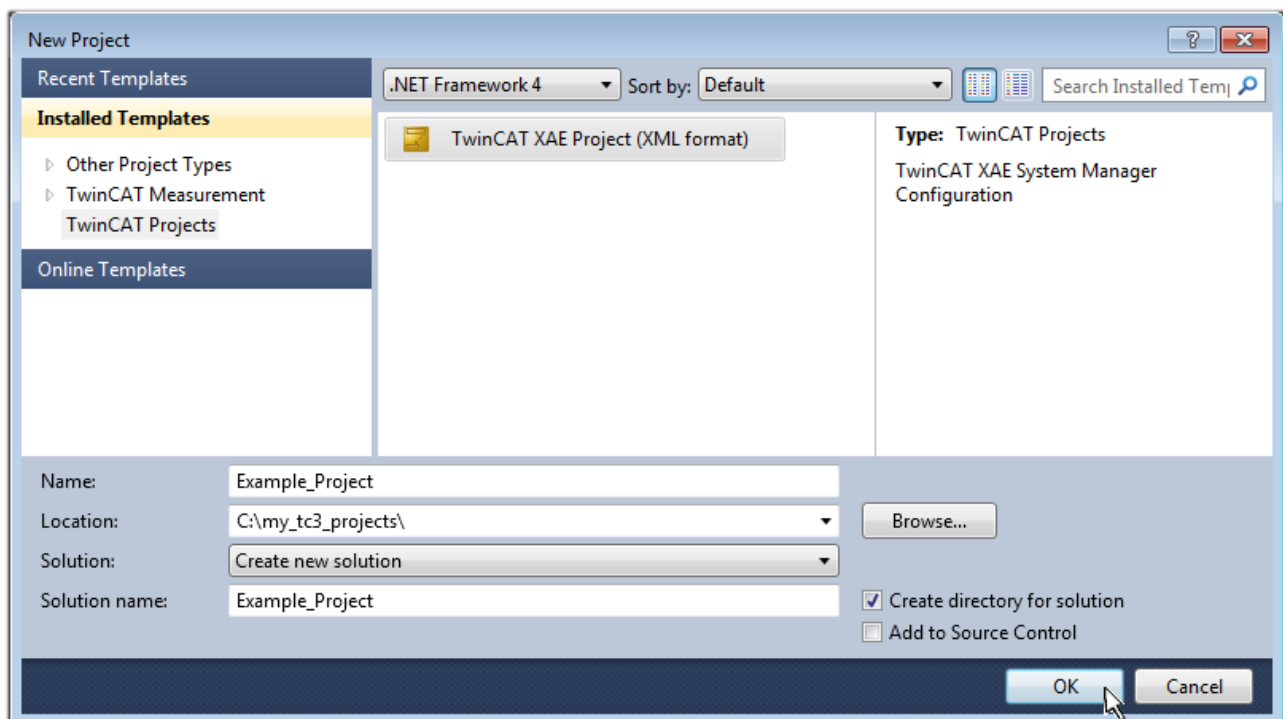


Fig. 103: Create new TwinCAT 3 project

The new project is then available in the project folder explorer:

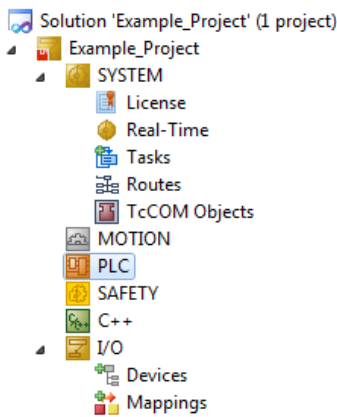
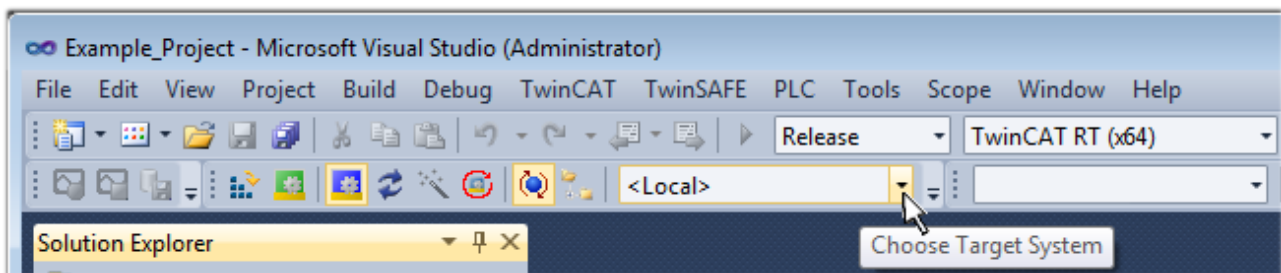


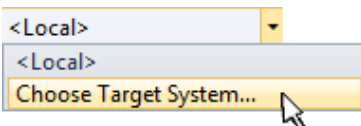
Fig. 104: New TwinCAT 3 project in the project folder explorer

Generally, TwinCAT can be used in local or remote mode. Once the TwinCAT system including the user interface (standard) is installed on the respective PLC (locally), TwinCAT can be used in local mode and the process can be continued with the next step, “[Insert Device |> 97](#)”.

If the intention is to address the TwinCAT runtime environment installed on a PLC remotely from another system used as a development environment, the target system must be made known first. Via the symbol in the menu bar:



expand the pull-down menu:



and open the following window:

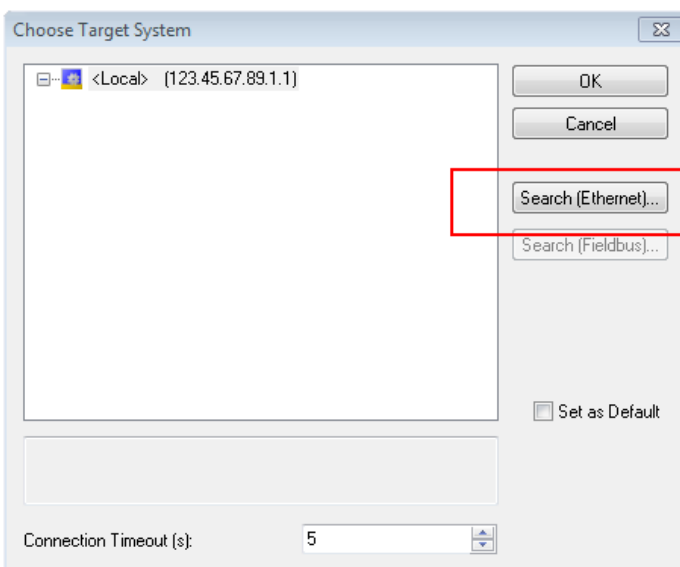


Fig. 105: Selection dialog: Choose the target system



Use “Search (Ethernet)...” to enter the target system. Thus another dialog opens to either:

- enter the known computer name after “Enter Host Name / IP:” (as shown in red)
- perform a “Broadcast Search” (if the exact computer name is not known)
- enter the known computer – IP or AmsNetId

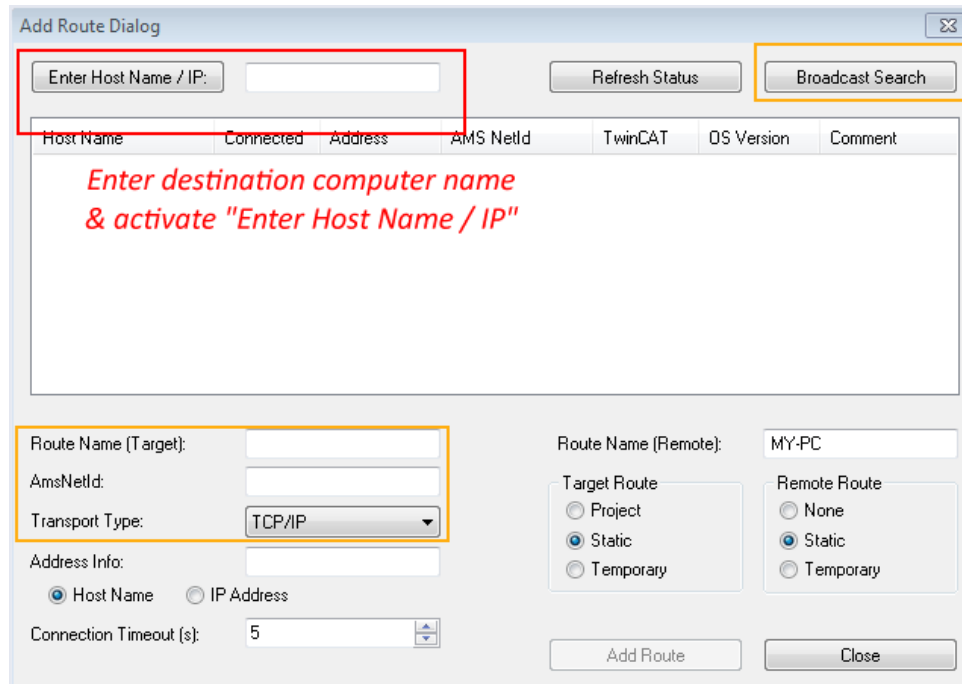
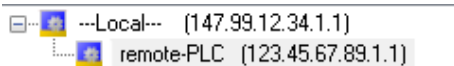


Fig. 106: specify the PLC for access by the TwinCAT System Manager: selection of the target system


Once the target system has been entered, it is available for selection as follows (the correct password may have to be entered beforehand):




After confirmation with “OK” the target system can be accessed via the Visual Studio shell.

## Adding devices

In the project folder explorer on the left of the Visual Studio shell user interface, select “Devices” within the

element “I/O”, then right-click to open a context menu and select “Scan” or start the action via  in the

menu bar. The TwinCAT System Manager may first have to be set to “Config mode” via  or via the menu “TwinCAT” → “Restart TwinCAT (Config Mode)”.

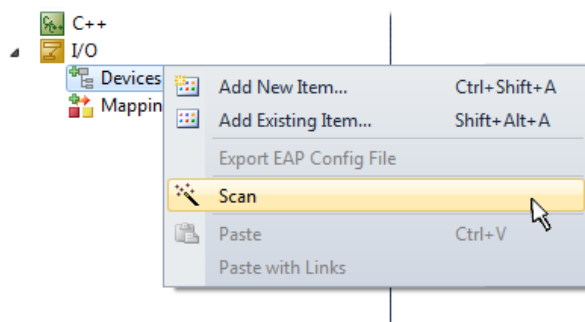


Fig. 107: Select “Scan”

Confirm the warning message, which follows, and select the “EtherCAT” devices in the dialog:

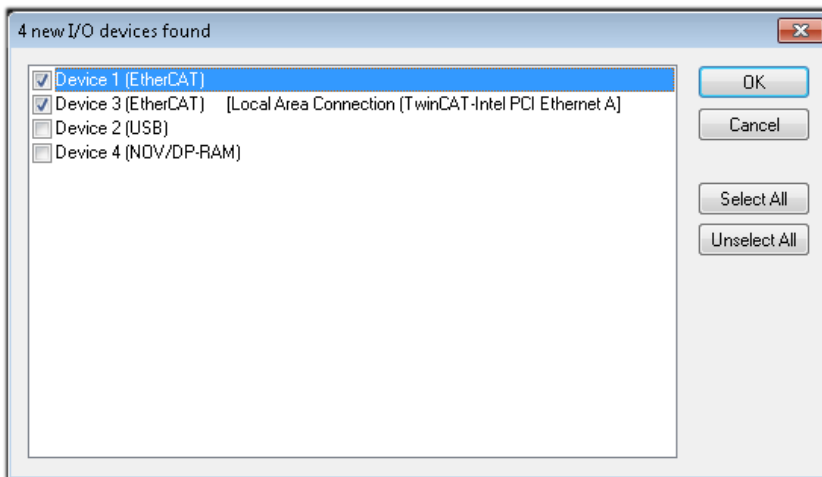


Fig. 108: Automatic detection of I/O devices: selection of the devices to be integrated

Confirm the message “Find new boxes”, in order to determine the terminals connected to the devices. “Free Run” enables manipulation of input and output values in “Config Mode” and should also be acknowledged.

Based on the [example configuration](#) [► 82] described at the beginning of this section, the result is as follows:

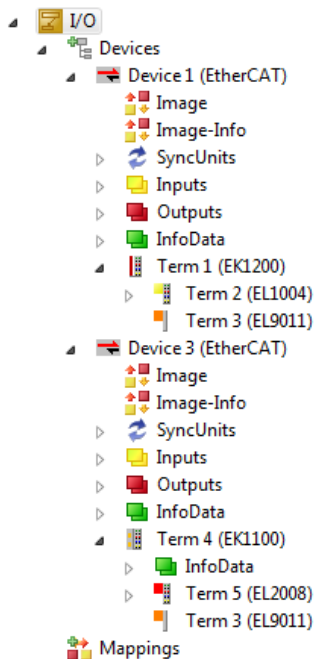


Fig. 109: Mapping of the configuration in VS shell of the TwinCAT 3 environment

The whole process consists of two stages, which can also be performed separately (first determine the devices, then determine the connected elements such as boxes, terminals, etc.). A scan (search function) can also be initiated by selecting “Device ...” from the context menu, which then only reads the elements below which are present in the configuration:

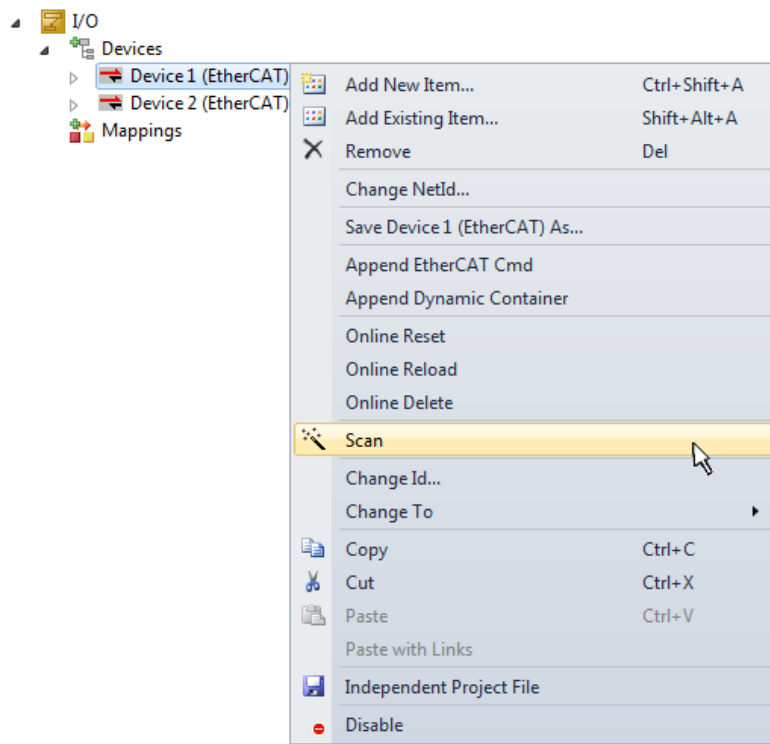


Fig. 110: Reading of individual terminals connected to a device

This functionality is useful if the actual configuration is modified at short notice.

## Programming the PLC

TwinCAT PLC Control is the development environment for generating the controller in different program environments: TwinCAT PLC Control supports all languages described in IEC 61131-3. There are two text-based languages and three graphical languages.

- **Text-based languages**
  - Instruction List (IL)
  - Structured Text (ST)
- **Graphical languages**
  - Function Block Diagram (FBD)
  - Ladder Diagram (LD)
  - The Continuous Function Chart Editor (CFC)
  - Sequential Function Chart (SFC)

The following section refers solely to Structured Text (ST).

In order to create a programming environment, a PLC subproject is added to the example project via the context menu of the “PLC” in the project folder explorer by selecting “Add New Item....”:

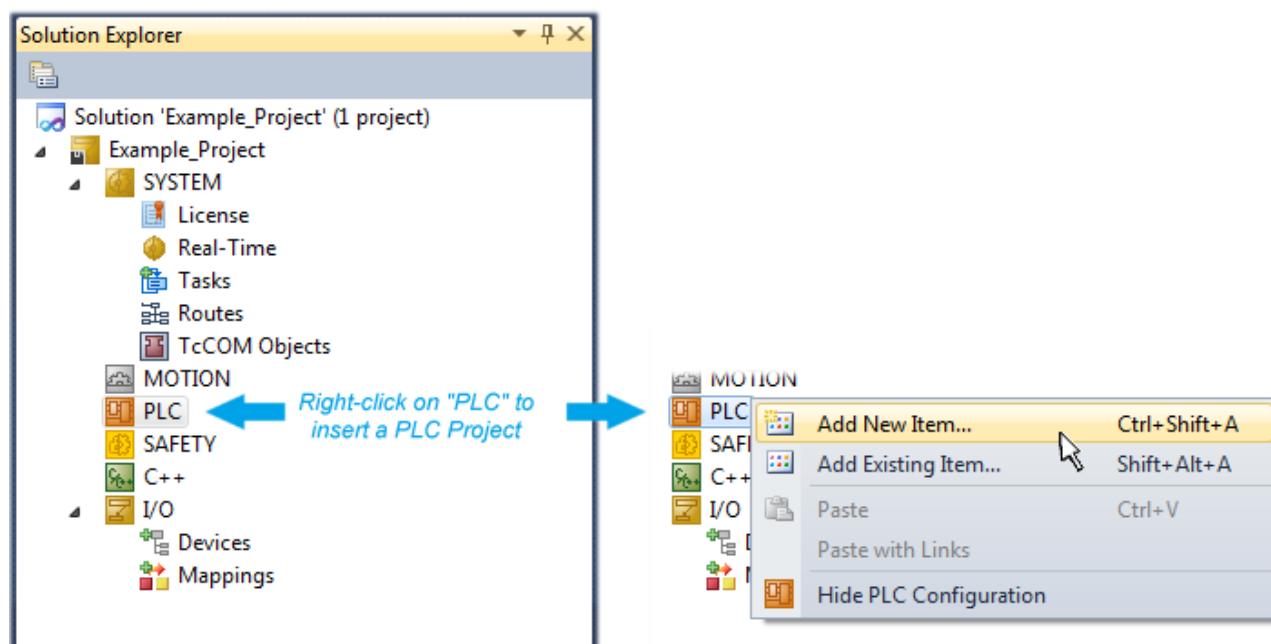


Fig. 111: Adding the programming environment in “PLC”

In the dialog that opens, select “Standard PLC project” and enter “PLC\_example” as project name, for example, and select a corresponding directory:

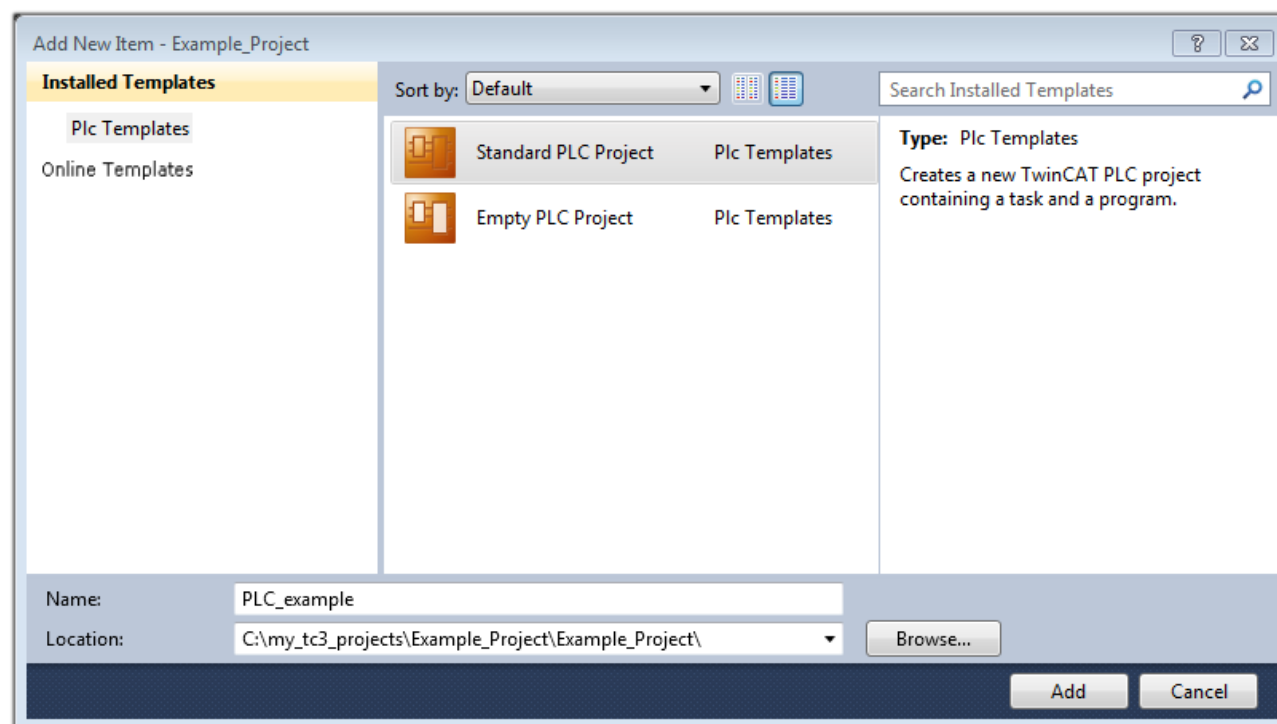


Fig. 112: Specifying the name and directory for the PLC programming environment

The “Main” program, which already exists due to selecting “Standard PLC project”, can be opened by double-clicking on “PLC\_example\_project” in “POUs”. The following user interface is shown for an initial project:

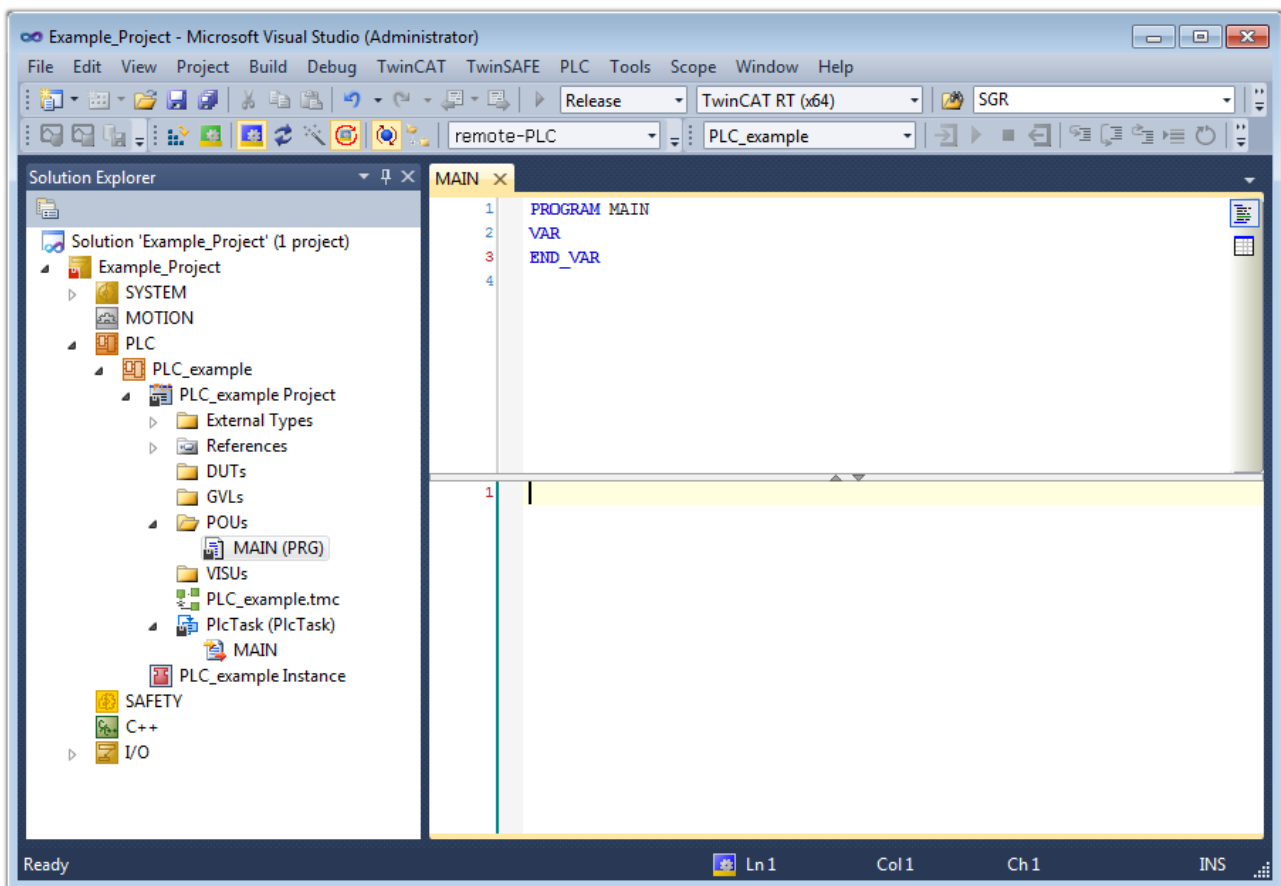


Fig. 113: Initial “Main” program for the standard PLC project

Now example variables and an example program have been created for the next stage of the process:

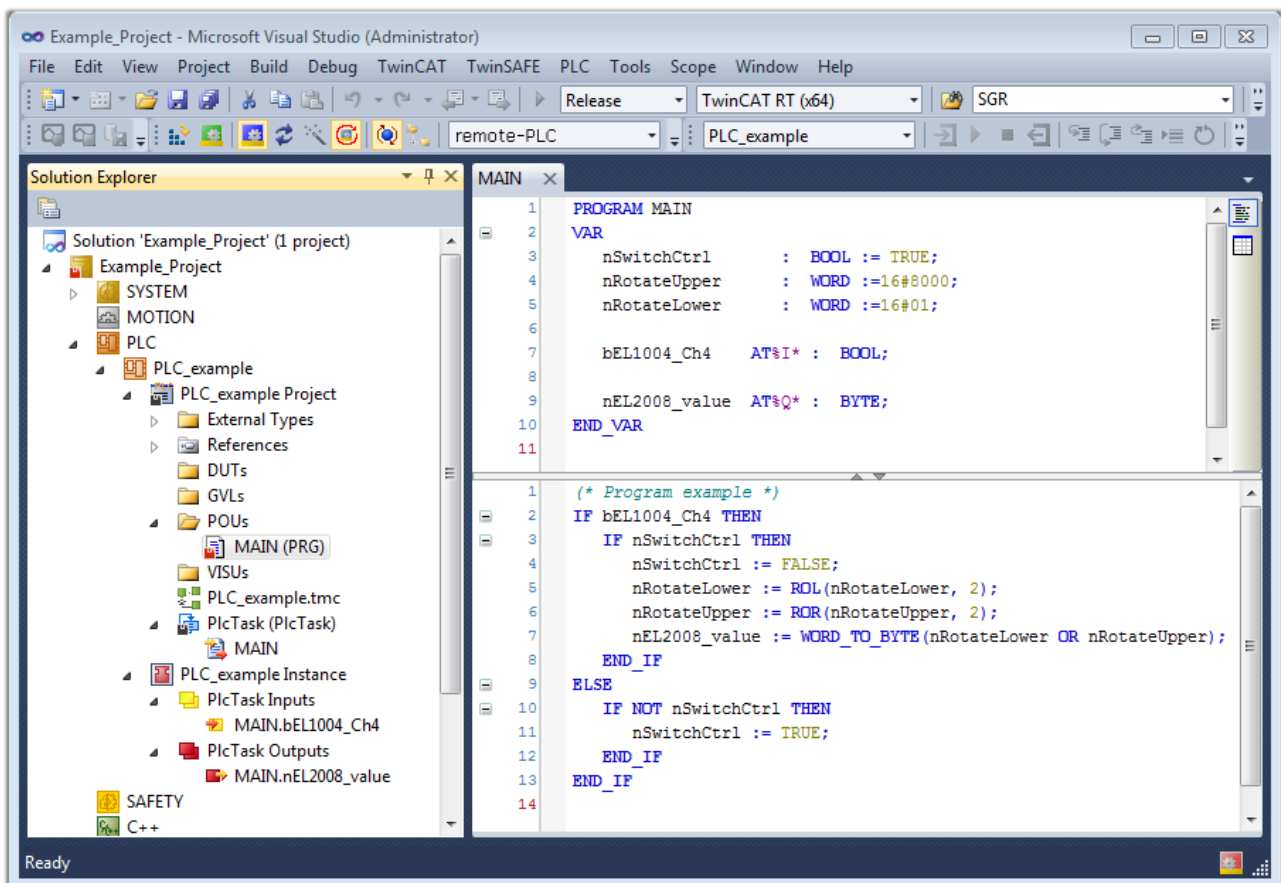


Fig. 114: Example program with variables after a compile process (without variable integration)

The control program is now created as a project folder, followed by the compile process:

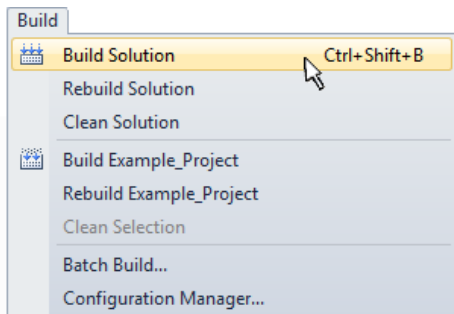
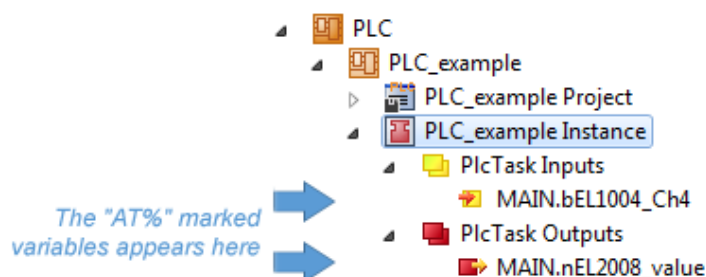


Fig. 115: Start program compilation

The following variables, identified in the ST/PLC program with “AT%”, are then available under “Assignments” in the project folder explorer:



### Assigning variables

Via the menu of an instance – variables in the “PLC” context, use the “Modify Link...” option to open a window to select a suitable process object (PDO) for linking:

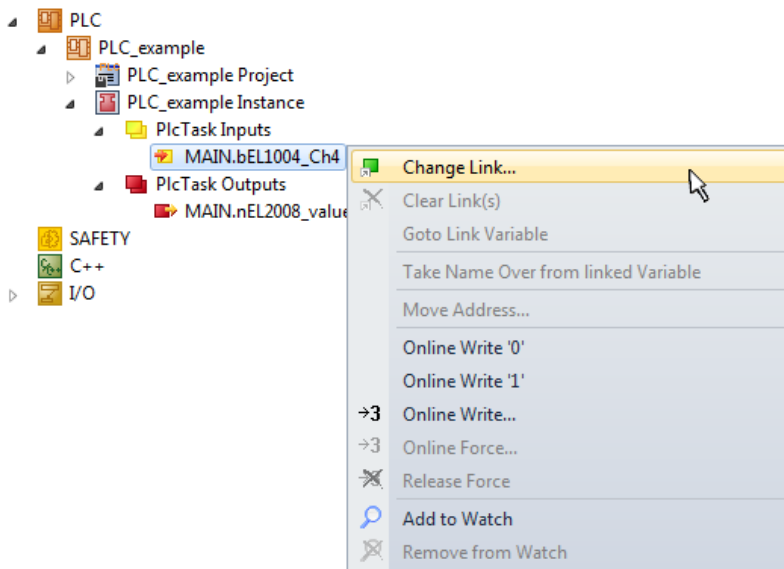


Fig. 116: Creating the links between PLC variables and process objects

In the window that opens, the process object for the “bEL1004\_Ch4” BOOL-type variable can be selected from the PLC configuration tree:

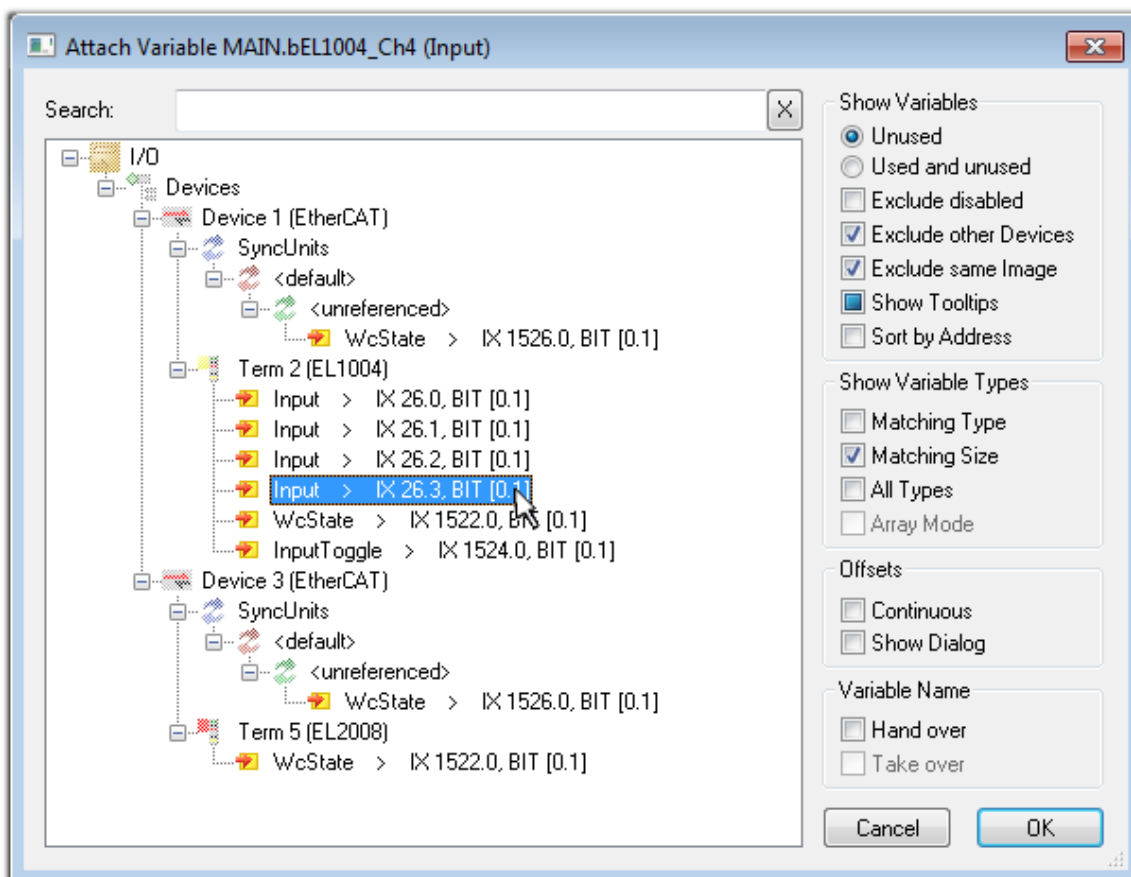


Fig. 117: Selecting BOOL-type PDO

According to the default setting, only certain PDO objects are now available for selection. In this example, the input of channel 4 of the EL1004 terminal is selected for linking. In contrast, the checkbox “All types” must be ticked to create the link for the output variables, in order to allocate a set of eight separate output bits to a byte variable in this case. The following diagram shows the whole process:

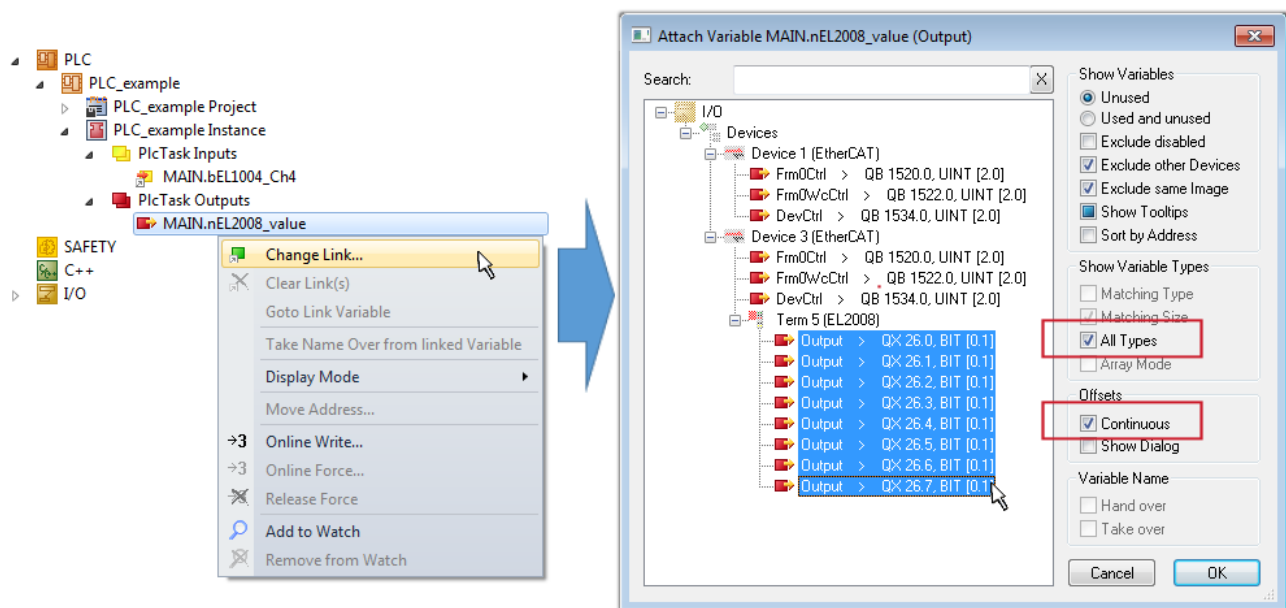



Fig. 118: Selecting several PDOs simultaneously: activate “Continuous” and “All types”

Note that the “Continuous” checkbox was also activated. This is designed to allocate the bits contained in the byte of the “nEL2008\_value” variable sequentially to all eight selected output bits of the EL2008 Terminal. It is thus possible to subsequently address all eight outputs of the terminal in the program with a byte corresponding to bit 0 for channel 1 to bit 7 for channel 8 of the PLC. A special symbol (  ) on the yellow or red object of the variable indicates that a link exists. The links can also be checked by selecting “Goto Link Variable” from the context menu of a variable. The opposite linked object, in this case the PDO, is automatically selected:

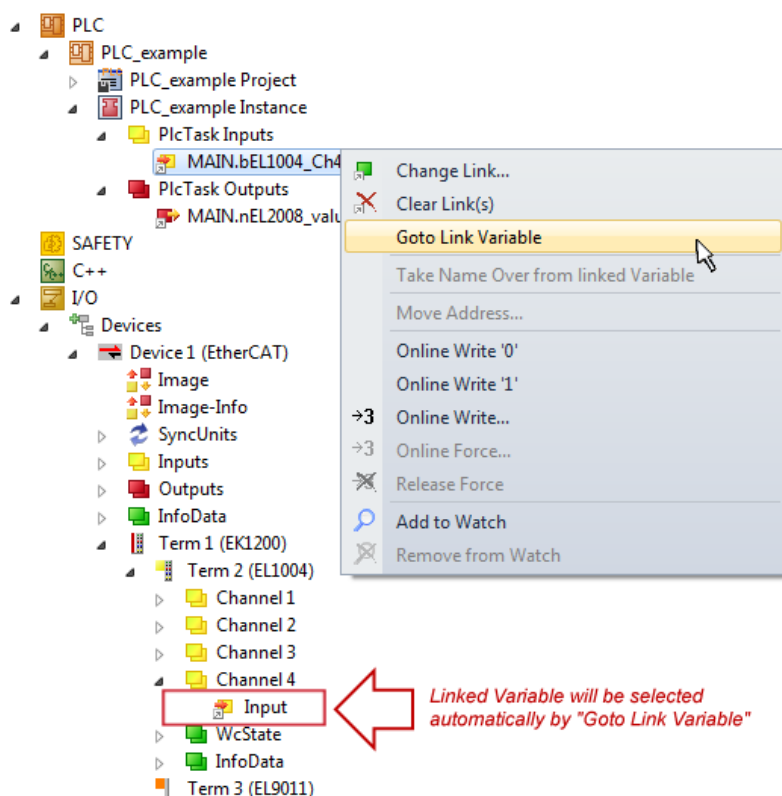


Fig. 119: Application of a “Goto Link Variable”, using “MAIN.bEL1004\_Ch4” as an example

The process of creating links can also be performed in the opposite direction, i.e. starting with individual PDOs to a variable. However, in this example, it would not be possible to select all output bits for the EL2008, since the terminal only makes individual digital outputs available. If a terminal has a byte, word,



integer or similar PDO, it is also possible to allocate this to a set of bit-standardized variables. Here, too, a “Goto Link Variable” can be executed in the other direction, so that the respective PLC instance can then be selected.

### ● Note on type of variable assignment

**1** The following type of variable assignment can only be used from TwinCAT version V3.1.4024.4 onwards and is only available for terminals with a microcontroller.

In TwinCAT, a structure can be created from the mapped process data of a terminal. An instance of this structure can then be created in the PLC, so it is possible to access the process data directly from the PLC without having to declare own variables.

The procedure for the EL3001 1-channel analog input terminal -10...+10 V is shown as an example.

1. First, the required process data must be selected in the “Process data” tab in TwinCAT.
2. After that, the PLC data type must be generated in the “PLC” tab via the check box.
3. The data type in the “Data Type” field can then be copied using the “Copy” button.

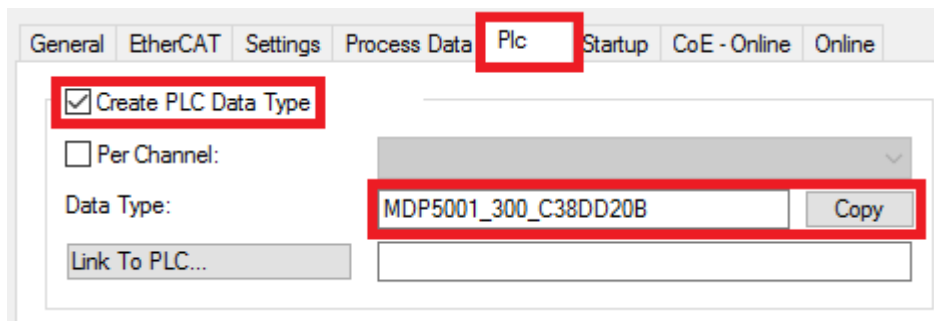


Fig. 120: Creating a PLC data type

4. An instance of the data structure of the copied data type must then be created in the PLC.

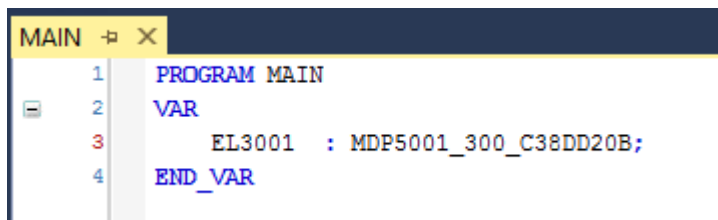


Fig. 121: Instance\_of\_struct

5. Then the project folder must be created. This can be done either via the key combination “CTRL + Shift + B” or via the “Build” tab in TwinCAT.
6. The structure in the “PLC” tab of the terminal must then be linked to the created instance.

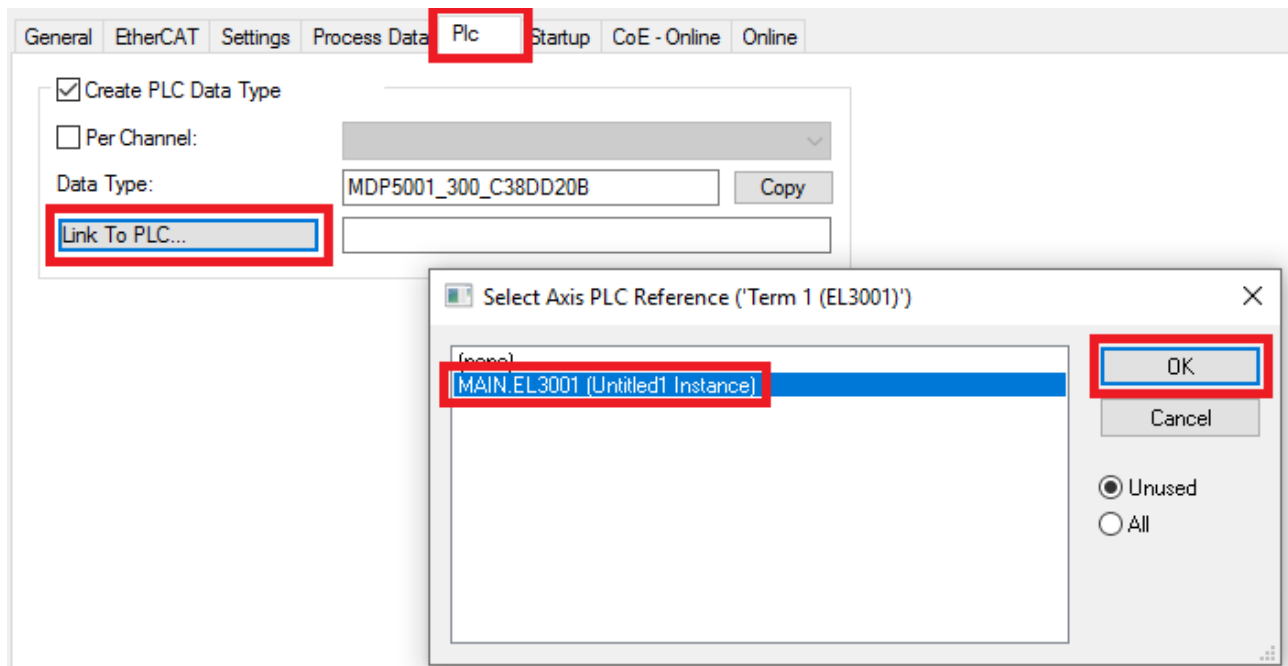


Fig. 122: Linking the structure

7. In the PLC, the process data can then be read or written via the structure in the program code.

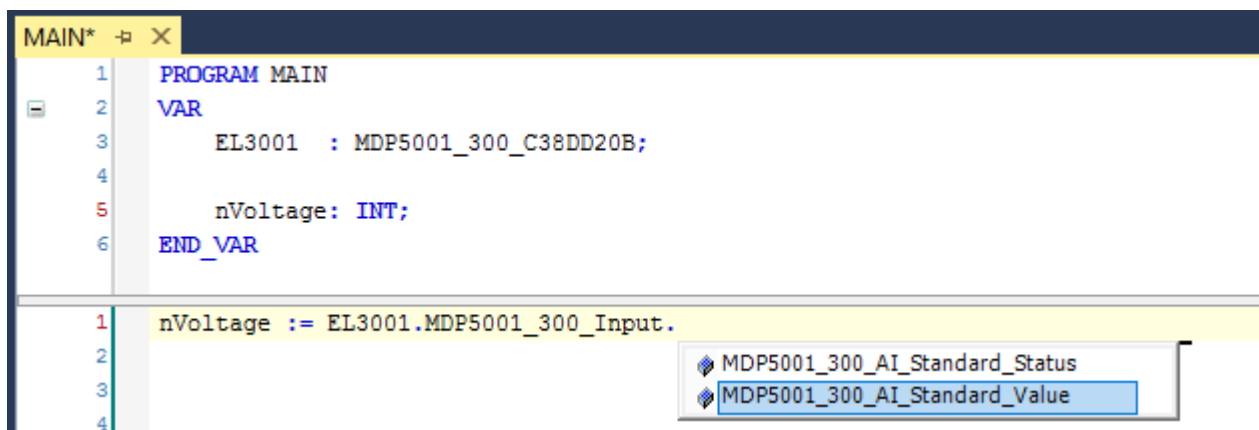

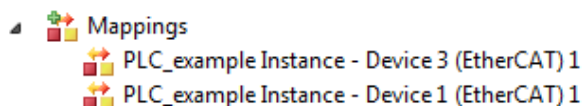


Fig. 123: Reading a variable from the structure of the process data

### Activation of the configuration

The allocation of PDO to PLC variables has now established the connection from the controller to the inputs

and outputs of the terminals. The configuration can now be activated with  or via the menu under "TwinCAT" in order to transfer the settings of the development environment to the runtime system. Confirm the messages "Old configurations will be overwritten!" and "Restart TwinCAT system in Run mode" with "OK". The corresponding assignments can be seen in the project folder explorer:





A few seconds later, the corresponding status of the Run mode is displayed in the form of a rotating symbol



at the bottom right of the VS shell development environment. The PLC system can then be started as described below.

## Starting the controller

Select the menu option “PLC” → “Login” or click on  to link the PLC with the real-time system and load the control program for execution. This results in the message “No program on the controller! Should the new program be loaded?”, which should be acknowledged with “Yes”. The runtime environment is ready for

the program to be started by clicking on symbol , the “F5” key or via “PLC” in the menu, by selecting “Start”. The started programming environment shows the runtime values of individual variables:

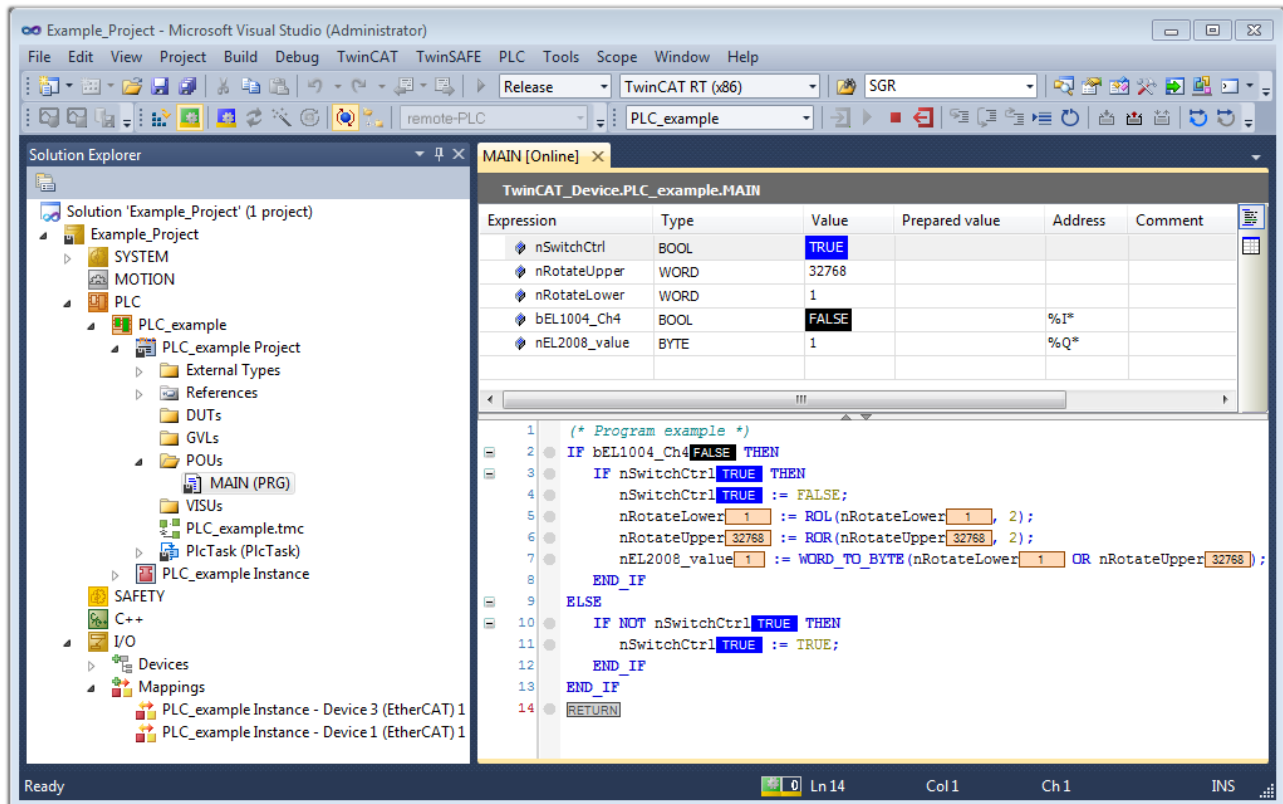




Fig. 124: TwinCAT 3 development environment (VS shell): logged-in, after program startup

The two operator control elements for stopping  and logout  result in the required action (also, “Shift + F5” can be used for stop, or both actions can be selected via the PLC menu).

## 5.2 Notes on commissioning

- Please note that a correct display of the LEDs is only possible in EtherCAT State OP. If the terminal changes to another state or if the power supply on the field side is lost, no frame is sent any more. It is possible that a last frame sent is faulty due to interference. This error will be present until the terminal is in the OP again and new data is sent. **Notice** Since the LEDs retain the last state or color (even if faulty due to malfunction) as long as the power supply is present, these LEDs must never be used for safety-related applications.
- Resetting to factory settings via the CoE object *0x1011:01 Restore default parameters* is only possible in PREOP state.
- To minimize interference and thus incorrect display of the pixels, we recommend the use of an external filter
- The cyclic frame output is enabled by default in the CoE object *0x80p0:02 Enable Cyclic Frame Output*. The data from the frame buffer is sent to the strip every 250 ms. Without an additional command, all data from the frame buffer will be sent cyclically.

### 5.2.1 Connection

Before commissioning in TwinCAT, connect the terminal as follows:

1. Supply the EL2574 via the power contacts with 24 V<sub>DC</sub>.
2. Connect the data line (and clock line) to the appropriate terminal points.
3. Connect the LED supply via the terminal or directly to the LEDs (see note from chapter "[Supply of the connected LEDs](#) [► 108]"). Note here that the terminal only serves as a rail.

#### **i**

#### Connection instructions

- Make sure that the supply of the LED has the same 0 V potential as the 0 V power contact, as this is the reference ground of the terminal. The 0 V power contact is also internally connected to terminal points GND.
- Shielded cables are mandatory for connecting the LED.
- For stable data transmission, the data and supply lines for the LED should be kept as short as possible and should not be routed past switching elements.

### 5.2.2 Supply of the connected LEDs

The supply voltage for the connected LEDs is not generated from the EL2574. This supply must be provided externally. Either the connected LEDs can be supplied directly or the supply voltage for the LED is optionally routed via the connection points "Ext. 5...24 V" and "GND". A derating of the current must be taken into account for the supply via the corresponding connection points of the EL2574 (applies to HW < 02).

- < 45°C ambient temperature: with 6 A supply
- 45...50°C ambient temperature: with 3 A supply
- 50°C ambient temperature: no supply
- With the ZB8610 8 A fan cartridge: supply over the entire temperature range

### 5.2.3 Adjustable parameters

The following parameters can be set:

- [Number of Pixels \[► 109\]](#) (0x80p0:11 "Number of Pixel")
- [Chip Type \[► 109\]](#) (0x80p0:12 "Chip Type")
- [Color Format \[► 109\]](#) (0x80p0:13 "Color Format")
- [Gamma Correction \[► 109\]](#) (0x80p0:1E "Gamma Correction")
- [Total brightness \[► 110\]](#) (0x80p0:1F "Brightness Scale")
- [Behavior in watchdog case \[► 110\]](#) (0x80p0:03 "Enable Watchdog", 0x80p0:24 to 0x80p0:027 " Watchdog Default Color")
- [Cyclic Fame Output \[► 110\]](#) (0x80p0:02 "Enable Cyclic Frame Output")
- [Custom Settings \[► 110\]](#) (0x80p0:01 "Enable Custom Settings")
- [Current setting \[► 110\]](#) (0x80p0:20 to 0x80p0:23 "Set Driver Current")

#### Number of pixels

The connected number of pixels must be specified per channel in the CoE object *0x80p0:11 Number of Pixel*. An error message will be output if pixels outside of this range are written by the process data.

A total of 2048 pixels can be written by the EL2574.

- These can be operated on one channel (1x 2048 pixels) or
- can be distributed to two channels (2x 1024 pixels), three channels (3x 512 pixels) or four channels (4x 512 pixels).

The maximum number of pixels per channel is determined by the number of configured channels.

#### Chip type

The LED chip type used must be specified per channel in the CoE object *0x80p0:12 Chip Type*. Only then the high, low and reset timings specified by the chip type are observed.

Depending on whether a channel is configured for a synchronous chip type (with clock) or an asynchronous type, different chip types are available for selection in the drop-down list.

#### Color format

The LEDs differ in their number of colors (RGB or RGBW) and in the sequence or format of the interconnected colors.

Here the connected color format must be specified in the CoE object *0x80p0:13 Color Format* so that the respective color value is also correctly displayed in the process data.

#### Gamma correction

The gamma value can be used to adjust the scaling of the output. The brightness behavior can, for example, be approximated to the perception of the human eye.

Gamma can be set per channel in the CoE object *0x80p0:1E Gamma Correction*. If this value is set to 1.0, the scaling is linear.

The output is calculated using the following relationship:  $(\text{Current\_process\_data\_value} / \text{Maximum\_process\_data\_value})^{\text{Gamma}}$

The adjustable value is between 0.2 and 5. The specified gamma value is applied to all colors.

#### Example application for the linearization of brightness for the human eye:

The brightness perceived by humans increases more steeply in dark areas and less steeply in bright areas. The human eye is assigned a gamma value of approx. 0.3 to 0.5. If you want to perceive the brightness signal of a display device (e.g. a monitor or an illumination device) linearly, you have to pre-distort it with the

reciprocal of the gamma value (approx.  $1/0.3 = 3.3$  to  $1/0.5 = 2$ ) so that the two non-linearities cancel each other out and the course appears linear to the observer.

For the gamma values, a value between 2 and 3.3 would thus have to be specified for the human eye.

### Total brightness

The total brightness of a channel can be specified via the CoE object *0x80p0:1F Brightness Scale*.

This value is set to 1.0 by default, which corresponds to 100 % brightness. The brightness can be set between 0 and 100%.

### Behavior in watchdog case

Normally, the last status is maintained in the watchdog case, since no new data is sent from the terminal. However, it is also possible to display a specified color on all connected LEDs on a channel in the event of a bus communication failure.

- To this end, the object *0x80p0:03 Enable Watchdog Default Color* must be set to TRUE in the CoE.
- The color is then specified via the objects *0x80p0:24 - 0x80p0:27 Watchdog Default Color*.

This can be used, for example, for error indication, so that in the watchdog case the LEDs light up red, for example, to signal the error. In the watchdog case, the values entered in the CoE for gamma and total brightness are retained.

### Cyclic frame output

The cyclic frame output is enabled by default in the CoE object *0x80p0:02 Enable Cyclic Frame Output*.

The data from the frame buffer is sent to the strip every 250 ms. As a result, possible false images caused by interference and influences on the cable are visible for a maximum of 250 ms. This cyclic sending can also be switched off if the application requires it.

### Custom Settings

Custom settings can be made for the data rate, duty cycle, reset time and level.

These must be enabled via the CoE object *0x80p0:01 Enable Custom Settings*.

### Current specification

Some chip types require the specification of a current (e.g. TM1814).

This can be specified via the CoE objects *0x80p0:20 - 0x80p0:23 Set Driver Current*. If this information is not required by the chip type it is ignored.

## 5.2.4 Updating procedure

The terminal has different buffers internally.

The data from the PLC is written into the first buffer (pixel buffer) in order to build the "image" for connected LEDs. The data is sent from the PLC either via a Command or a Write command together with an Execute.

From this buffer the finished data is written to the next buffer (frame buffer). For this purpose, the update command is used with an execute. In order to finally send the data out of this buffer to the LEDs, a Send command must be executed with the Execute. This process is shown graphically below.

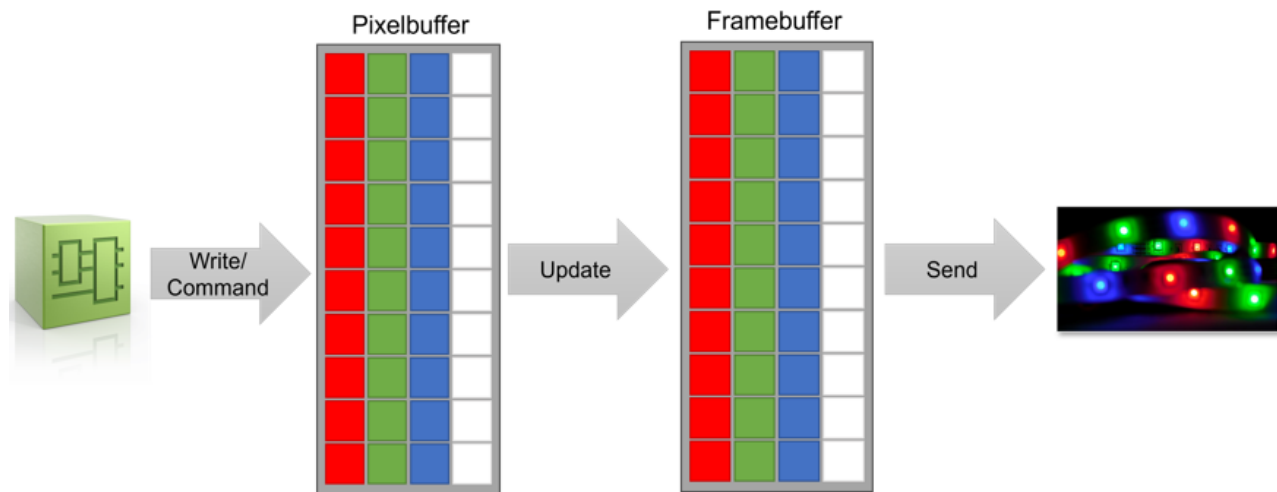


Fig. 125: Buffer sequence

For the timings to transfer data from the PLC to the connected LEDs different times have to be considered.

### PLC to pixel buffer (Write/Command)

The duration for the transfer of data from the PLC to the pixel buffer depends on the selected operation mode (Command, Extended) and the set cycle time.

All four channels can be written in parallel from the PLC, so that no addition of times is necessary.

- **Command mode**

In the command process image, one command can be sent per cycle. So it depends on the number of commands. If all pixels are to be written with one color, it is possible here to write all data in one cycle.

- **Extended mode**

One segment can be written per cycle in the extended process image. A segment comprises 8 individual pixels.

Example:

To write to 512 pixels, 64 cycles are required until everything has been written. With a cycle time of 500 µs it takes 32 ms to write the data from the PLC into the first buffer.

### Frame buffer to LEDs (Send)

The duration for the transfer of data between the buffers depends on whether RGB or RGBW LEDs are used. For this purpose, the following calculation can be used with the data sheet values for the LED type used.

n: Number of LEDs to be manipulated

$f_D$ : Data rate (800 kHz for WS2812B)

$T_{RST}$ : Reset Time (~300 kHz for WS2812B)

Frame rate: 
$$f_{RGB} = \frac{1}{\frac{24 \cdot n}{f_D} + T_{RST}}$$

$$f_{RGBW} = \frac{1}{\frac{32 \cdot n}{f_D} + T_{RST}}$$

The update of the data to the terminal and the sending to the strip can run in parallel, because two different buffers in the terminal are accessed.

In addition, all four channels can always be processed in parallel.

## 5.3 Function and parameterization

### 5.3.1 Basics of the "Modules/Slots" procedure

The modules/slots procedure enables simplified configuration and parameterization of multifunctional EtherCAT devices. The configuration is carried out in TwinCAT via the "Slots" tab.

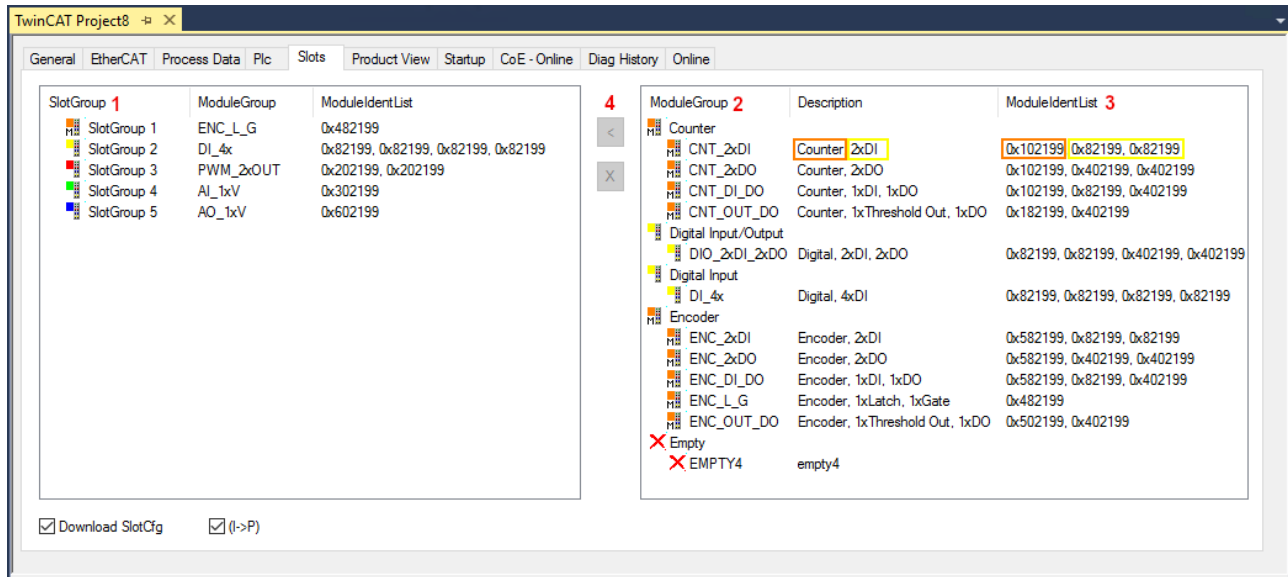



Fig. 126: Configuration in the "Slots" tab using an EL8601-8411 as example

1. The operating mode is defined for each **"SlotGroup"** by assigning a maximum of one **"ModuleGroup"**. When the **"SlotGroup"** is selected in the left-hand field, the **"ModuleGroups"** available for this **"SlotGroup"** with the associated **"Modules"** are displayed in the right-hand field.
2. **"ModuleGroups"** describe the possible combinations of the individual **"Modules"**. In this way, the configuration of invalid combinations can be excluded. The operation mode of the group is explained in short form in the **"Description"** section.
3. Each **"Module"** has a defined **"ModuleIdent"** number that is fixed to an operation mode and the corresponding process data and CoE objects. In the **"ModuleIdentList"**, all **"ModuleIdents"** of a **"ModuleGroup"** are displayed according to the frequency of their use (see Fig. above CNT\_2xDI).
4. Each **SlotGroup** must be assigned exactly one **"ModuleGroup"** using the **"<" (assign)** and **"X" (remove)** buttons.
  - ⇒ The process data and the CoE objects are automatically adjusted according to the selected **"ModuleGroup"**, in the respective **SlotGroup**.
  - ⇒ In the TwinCAT **"Product View"** tab, the pin assignment is displayed according to the configuration (see chapter [Connections in the "Product View" tab](#)) [► 113].





### Notes on configuration with the modules/slots procedure

- The TwinCAT system must be in "ConfigMode"  to perform the configuration.
- If a "SlotGroup" is to be operated without function, the "ModuleGroup" "Empty" must be selected. A "SlotGroup" without an assigned "ModuleGroup" is not allowed.
- If the configuration is changed, the settings in the Setting objects of the changed "ModuleGroups" are reset to the factory settings.
- To reset the product to the delivery state (factory settings), the "PreOP" state is required before the "Restore default parameters" object is used.

⇒ Please refer to the detailed description in the chapter "[Modules/Slots configuration EL2574](#)".  
[▶ 115]

#### 5.3.1.1 Connections in the "Product View" tab


The "Product View" tab shows the connections of the product according to the current configuration. This makes it easier to assign the individual signal types to the connection points, especially for multi-interface products. To make it easier to assign the connections, the designation also contains the corresponding SlotGroup in addition to the function.

For some products, the LED status is also displayed in real-time in the "Product View" tab. The LED status display is currently only supported for products that have the "LED status" CoE object.

Requirement for displaying the "Product View" tab:

- Development environment TwinCAT 3.1 Build 4024.59

After making changes in the "Slots" area, refresh the view as follows:

- The project must be saved for offline configuration.
- With an online configuration, a "Reload Devices"  is sufficient to refresh the view.

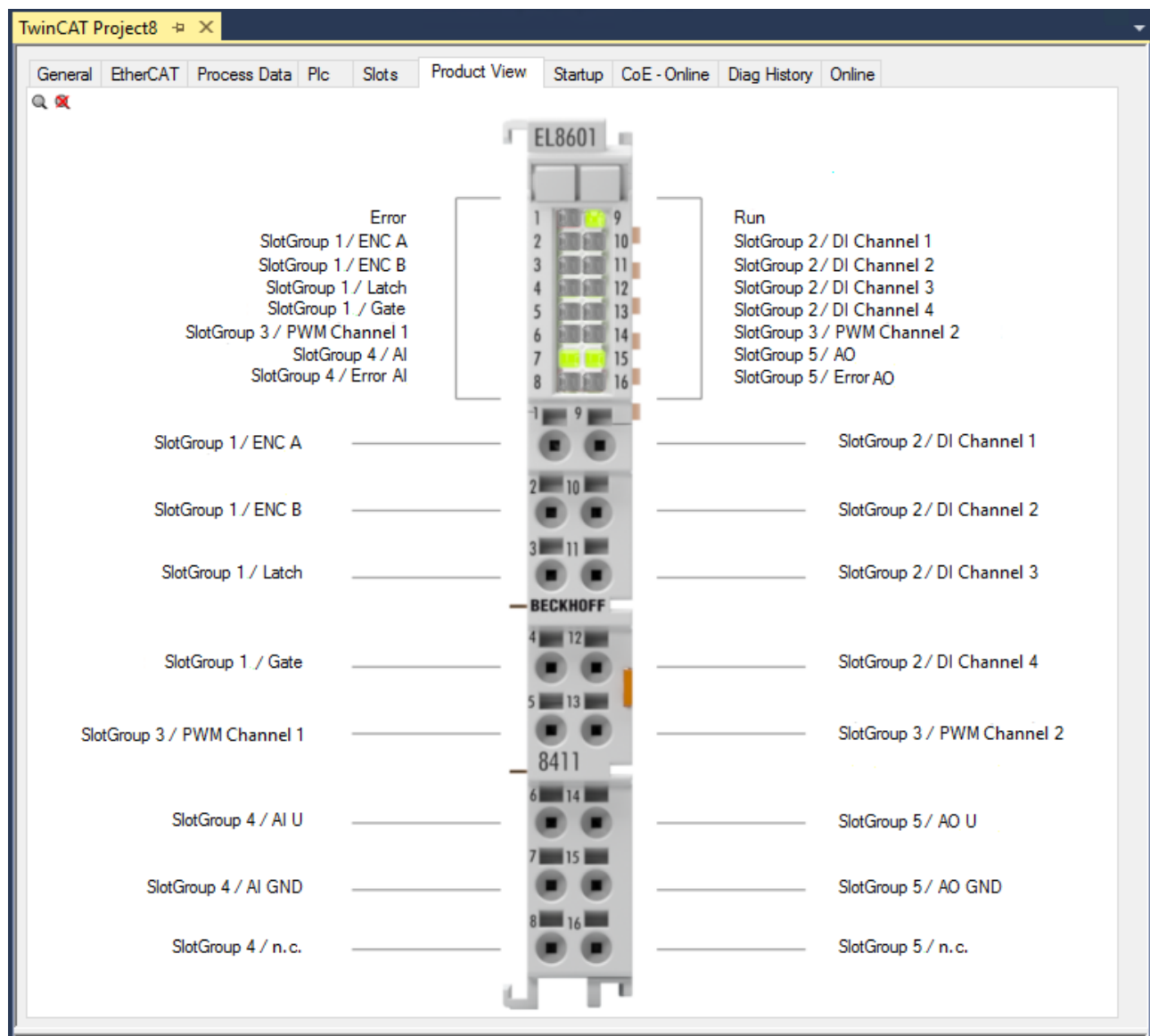


Fig. 127: View in the "Product View" tab using the EL8601-8411 as an example

A detailed description of the connections can be found in the chapter "Connections".

### 5.3.2 Modules/Slots configuration EL2574

Two slot groups are available for configuring the individual channels, each containing two slots, one for each channel. Depending on which type of pixel LEDs are used (synchronous with CLK or asynchronous without CLK), two channels will be combined into one due to the pin assignment.

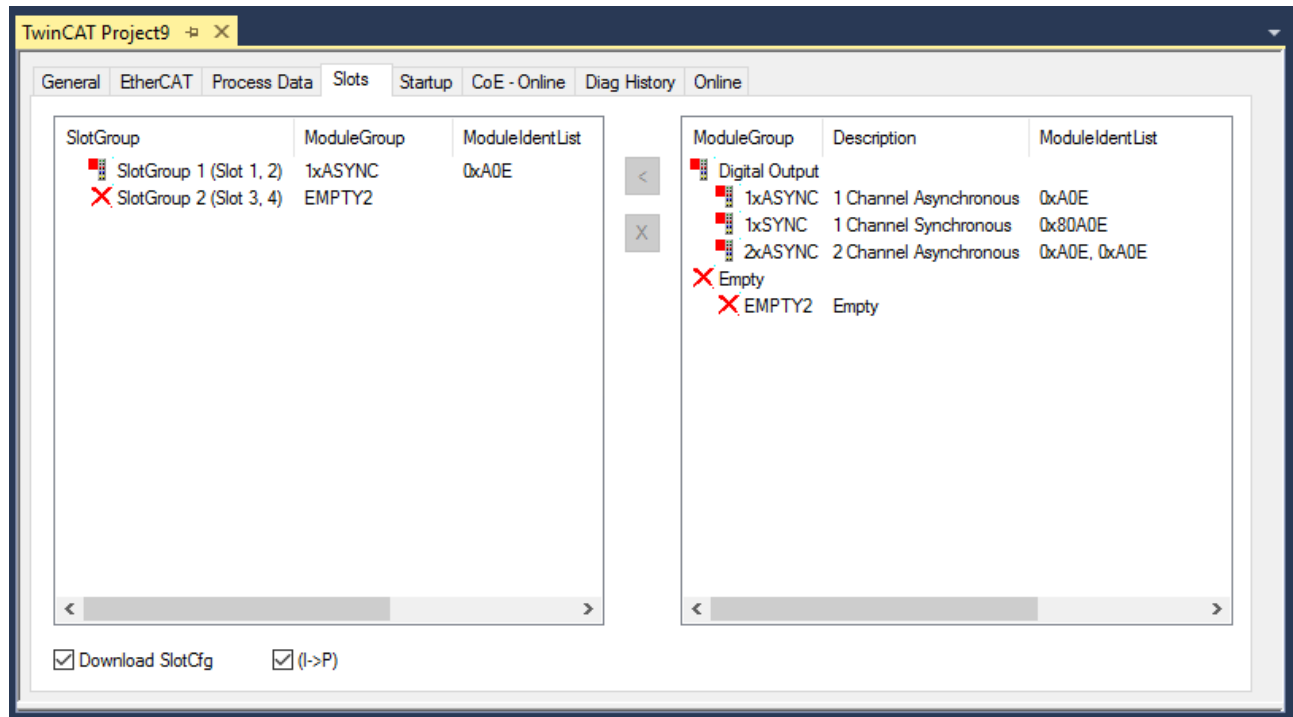


Fig. 128: EL2574 - Rider Slots

Modules can be assigned to a specific slot with the < button, or removed again with x.

Either 2x ALED or 1x SLED can be assigned per SlotGroup. An unused channel must always be assigned an "Empty Module".

Both the CoE objects and the process data are automatically generated from the configuration of the modules.

### 5.3.3 Command mode

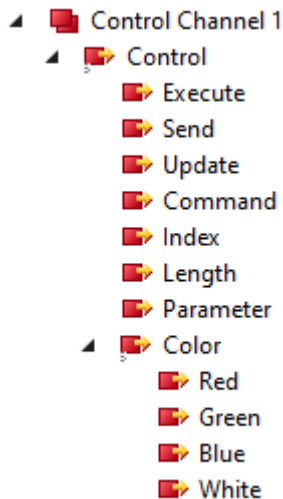


Fig. 129: EL2574 - Default - Process data display in TwinCAT tree "Control Channel 1"

In the default configuration the ALED/SLED Control output data are enabled. Here, the pixels can be controlled by means of commands.

#### 5.3.3.1 Commands

These commands can be used to fill, move or modify entire areas.

##### Overview of commands

Action	Command	Variables			
		Index	Length	Color	Parameter
No action	0x00 "Nop"	-	-	-	-
Fill an area	0x01 "Fill" [► 117]	Start position	Number of pixels	Fill color	-
Turn off all pixels	0x02 "Clear" [► 117]	-	-	-	-
Copy pixel	0x03 "Copy" [► 117]	Start position	Number of pixels	-	Target position
Move pixels / areas	0x04 "Move" [► 118]	Start position	Number of pixels	-	Target position
	0x05 "Rotate Left" [► 118]				Shift
	0x06 "Rotate Right" [► 118]				Shift
Invert order	0x07 "Reverse" [► 119]	Start position	Number of pixels	-	-
Create color gradient	0x08/0x09 "Gradient" [► 119]	Start position	Number of pixels	0x08: color first pixel 0x09: color last pixel	-

**Command 0x01 "Fill", Fill an area**

A specified area is filled with a color.

Command	Variables			
	Index	Length	Color	Parameter
0x01 "Fill"	Start position	Number of pixels	Fill color	-

Command: 0x01, Index: 4, Length: 6, Color 1: #FF0000

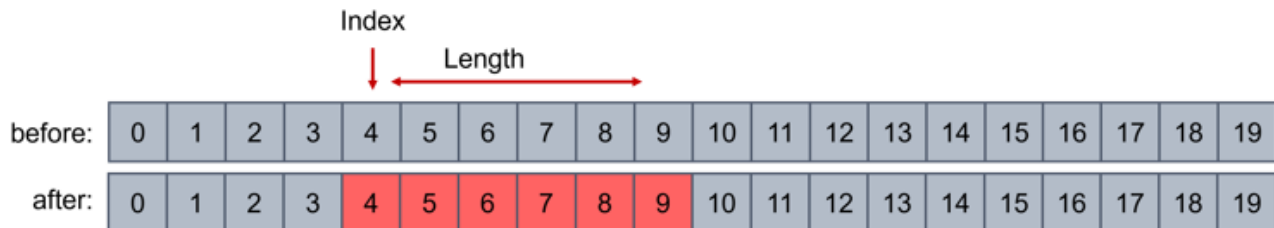


Fig. 130: Example Command 0x01 "Fill"

**Command 0x02 "Clear", Clear all pixels**

All pixels are deleted.

Command	Variables			
	Index	Length	Color	Parameter
0x02 "Clear"	-	-	-	-

Command: 0x02



Fig. 131: Example Command 0x02 "Clear"

**Command 0x03 "Copy", Copy number of pixels**

A number of pixels are copied to a new position (overwrite).

Command	Variables			
	Index	Length	Color	Parameter
0x03 "Copy"	Start position	Number of pixels	-	Target position

Command: 0x03, Index: 0, Length: 6, Parameter: 11

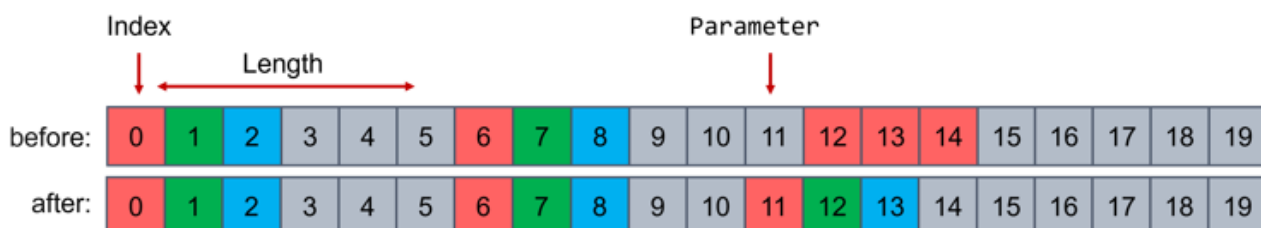


Fig. 132: Example Command 0x03 "Copy"

**Command 0x04 "Move", Move number of pixels to target position**

A number of pixels is moved to a new position (overwrite).

Command	Variables			
	Index	Length	Color	Parameter
0x04 "Move"	Start position	Number of pixels	-	Target position

Command: 0x04, Index: 0, Length: 6, Parameter: 11

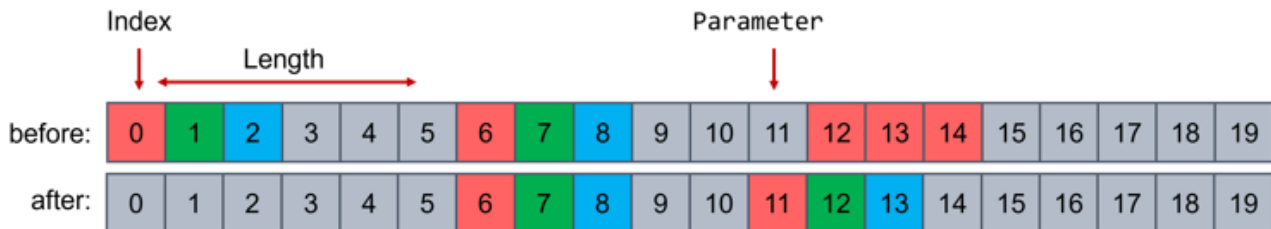


Fig. 133: Example Command 0x04 "Move"

**Command 0x05 "Rotate Left", 0x06 "Rotate Right", shift within a range**

The "Rotate" command moves the pixels within the selected range (from the start position "Index" over the selected number of pixels "Length") (see the following example: range = pixels 3 to 8).

Via "Parameters" the number of shifted positions (see the following example: shifted by 2 pixels).

Pixels that are moved beyond the range limit are added again at the other end of the range (see the following example: Before: pixels 7 and 8 -> After: pixels 3 and 4).

**Command 0x05:**

Within the selected range is shifted to the left by the specified number of positions.

**Command 0x06:**

Shift to the right by the specified number of positions within the selected range.

Command	Variables			
	Index	Length	Color	Parameter
0x05 "Rotate Left"	Start position	Number of pixels	-	Shift
0x06 "Rotate Right"				Shift

Command: 0x06, Index: 3, Length: 6, Parameter: 2

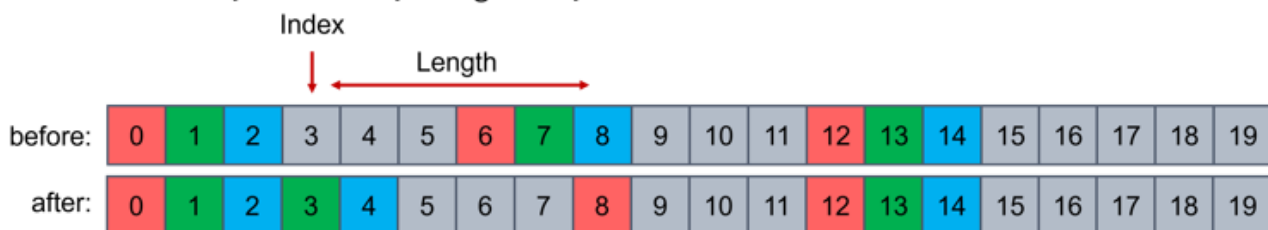


Fig. 134: Example Command 0x06 "Rotate Right"

### Command 0x07 "Reverse", invert pixels in an area

In the selected area, the order of the pixels is inverted.

Command	Variables			
	Index	Length	Color	Parameter
0x07 "Reverse"	Start position	Number of pixels	-	-

Command: 0x07, Index: 3, Length: 6

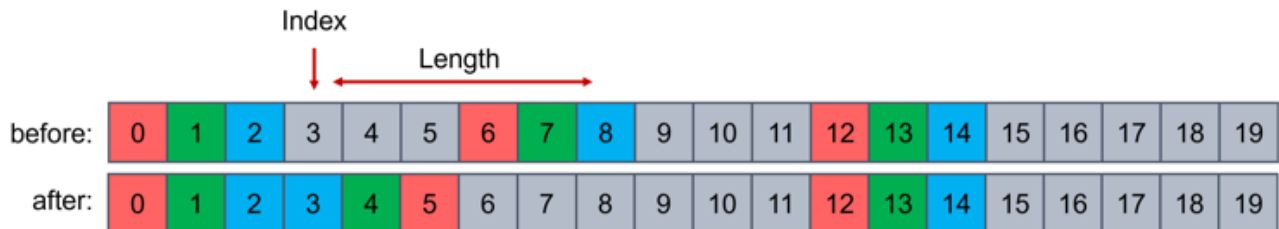


Fig. 135: Example Command 0x07 "Reverse"

### Command 0x08/0x09 "Gradient", create color gradient in an area

A color gradient is created in the selected area (linear interpolated).

- Via command 0x08 the color of the first pixel is set
- Via command 0x09 the color of the last pixel is set

Command	Variables			
	Index	Length	Color	Parameter
0x08/0x09 "Gradient"	Start position	Number of pixels	0x08: color first pixel 0x09: color last pixel	-

Command: 0x08, Index: 0, Length: 9, Color: #FF0000

Command: 0x09, Index: 0, Length: 9, Color: #0000FF



Fig. 136: Example Command 0x08 / 0x09 "Gradient"

### 5.3.3.2 Procedure

The following procedure is required to specify the data and change the connected pixels:

1. Specification of the "Command" with the required variables ("Index", "Length", "Color", "Parameter")
2. Writing the data into the pixel buffer via toggling the "Execute" bit
3. if necessary, repeat steps 1 and 2 until all pixel data are completely prepared
4. Update the data from the pixel buffer into the frame buffer by setting the "Update" bit to 1
5. toggle the "Execute" bit to execute the update
6. Transfer data:
  - ⇒ If the object 0x8pp0:02 "Enable Cyclic Frame Output" is enabled in the CoE, the data from the frame buffer is automatically transferred to the connected LEDs.
  - ⇒ Alternatively, sending can be triggered manually by setting the "Send" bit to 1
7. Toggle the "Execute" bit to execute the transmission
  - ⇒ The changes on the connected LEDs are visible

The transmission to the buffers and to the connected LEDs can also be done in one cycle. For this, "Update" and "Send" bit must be set to 1 at the same time and executed together with a toggling of the "Execute" bit.

#### NOTICE

##### Loss of data possible

Note the status of the Busy and Transmit bits before transmitting new data.

- Busy bit = FALSE  
Always make sure that the busy bit in the status object is FALSE, because otherwise no commands are executed.
- Transmit bit = FALSE  
When sending data to the LEDs, the frame buffer is read. This is indicated via the "Transmit" bit.  
To ensure data consistency, simultaneous reading and writing is not possible.  
During the transfer of data from the frame buffer the "Transmit" bit is TRUE, so that it is necessary to wait until the bit is FALSE again in order to write new data.



### 5.3.4 Extended mode

For high update rates and animations, the extended process image can be used. These data must be added in the tab "Process data" per channel (Outputs - 0x16n1).

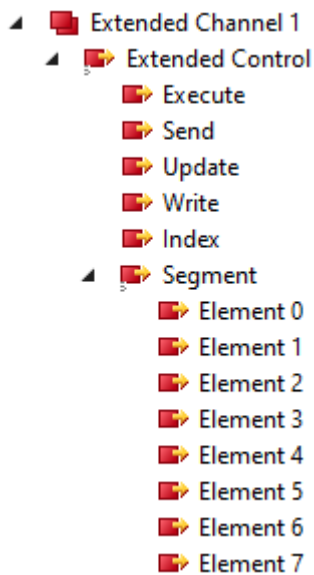


Fig. 137: EL2574 - Process data display in TwinCAT tree "Extended Channel 1

Here, the LEDs are divided into segments of eight LEDs, named as elements. For each of the eight LEDs 4 bytes of output data are available. Thus, the color and brightness of each LED can be specified. Each Byte corresponds to a color component with the sequence red, green, blue, white (0xWWBBGGRR).

Due to the fact that eight LEDs always correspond to one segment and the numbering starts at 0, the first LED of a segment results by:

Index \* 8.

Subsequently, the individual LEDs can be defined via the eight elements in the segment. Thus, eight LEDs can be rewritten in the pixel buffer per cycle.

#### Procedure

The following procedure is required to specify the data and change the connected pixels:

1. Presetting the segment number via "Index"
2. Specification of the setpoints for the eight LEDs in the selected segment via the eight available elements
3. Writing the data into the pixel buffer via "Write" bit toggling the "Execute" bit.
4. If necessary, repeat steps 1 and 2 until all required segments are completely prepared.
5. Update the data from the pixel buffer into the frame buffer by setting the "Update" bit to 1
6. Toggling the "Execute" bit to execute the update
7. Transfer data:
  - ⇒ If the object 0x80p0:02 Enable Cyclic Frame Output is enabled in the CoE, the data from the frame buffer is automatically transferred to the connected LEDs.
  - ⇒ Alternatively, sending can be triggered manually by setting the "Send" bit to 1.
8. Toggle the "Execute" bit to execute the transmission.
9. The changes on the connected LEDs are visible.

The transfer to the buffers or between the buffers and to the strip can also be triggered in one cycle. For this purpose, the desired commands "Write", "Update" and "Send" must be set to 1. The commands are triggered by toggling the "Execute" bit.

**NOTICE****Loss of data possible**

Note the status of the Busy and Transmit bits before transmitting new data.

- Busy bit = FALSE  
Always make sure that the busy bit in the status object is FALSE, because otherwise no commands are executed.
- Transmit bit = FALSE  
When sending data to the LEDs, the frame buffer is read. This is indicated via the "Transmit" bit.  
To ensure data consistency, simultaneous reading and writing is not possible.  
During the transfer of data from the frame buffer the "Transmit" bit is TRUE, so that it is necessary to wait until the bit is FALSE again in order to write new data.

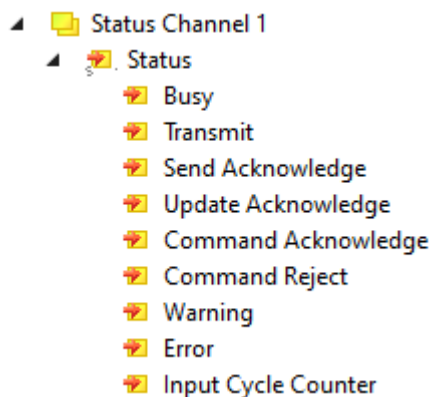
**5.3.5 Status information**

Fig. 138: EL2574 - Display of status information in the TwinCAT tree

Name	Meaning
Busy	Feedback to toggle the Execute bit. A command is currently being executed. It is necessary to wait until the device is ready. <b>No</b> new command can be sent or executed during this time.
Transmit	A frame is currently being sent (by manual sending or by automatic frame output ("Cyclic Frame Output" in CoE)). The frame buffer is used during sending. To prevent the frame buffer from being sent inconsistently, it is not possible to update it or trigger sending again. While the "Transmit" bit is active, commands that do not access the frame buffer can still be processed (without Update and Send).
Send Acknowledge	This toggling bit is a confirmation of the successful execution of the "Send" bit. If "Send" is executed while the "Transmit" bit was active, then the bit does not toggle.
Update Acknowledge	This toggling bit is a confirmation for the successful execution of the "Update" bit. If "Update" is executed while the "Transmit" bit was active, then the bit does not toggle.
Command Acknowledge	This toggling bit is an acknowledgement for the successful execution of the "Command".
Command Reject	If a wrong command is executed or the parameters are not correct (e.g. device is busy, length = 0, pixels addressed are outside the set Number of Pixels), the command is rejected.

## 5.4 Process data

### 5.4.1 Sync-Manager (SM)

The extent of the process data that is made available can be changed through the "Process data" tab (see following Fig.).

#### Output SyncManager (SM2)

The PDOs from the range 0x16nm (0x1600 to 0x1631) can be assigned to the Output SyncManager 2.

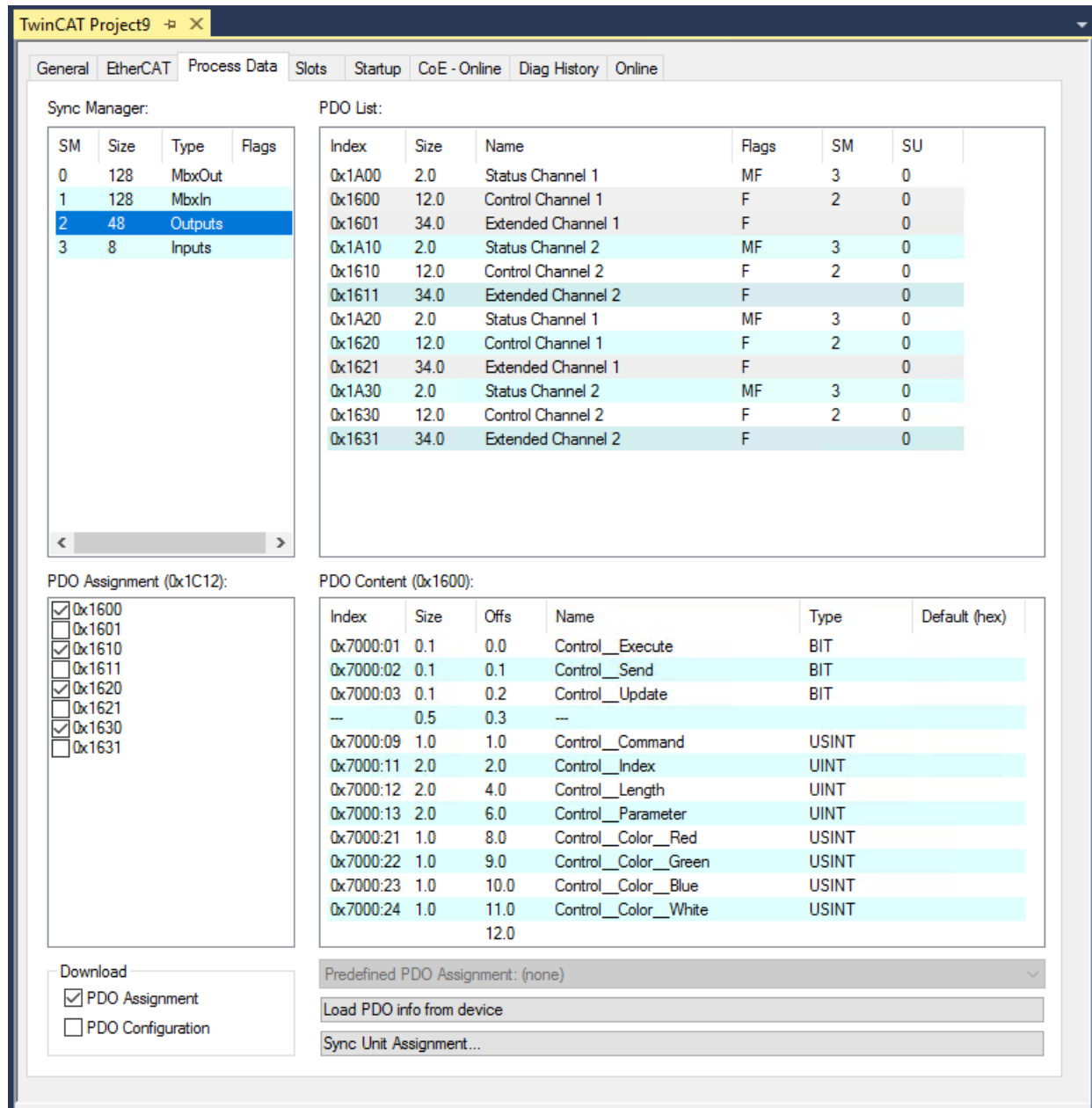


Fig. 139: EL2574 - Process data Output SyncManager (SM2)

### Input SyncManager (SM 3)

The PDOs from the range 0x1An0 (0x1A00 to 0x1A30) can be assigned to the Input SyncManager 3.

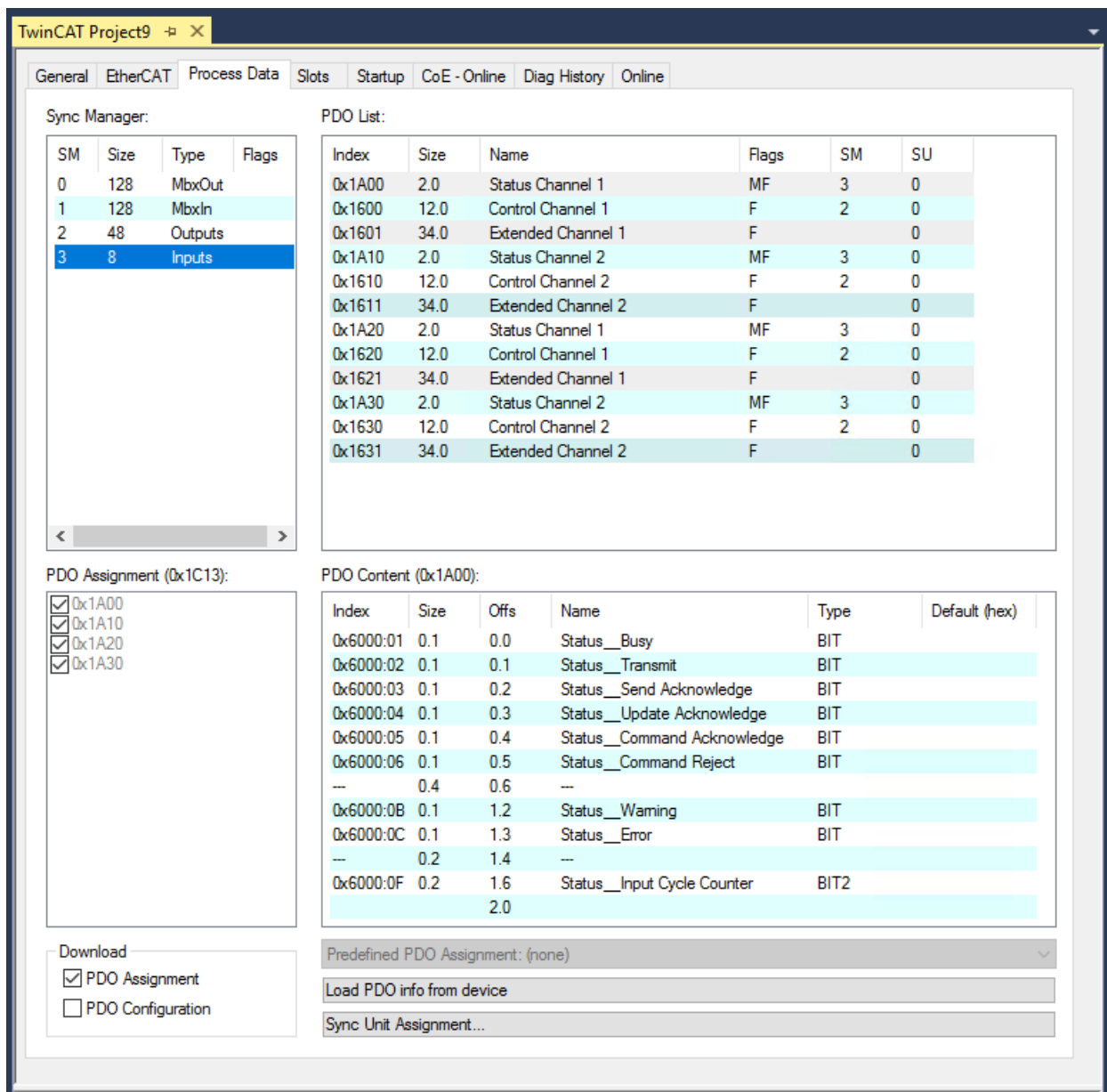


Fig. 140: EL2574 - Process Data Input SyncManager (SM3)

## 5.4.2 Manual PDO Assignment

To configure the process data,

1. select the desired Sync Manager (SM 2 and SM 3 can be edited) in the upper left-hand "Sync Manager" box.
2. The process data assigned to this Sync Manager can then be switched on or off in the "PDO Assignment" box underneath.
3. Restarting the EtherCAT system, or reloading the configuration in Config mode (F4), causes the EtherCAT communication to restart, and the process data is transferred from the terminal.

SM2, PDO Assignment 0x1C12				
Index	Index of excluded PDOs	Size (byte.bit)	Name	PDO content Index - Name
0x16n0	-	12.0	Control Channel n	0x70n00:01 – Control_Execute 0x70n00:02 – Control_Send 0x70n00:03 – Control_Update  0x70n00:09 – Control_Command 0x70n00:11 – Control_Index 0x70n00:12 – Control_Length 0x70n00:13 – Control_Parameter 0x70n00:21 – Control_Color_Red 0x70n00:22 – Control_Color_Green 0x70n00:23 – Control_Color_Blue 0x70n00:24 – Control_Color_White
0x16n1	-	34.0	Extended Channel n	0x70n1:01 – Extended Control_Execute 0x70n1:02 – Extended Control_Send 0x70n1:03 – Extended Control_Update 0x70n1:04 – Extended Control_Write  0x70n1:09 – Extended Control_Index 0x70n1:11 – Extended Control_Segment_Element 0 0x70n1:12 – Extended Control_Segment_Element 1 0x70n1:13 – Extended Control_Segment_Element 2 0x70n1:14 – Extended Control_Segment_Element 3 0x70n1:15 – Extended Control_Segment_Element 4 0x70n1:16 – Extended Control_Segment_Element 5 0x70n1:17 – Extended Control_Segment_Element 6 0x70n1:18 – Extended Control_Segment_Element 7

SM3, PDO Assignment 0x1C13				
Index	Index of excluded PDOs	Size (byte.bit)	Name	PDO content Index - Name
0x1An0	-	2.0	Status Channel n	0x60n00:01 – Status_Busy 0x60n00:02 – Status_Transmit 0x60n00:03 – Status_Send Acknowledge 0x60n00:04 – Status_Update Acknowledge 0x60n00:05 – Status_Command Acknowledge 0x60n00:06 – Status_Command Reject  0x60n0:0B – Status_Warning 0x60n0:0C – Status_Error  0x60n0:0F – Status_Input Cycle Counter

## 5.5 Diagnosis

### 5.5.1 Diagnostics in the CoE

The EL2574 has an internal diagnosis of the temperature and the supply voltage via the power contacts. The diagnostics can be viewed in the CoE objects under 0xFA15 "PLED Diag Data".

FA15:0	PLED Diag data	R0	> 17 <	
FA15:01	Field Power Supply	R0	TRUE	
FA15:11	PCB Temperature	R0	254	0.1 °C

Fig. 141: Diagnostics in CoE object 0xFA15

A warning or an error is indicated in the process data in the "Status" via the "Warning" or "Error" bit.

Object	Name	Description	Correction
FA15:01	Field Power Supply	Supply voltage via the power contacts is not sufficient.	Check supply at the power contacts
FA15:02	PCB Temperature	Warning at 80°C PCB temperature Error at 100°C PCB temperature	Terminal must cool down

### 5.5.2 Diagnostics in the Diag Messages

There are special messages for the device in the Diag Messages for errors and warnings, which are listed in the following table:

Text-ID	Type	Message
0x4616	Warning	Invalid command
0x4617	Warning	Invalid area
0x4619	Warning	The color channel is ignored due to the configuration
0x810B	Error	Undervoltage Up
0x81B5	Error	Invalid Module and Slot configuration
0x8625	Error	Set number of pixels is greater than the Frame buffer

The basics principles of Diag Messages can be found in chapter „[Diagnostics – basic principles of diag messages](#) [► 126]“.

### 5.5.3 Diagnostics - basic principles of diag messages

*DiagMessages* designates a system for the transmission of messages from the EtherCAT Slave to the EtherCAT Master/TwinCAT. The messages are stored by the device in its own CoE under 0x10F3 and can be read by the application or the System Manager. An error message referenced via a code is output for each event stored in the device (warning, error, status change).

#### Definition

The *DiagMessages* system is defined in the ETG (EtherCAT Technology Group) in the guideline ETG.1020, chapter 13 "Diagnosis handling". It is used so that pre-defined or flexible diagnostic messages can be conveyed from the EtherCAT Slave to the Master. In accordance with the ETG, the process can therefore be implemented supplier-independently. Support is optional. The firmware can store up to 250 DiagMessages in its own CoE.

Each DiagMessage consists of

- Diag Code (4-byte)

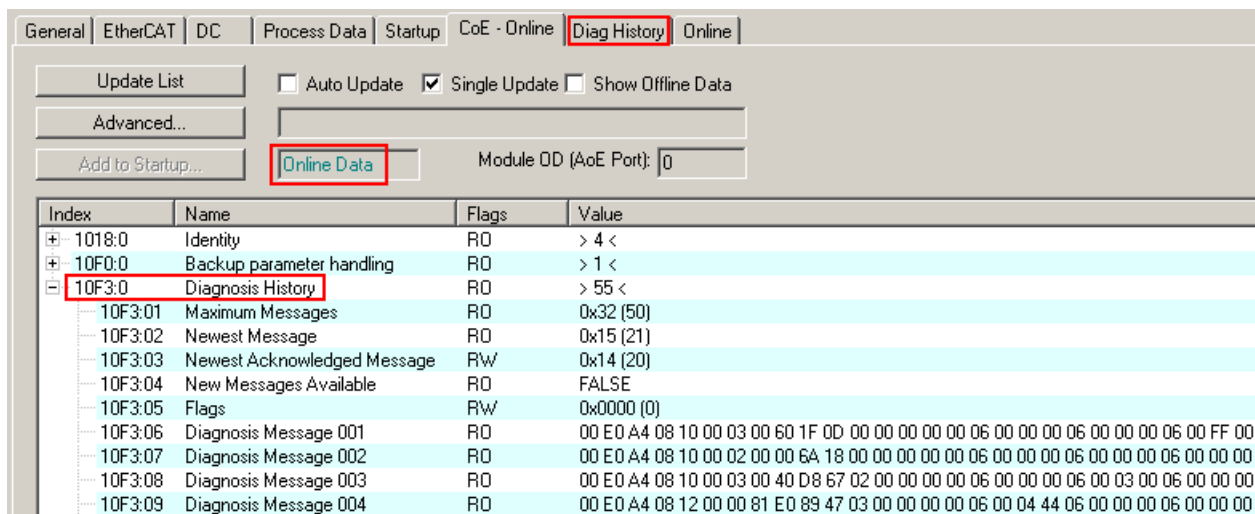
- Flags (2-byte; info, warning or error)
- Text ID (2-byte; reference to explanatory text from the ESI/XML)
- Timestamp (8-byte, local slave time or 64-bit Distributed Clock time, if available)
- Dynamic parameters added by the firmware

The DiagMessages are explained in text form in the ESI/XML file belonging to the EtherCAT device: on the basis of the Text ID contained in the DiagMessage, the corresponding plain text message can be found in the languages contained in the ESI/XML. In the case of Beckhoff products these are usually German and English.

Via the entry *NewMessagesAvailable* the user receives information that new messages are available.

DiagMessages can be confirmed in the device: the last/latest unconfirmed message can be confirmed by the user.

In the CoE both the control entries and the history itself can be found in the CoE object 0x10F3:



Index	Name	Flags	Value
1018:0	Identity	RO	> 4 <
10F0:0	Backup parameter handling	RO	> 1 <
10F3:0	Diagnosis History	RO	> 55 <
10F3:01	Maximum Messages	RO	0x32 (50)
10F3:02	Newest Message	RO	0x15 (21)
10F3:03	Newest Acknowledged Message	R/W	0x14 (20)
10F3:04	New Messages Available	RO	FALSE
10F3:05	Flags	R/W	0x0000 (0)
10F3:06	Diagnosis Message 001	RO	00 E0 A4 08 10 00 03 00 60 1F 0D 00 00 00 00 00 06 00 00 00 06 00 00 00 06 00 FF 00
10F3:07	Diagnosis Message 002	RO	00 E0 A4 08 10 00 02 00 00 6A 18 00 00 00 00 00 06 00 00 00 06 00 00 00 06 00 00 00
10F3:08	Diagnosis Message 003	RO	00 E0 A4 08 10 00 03 00 40 D8 67 02 00 00 00 00 06 00 00 00 06 00 03 00 06 00 00 00
10F3:09	Diagnosis Message 004	RO	00 E0 A4 08 12 00 00 81 E0 89 47 03 00 00 00 00 06 00 04 44 06 00 00 00 06 00 00 00

Fig. 142: DiagMessages in the CoE

The subindex of the latest *DiagMessage* can be read under 0x10F3:02.

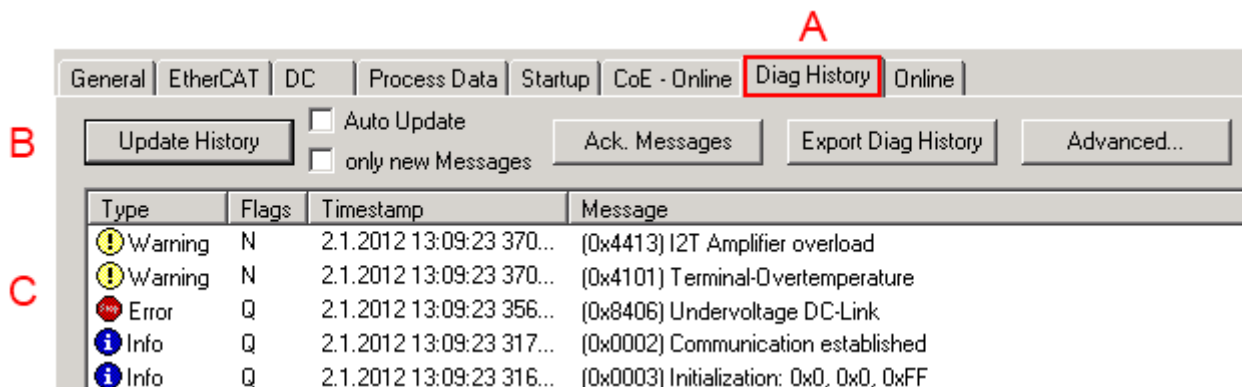


### Support for commissioning

The DiagMessages system is to be used above all during the commissioning of the plant. The diagnostic values e.g. in the StatusWord of the device (if available) are helpful for online diagnosis during the subsequent continuous operation.

### TwinCAT System Manager implementation

From TwinCAT 2.11 DiagMessages, if available, are displayed in the device's own interface. Operation (collection, confirmation) also takes place via this interface.



Type	Flags	Timestamp	Message
Warning	N	2.1.2012 13:09:23 370...	(0x4413) I2T Amplifier overload
Warning	N	2.1.2012 13:09:23 370...	(0x4101) Terminal-Overtemperature
Error	Q	2.1.2012 13:09:23 356...	(0x8406) Undervoltage DC-Link
Info	Q	2.1.2012 13:09:23 317...	(0x0002) Communication established
Info	Q	2.1.2012 13:09:23 316...	(0x0003) Initialization: 0x0, 0x0, 0xFF

Fig. 143: Implementation of the DiagMessage system in the TwinCAT System Manager

The operating buttons (B) and the history read out (C) can be seen on the Diag History tab (A). The components of the message:

- Info/Warning/Error
- Acknowledge flag (N = unconfirmed, Q = confirmed)
- Time stamp
- Text ID
- Plain text message according to ESI/XML data

The meanings of the buttons are self-explanatory.

### DiagMessages within the ADS Logger/Eventlogger

From TwinCAT 3.1 build 4022 onwards, DiagMessages sent by the terminal are shown by the TwinCAT ADS Logger. Given that DiagMessages are represented IO- comprehensive at one place, commissioning will be simplified. In addition, the logger output could be stored into a data file – hence DiagMessages are available long-term for analysis.

DiagMessages are actually only available locally in CoE 0x10F3 in the terminal and can be read out manually if required, e.g. via the DiagHistory mentioned above.

In the latest developments, the EtherCAT Terminals are set by default to report the presence of a DiagMessage as emergency via EtherCAT; the event logger can then retrieve the DiagMessage. The function is activated in the terminal via 0x10F3:05, so such terminals have the following entry in the StartUp list by default:




General	EtherCAT	Settings	Filter	DC	Process Data	Plc	Startup	CoE - Online	Diag History	Online
Transition	Protocol	Index	Data	Comment						
 <PS>	CoE	0x1C12 C 0	00 00	download pdo 0x1C12 index						
 <PS>	CoE	0x1C13 C 0	05 00 00 1A 01 1A 10 1A ...	download pdo 0x1C13 index						
 IP	CoE	0x10F3:05	0x0001 (1)							

Fig. 144: Startup List

If the function is to be deactivated because, for example, many messages come in or the EventLogger is not used, the StartUp entry can be deleted or set to 0. The value can then be set back to 1 later from the PLC via CoE access if required.

### Reading messages into the PLC

- In preparation -

### Interpretation

#### Time stamp

The time stamp is obtained from the local clock of the terminal at the time of the event. The time is usually the distributed clock time (DC) from register x910.

Please note: When EtherCAT is started, the DC time in the reference clock is set to the same time as the local IPC/TwinCAT time. From this moment the DC time may differ from the IPC time, since the IPC time is not adjusted. Significant time differences may develop after several weeks of operation without a EtherCAT restart. As a remedy, external synchronization of the DC time can be used, or a manual correction calculation can be applied, as required: The current DC time can be determined via the EtherCAT master or from register x901 of the DC slave.

### Structure of the Text ID

The structure of the MessageID is not subject to any standardization and can be supplier-specifically defined. In the case of Beckhoff EtherCAT devices (EL, EP) it usually reads according to **xyzz**:



x	y	zz
0: Systeminfo 2: reserved 1: Info 4: Warning 8: Error	0: System 1: General 2: Communication 3: Encoder 4: Drive 5: Inputs 6: I/O general 7: reserved	Error number

Example: Message 0x4413 --> Drive Warning Number 0x13

Overview of text IDs

Specific text IDs are listed in the device documentation.

Text ID	Type	Place	Text Message	Additional comment
0x0001	Information	System	No error	No error
0x0002	Information	System	Communication established	Connection established
0x0003	Information	System	Initialization: 0x%X, 0x%X, 0x%X	General information; parameters depend on event. See device documentation for interpretation.
0x1000	Information	System	Information: 0x%X, 0x%X, 0x%X	General information; parameters depend on event. See device documentation for interpretation.
0x1012	Information	System	EtherCAT state change Init - PreOp	
0x1021	Information	System	EtherCAT state change PreOp - Init	
0x1024	Information	System	EtherCAT state change PreOp - Safe-Op	
0x1042	Information	System	EtherCAT state change SafeOp - PreOp	
0x1048	Information	System	EtherCAT state change SafeOp - Op	
0x1084	Information	System	EtherCAT state change Op - SafeOp	
0x1100	Information	General	Detection of operation mode completed: 0x%X, %d	Detection of the mode of operation ended
0x1135	Information	General	Cycle time o.k.: %d	Cycle time OK
0x1157	Information	General	Data manually saved (Idx: 0x%X, SubIdx: 0x%X)	Data saved manually
0x1158	Information	General	Data automatically saved (Idx: 0x%X, SubIdx: 0x%X)	Data saved automatically
0x1159	Information	General	Data deleted (Idx: 0x%X, SubIdx: 0x%X)	Data deleted
0x117F	Information	General	Information: 0x%X, 0x%X, 0x%X	Information
0x1201	Information	Communication	Communication re-established	Communication to the field side restored This message appears, for example, if the voltage was removed from the power contacts and re-applied during operation.
0x1300	Information	Encoder	Position set: %d, %d	Position set - StartInputhandler
0x1303	Information	Encoder	Encoder Supply ok	Encoder power supply unit OK
0x1304	Information	Encoder	Encoder initialization successfully, channel: %X	Encoder initialization successfully completed
0x1305	Information	Encoder	Sent command encoder reset, channel: %X	Send encoder reset command
0x1400	Information	Drive	Drive is calibrated: %d, %d	Drive is calibrated
0x1401	Information	Drive	Actual drive state: 0x%X, %d	Current drive status
0x1705	Information		CPU usage returns in normal range (< 85%%)	Processor load is back in the normal range
0x1706	Information		Channel is not in saturation anymore	Channel is no longer in saturation
0x1707	Information		Channel is not in overload anymore	Channel is no longer overloaded
0x170A	Information		No channel range error anymore	A measuring range error is no longer active
0x170C	Information		Calibration data saved	Calibration data were saved
0x170D	Information		Calibration data will be applied and saved after sending the command "0x5AFE"	Calibration data are not applied and saved until the command "0x5AFE" is sent.

Text ID	Type	Place	Text Message	Additional comment
0x2000	Information	System	%s: %s	
0x2001	Information	System	%s: Network link lost	Network connection lost
0x2002	Information	System	%s: Network link detected	Network connection found
0x2003	Information	System	%s: no valid IP Configuration - Dhcp client started	Invalid IP configuration
0x2004	Information	System	%s: valid IP Configuration (IP: %d.%d.%d.%d) assigned by Dhcp server %d.%d.%d.%d	Valid IP configuration, assigned by the DHCP server
0x2005	Information	System	%s: Dhcp client timed out	DHCP client timeout
0x2006	Information	System	%s: Duplicate IP Address detected (%d.%d.%d.%d)	Duplicate IP address found
0x2007	Information	System	%s: UDP handler initialized	UDP handler initialized
0x2008	Information	System	%s: TCP handler initialized	TCP handler initialized
0x2009	Information	System	%s: No more free TCP sockets available	No free TCP sockets available.

Text ID	Type	Place	Text Message	Additional comment
0x4000	Warning		Warning: 0x%X, 0x%X, 0x%X	General warning; parameters depend on event. See device documentation for interpretation.
0x4001	Warning	System	Warning: 0x%X, 0x%X, 0x%X	
0x4002	Warning	System	%s: %s Connection Open (IN:%d OUT:%d API:%dms) from %d. %d.%d.%d successful	
0x4003	Warning	System	%s: %s Connection Close (IN:%d OUT:%d) from %d.%d.%d.%d successful	
0x4004	Warning	System	%s: %s Connection (IN:%d OUT:%d) with %d.%d.%d.%d timed out	
0x4005	Warning	System	%s: %s Connection Open (IN:%d OUT:%d) from %d.%d.%d.%d denied (Error: %u)	
0x4006	Warning	System	%s: %s Connection Open (IN:%d OUT:%d) from %d.%d.%d.%d denied (Input Data Size expected: %d Byte(s) received: %d Byte(s))	
0x4007	Warning	System	%s: %s Connection Open (IN:%d OUT:%d) from %d.%d.%d.%d denied (Output Data Size expected: %d Byte(s) received: %d Byte(s))	
0x4008	Warning	System	%s: %s Connection Open (IN:%d OUT:%d) from %d.%d.%d.%d denied (RPI:%dms not supported -> API:%dms)	
0x4101	Warning	General	Terminal-Overtemperature	Overtemperature. The internal temperature of the terminal exceeds the parameterized warning threshold.
0x4102	Warning	General	Discrepancy in the PDO-Configuration	The selected PDOs do not match the set operating mode.  Sample: Drive operates in velocity mode, but the velocity PDO is but not mapped in the PDOs.
0x417F	Warning	General	Warning: 0x%X, 0x%X, 0x%X	
0x428D	Warning	General	Challenge is not Random	
0x4300	Warning	Encoder	Subincrements deactivated: %d, %d	Sub-increments deactivated (despite activated configuration)
0x4301	Warning	Encoder	Encoder-Warning	General encoder error
0x4302	Warning	Encoder	Maximum frequency of the input signal is nearly reached (channel %d)	
0x4303	Warning	Encoder	Limit counter value was reduced because of the PDO configuration (channel %d)	
0x4304	Warning	Encoder	Reset counter value was reduced because of the PDO configuration (channel %d)	
0x4400	Warning	Drive	Drive is not calibrated: %d, %d	Drive is not calibrated
0x4401	Warning	Drive	Starttype not supported: 0x%X, %d	Start type is not supported
0x4402	Warning	Drive	Command rejected: %d, %d	Command rejected
0x4405	Warning	Drive	Invalid modulo subtype: %d, %d	Modulo sub-type invalid
0x4410	Warning	Drive	Target overrun: %d, %d	Target position exceeded
0x4411	Warning	Drive	DC-Link undervoltage (Warning)	The DC link voltage of the terminal is lower than the parameterized minimum voltage. Activation of the output stage is prevented.
0x4412	Warning	Drive	DC-Link overvoltage (Warning)	The DC link voltage of the terminal is higher than the parameterized maximum voltage. Activation of the output stage is prevented.
0x4413	Warning	Drive	I2T-Model Amplifier overload (Warning)	<ul style="list-style-type: none"> <li>The amplifier is being operated outside the specification.</li> <li>The I2T-model of the amplifier is incorrectly parameterized.</li> </ul>
0x4414	Warning	Drive	I2T-Model Motor overload (Warning)	<ul style="list-style-type: none"> <li>The motor is being operated outside the parameterized rated values.</li> </ul>

Text ID	Type	Place	Text Message	Additional comment
				<ul style="list-style-type: none"> <li>The I2T-model of the motor is incorrectly parameterized.</li> </ul>
0x4415	Warning	Drive	Speed limitation active	The maximum speed is limited by the parameterized objects (e.g. velocity limitation, motor speed limitation). This warning is output if the set velocity is higher than one of the parameterized limits.
0x4416	Warning	Drive	Step lost detected at position: 0x%X%X	Step loss detected
0x4417	Warning	Drive	Motor overtemperature	The internal temperature of the motor exceeds the parameterized warning threshold
0x4418	Warning	Drive	Limit: Current	Limit: current is limited
0x4419	Warning	Drive	Limit: Amplifier I2T-model exceeds 100%%	The threshold values for the maximum current were exceeded.
0x441A	Warning	Drive	Limit: Motor I2T-model exceeds 100%%	Limit: Motor I2T-model exceeds 100%
0x441B	Warning	Drive	Limit: Velocity limitation	The threshold values for the maximum speed were exceeded.
0x441C	Warning	Drive	STO while the axis was enabled	An attempt was made to activate the axis, despite the fact that no voltage is present at the STO input.
0x4600	Warning	General IO	Wrong supply voltage range	Supply voltage not in the correct range
0x4610	Warning	General IO	Wrong output voltage range	Output voltage not in the correct range
0x4705	Warning		Processor usage at %d %%	Processor load at %d %%
0x470A	Warning		EtherCAT Frame missed (change Settings or DC Operation Mode or Sync0 Shift Time)	EtherCAT frame missed (change DC Operation Mode or Sync0 Shift Time under Settings)

Text ID	Type	Place	Text Message	Additional comment
0x8000	Error	System	%s: %s	
0x8001	Error	System	Error: 0x%X, 0x%X, 0x%X	General error; parameters depend on event. See device documentation for interpretation.
0x8002	Error	System	Communication aborted	Communication aborted
0x8003	Error	System	Configuration error: 0x%X, 0x%X, 0x%X	General; parameters depend on event. See device documentation for interpretation.
0x8004	Error	System	%s: Unsuccessful FwdOpen-Response received from %d.%d.%d.%d (%s) (Error: %u)	
0x8005	Error	System	%s: FwdClose-Request sent to %d.%d.%d.%d (%s)	
0x8006	Error	System	%s: Unsuccessful FwdClose-Response received from %d.%d.%d.%d (%s) (Error: %u)	
0x8007	Error	System	%s: Connection with %d.%d.%d.%d (%s) closed	
0x8100	Error	General	Status word set: 0x%X, %d	Error bit set in the status word
0x8101	Error	General	Operation mode incompatible to PDO interface: 0x%X, %d	Mode of operation incompatible with the PDO interface
0x8102	Error	General	Invalid combination of Inputs and Outputs PDOs	Invalid combination of input and output PDOs
0x8103	Error	General	No variable linkage	No variables linked
0x8104	Error	General	Terminal-Overtemperature	The internal temperature of the terminal exceeds the parameterized error threshold. Activation of the terminal is prevented
0x8105	Error	General	PD-Watchdog	Communication between the fieldbus and the output stage is secured by a Watchdog. The axis is stopped automatically if the fieldbus communication is interrupted. <ul style="list-style-type: none"> <li>The EtherCAT connection was interrupted during operation.</li> <li>The Master was switched to Config mode during operation.</li> </ul>
0x8135	Error	General	Cycle time has to be a multiple of 125 µs	The IO or NC cycle time divided by 125 µs does not produce a whole number.
0x8136	Error	General	Configuration error: invalid sampling rate	Configuration error: Invalid sampling rate
0x8137	Error	General	Electronic type plate: CRC error	Content of the external name plate memory invalid.
0x8140	Error	General	Sync Error	Real-time violation
0x8141	Error	General	Sync%X Interrupt lost	Sync%X Interrupt lost
0x8142	Error	General	Sync Interrupt asynchronous	Sync Interrupt asynchronous
0x8143	Error	General	Jitter too big	Jitter limit violation
0x817F	Error	General	Error: 0x%X, 0x%X, 0x%X	
0x8200	Error	Communication	Write access error: %d, %d	Error while writing
0x8201	Error	Communication	No communication to field-side (Auxiliary voltage missing)	<ul style="list-style-type: none"> <li>There is no voltage applied to the power contacts.</li> <li>A firmware update has failed.</li> </ul>
0x8281	Error	Communication	Ownership failed: %X	
0x8282	Error	Communication	To many Keys founded	
0x8283	Error	Communication	Key Creation failed: %X	
0x8284	Error	Communication	Key loading failed	
0x8285	Error	Communication	Reading Public Key failed: %X	
0x8286	Error	Communication	Reading Public EK failed: %X	
0x8287	Error	Communication	Reading PCR Value failed: %X	
0x8288	Error	Communication	Reading Certificate EK failed: %X	
0x8289	Error	Communication	Challenge could not be hashed: %X	
0x828A	Error	Communication	Tickstamp Process failed	
0x828B	Error	Communication	PCR Process failed: %X	
0x828C	Error	Communication	Quote Process failed: %X	
0x82FF	Error	Communication	Bootmode not activated	Boot mode not activated
0x8300	Error	Encoder	Set position error: 0x%X, %d	Error while setting the position

Text ID	Type	Place	Text Message	Additional comment
0x8301	Error	Encoder	Encoder increments not configured: 0x%X, %d	Encoder increments not configured
0x8302	Error	Encoder	Encoder error	The amplitude of the resolver is too small
0x8303	Error	Encoder	Encoder power missing (channel %d)	
0x8304	Error	Encoder	Encoder communication error, channel: %X	Encoder communication error
0x8305	Error	Encoder	EnDat2.2 is not supported, channel: %X	EnDat2.2 is not supported
0x8306	Error	Encoder	Delay time, tolerance limit exceeded, 0x%X, channel: %X	Runtime measurement, tolerance exceeded
0x8307	Error	Encoder	Delay time, maximum value exceeded, 0x%X, channel: %X	Runtime measurement, maximum value exceeded
0x8308	Error	Encoder	Unsupported ordering designation, 0x%X, channel: %X (only 02 and 22 is supported)	Wrong EnDat order ID
0x8309	Error	Encoder	Encoder CRC error, channel: %X	Encoder CRC error
0x830A	Error	Encoder	Temperature %X could not be read, channel: %X	Temperature cannot be read
0x830C	Error	Encoder	Encoder Single-Cycle-Data Error, channel: %X	CRC error detected. Check the transmission path and the CRC polynomial
0x830D	Error	Encoder	Encoder Watchdog Error, channel: %X	The sensor has not responded within a predefined time period
0x8310	Error	Encoder	Initialisation error	
0x8311	Error	Encoder	Maximum frequency of the input signal is exceeded (channel %d)	
0x8312	Error	Encoder	Encoder plausibility error (channel %d)	
0x8313	Error	Encoder	Configuration error (channel %d)	
0x8314	Error	Encoder	Synchronisation error	
0x8315	Error	Encoder	Error status input (channel %d)	
0x8400	Error	Drive	Incorrect drive configuration: 0x%X, %d	Drive incorrectly configured
0x8401	Error	Drive	Limiting of calibration velocity: %d, %d	Limitation of the calibration velocity
0x8402	Error	Drive	Emergency stop activated: 0x%X, %d	Emergency stop activated
0x8403	Error	Drive	ADC Error	Error during current measurement in the ADC
0x8404	Error	Drive	Overcurrent	Overcurrent in phase U, V or W
0x8405	Error	Drive	Invalid modulo position: %d	Modulo position invalid
0x8406	Error	Drive	DC-Link undervoltage (Error)	The DC link voltage of the terminal is lower than the parameterized minimum voltage. Activation of the output stage is prevented.
0x8407	Error	Drive	DC-Link overvoltage (Error)	The DC link voltage of the terminal is higher than the parameterized maximum voltage. Activation of the output stage is prevented.
0x8408	Error	Drive	I2T-Model Amplifier overload (Error)	<ul style="list-style-type: none"> <li>The amplifier is being operated outside the specification.</li> <li>The I2T-model of the amplifier is incorrectly parameterized.</li> </ul>
0x8409	Error	Drive	I2T-Model motor overload (Error)	<ul style="list-style-type: none"> <li>The motor is being operated outside the parameterized rated values.</li> <li>The I2T-model of the motor is incorrectly parameterized.</li> </ul>
0x840A	Error	Drive	Overall current threshold exceeded	Total current exceeded
0x8415	Error	Drive	Invalid modulo factor: %d	Modulo factor invalid
0x8416	Error	Drive	Motor overtemperature	The internal temperature of the motor exceeds the parameterized error threshold. The motor stops immediately. Activation of the output stage is prevented.
0x8417	Error	Drive	Maximum rotating field velocity exceeded	Rotary field speed exceeds the value specified for dual use (EU 1382/2014).
0x841C	Error	Drive	STO while the axis was enabled	An attempt was made to activate the axis, despite the fact that no voltage is present at the STO input.

Text ID	Type	Place	Text Message	Additional comment
0x8550	Error	Inputs	Zero crossing phase %X missing	Zero crossing phase %X missing
0x8551	Error	Inputs	Phase sequence Error	Wrong direction of rotation
0x8552	Error	Inputs	Overcurrent phase %X	Overcurrent phase %X
0x8553	Error	Inputs	Overcurrent neutral wire	Overcurrent neutral wire
0x8581	Error	Inputs	Wire broken Ch %D	Wire broken Ch %d
0x8600	Error	General IO	Wrong supply voltage range	Supply voltage not in the correct range
0x8601	Error	General IO	Supply voltage to low	Supply voltage too low
0x8602	Error	General IO	Supply voltage to high	Supply voltage too high
0x8603	Error	General IO	Over current of supply voltage	Overcurrent of supply voltage
0x8610	Error	General IO	Wrong output voltage range	Output voltage not in the correct range
0x8611	Error	General IO	Output voltage to low	Output voltage too low
0x8612	Error	General IO	Output voltage to high	Output voltage too high
0x8613	Error	General IO	Over current of output voltage	Overcurrent of output voltage
0x8700	Error		Channel/Interface not calibrated	Channel/interface not synchronized
0x8701	Error		Operating time was manipulated	Operating time was manipulated
0x8702	Error		Oversampling setting is not possible	Oversampling setting not possible
0x8703	Error		No slave controller found	No slave controller found
0x8704	Error		Slave controller is not in Bootstrap	Slave controller is not in bootstrap
0x8705	Error		Processor usage to high (>= 100%%)	Processor load too high (>= 100%%)
0x8706	Error		Channel in saturation	Channel in saturation
0x8707	Error		Channel overload	Channel overload
0x8708	Error		Overloadtime was manipulated	Overload time was manipulated
0x8709	Error		Saturationtime was manipulated	Saturation time was manipulated
0x870A	Error		Channel range error	Measuring range error for the channel
0x870B	Error		no ADC clock	No ADC clock available
0xFFFF	Information		Debug: 0x%X, 0x%X, 0x%X	Debug: 0x%X, 0x%X, 0x%X



## 5.6 Object description and parameterization



### EtherCAT XML Device Description

The display matches that of the CoE objects from the EtherCAT ESI Device Description ([XML](#)). We recommend downloading the latest XML file from the download area of the [Beckhoff website](#) and installing it according to installation instructions.



### Parameterization via the CoE list (CAN over EtherCAT)

The EtherCAT device is parameterized via the [CoE-Online tab](#) [[▶ 69](#)] (double-click on the respective object) or via the [Process Data tab](#) [[▶ 66](#)] (allocation of PDOs). Please note the following general [CoE notes](#) [[▶ 24](#)] when using/manipulating the CoE parameters:

- Keep a startup list if components have to be replaced
- Differentiation between online/offline dictionary, existence of current XML description
- use “[CoE reload](#) [[▶ 177](#)]” for resetting changes

### 5.6.1 Profile-specific objects

#### Index 60p0 Status Ch. p

Index (hex)	Name	Meaning	Data type	Flags	Default
0x60p0:00	State	Max. Subindex	UINT8	RO	0x0F (15 <sub>dec</sub> )
0x60p0:01	Busy	The channel executes a command. Further commands are ignored when triggered.	BOOLEAN	RO	0x00 (0 <sub>dec</sub> )
0x60p0:02	Transmit	Indicates a frame transmission in progress. Frame buffer is in use and cannot be updated or sent while this bit is TRUE.	BOOLEAN	RO	0x00 (0 <sub>dec</sub> )
0x60p0:03	Send Acknowledge	Toggles when the send command flag is confirmed and data is transmitted to the LEDs.  Does not switch if the command was invalid or the device was in an invalid state (e.g. updating the frame buffer while transmitting a frame).	BOOLEAN	RO	0x00 (0 <sub>dec</sub> )
0x60p0:04	Update Acknowledge	Toggles when the update command flag is confirmed and data is copied to the frame buffer.  Does not switch if the command was invalid or the device was in an invalid state (e.g. updating the frame buffer while transmitting a frame).	BOOLEAN	RO	0x00 (0 <sub>dec</sub> )
0x60p0:05	Command Acknowledge	Toggles when the command was executed successfully.  Does not switch if the command was invalid or if the device was in an invalid state (e.g. updating the frame buffer while frame transmission was in progress).	BOOLEAN	RO	0x00 (0 <sub>dec</sub> )
0x60p0:06	Command Reject	Indicates that the command was rejected. This bit remains TRUE until the next command is executed. The rejection can have various reasons: invalid command, index, parameter... Check the diagnosis messages for more information.	BOOLEAN	RO	0x00 (0 <sub>dec</sub> )
0x60p0:0B	Warning	A warning has occurred. Check the diagnosis messages for more information.	BOOLEAN	RO	0x0000 (0 <sub>dec</sub> )
0x60p0:0C	Error	An error has occurred. Check the diagnosis messages for more information.	BOOLEAN	RO	0x0000 (0 <sub>dec</sub> )
0x60p0:0F	Input CycleCounter	Is incremented with each process data cycle and switches to 0 after its maximum value of 3.	BIT2	RO	0x0000 (0 <sub>dec</sub> )

## Index 70p0 Control Ch. p

Index (hex)	Name	Meaning	Data type	Flags	Default
0x70p0:00	Control	Max. Subindex	UINT8	RO	0x24 (36 <sub>dec</sub> )
0x70p0:01	Execute	Execute the command when toggling ("Busy" bit must be FALSE, otherwise the command will be rejected).	BOOLEAN	RW	0x00 (0 <sub>dec</sub> )
0x70p0:02	Send	Send frame buffer after command execution ("Transmit" bit must be FALSE, otherwise frame output is delayed).	BOOLEAN	RW	0x00 (0 <sub>dec</sub> )
0x70p0:03	Update	Update the frame buffer after executing the command ("Transmit" bit must be FALSE, otherwise the frame buffer update will be delayed).	BOOLEAN	RW	0x00 (0 <sub>dec</sub> )
0x70p0:09	Command	Specifies the command to be executed. The command uses "index", "length", "parameter" and "color": 0: No Operation 1: Fill (Index, Length, Color) 2: Clear (Index, Length) 3: Copy (Index, Length, Parameter) 4: Move (Index, Length, Parameter) 5: Rotate Left (Index, Length, Parameter) 6: Rotate Right (Index, Length, Parameter) 7: Reverse (Index, Length) 8: Gradient Color 1 (Index, Length, Color) 9: Gradient Color 2 (Index, Length, Color)	UINT16	RW	0x00 (0 <sub>dec</sub> )
0x70p0:11	Index	Specifies the area of pixels to be processed with the command. The first pixel is index 0.	UINT16	RW	0x0000 (0 <sub>dec</sub> )
0x70p0:12	Length	Specifies the area of pixels to be processed with the command. The length is the number of pixels.	UINT16	RW	0x0000 (0 <sub>dec</sub> )
0x70p0:13	Parameter	The parameter depends on the command. • For Copy, Move, it specifies the destination. • In the case of rotate, it describes the number of pixels to be moved left or right.	UINT8	RW	0x0000 (0 <sub>dec</sub> )
0x70p0:21	Color__Red	Red part for RGBW color	UINT8	RW	0x00 (0 <sub>dec</sub> )
0x70p0:22	Color__Green	Green part for RGBW color	UINT8	RW	0x00 (0 <sub>dec</sub> )
0x70p0:23	Color__Blue	Blue part for RGBW color	UINT8	RW	0x00 (0 <sub>dec</sub> )
0x70p0:24	Color__White	White part for RGBW color (Only used if 4-byte color format is used (RGBW, GRBW, WRGB,...), otherwise ignored).	UINT8	RO	0x00 (0 <sub>dec</sub> )

## Index 70p1 Extended Ch. p

Index (hex)	Name	Meaning	Data type	Flags	Default
0x70p1:00	Extended	Max. Subindex	UINT8	RO	0x18 (24 <sub>dec</sub> )
0x70p1:01	Execute	Execute the command when toggling ("Busy" bit must be FALSE, otherwise the command will be rejected).	BOOLEAN	RW	0x00 (0 <sub>dec</sub> )
0x70p1:02	Send	Send frame buffer after command execution ("Transmit" bit must be FALSE, otherwise frame output is delayed).	BOOLEAN	RW	0x00 (0 <sub>dec</sub> )
0x70p1:03	Update	Update the frame buffer after executing the command ("Transmit" bit must be FALSE, otherwise the frame buffer update will be delayed).	BOOLEAN	RW	0x00 (0 <sub>dec</sub> )
0x70p1:04	Write	Write segment to the segment index.	BOOLEAN	RW	0x00 (0 <sub>dec</sub> )
0x70p1:09	Index	Specifies the segment index. The first segment has index 0.	UINT8	RW	0x00 (0 <sub>dec</sub> )
0x70p1:11	Segment__Element[0]	Segment color data with 8 bit RGBW components. The order of the colors is red, green, blue, white: 0xWWBBGGRR	UINT32	RW	0x00000000 (0 <sub>dec</sub> )
0x70p1:12	Segment__Element[1]		UINT32	RW	0x00000000 (0 <sub>dec</sub> )
0x70p1:13	Segment__Element[2]		UINT32	RW	0x00000000 (0 <sub>dec</sub> )
0x70p1:14	Segment__Element[3]		UINT32	RW	0x00000000 (0 <sub>dec</sub> )
0x70p1:15	Segment__Element[4]		UINT32	RW	0x00000000 (0 <sub>dec</sub> )
0x70p1:16	Segment__Element[5]		UINT32	RW	0x00000000 (0 <sub>dec</sub> )
0x70p1:17	Segment__Element[6]		UINT32	RW	0x00000000 (0 <sub>dec</sub> )
0x70p1:18	Segment__Element[7]		UINT32	RW	0x00000000 (0 <sub>dec</sub> )

## Index 80p0 Settings Ch. p

Index (hex)	Name	Meaning	Data type	Flags	Default
0x80p0:00	Settings	Max. Subindex	UINT8	RO	0x27 (39 <sub>dec</sub> )
0x80p0:01	Enable Custom Settings	Enables the custom settings (data rate, duty cycle, reset time and level).	BOOLEAN	RW	0x00 (0 <sub>dec</sub> )
0x80p0:02	Enable Cyclic Frame Output	Enables the cyclic frame output for.	BOOLEAN	RW	0x01 (1 <sub>dec</sub> )
0x80p0:03	Enable Watchdog Default Color	Defines the behavior during the transition to the Safe-Op state. <ul style="list-style-type: none"> <li>If enabled, all pixels will output the defined color.</li> <li>Otherwise, the last frame buffer is used.</li> </ul>	BOOLEAN	RW	0x00 (0 <sub>dec</sub> )
0x80p0:11	Number Of Pixel	Number of pixels to be transferred.	UINT16	RW	0x0100 (256 <sub>dec</sub> )
0x80p0:12	Chip type	Preset for LED chipset / protocol. <b>Asynchronous Protocol:</b> 0 = APA-104 1 = APA-109 2 = CS8812 3 = GS8206 4 = GS8208 5 = INK1002 6 = INK1003 8 = SK6812 9 = SK6813 10 = SK6822 11 = SM16703 12 = SM16704 13 = TM1803 14 = TM1804 15 = TM1809 16 = TM1812 17 = TM1814 20 = UCS1903 21 = UCS1912 22 = UCS2903 23 = UCS2912 24 = UCS2904 27 = WS2811 28 = WS2812(B) 29 = WS2813 30 = WS2815 31 = WS2818 <b>Synchronous Protocol:</b> 64 = APA101 65 = APA102 67 = GE8822 68 = HD107S 69 = P9813 70 = SK9822 73 = WS2801 74 = WS2803	ENUM[8]	RW	0x1C (28 <sub>dec</sub> )

Index (hex)	Name	Meaning	Data type	Flags	Default
0x80p0:13	Color Format	Color sequence for individual LEDs. 26 = RGB 27 = RGBW 30 = RGWB 37 = RBG 39 = RBGW 45 = RWGB 54 = RBWG 57 = RWBG 74 = GRB 75 = GRBW 78 = GRWB 96 = BRG 99 = BRGW 108 = WRGB 114 = BRWG 120 = WRBG 133 = GBR 135 = GBRW 141 = GWRB 144 = BGR 147 = BGRW 156 = WGRB 177 = BWRG 180 = WBRG 198 = GBWR 201 = GWBR 210 = BGWR 216 = WGBR 225 = BWGR 228 = WBGR	ENUM[8]	RW	0x1A (26 <sub>dec</sub> )
0x80p0:15	Custom Data Rate	Defines the data rate (custom setting). (Applied only when custom settings are enabled) 1 = 1 Mbit/s 2 = 2 Mbit/s 3 = 3 Mbit/s 4 = 4 Mbit/s 40 = 400 kbit/s 80 = 800 kbit/s	ENUM[8]	RW	0x50 (80 <sub>dec</sub> )

Index (hex)	Name	Meaning	Data type	Flags	Default
0x80p0:16	Custom Reset Time	Defines the time for the minimum reset duration (custom setting). Only for asynchronous pixel chipsets! (Applied only when custom settings are enabled) 10 = 100 µs 20 = 200 µs 30 = 300 µs 40 = 400 µs 50 = 500 µs 60 = 600 µs 70 = 700 µs 80 = 800 µs 90 = 900 µs 100 = 1000 µs	ENUM[8]	RW	0x32 (50 <sub>dec</sub> )
0x80p0:17	Custom Reset Level	Defines the logic level for the reset sequence (custom setting). Only for asynchronous pixel chipsets! (Applied only when custom settings are enabled) 0 = Low 1 = High	ENUM[8]	RW	0x00 (0 <sub>dec</sub> )
0x80p0:1A	Custom Duty Cycle High Bit	Defines the duty cycle for High Bit (custom setting). Only for asynchronous pixel chipsets! (Applied only when custom settings are enabled)	REAL32	RW	0x3F000000 (1056964608 <sub>dec</sub> )
0x80p0:1B	Custom Duty Cycle Low Bit	Defines the duty cycle for Low Bit (custom setting). Only for asynchronous pixel chipsets!	REAL32	RW	0x3E4CCCCD (1045220557 <sub>dec</sub> )
0x80p0:1C	Custom Start Frame	Defines a 4-byte start frame. Only for synchronous pixel chipsets! (Applied only when custom settings are enabled)	UINT32	RW	0x00 (0 <sub>dec</sub> )
0x80p0:1D	Custom Stop Frame	Defines a 4-byte stop frame. Only for synchronous pixel chipsets! (Applied only when custom settings are enabled)	UINT32	RW	0x00 (0 <sub>dec</sub> )
0x80p0:1E	Gamma Correction	Value for gamma correction	REAL32	RW	0x3F800000 (1065353216 <sub>dec</sub> )
0x80p0:1F	Brightness Scale	Global scale for pixel brightness	REAL32	RW	0x3F800000 (1065353216 <sub>dec</sub> )
0x80p0:20	Current Setting Red	Driver current for the red LED	UINT8	RW	0x00000000 (0 <sub>dec</sub> )
0x80p0:21	Current Setting Green	Driver current for the green LED	UINT8	RW	0x00000000 (0 <sub>dec</sub> )
0x80p0:22	Current Setting Blue	Driver current for the blue LED	UINT8	RW	0x00000000 (0 <sub>dec</sub> )
0x80p0:23	Current Setting White	Driver current for the white LED	UINT8	RW	0x00000000 (0 <sub>dec</sub> )
0x80p0:24	Watchdog Default Color Red	Color component red in watchdog case	UINT8	RW	0x00 (0 <sub>dec</sub> )
0x80p0:25	Watchdog Default Color Green	Color component green in watchdog case	UINT8	RW	0x00 (0 <sub>dec</sub> )
0x80p0:26	Watchdog Default Color Blue	Color component blue in watchdog case	UINT8	RW	0x00 (0 <sub>dec</sub> )
0x80p0:27	Watchdog Default Color White	Color component white in watchdog case	UINT8	RW	0x00 (0 <sub>dec</sub> )

### Index 90p0 Info data

Index (hex)	Name	Meaning	Data type	Flags	Default
90p0:0	Info data	Max. Subindex	UINT8	RO	0x11 (17 <sub>dec</sub> )
90p0:11	Framebuffer Size	Assigned number of pixels of the channel	UINT16	RO	0x0000 (0 <sub>dec</sub> )

### Index F000 Modular device profile

Index (hex)	Name	Meaning	Data type	Flags	Default
F000:0	Modular device profile	General information for the Modular Device Profile	UINT8	RO	0x02 (2 <sub>dec</sub> )
F000:01	Module index distance	Index distance of the objects of the individual channels	UINT16	RO	0x0010 (16 <sub>dec</sub> )
F000:02	Maximum number of modules	Number of channels	UINT16	RO	0x0004 (4 <sub>dec</sub> )

### Index F008 Code word

Index (hex)	Name	Meaning	Data type	Flags	Default
F008:0	Code word	reserved	UINT32	RW	0x00000000 (0 <sub>dec</sub> )

### Index F081 Download revision

Index (hex)	Name	Meaning	Data type	Flags	Default
F081:0	Download revision	Max. Subindex	UINT8	RO	0x01 (1 <sub>dec</sub> )
F081:01	Revision number	The subindex 0xF081:01 (Download revision) describes the revision level of the terminal / module.	UINT32	RW	0x00000000 (0 <sub>dec</sub> )

### Index F815 PLED Vendor data

Index (hex)	Name	Meaning	Data type	Flags	Default
F815:0	PLED Vendor data	Max. Subindex	UINT8	RO	0x14 (20 <sub>dec</sub> )
F815:11	PCB Temperature Warn Level	Threshold value for overtemperature warning [0.1°C]	UINT16	RW	0x0320 (800 <sub>dec</sub> )
F815:12	PCB Temperature Error Level	Threshold value for overtemperature error. [0.1°C]	UINT16	RW	0x03E8 (1000 <sub>dec</sub> )
F815:13	LED Pixel Update Interval	Update interval for cyclic frame output. [ms]	UINT16	RW	0x00FA (250 <sub>dec</sub> )
F815:14	Hardware Variant ID		REAL32	RW	0x00001313 (4883 <sub>dec</sub> )

### Index FA15 PLED Diag data

Index (hex)	Name	Meaning	Data type	Flags	Default
FA15:0	PLED Diag data	Max. Subindex	UINT8	RO	0x11 (17 <sub>dec</sub> )
FA15:01	Field Power Supply	Indicator for the field power supply	BOOLEAN	RO	
FA15:11	PCB Temperature	Actual PCB temperature [0.1°C]	UINT16	RO	

### Index FB00 PLED Command

Index (hex)	Name	Meaning	Data type	Flags	Default
FB00:0	PLED Command	Terminal-specific commands can be executed via PLED Command.	UINT8	RO	0x03 (3 <sub>dec</sub> )
FB00:01	Request		OCTET-STRING[2]	RW	{0}
FB00:02	State		UINT8	RO	0x00 (0 <sub>dec</sub> )
FB00:03	Response		OCTET-STRING[4]	RO	{0}

### Index FB40 Memory interface

Index (hex)	Name	Meaning	Data type	Flags	Default
FB40:0	Memory interface	Max. Subindex	UINT8	RO	0x03 (3 <sub>dec</sub> )
FB40:01	Address		UINT32	RW	0x00 (0 <sub>dec</sub> )
FB40:02	Length		UINT32	RO	0x00 (0 <sub>dec</sub> )
FB40:03	Data		OCTET-STRING[8]	RO	{0}

## 5.6.2 Standard objects

### Index 1000 Device type

Index (hex)	Name	Meaning	Data type	Flags	Default
1000:0	Device type	Device type of the EtherCAT slave: the Lo-Word contains the used CoE profile (5001). The Hi-Word contains the module profile according to the modular device profile.	UINT32	RO	0x00001389 (5001 <sub>dec</sub> )

### Index 1008 Device name

Index (hex)	Name	Meaning	Data type	Flags	Default
1008:0	Device name	Device name of the EtherCAT slave	STRING	RO	EL2574

### Index 1009 Hardware version

Index (hex)	Name	Meaning	Data type	Flags	Default
1009:0	Hardware version	Hardware version of the EtherCAT slave	STRING	RO	

### Index 100A Software version

Index (hex)	Name	Meaning	Data type	Flags	Default
100A:0	Software version	Firmware version of the EtherCAT slave	STRING	RO	01

### Index 100B Bootloader version

Index (hex)	Name	Meaning	Data type	Flags	Default
100B:0	Bootloader version	Bootloader version of the EtherCAT slave	STRING	RO	N/A

### Index 1011 Restore default parameters

Index (hex)	Name	Meaning	Data type	Flags	Default
1011:0	<a href="#">Restore default parameters</a> <a href="#">▶ 177</a>	Restore default parameters	UINT8	RO	0x01 (1 <sub>dec</sub> )
1011:01	SubIndex 001	If this object is set to "0x64616F6C" in the Set Value dialog, all backup objects are reset to their delivery state.	UINT32	RW	0x00000000 (0 <sub>dec</sub> )

### Index 1018 Identity

Index (hex)	Name	Meaning	Data type	Flags	Default
1018:0	Identity	Information for identifying the slave	UINT8	RO	0x04 (4 <sub>dec</sub> )
1018:01	Vendor ID	Vendor ID of the EtherCAT slave	UINT32	RO	0x00000002 (2 <sub>dec</sub> )
1018:02	Product code	Product code of the EtherCAT slave	UINT32	RO	0x0A0E3052 (168702034 <sub>dec</sub> )
1018:03	Revision	Revision number of the EtherCAT slave; the Low Word (bit 0-15) indicates the special terminal number, the High Word (bit 16-31) refers to the device description	UINT32	RO	0x00000000 (0 <sub>dec</sub> )
1018:04	Serial number	Serial number of the EtherCAT slave; the Low Byte (bit 0-7) of the Low Word contains the year of production, the High Byte (bit 8-15) of the Low Word contains the week of production, the High Word (bit 16-31) is 0	UINT32	RO	0x00000000 (0 <sub>dec</sub> )

### Index 10F0 Backup parameter handling

Index (hex)	Name	Meaning	Data type	Flags	Default
10F0:0	Backup parameter handling	Information for standardized loading and saving of backup entries	UINT8	RO	0x01 (1 <sub>dec</sub> )
10F0:01	Checksum	Checksum across all backup entries of the EtherCAT slave	UINT32	RO	0x00000000 (0 <sub>dec</sub> )



**Index 10F3 Diagnosis History**

Index (hex)	Name	Meaning	Data type	Flags	Default
10F3:0	Diagnosis History	Max. Subindex	UINT8	RO	0x1E (30 <sub>dec</sub> )
10F3:01	Maximum Messages	Maximum number of stored messages A maximum of 50 messages can be stored	UINT8	RO	0x00 (0 <sub>dec</sub> )
10F3:02	Newest Message	Subindex of the latest message	UINT8	RO	0x00 (0 <sub>dec</sub> )
10F3:03	Newest Acknowledged Message	Subindex of the last confirmed message	UINT8	RW	0x00 (0 <sub>dec</sub> )
10F3:04	New Messages Available	Indicates that a new message is available	BOOLEAN	RO	0x00 (0 <sub>dec</sub> )
10F3:05	Flags	not used	UINT16	RW	0x0000 (0 <sub>dec</sub> )
10F3:06	Diagnosis Message 001	Message 1	OCTET-STRING[28]	RO	{0}
...	...	...	OCTET-STRING[28]	RO	{0}
10F3:1E	Diagnosis Message 030	Message 30	OCTET-STRING[28]	RO	{0}

**Index 10F8 Actual Time Stamp**

Index (hex)	Name	Meaning	Data type	Flags	Default
10F8:0	Actual Time Stamp	Timestamp	UINT64	RO	

**Index 1C00 Sync manager type**

Index (hex)	Name	Meaning	Data type	Flags	Default
1C00:0	Sync manager type	Using the Sync Managers	UINT8	RO	0x04 (4 <sub>dec</sub> )
1C00:01	SubIndex 001	Sync-Manager Type Channel 1: Mailbox Write	UINT8	RO	0x01 (1 <sub>dec</sub> )
1C00:02	SubIndex 002	Sync-Manager Type Channel 2: Mailbox Read	UINT8	RO	0x02 (2 <sub>dec</sub> )
1C00:03	SubIndex 003	Sync-Manager Type Channel 3: Process Data Write (Outputs)	UINT8	RO	0x03 (3 <sub>dec</sub> )
1C00:04	SubIndex 004	Sync-Manager Type Channel 4: Process Data Read (Inputs)	UINT8	RO	0x04 (4 <sub>dec</sub> )

**Index 1C12 RxPDO assign**

Index (hex)	Name	Meaning	Data type	Flags	Default
1C12:0	RxPDO assign	PDO Assign Outputs	UINT8	RW	0x01 (1 <sub>dec</sub> )
1C12:01	Subindex 001	1. allocated RxPDO (contains the index of the associated RxPDO mapping object)	UINT16	RW	0x1600 (5632 <sub>dec</sub> )
1C12:02	Subindex 002	2. allocated RxPDO (contains the index of the associated RxPDO mapping object)	UINT16	RW	0x0000 (0 <sub>dec</sub> )
1C12:03	Subindex 003	3. allocated RxPDO (contains the index of the associated RxPDO mapping object)	UINT16	RW	0x0000 (0 <sub>dec</sub> )
1C12:04	Subindex 004	4. allocated RxPDO (contains the index of the associated RxPDO mapping object)	UINT16	RW	0x0000 (0 <sub>dec</sub> )
1C12:05	Subindex 005	5. allocated RxPDO (contains the index of the associated RxPDO mapping object)	UINT16	RW	0x0000 (0 <sub>dec</sub> )
1C12:06	Subindex 006	6. allocated RxPDO (contains the index of the associated RxPDO mapping object)	UINT16	RW	0x0000 (0 <sub>dec</sub> )
1C12:07	Subindex 007	7. allocated RxPDO (contains the index of the associated RxPDO mapping object)	UINT16	RW	0x0000 (0 <sub>dec</sub> )

**Index 1C13 TxPDO assign**

Index (hex)	Name	Meaning	Data type	Flags	Default
1C13:0	TxPDO assign	PDO Assign Inputs	UINT8	RW	0x01 (1 <sub>dec</sub> )
1C13:01	Subindex 001	1. allocated TxPDO (contains the index of the associated TxPDO mapping object)	UINT16	RW	0x1A00 (6656 <sub>dec</sub> )
1C13:02	Subindex 002	2. allocated TxPDO (contains the index of the associated TxPDO mapping object)	UINT16	RW	0x0000 (0 <sub>dec</sub> )

## Index 1C32 SM output parameter

Index (hex)	Name	Meaning	Data type	Flags	Default
1C32:0	SM output parameter	Synchronization parameters for the outputs	UINT8	RO	0x20 (32 <sub>dec</sub> )
1C32:01	Sync mode	Current synchronization mode: <ul style="list-style-type: none"> <li>0: Free Run</li> <li>1: Synchron with SM 2 Event</li> <li>2: DC-Mode - Synchron with SYNC0 Event</li> <li>3: DC-Mode - Synchron with SYNC1 Event</li> </ul>	UINT16	RW	0x0001 (1 <sub>dec</sub> )
1C32:02	Cycle time	Cycle time (in ns): <ul style="list-style-type: none"> <li>Free Run: cycle time of the local timer</li> <li>Synchron with SM 2 Event: cycle time of the master</li> <li>DC-Mode: SYNC0/SYNC1 Cycle Time</li> </ul>	UINT32	RW	0x000F4240 (1000000 <sub>dec</sub> )
1C32:03	Shift time	Time between SYNC0 event and output of the outputs (in ns, DC mode only)	UINT32	RO	0x00000000 (0 <sub>dec</sub> )
1C32:04	Sync modes supported	Supported synchronization modes: <ul style="list-style-type: none"> <li>Bit 0 = 1: Free Run is supported</li> <li>Bit 1 = 1: Synchron with SM 2 Event is supported</li> <li>Bit 2-3 = 01: DC-Mode is supported</li> <li>Bit 4-5 = 10: Output Shift with SYNC1 Event (only DC mode)</li> <li>Bit 14 = 1: dynamic times (measurement through writing of 1C32:08)</li> </ul>	UINT16	RO	0x0002 (2 <sub>dec</sub> )
1C32:05	Minimum cycle time	Minimum cycle time (in ns)	UINT32	RO	0x000186A0 (100000 <sub>dec</sub> )
1C32:06	Calc and copy time	Minimum time between SYNC0 and SYNC1 event (in ns, DC mode only)	UINT32	RO	0x00000000 (0 <sub>dec</sub> )
1C32:07	Minimum delay time		UINT32	RO	0x00000000 (0 <sub>dec</sub> )
1C32:08	Get Cycle Time	<ul style="list-style-type: none"> <li>0: Measurement of the local cycle time is stopped</li> <li>1: Measurement of the local cycle time is started</li> </ul> <p>The entries 0x1C32:03, 0x1C32:05, 0x1C32:06, 0x1C32:09, 0x1C33:03, 0x1C33:06, 0x1C33:09 <a href="#">▶ 147</a> are updated with the maximum measured values. For a subsequent measurement the measured values are reset</p>	UINT16	RW	0x0000 (0 <sub>dec</sub> )
1C32:09	Maximum delay time	Time between SYNC1 event and output of the outputs (in ns, DC mode only)	UINT32	RO	0x00000000 (0 <sub>dec</sub> )
1C32:0B	SM event missed counter	Number of missed SM events in OPERATIONAL (DC mode only)	UINT16	RO	0x0000 (0 <sub>dec</sub> )
1C32:0C	Cycle exceeded counter	Number of occasions the cycle time was exceeded in OPERATIONAL (cycle was not completed in time or the next cycle began too early)	UINT16	RO	0x0000 (0 <sub>dec</sub> )
1C32:0D	Shift too short counter	Number of occasions that the interval between SYNC0 and SYNC1 event was too short (DC mode only)	UINT16	RO	0x0000 (0 <sub>dec</sub> )
1C32:20	Sync error	The synchronization was not correct in the last cycle (outputs were output too late; DC mode only)	BOOLEAN	RO	0x00 (0 <sub>dec</sub> )

# Index 1C33 SM input parameter

Index (hex)	Name	Meaning	Data type	Flags	Default
1C33:0	SM input parameter	Synchronization parameters for the inputs	UINT8	RO	0x20 (32 <sub>dec</sub> )
1C33:01	Sync mode	Current synchronization mode: <ul style="list-style-type: none"> <li>• 0: Free Run</li> <li>• 1: Synchron with SM 3 Event (no outputs available)</li> <li>• 2: DC - Synchron with SYNC0 Event</li> <li>• 3: DC - Synchron with SYNC1 Event</li> <li>• 34: Synchron with SM 2 Event (outputs available)</li> </ul>	UINT16	RW	0x0022 (34 <sub>dec</sub> )
1C33:02	Cycle time	as 0x1C32:02 [► 146]	UINT32	RW	0x000F4240 (1000000 <sub>dec</sub> )
1C33:03	Shift time	Time between SYNC0 event and reading of the inputs (in ns, DC mode only)	UINT32	RO	0x00000000 (0 <sub>dec</sub> )
1C33:04	Sync modes supported	Supported synchronization modes: <ul style="list-style-type: none"> <li>• Bit 0: Free Run is supported</li> <li>• Bit 1: Synchron with SM 2 Event is supported (outputs available)</li> <li>• Bit 1: Synchron with SM 3 Event is supported (no outputs available)</li> <li>• Bit 2-3 = 01: DC-Mode is supported</li> <li>• Bit 4-5 = 01: Input Shift through local event (outputs available)</li> <li>• Bit 4-5 = 10: Input Shift with SYNC1 Event (no outputs available)</li> <li>• Bit 14 = 1: dynamic times (measurement through writing of 0x1C32:08 [► 146] or 0x1C33:08)</li> </ul>	UINT16	RO	0x0002 (2 <sub>dec</sub> )
1C33:05	Minimum cycle time	as 0x1C32:05 [► 146]	UINT32	RO	0x000186A0 (100000 <sub>dec</sub> )
1C33:06	Calc and copy time	Time between reading of the inputs and availability of the inputs for the master (in ns, only DC mode)	UINT32	RO	0x00000000 (0 <sub>dec</sub> )
1C33:07	Minimum delay time		UINT32	RO	0x00000000 (0 <sub>dec</sub> )
1C33:08	Command	as 0x1C32:08 [► 146]	UINT16	RW	0x0000 (0 <sub>dec</sub> )
1C33:09	Maximum delay time	Time between SYNC1 event and reading of the inputs (in ns, only DC mode)	UINT32	RO	0x00000000 (0 <sub>dec</sub> )
1C33:0B	SM event missed counter	as 0x1C32:11 [► 146]	UINT16	RO	0x0000 (0 <sub>dec</sub> )
1C33:0C	Cycle exceeded counter	as 0x1C32:12 [► 146]	UINT16	RO	0x0000 (0 <sub>dec</sub> )
1C33:0D	Shift too short counter	as 0x1C32:13 [► 146]	UINT16	RO	0x0000 (0 <sub>dec</sub> )
1C33:20	Sync error	as 0x1C32:32 [► 146]	BOOLEAN	RO	0x00 (0 <sub>dec</sub> )

## 5.7 General Commissioning Instructions for an EtherCAT Slave

This summary briefly deals with a number of aspects of EtherCAT Slave operation under TwinCAT. More detailed information on this may be found in the corresponding sections of, for instance, the [EtherCAT System Documentation](#).

### Diagnosis in real time: WorkingCounter, EtherCAT State and Status

Generally speaking an EtherCAT Slave provides a variety of diagnostic information that can be used by the controlling task.

This diagnostic information relates to differing levels of communication. It therefore has a variety of sources, and is also updated at various times.

Any application that relies on I/O data from a fieldbus being correct and up to date must make diagnostic access to the corresponding underlying layers. EtherCAT and the TwinCAT System Manager offer comprehensive diagnostic elements of this kind. Those diagnostic elements that are helpful to the controlling task for diagnosis that is accurate for the current cycle when in operation (not during commissioning) are discussed below.

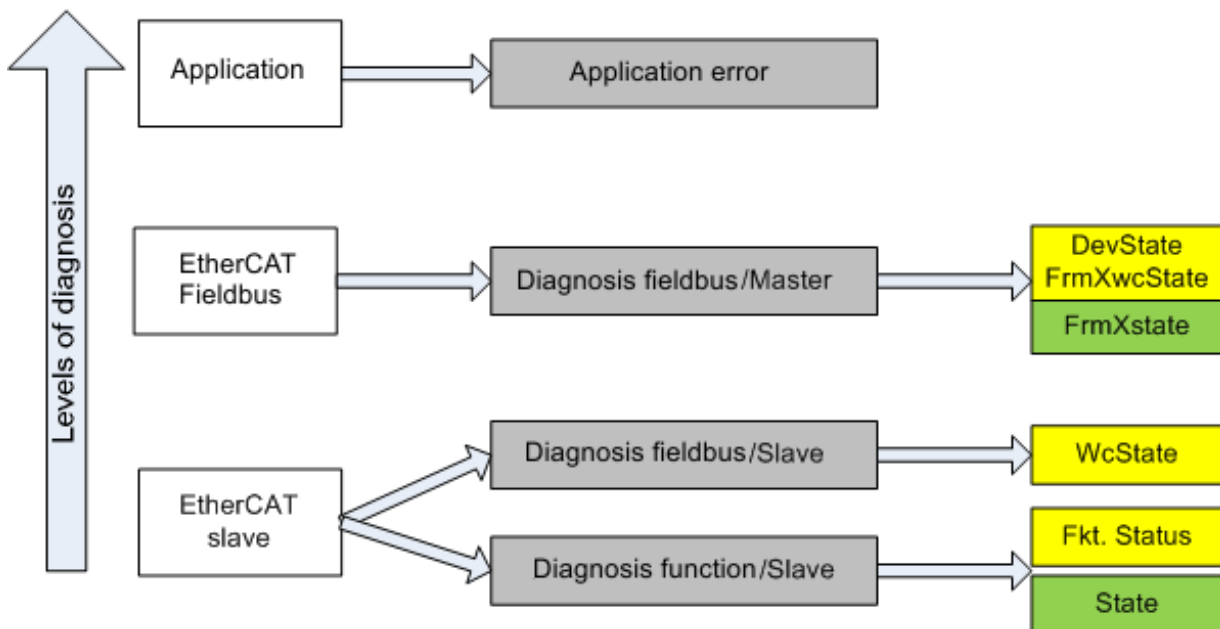


Fig. 145: Selection of the diagnostic information of an EtherCAT Slave

In general, an EtherCAT Slave offers

- communication diagnosis typical for a slave (diagnosis of successful participation in the exchange of process data, and correct operating mode)  
This diagnosis is the same for all slaves.

as well as

- function diagnosis typical for a channel (device-dependent)  
See the corresponding device documentation

The colors in Fig. *Selection of the diagnostic information of an EtherCAT Slave* also correspond to the variable colors in the System Manager, see Fig. *Basic EtherCAT Slave Diagnosis in the PLC*.

Colour	Meaning
yellow	Input variables from the Slave to the EtherCAT Master, updated in every cycle
red	Output variables from the Slave to the EtherCAT Master, updated in every cycle
green	Information variables for the EtherCAT Master that are updated acyclically. This means that it is possible that in any particular cycle they do not represent the latest possible status. It is therefore useful to read such variables through ADS.

Fig. Basic EtherCAT Slave Diagnosis in the PLC shows an example of an implementation of basic EtherCAT Slave Diagnosis. A Beckhoff EL3102 (2-channel analogue input terminal) is used here, as it offers both the communication diagnosis typical of a slave and the functional diagnosis that is specific to a channel. Structures are created as input variables in the PLC, each corresponding to the process image.

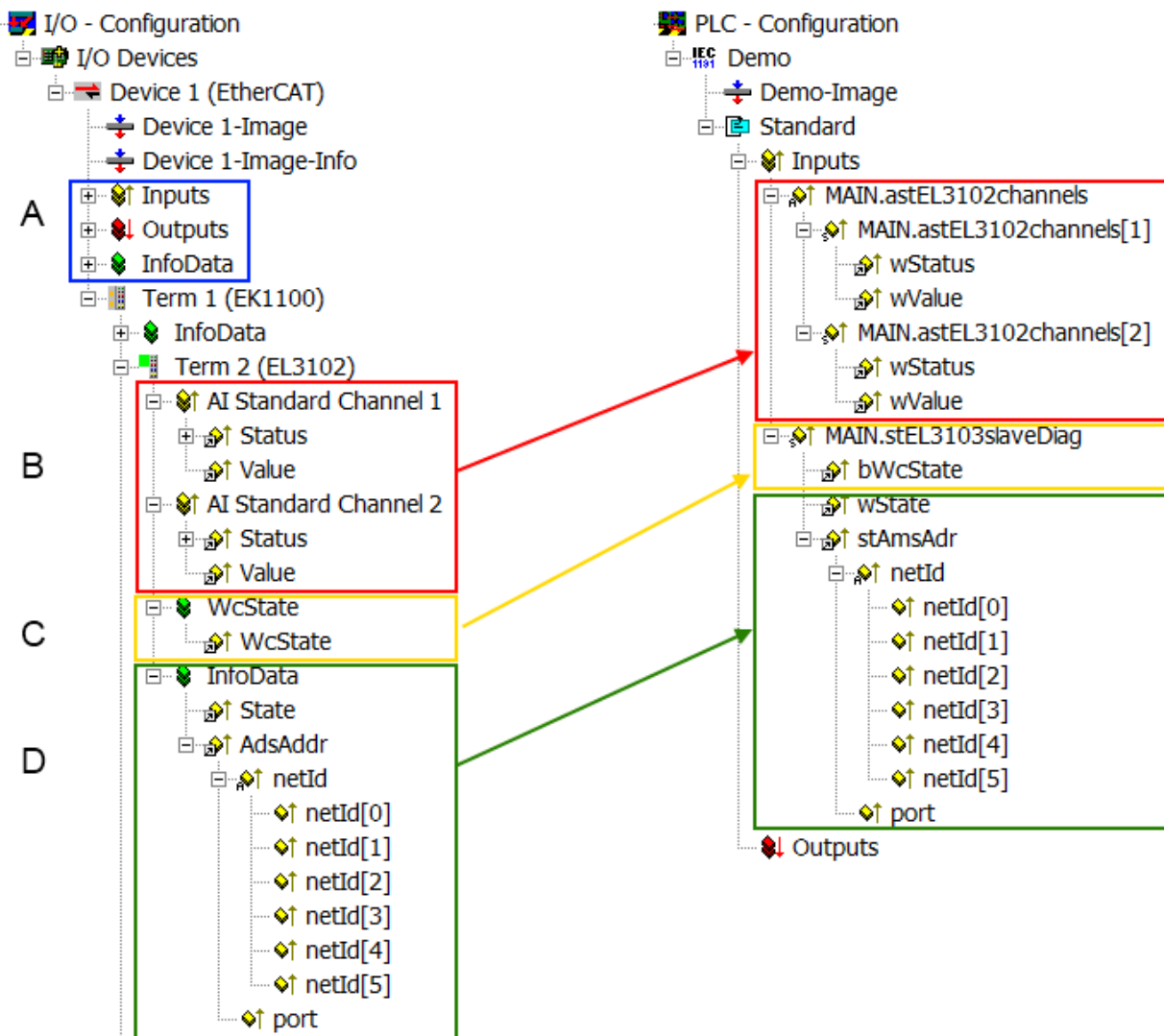


Fig. 146: Basic EtherCAT Slave Diagnosis in the PLC

The following aspects are covered here:

Code	Function	Implementation	Application/evaluation
A	The EtherCAT Master's diagnostic information updated cyclically (yellow) or provided acyclically (green).		At least the DevState is to be evaluated for the most recent cycle in the PLC.  The EtherCAT Master's diagnostic information offers many more possibilities than are treated in the EtherCAT System Documentation. A few keywords: <ul style="list-style-type: none"> <li>• CoE in the Master for communication with/through the Slaves</li> <li>• Functions from <i>TcEtherCAT.lib</i></li> <li>• Perform an OnlineScan</li> </ul>
B	In the example chosen (EL3102) the EL3102 comprises two analogue input channels that transmit a single function status for the most recent cycle.	Status <ul style="list-style-type: none"> <li>• the bit significations may be found in the device documentation</li> <li>• other devices may supply more information, or none that is typical of a slave</li> </ul>	In order for the higher-level PLC task (or corresponding control applications) to be able to rely on correct data, the function status must be evaluated there. Such information is therefore provided with the process data for the most recent cycle.
C	For every EtherCAT Slave that has cyclic process data, the Master displays, using what is known as a WorkingCounter, whether the slave is participating successfully and without error in the cyclic exchange of process data. This important, elementary information is therefore provided for the most recent cycle in the System Manager  1. at the EtherCAT Slave, and, with identical contents 2. as a collective variable at the EtherCAT Master (see Point A) for linking.	WcState (Working Counter) 0: valid real-time communication in the last cycle 1: invalid real-time communication This may possibly have effects on the process data of other Slaves that are located in the same SyncUnit	In order for the higher-level PLC task (or corresponding control applications) to be able to rely on correct data, the communication status of the EtherCAT Slave must be evaluated there. Such information is therefore provided with the process data for the most recent cycle.
D	Diagnostic information of the EtherCAT Master which, while it is represented at the slave for linking, is actually determined by the Master for the Slave concerned and represented there. This information cannot be characterized as real-time, because it <ul style="list-style-type: none"> <li>• is only rarely/never changed, except when the system starts up</li> <li>• is itself determined acyclically (e.g. EtherCAT Status)</li> </ul>	State current Status (INIT..OP) of the Slave. The Slave must be in OP (=8) when operating normally. <i>AdsAddr</i> The ADS address is useful for communicating from the PLC/task via ADS with the EtherCAT Slave, e.g. for reading/writing to the CoE. The AMS-NetID of a slave corresponds to the AMS-NetID of the EtherCAT Master; communication with the individual Slave is possible via the <i>port</i> (= EtherCAT address).	Information variables for the EtherCAT Master that are updated acyclically. This means that it is possible that in any particular cycle they do not represent the latest possible status. It is therefore possible to read such variables through ADS.

## NOTICE

### Diagnostic information

It is strongly recommended that the diagnostic information made available is evaluated so that the application can react accordingly.

### CoE Parameter Directory

The CoE parameter directory (CanOpen-over-EtherCAT) is used to manage the set values for the slave concerned. Changes may, in some circumstances, have to be made here when commissioning a relatively complex EtherCAT Slave. It can be accessed through the TwinCAT System Manager, see Fig. *EL3102, CoE directory*:

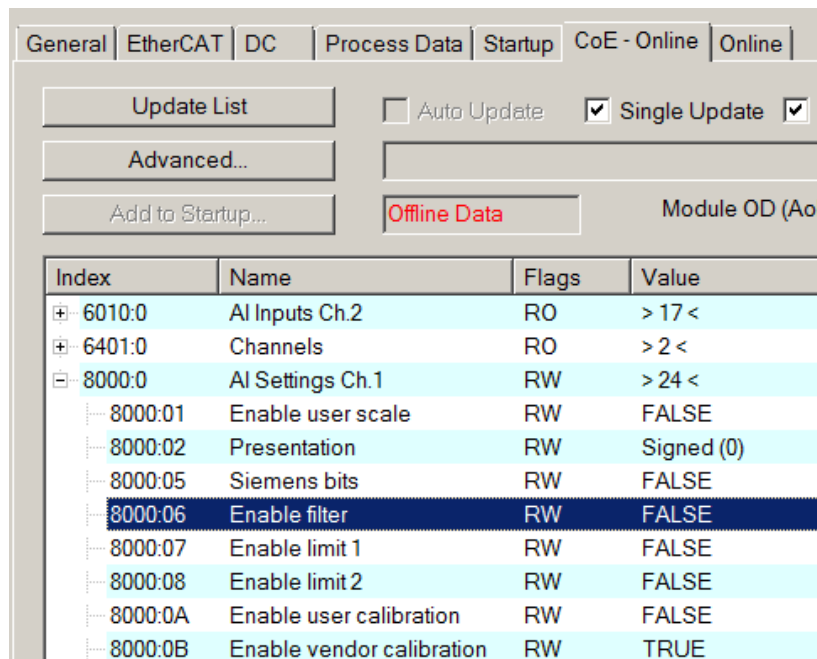


Fig. 147: EL3102, CoE directory

## **i** EtherCAT System Documentation

The comprehensive description in the [EtherCAT System Documentation](#) (EtherCAT Basics --> CoE Interface) must be observed!

A few brief extracts:

- Whether changes in the online directory are saved locally in the slave depends on the device. EL terminals (except the EL66xx) are able to save in this way.
- The user must manage the changes to the StartUp list.

## Commissioning aid in the TwinCAT System Manager

Commissioning interfaces are being introduced as part of an ongoing process for EL/EP EtherCAT devices. These are available in TwinCAT System Managers from TwinCAT 2.11R2 and above. They are integrated into the System Manager through appropriately extended ESI configuration files.

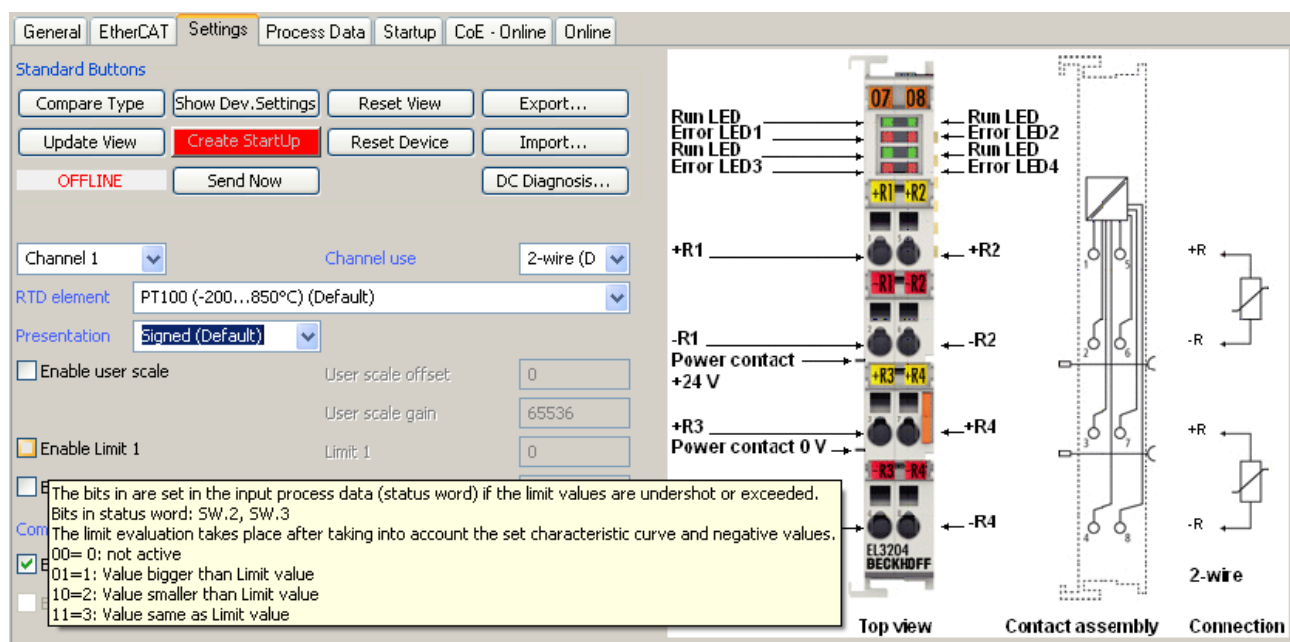


Fig. 148: Example of commissioning aid for a EL3204



This commissioning process simultaneously manages

- CoE Parameter Directory
- DC/FreeRun mode
- the available process data records (PDO)

Although the “Process Data”, “DC”, “Startup” and “CoE-Online” that used to be necessary for this are still displayed, it is recommended that, if the commissioning aid is used, the automatically generated settings are not changed by it.

The commissioning tool does not cover every possible application of an EL/EP device. If the available setting options are not adequate, the user can make the DC, PDO and CoE settings manually, as in the past.

### EtherCAT State: automatic default behaviour of the TwinCAT System Manager and manual operation

After the operating power is switched on, an EtherCAT Slave must go through the following statuses

- INIT
- PREOP
- SAFEOP
- OP

to ensure sound operation. The EtherCAT Master directs these statuses in accordance with the initialization routines that are defined for commissioning the device by the ES/XML and user settings (Distributed Clocks (DC), PDO, CoE). See also the section on "Principles of [Communication, EtherCAT State Machine \[► 22\]](#)" in this connection. Depending how much configuration has to be done, and on the overall communication, booting can take up to a few seconds.

The EtherCAT Master itself must go through these routines when starting, until it has reached at least the OP target state.

The target state wanted by the user, and which is brought about automatically at start-up by TwinCAT, can be set in the System Manager. As soon as TwinCAT reaches the status RUN, the TwinCAT EtherCAT Master will approach the target states.

### Standard setting

The advanced settings of the EtherCAT Master are set as standard:

- EtherCAT Master: OP
  - Slaves: OP
- This setting applies equally to all Slaves.

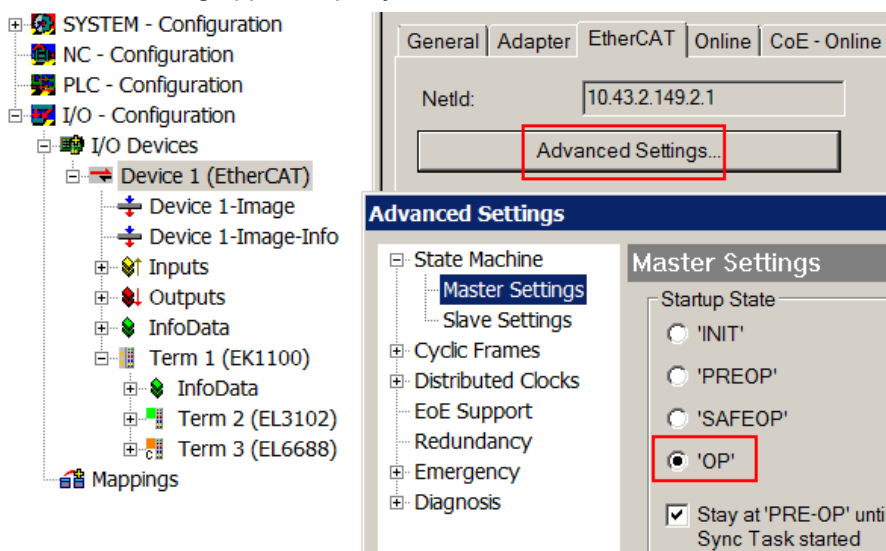


Fig. 149: Default behaviour of the System Manager



In addition, the target state of any particular Slave can be set in the “Advanced Settings” dialogue; the standard setting is again OP.

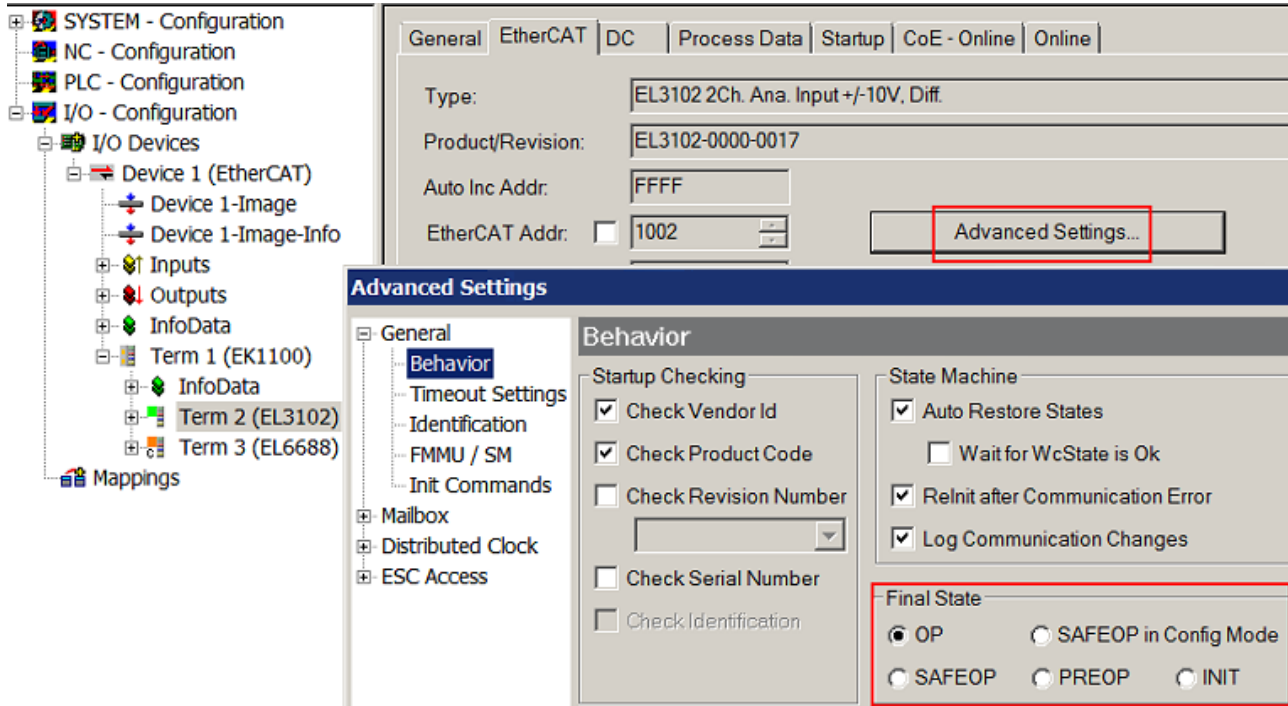


Fig. 150: Default target state in the Slave

## Manual Control

There are particular reasons why it may be appropriate to control the states from the application/task/PLC. For instance:

- for diagnostic reasons
- to induce a controlled restart of axes
- because a change in the times involved in starting is desirable

In that case it is appropriate in the PLC application to use the PLC function blocks from the *TcEtherCAT.lib*, which is available as standard, and to work through the states in a controlled manner using, for instance, *FB\_EcSetMasterState*.

It is then useful to put the settings in the EtherCAT Master to INIT for master and slave.

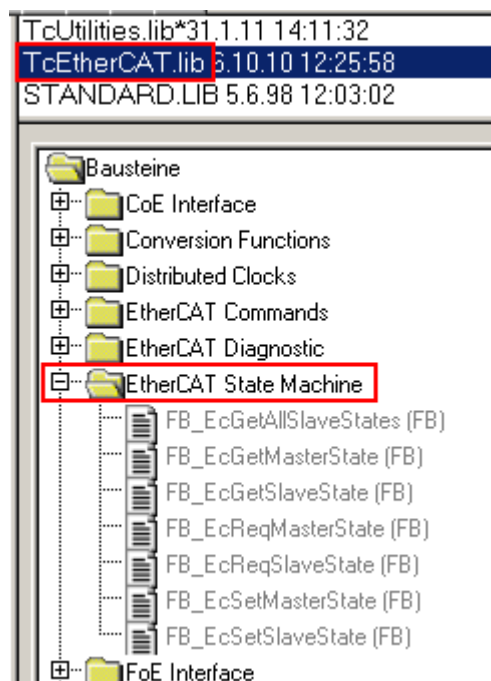


Fig. 151: PLC function blocks

### Note regarding E-Bus current

EL/ES terminals are placed on the DIN rail at a coupler on the terminal strand. A Bus Coupler can supply the EL terminals added to it with the E-bus system voltage of 5 V; a coupler is thereby loadable up to 2 A as a rule. Information on how much current each EL terminal requires from the E-bus supply is available online and in the catalogue. If the added terminals require more current than the coupler can supply, then power feed terminals (e.g. EL9410) must be inserted at appropriate places in the terminal strand.

The pre-calculated theoretical maximum E-Bus current is displayed in the TwinCAT System Manager as a column value. A shortfall is marked by a negative total amount and an exclamation mark; a power feed terminal is to be placed before such a position.

General   Adapter   <b>EtherCAT</b>   Online   CoE - Online						
NetId:		10.43.2.149.2.1		Advanced Settings...		
Number	Box Name	Address	Type	In Size	Out S...	E-Bus (..
1	Term 1 (EK1100)	1001	EK1100			
2	Term 2 (EL3102)	1002	EL3102	8.0		1830
3	Term 4 (EL2004)	1003	EL2004		0.4	1730
4	Term 5 (EL2004)	1004	EL2004		0.4	1630
5	Term 6 (EL7031)	1005	EL7031	8.0	8.0	1510
6	Term 7 (EL2808)	1006	EL2808		1.0	1400
7	Term 8 (EL3602)	1007	EL3602	12.0		1210
8	Term 9 (EL3602)	1008	EL3602	12.0		1020
9	Term 10 (EL3602)	1009	EL3602	12.0		830
10	Term 11 (EL3602)	1010	EL3602	12.0		640
11	Term 12 (EL3602)	1011	EL3602	12.0		450
12	Term 13 (EL3602)	1012	EL3602	12.0		260
13	Term 14 (EL3602)	1013	EL3602	12.0		70
14	Term 3 (EL6688)	1014	EL6688	22.0		-240 !

Fig. 152: Illegally exceeding the E-Bus current

From TwinCAT 2.11 and above, a warning message “E-Bus Power of Terminal...” is output in the logger window when such a configuration is activated:

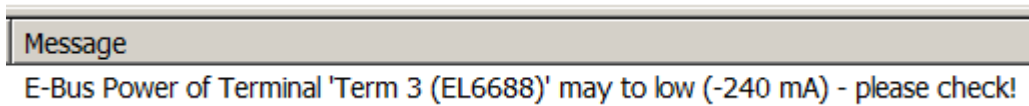


Fig. 153: Warning message for exceeding E-Bus current

#### NOTICE

##### **Caution! Malfunction possible!**

The same ground potential must be used for the E-Bus supply of all EtherCAT terminals in a terminal block!

## 6 Sample program EL2574

The *FB\_Pixel\_LED* function block provides simple control of LED strips and matrices connected to the EL2574. This sample also includes two conversion tools. One tool can be used to convert images (\*.png, \*.jpg, \*.bmp, ...) into an array, which can be displayed on the matrix. The other tool is used to convert fonts in order to display text in different fonts and sizes.

### Download

There are various ways to download the function block:

1. Function block only

This download contains the function block for integration in the PLC in PLCopen XML format.



<https://infosys.beckhoff.com/content/1033/el2574/Resources/19201701899.zip>

2. Conversion tools

This download only contains the two tools for converting images and fonts into displayable PLC code.



<https://infosys.beckhoff.com/content/1033/el2574/Resources/19201700235.zip>

3. Solution, function block and conversion tools

This download link contains a TwinCAT solution in \*.tzip format, in which the function block is used as an example, the function block for integration in your own PLC program in PLCopen XML format and the conversion tools for images and fonts.



<https://infosys.beckhoff.com/content/1033/el2574/Resources/19201703563.zip>

## 6.1 General Information

- The function block must be called in every cycle.
- It is possible to use different methods in a single cycle, as well as multiple calls to the same method in a cycle.
- The methods only apply to an internal buffer (no direct influence on the terminal with the connected hardware) (exception: DirectCommands). To display the changes, a rising edge must be applied to the bShow input of the function block.
- You must wait as long as bBusy is TRUE.

### Linking the function block

The function block provides variables that can be linked to the terminal via the "PLC" tab in the System Manager. The following adjustments must be made in preparation:

#### 1. Configuring the slots depending on the connected hardware

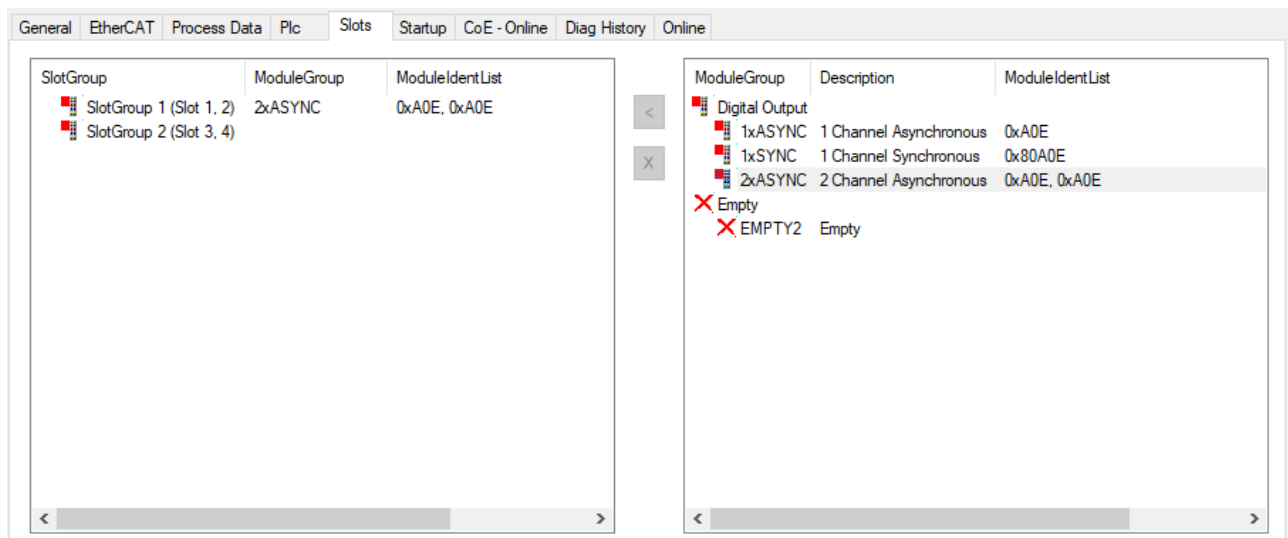


Fig. 154: 1. Configuring the slots depending on the connected hardware

#### 2. Activating all process data

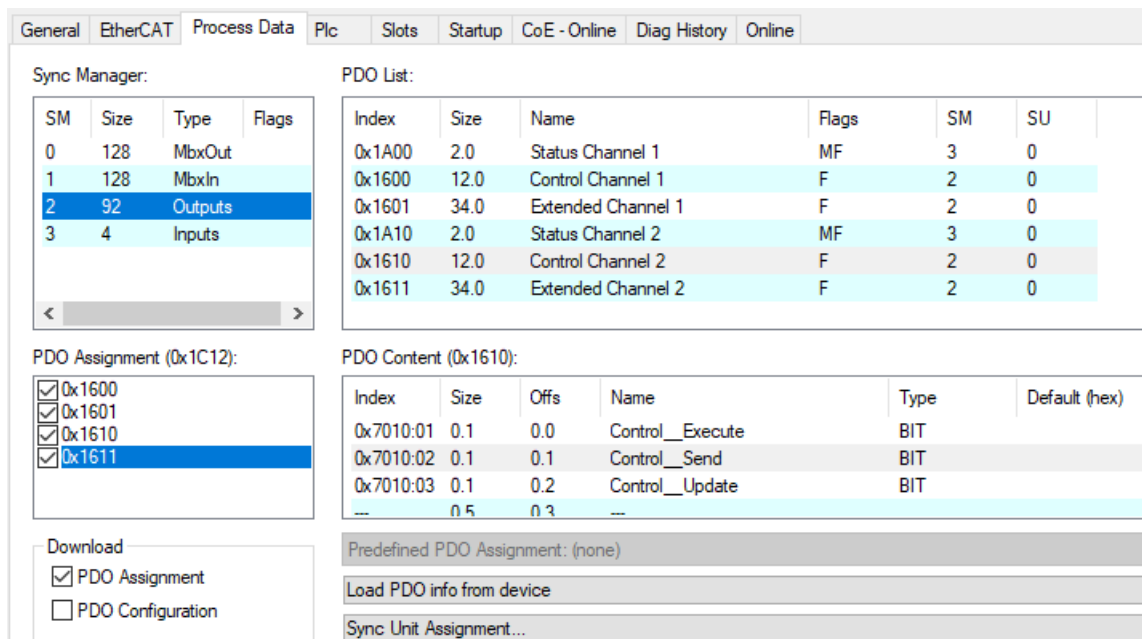


Fig. 155: 2. Activating all process data

## 3. Creating the "PLC" data type per channel and linking the function block

General EtherCAT Process Data **Plc** Slots Startup CoE - Online Diag History Online

☒ Create PLC Data Type

☒ Per Channel: 1

Data Type: MDP5001\_260\_D74B327E Copy

Link To PLC... MAIN.fbPixelLED.stEL2574[0] (EL2574\_ExampleCodeV)

Fig. 156: 3. Creating the "PLC" data type per channel and linking the function block

## 6.2 Function block

### FB\_Pixel\_LED

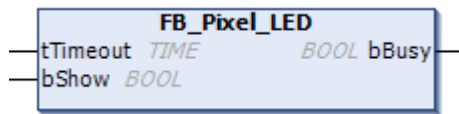


Fig. 157: FB\_Pixel\_LED

#### Inputs

Name	Type	Description
tTimeout	TIME	Timeout for the transmission process to the terminal
bShow	BOOL	Rising edge starts the transmission and display of data

#### Outputs

Name	Type	Description
bBusy	BOOL	Is TRUE if the FB transmits data to the terminal. Data cannot be changed during this time

#### Definition of constants

The following constants must be defined in the function block before the program is started:

```

1 FUNCTION_BLOCK FB_Pixel_LED
2   VAR CONSTANT
3     nWidth          : INT := 32;          (* Width of the entire matrix *)
4     nHeight         : INT := 16;         (* Height of the entire matrix *)
5     XTiles          : INT := 2;          (* Number of individual matrices in X direction (Width) *)
6     YTiles          : INT := 1;          (* Number of individual matrices in Y direction (Height) *)
7     bMatrixSerpentineLayout : INT(0..1) := 1; (* Layout type (see M_CursorToIndex) *)
8     bMatrixVertical  : INT(0..1) := 1;    (* Layout of LED lines is vertical (see M_CursorToIndex) *)
9     nNumberOfChannels : INT := 2;        (* Number of EL2574 Channels to use (only use identical matrices per channel) *)
10  END_VAR

```

Fig. 158: Definition of constants

Name	Type	Description
nWidth	INT	Width of the entire matrix
nHeight	INT	Height of the entire matrix
XTiles	INT	Number of individual matrices in X direction (width)
YTiles	INT	Number of individual matrices in Y-direction (height)
bMatrixSerpentineLayout	INT(0..1)	Layout type (see M_CursorToIndex)
bMatrixVertical	INT(0..1)	Layout of the LED lines is vertical (see M_CursorToIndex)
nNumberOfChannels	INT	Number of EL2574 channels used

#### Methods

##### • M\_ClearAll

This method can be used to set all elements of the array to the value 0.

##### • M\_Clear

This method can be used to set all elements of the array in a given rectangle to the value 0.

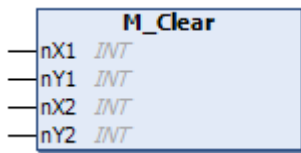


Fig. 159: M\_Clear

### Inputs

Name	Type	Description
nX1	INT	Start position of the pixels to be deleted in X direction
nY1	INT	Start position of the pixels to be deleted in Y direction
nX2	INT	End position of the pixels to be deleted in X direction
nY2	INT	End position of the pixels to be deleted in Y direction

### • M\_DrawImage

This method can be used to display an image statically on a matrix.

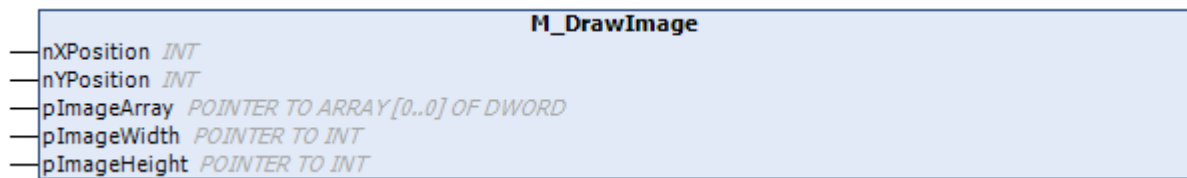


Fig. 160: M\_DrawImage

### Inputs

Name	Type	Description
nXPosition	INT	Start position for displaying the image in X direction
nYPosition	INT	Start position for displaying the image in Y direction
pImageArray	POINTER TO ARRAY [0..0] OF DWORD	Pointer to the array containing the image data
pImageWidth	POINTER TO INT	Pointer to the width of the image
pImageHeight	POINTER TO INT	Pointer to the height of the image

The supplied HTML page ImageConverter.html (see chapter "ImageConverter.html") can be used to display images. The image uploaded there should not exceed the resolution of the pixel matrix; smaller images can be used without restriction. After uploading an image file, the web page will download a text file. This file contains PLC variable definitions that must be included in the PLC project. This can be done, for example, by copying the content into a GVL. The "M\_DrawImage" method can be used to display the image on the matrix by assigning the addresses of the copied variables to the input pointers of the method.

### • M\_DrawText

This method can be used to display a string statically on a matrix.

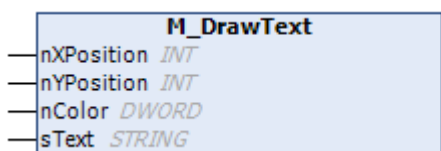


Fig. 161: M\_DrawText

### Inputs



Name	Type	Description
nXPosition	INT	Start position for displaying the text in X direction
nYPosition	INT	Start position for displaying the text in Y direction
nColor	DWORD	Text color (0xWWBBGGRR)
sText	STRING	Display text

A basic font is stored in the declaration section of the constants in the function block. To use a different font and / or font size, the supplied HTML page FontConverter.html (see chapter "FontConverter.html") can be used.

```

31  VAR CONSTANT
32  (----- Font Definition -----)
33  StartChar          : BYTE := 16#20;
34  EndChar            : BYTE := 16#7A;
35  NewLineOffset      : USINT := 6;
36  MaxBaseLine        : INT := 4;
37  Bitmaps            : ARRAY[0..1534] OF BYTE := [ 16#00, 16#00, 16#00, 16#E8, 16#00, 16#B4, 16#00, 16#00, 16#57, 16#D5,
38  glyphs              : ARRAY[0..90] OF ARRAY[0..5] OF INT := [[0, 3, 6, 4, 0, -4],[3, 1, 5, 2, 0, -4],[5, 3, 5, 4, 0, -
39  END_VAR

```

Fig. 162: PLC variable definitions

### • M\_ScrollText

This method can be used to display a string from right to left on a matrix.

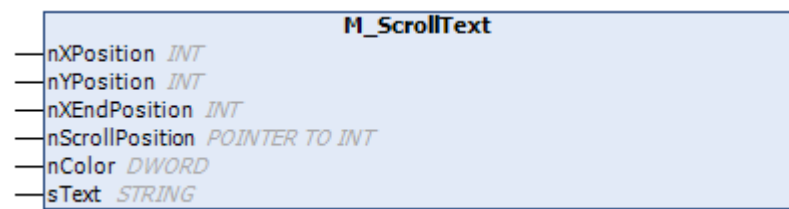


Fig. 163: M\_ScrollText

### Inputs

Name	Type	Description
nXPosition	INT	Start position for displaying the text in X direction
nYPosition	INT	Start position for displaying the text in Y direction
nXEndPosition	INT	End position to display the text in X direction <i>Note: If nXEndPosition = -1, the text is output from the start position across the entire width of the matrix</i>
nScrollPosition	POINTER TO INT	Pointer to auxiliary variable to track the scroll position (an auxiliary variable is required for each text)
nColor	DWORD	Text color (0xWWBBGGRR)
sText	STRING	Display text

A basic font is stored in the declaration section of the constants in the function block. To use a different font and / or font size, the supplied HTML page FontConverter.html (see section 6.3.2) can be used.

### • M\_RotateAll

This method rotates the entire content of the matrix in a given direction.

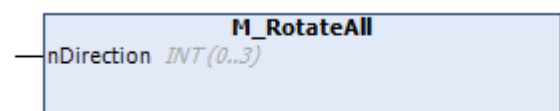


Fig. 164: M\_RotateAll

### Inputs

Name	Type	Description
nDirection	INT(0...3)	Direction preset 0: Left, 1: Right, 2: Up, 3: Down

### • M\_Rotate

This method rotates the contents of the matrix in a given rectangle in a given direction.

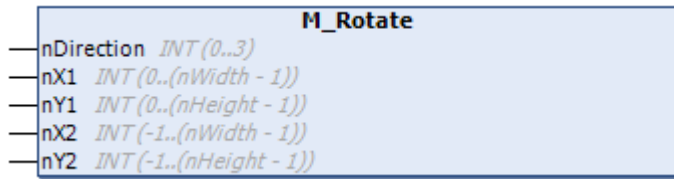


Fig. 165: M\_Rotate

#### Inputs

Name	Type	Description
nDirection	INT(0...3)	Direction preset 0: Left, 1: Right, 2: Up, 3: Down
nX1	INT	Start position of the pixels to be rotated in X direction
nY1	INT	Start position of the pixels to be rotated in Y direction
nX2	INT	End position of the pixels to be rotated in X direction <i>Note: If nX2 = -1, the entire matrix width is rotated from nX1</i>
nY2	INT	End position of the pixels to be rotated in Y direction <i>Note: If nY2 = -1, the entire matrix height is rotated from nY1</i>

### • M\_SetPixel

This method allows individual pixels to be switched on in a specific color.



Fig. 166: M\_SetPixel

#### Inputs

Name	Type	Description
nXPosition	INT	Pixel position in X direction
nYPosition	INT	Pixel position in Y direction
nColor	DWORD	Color of the pixel (0xWWBBGGRR)

### • M\_GetPixel

This method reads the color of a pixel at a specified position from the buffer.

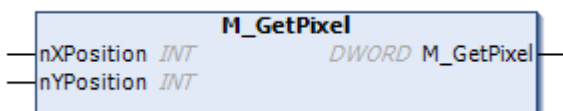


Fig. 167: M\_GetPixel

#### Inputs

Name	Type	Description
nXPosition	INT	Pixel position in X direction
nYPosition	INT	Pixel position in Y direction
nColor	DWORD	Color of the pixel (0xWWBBGGRR)

#### Outputs

Name	Type	Description
M_GetPixel	DWORD	Color of the pixel (0xWWBBGGRR)

## 6.3 Conversion tools

### ImageConverter.html

The ImageConverter is used to convert graphics into PLC code. The \*.html must be opened in the browser.



Fig. 168: Image-Pixel-Converter

In the tool there is a link "Pixel Image Editor" to the free editor "Piskel" to create pixel graphics and animations. The graphics created here or existing graphics can then be chosen using the "Choose Files" button. Typical image formats are supported. Make sure that the graphic has the same resolution as the pixel matrix, otherwise the complete graphic will not be displayed. The graphic should not exceed the resolution of the pixel matrix; smaller images can be used without restriction.

After uploading the selected file, a text file (\*.txt) is automatically downloaded. The file contains PLC variable definitions (width and height of the graphic, value of the individual pixels) that must be included in the PLC project. This can be done, for example, by copying the content into a GVL. The "M\_DrawImage" method can then be used to display the graphic on the matrix by assigning the addresses of the copied variables to the input pointers of the method.

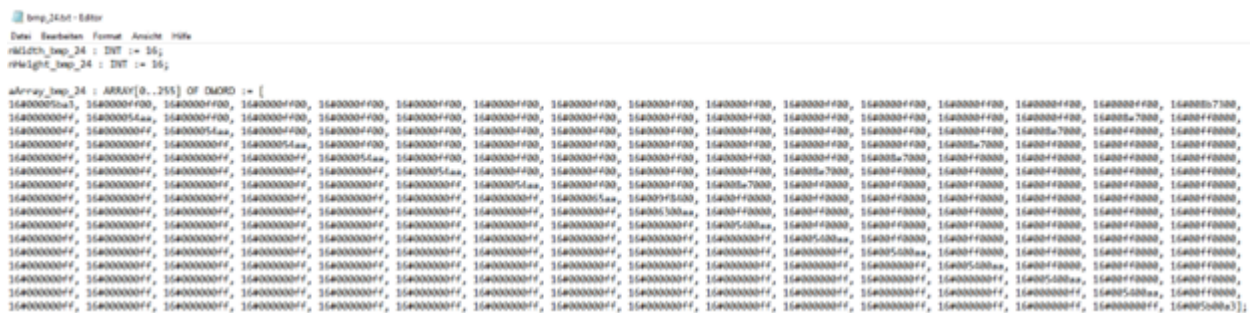


Fig. 169: Generated text file Image-Pixel-Converter

### FontConverter.html

The FontConverter is used to convert fonts into PLC code. The \*.html must be opened in the browser.

## GFXfont to PixelFont

[Example Fonts](#)

[Font Editor](#)

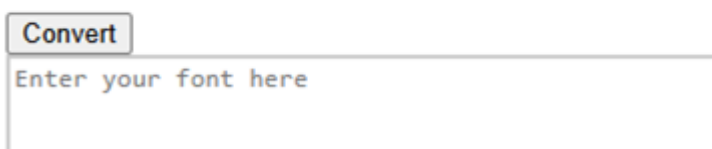


Fig. 170: Font-Pixel-Converter

In the tool there is a link "Example Fonts" to a Github directory that contains fonts in the \*.h file format in various font sizes and designs (bold, italic). The text from the \*.h file must then be copied and pasted into the "Enter your font here" text field on the \*.html page. A text file (\*.txt) is downloaded via the "Convert" button. This file contains PLC variable definitions that must be included in FB\_Pixel\_LED under VAR CONSTANT for the "Font Definiton".

```

31  VAR CONSTANT
32  (----- Font Definition -----)
33  StartChar          : BYTE := 16#20;
34  EndChar            : BYTE := 16#7A;
35  NewLineOffset      : USINT := 6;
36  MaxBaseline        : INT := 4;
37  Bitmaps            : ARRAY[0..1534] OF BYTE := [ 16#00, 16#00, 16#00, 16#EB, 16#00, 16#B4, 16#00, 16#00, 16#57, 16#D5,
38  glyphs              : ARRAY[0..90] OF ARRAY[0..5] OF INT := [[0, 3, 6, 4, 0, -4],[3, 1, 5, 2, 0, -4],[5, 3, 5, 4, 0, -
39  END_VAR

```

Fig. 171: PLC variable definitions

The "Font Editor" link opens a free online tool for creating your own fonts. The output here is also a text in \*.h format. The same procedure must be followed for the conversion.

FreeFont.txt - Editor

Datei Bearbeiten Format Ansicht Hilfe

```

StartChar          : BYTE := 16#20;
EndChar            : BYTE := 16#7E;
NewLineOffset      : USINT := 2;
MaxBaseline        : INT := 16;
Bitmaps            : ARRAY[0..1978] OF BYTE := [16#1C, 16#F3, 16#CE, 16#38, 16#E7, 16#1C, 16#61, 16#86, 16#00, 16#63, 16#8C,
16#1C, 16#F3, 16#CE, 16#38, 16#E7, 16#1C, 16#61, 16#86, 16#00, 16#63, 16#8C, 16#00, 16#E7, 16#E7, 16#E6, 16#C6, 16#C6, 16#C4,
16#03, 16#30, 16#19, 16#81, 16#DC, 16#0C, 16#E0, 16#66, 16#1F, 16#FC, 16#FF, 16#E1, 16#98, 16#0C, 16#C0, 16#EE, 16#06, 16#70,
16#CF, 16#FE, 16#1D, 16#C0, 16#CC, 16#06, 16#60, 16#77, 16#03, 16#30, 16#00, 16#01, 16#00, 16#70, 16#0C, 16#07, 16#F1, 16#FE,
16#CC, 16#11, 16#80, 16#3F, 16#03, 16#F0, 16#0F, 16#20, 16#6E, 16#0D, 16#C3, 16#3F, 16#E7, 16#F8, 16#1C, 16#03, 16#00, 16#60,
16#00, 16#0E, 16#03, 16#E0, 16#C4, 16#10, 16#82, 16#30, 16#7C, 16#07, 16#78, 16#7C, 16#7F, 16#19, 16#F0, 16#62, 16#08, 16#41,
16#3E, 16#03, 16#80, 16#07, 16#C1, 16#F8, 16#62, 16#0C, 16#01, 16#80, 16#38, 16#0F, 16#03, 16#F7, 16#6F, 16#D8, 16#F3, 16#1E,
16#E7, 16#F8, 16#FF, 16#6D, 16#20, 16#06, 16#1C, 16#70, 16#C3, 16#06, 16#18, 16#30, 16#C1, 16#83, 16#06, 16#0C, 16#18, 16#30,
16#60, 16#C1, 16#00, 16#0C, 16#18, 16#38, 16#30, 16#60, 16#C1, 16#83, 16#06, 16#0C, 16#30, 16#61, 16#C3, 16#0E, 16#38, 16#61,
16#00, 16#06, 16#00, 16#C0, 16#18, 16#3F, 16#7F, 16#FE, 16#FF, 16#07, 16#81, 16#F8, 16#77, 16#0C, 16#60, 16#03, 16#00, 16#70,
16#00, 16#60, 16#06, 16#0F, 16#FF, 16#FF, 16#F0, 16#E0, 16#0C, 16#00, 16#C0, 16#0C, 16#01, 16#C0, 16#18, 16#00, 16#1C, 16#E3,
16#63, 16#08, 16#00, 16#7F, 16#FF, 16#FF, 16#C0, 16#7F, 16#00, 16#00, 16#08, 16#00, 16#70, 16#01, 16#80, 16#0E, 16#00, 16#70,

```

Fig. 172: Generated text file Font-Pixel-Converter

## 7 Appendix

### 7.1 Firmware Update EL/ES/EM/ELM/EP/EPP/ERPxxxx

This section describes the device update for Beckhoff EtherCAT slaves from the EL/ES, ELM, EM, EK, EP, EPP and ERP series. A firmware update should only be carried out after consultation with Beckhoff support.

#### NOTICE

##### Only use TwinCAT 3 software!

A firmware update of Beckhoff IO devices must only be performed with a TwinCAT 3 installation. It is recommended to build as up-to-date as possible, available for free download on the [Beckhoff website](#).

To update the firmware, TwinCAT can be operated in the so-called FreeRun mode, a paid license is not required.

The device to be updated can usually remain in the installation location, but TwinCAT has to be operated in the FreeRun. Please make sure that EtherCAT communication is trouble-free (no LostFrames etc.).

Other EtherCAT master software, such as the EtherCAT Configurator, should not be used, as they may not support the complexities of updating firmware, EEPROM and other device components.

#### Storage locations

An EtherCAT slave stores operating data in up to three locations:

- Each EtherCAT slave has a device description, consisting of identity (name, product code), timing specifications, communication settings, etc.  
This device description (ESI; EtherCAT Slave Information) can be downloaded from the Beckhoff website in the download area as a [zip file](#) and used in EtherCAT masters for offline configuration, e.g. in TwinCAT.  
Above all, each EtherCAT slave carries its device description (ESI) electronically readable in a local memory chip, the so-called **ESI EEPROM**. When the slave is switched on, this description is loaded locally in the slave and informs it of its communication configuration; on the other hand, the EtherCAT master can identify the slave in this way and, among other things, set up the EtherCAT communication accordingly.

#### NOTICE

##### Application-specific writing of the ESI-EEPROM

The ESI is developed by the device manufacturer according to ETG standard and released for the corresponding product.

- Meaning for the ESI file: Modification on the application side (i.e. by the user) is not permitted.
- Meaning for the ESI EEPROM: Even if a writeability is technically given, the ESI parts in the EEPROM and possibly still existing free memory areas must not be changed beyond the normal update process. Especially for cyclic memory processes (operating hours counter etc.), dedicated memory products such as EL6080 or IPC's own NOVDRAM must be used.

- Depending on functionality and performance EtherCAT slaves have one or several local controllers for processing I/O data. The corresponding program is the so-called **firmware** in \*.efw format.
- In some EtherCAT slaves the EtherCAT communication may also be integrated in these controllers. In this case the controller is usually a so-called **FPGA** chip with \*.rbf firmware.

Customers can access the data via the EtherCAT fieldbus and its communication mechanisms. Acyclic mailbox communication or register access to the ESC is used for updating or reading of these data.

The TwinCAT System Manager offers mechanisms for programming all three parts with new data, if the slave is set up for this purpose. Generally the slave does not check whether the new data are suitable, i.e. it may no longer be able to operate if the data are unsuitable.

### Simplified update by bundle firmware

The update using so-called **bundle firmware** is more convenient: in this case the controller firmware and the ESI description are combined in a \*.efw file; during the update both the firmware and the ESI are changed in the terminal. For this to happen it is necessary

- for the firmware to be in a packed format: recognizable by the file name, which also contains the revision number, e.g. ELxxx-xxx\_REV0016\_SW01.efw
- for password=1 to be entered in the download dialog. If password=0 (default setting) only the firmware update is carried out, without an ESI update.
- for the device to support this function. The function usually cannot be retrofitted; it is a component of many new developments from year of manufacture 2016.

Following the update, its success should be verified

- ESI/Revision: e.g. by means of an online scan in TwinCAT ConfigMode/FreeRun – this is a convenient way to determine the revision
- Firmware: e.g. by looking in the online CoE of the device

#### NOTICE

##### Risk of damage to the device!

- ✓ Note the following when downloading new device files

- Firmware downloads to an EtherCAT device must not be interrupted
  - Flawless EtherCAT communication must be ensured. CRC errors or LostFrames must be avoided.
  - The power supply must adequately dimensioned. The signal level must meet the specification.
- ⇒ In the event of malfunctions during the update process the EtherCAT device may become unusable and require re-commissioning by the manufacturer.

### 7.1.1 Device description ESI file/XML

#### NOTICE

##### Attention regarding update of the ESI description/EEPROM

Some slaves have stored calibration and configuration data from the production in the EEPROM. These are irretrievably overwritten during an update.

The ESI device description is stored locally on the slave and loaded on start-up. Each device description has a unique identifier consisting of slave name (9 characters/digits) and a revision number (4 digits). Each slave configured in the System Manager shows its identifier in the EtherCAT tab:

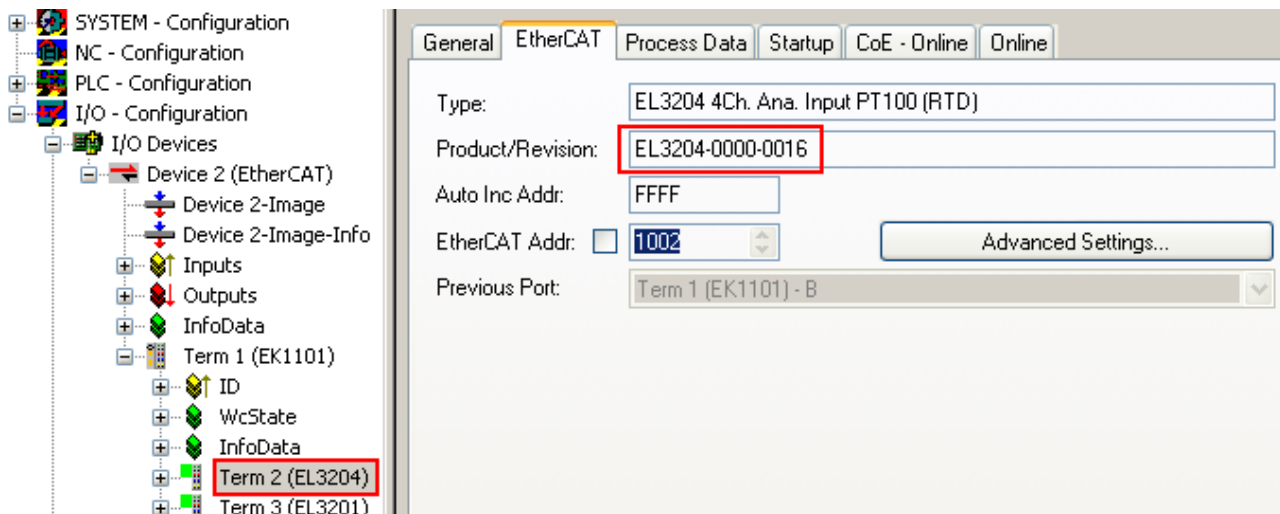


Fig. 173: Device identifier consisting of name EL3204-0000 and revision -0016

The configured identifier must be compatible with the actual device description used as hardware, i.e. the description which the slave has loaded on start-up (in this case EL3204). Normally the configured revision must be the same or lower than that actually present in the terminal network.

For further information on this, please refer to the [EtherCAT system documentation](#).

### **i Update of XML/ESI description**

The device revision is closely linked to the firmware and hardware used. Incompatible combinations lead to malfunctions or even final shutdown of the device. Corresponding updates should only be carried out in consultation with Beckhoff support.

### **Display of ESI slave identifier**

The simplest way to ascertain compliance of configured and actual device description is to scan the EtherCAT boxes in TwinCAT mode Config/FreeRun:

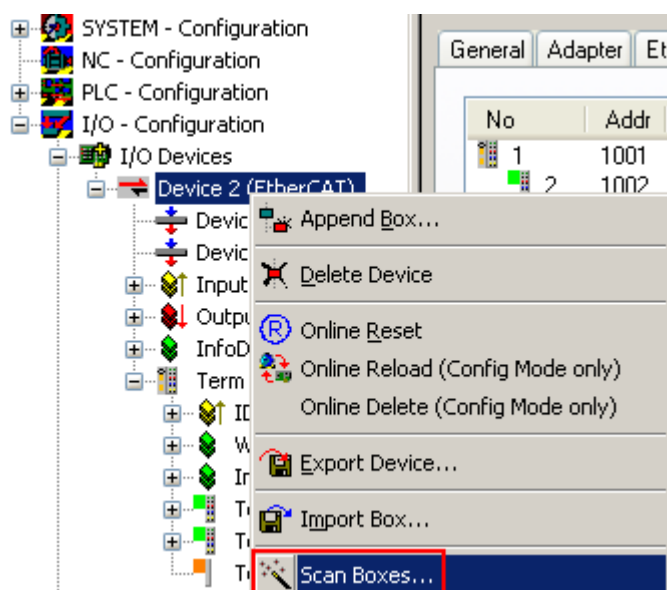


Fig. 174: Scan the subordinate field by right-clicking on the EtherCAT device

If the found field matches the configured field, the display shows



Fig. 175: Configuration is identical

otherwise a change dialog appears for entering the actual data in the configuration.



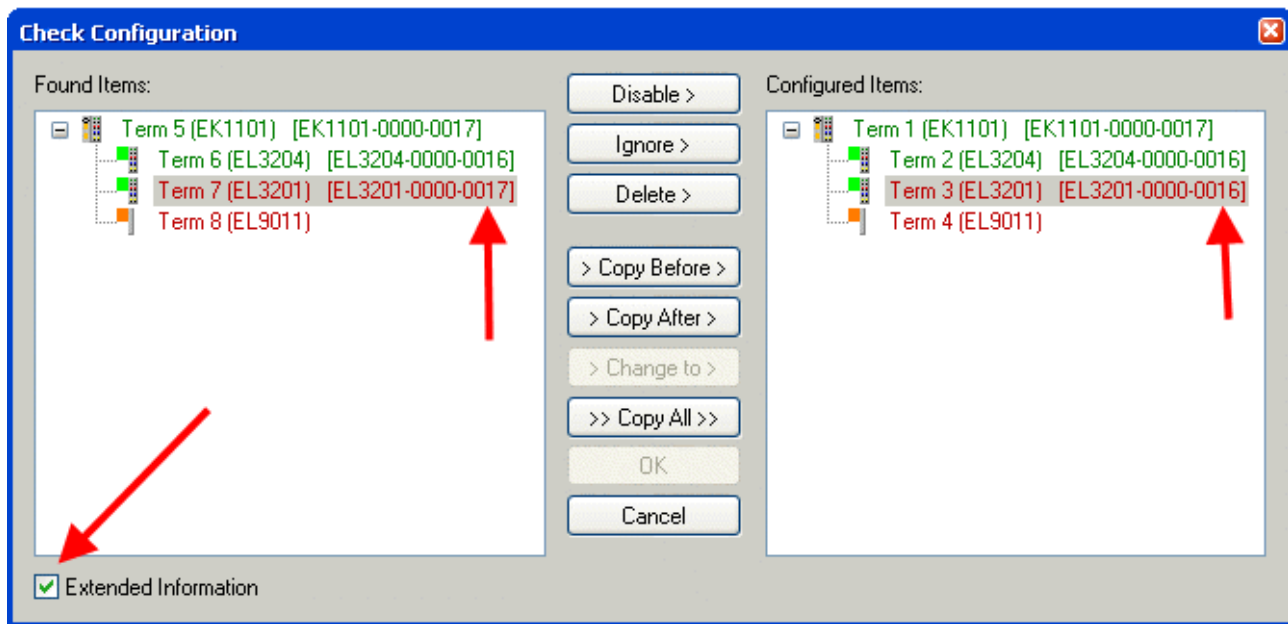


Fig. 176: Change dialog

In this example in Fig. *Change dialog*, an EL3201-0000-**0017** was found, while an EL3201-0000-**0016** was configured. In this case the configuration can be adapted with the *Copy Before* button. The *Extended Information* checkbox must be set in order to display the revision.

### Changing the ESI slave identifier

The ESI/EEPROM identifier can be updated as follows under TwinCAT:

- Trouble-free EtherCAT communication must be established with the slave.
- The state of the slave is irrelevant.
- Right-clicking on the slave in the online display opens the *EEPROM Update* dialog, Fig. *EEPROM Update*

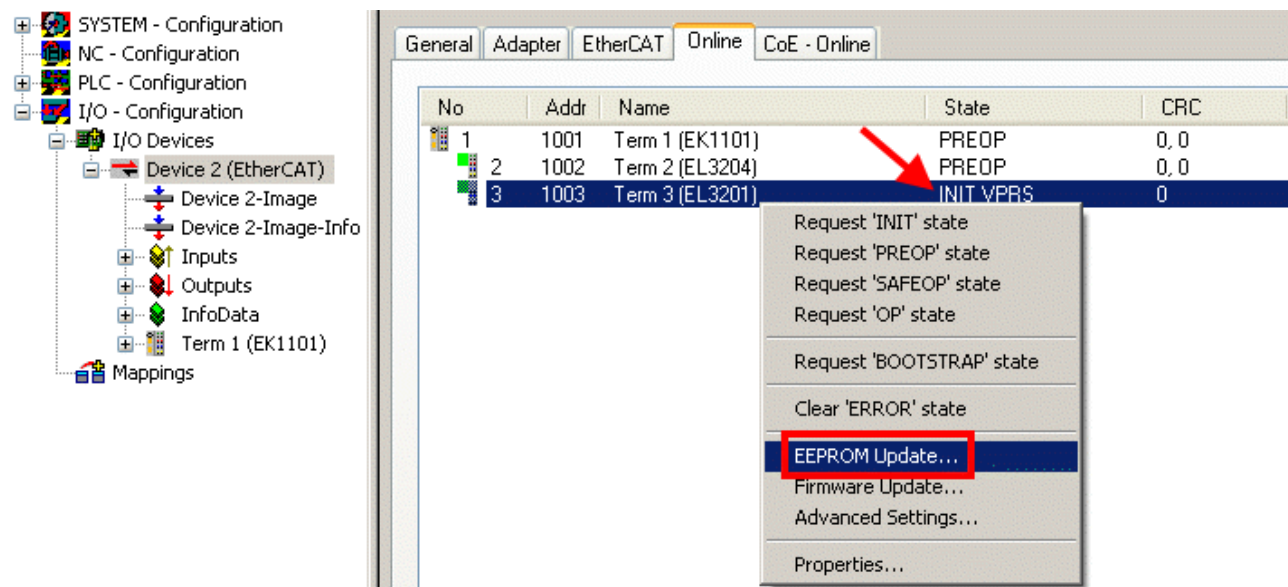


Fig. 177: EEPROM Update

The new ESI description is selected in the following dialog, see Fig. *Selecting the new ESI*. The checkbox *Show Hidden Devices* also displays older, normally hidden versions of a slave.



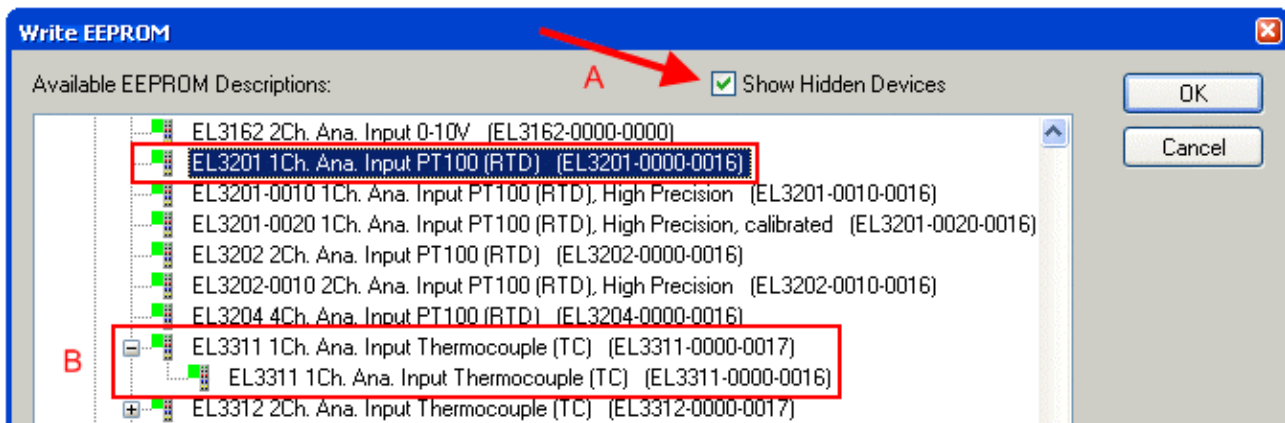


Fig. 178: Selecting the new ESI

A progress bar in the System Manager shows the progress. Data are first written, then verified.



### The change only takes effect after a restart.

Most EtherCAT devices read a modified ESI description immediately or after startup from the INIT. Some communication settings such as distributed clocks are only read during power-on. The EtherCAT slave therefore has to be switched off briefly in order for the change to take effect.

## 7.1.2 Firmware explanation

### Determining the firmware version

#### Determining the version via the TwinCAT System Manager

The TwinCAT System Manager shows the version of the controller firmware if the master can access the slave online. Click on the E-Bus Terminal whose controller firmware you want to check (in the example terminal 2 (EL3204)) and select the tab *CoE Online* (CAN over EtherCAT).



### CoE Online and Offline CoE

Two CoE directories are available:

- **online:** This is offered in the EtherCAT slave by the controller, if the EtherCAT slave supports this. This CoE directory can only be displayed if a slave is connected and operational.
- **offline:** The EtherCAT Slave Information ESI/XML may contain the default content of the CoE. This CoE directory can only be displayed if it is included in the ESI (e.g. "Beckhoff EL5xxx.xml").

The Advanced button must be used for switching between the two views.

In Fig. *Display of EL3204 firmware version* the firmware version of the selected EL3204 is shown as 03 in CoE entry 0x100A.

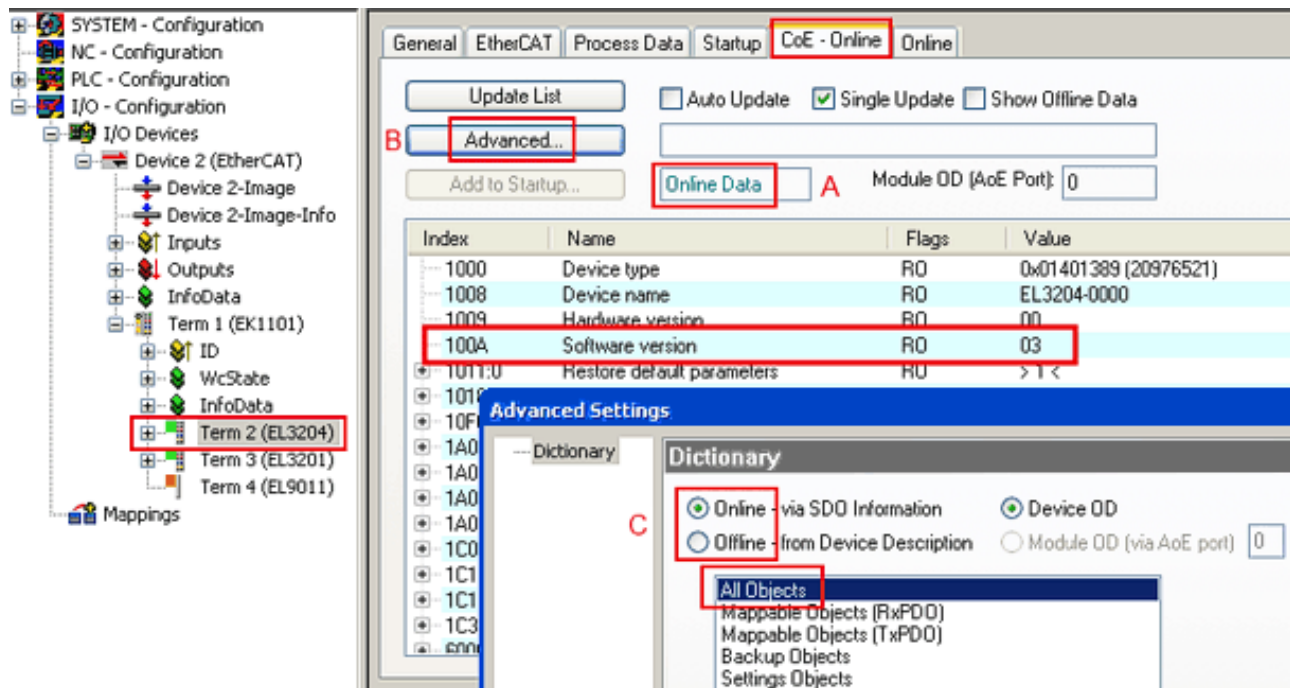


Fig. 179: Display of EL3204 firmware version

In (A) TwinCAT 2.11 shows that the Online CoE directory is currently displayed. If this is not the case, the Online directory can be loaded via the *Online* option in Advanced Settings (B) and double-clicking on *All Objects*.

### 7.1.3 Updating controller firmware \*.efw



#### CoE directory

The Online CoE directory is managed by the controller and stored in a dedicated EEPROM, which is generally not changed during a firmware update.

Switch to the *Online* tab to update the controller firmware of a slave, see Fig. *Firmware Update*.

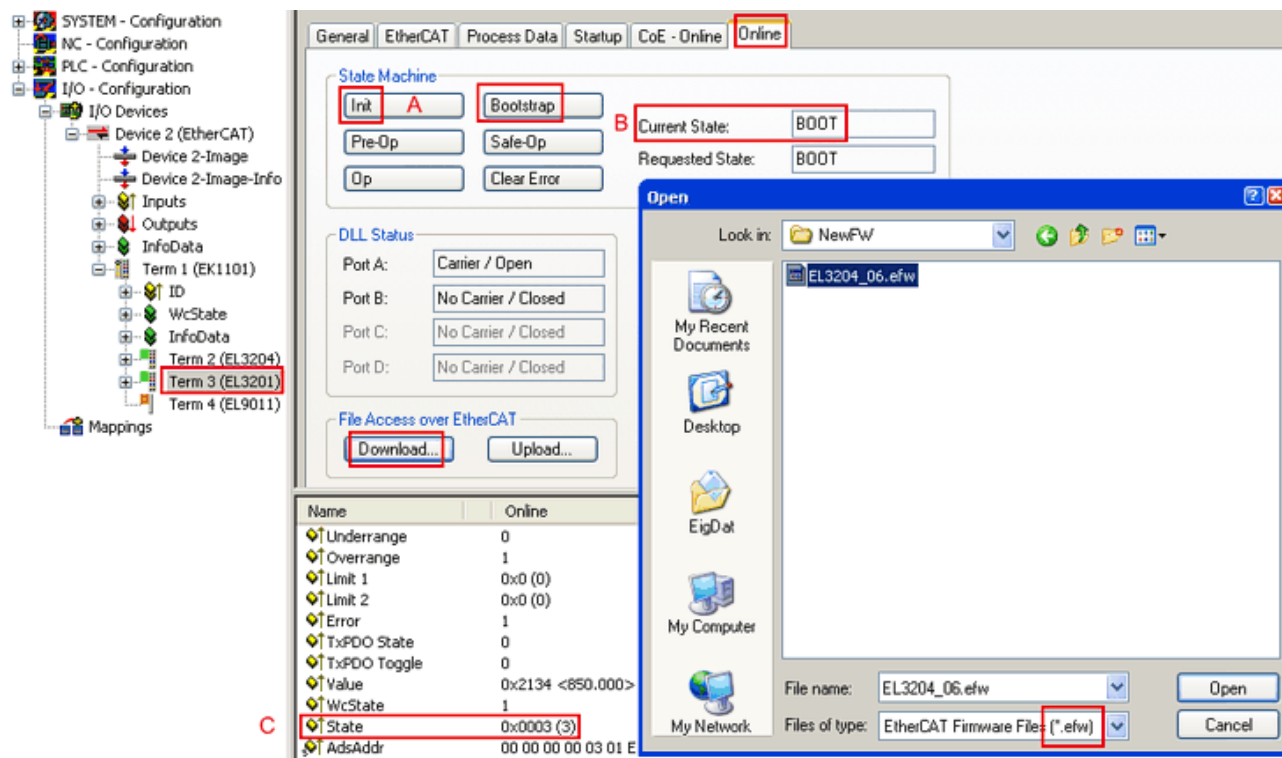
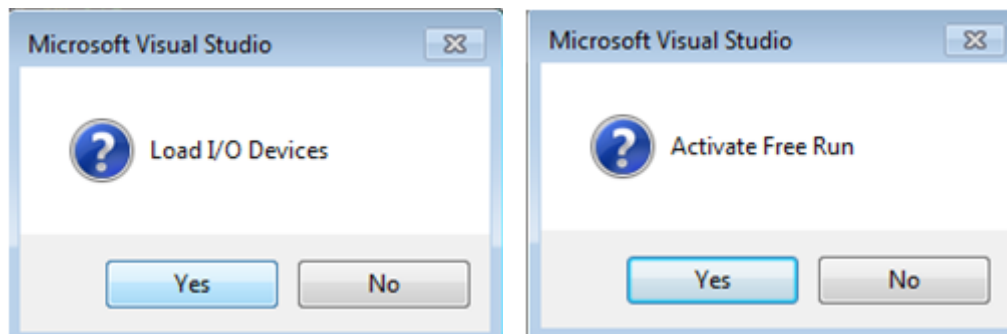


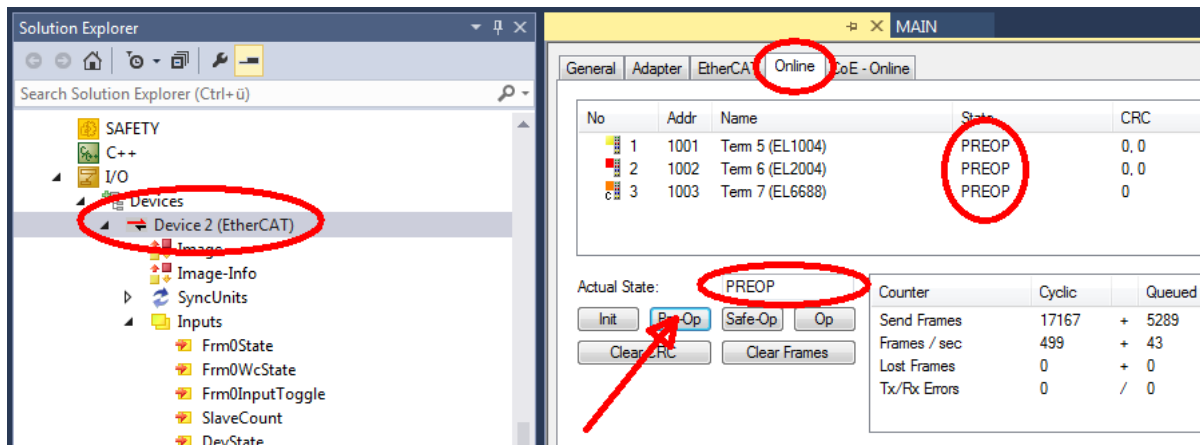
Fig. 180: Firmware Update

Proceed as follows, unless instructed otherwise by Beckhoff support. Valid for TwinCAT 2 and 3 as EtherCAT master.

- Switch TwinCAT system to ConfigMode/FreeRun with cycle time  $\geq 1$  ms (default in ConfigMode is 4 ms). A FW-Update during real time operation is not recommended.

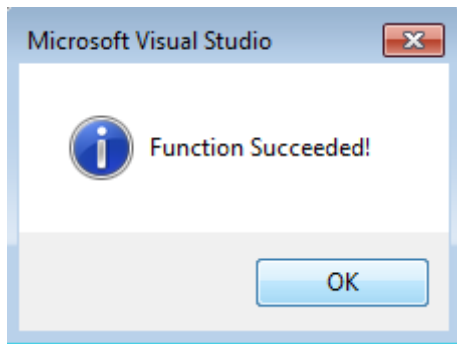


- Switch EtherCAT Master to PreOP



- Switch slave to INIT (A)
- Switch slave to BOOTSTRAP

- Check the current status (B, C)
- Download the new \*efw file (wait until it ends). A password will not be necessary usually.



- After the download switch to INIT, then PreOP
- Switch off the slave briefly (don't pull under voltage!)
- Check within CoE 0x100A, if the FW status was correctly overtaken.

### 7.1.4 FPGA firmware \*.rbf

If an FPGA chip deals with the EtherCAT communication an update may be accomplished via an \*.rbf file.

- Controller firmware for processing I/O signals
- FPGA firmware for EtherCAT communication (only for terminals with FPGA)

The firmware version number included in the terminal serial number contains both firmware components. If one of these firmware components is modified this version number is updated.

#### Determining the version via the TwinCAT System Manager

The TwinCAT System Manager indicates the FPGA firmware version. Click on the Ethernet card of your EtherCAT strand (Device 2 in the example) and select the *Online* tab.

The *Reg:0002* column indicates the firmware version of the individual EtherCAT devices in hexadecimal and decimal representation.

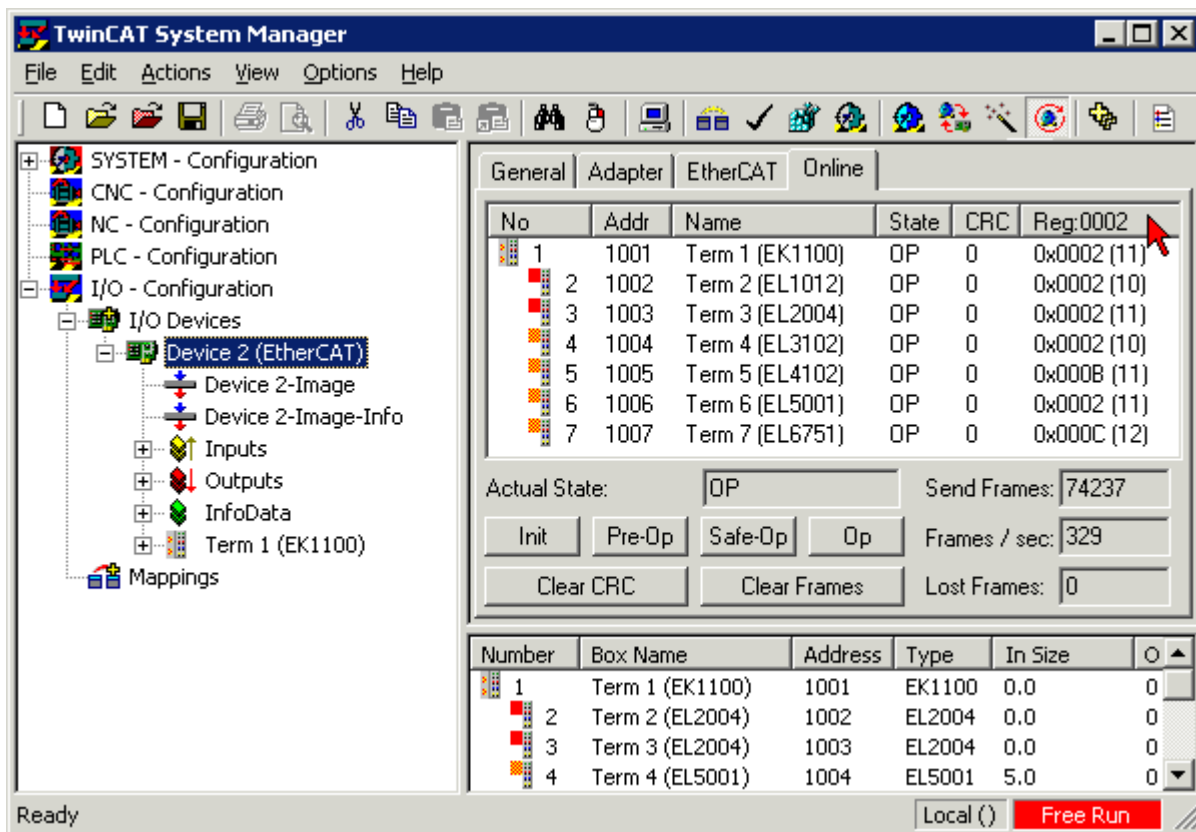
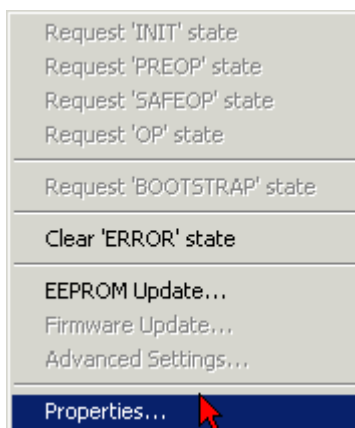


Fig. 181: FPGA firmware version definition

If the column *Reg:0002* is not displayed, right-click the table header and select *Properties* in the context menu.

Fig. 182: Context menu *Properties*

The *Advanced Settings* dialog appears where the columns to be displayed can be selected. Under *Diagnosis/Online View* select the '*0002 ETxxxx Build*' check box in order to activate the FPGA firmware version display.

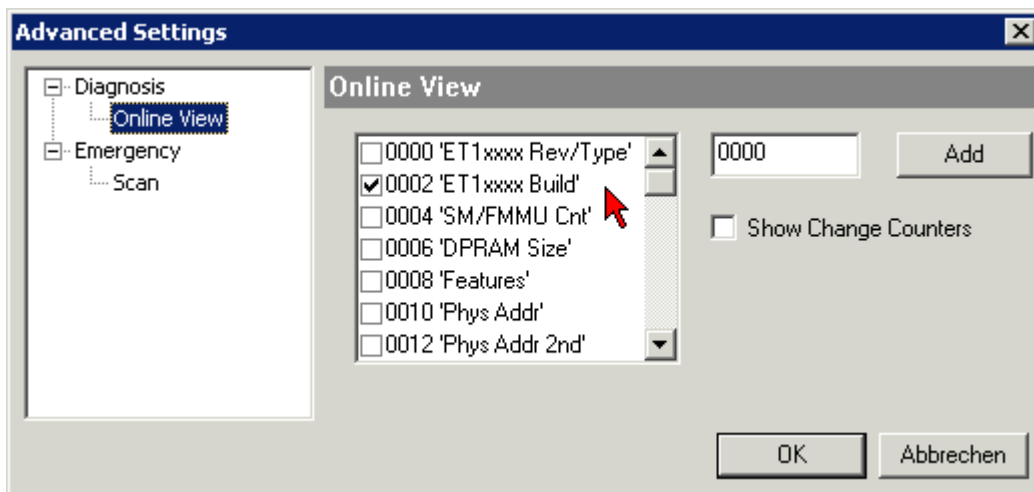


Fig. 183: Dialog *Advanced Settings*

## Update

For updating the FPGA firmware

- of an EtherCAT coupler the coupler must have FPGA firmware version 11 or higher;
- of an E-Bus Terminal the terminal must have FPGA firmware version 10 or higher.

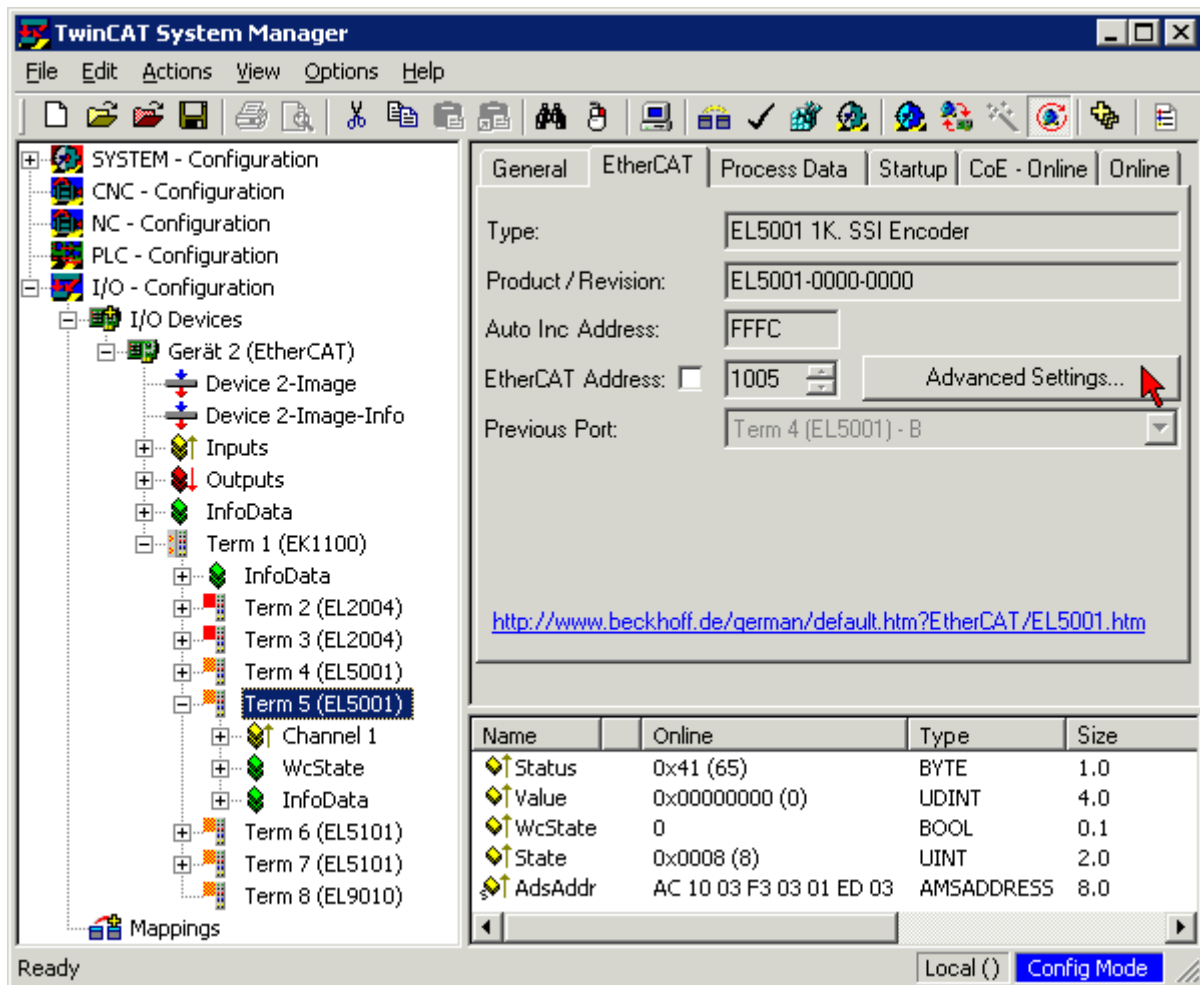
Older firmware versions can only be updated by the manufacturer!

## Updating an EtherCAT device

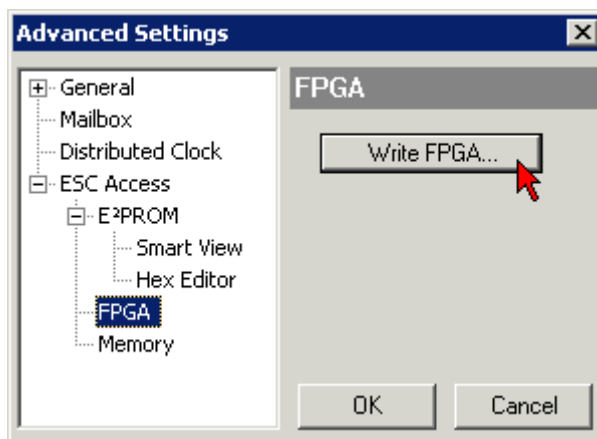
The following sequence order have to be met if no other specifications are given (e.g. by the Beckhoff support):

- Switch TwinCAT system to ConfigMode/FreeRun with cycle time  $\geq 1$  ms (default in ConfigMode is 4 ms). A FW-Update during real time operation is not recommended.

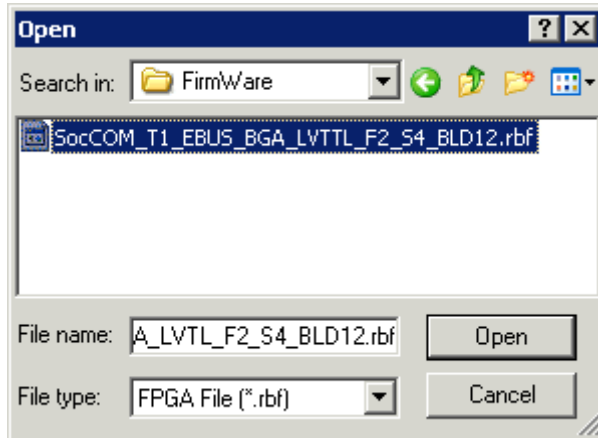
- In the TwinCAT System Manager select the terminal for which the FPGA firmware is to be updated (in the example: Terminal 5: EL5001) and click the *Advanced Settings* button in the *EtherCAT* tab:



- The *Advanced Settings* dialog appears. Under *ESC Access/E<sup>2</sup>PROM/FPGA* click on *Write FPGA* button:



- Select the file (\*.rbf) with the new FPGA firmware, and transfer it to the EtherCAT device:



- Wait until download ends
- Switch slave current less for a short time (don't pull under voltage!). In order to activate the new FPGA firmware a restart (switching the power supply off and on again) of the EtherCAT device is required.
- Check the new FPGA status

### NOTICE

#### Risk of damage to the device!

A download of firmware to an EtherCAT device must not be interrupted in any case! If you interrupt this process by switching off power supply or disconnecting the Ethernet link, the EtherCAT device can only be recommissioned by the manufacturer!

## 7.1.5 Simultaneous updating of several EtherCAT devices

The firmware and ESI descriptions of several devices can be updated simultaneously, provided the devices have the same firmware file/ESI.

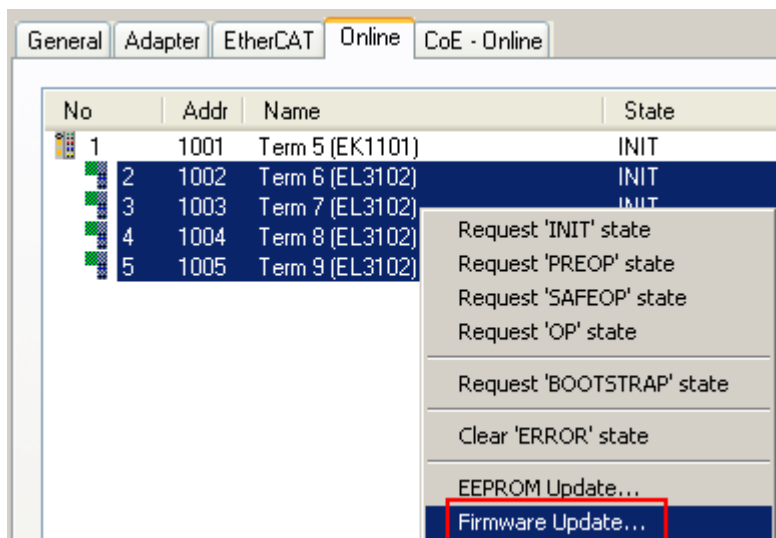


Fig. 184: Multiple selection and firmware update

Select the required slaves and carry out the firmware update in BOOTSTRAP mode as described above.



## 7.2 Firmware compatibility

Beckhoff EtherCAT devices are delivered with the latest available firmware version. Compatibility of firmware and hardware is mandatory; not every combination ensures compatibility. The overview below shows the hardware versions on which a firmware can be operated.

### Note

- It is recommended to use the newest possible firmware for the respective hardware
- Beckhoff is not under any obligation to provide customers with free firmware updates for delivered products.

### NOTICE

#### Risk of damage to the device!

Pay attention to the instructions for firmware updates on the [separate page \[p. 165\]](#).

If a device is placed in BOOTSTRAP mode for a firmware update, it does not check when downloading whether the new firmware is suitable.

This can result in damage to the device! Therefore, always make sure that the firmware is suitable for the hardware version!

#### EL2574

Hardware (HW)	Firmware (FW)	Revision no.	Release date
01*	01*	EL2574-0000-0016	2023/07

\*) This is the current compatible firmware/hardware version at the time of the preparing this documentation. Check on the Beckhoff web page whether more up-to-date [documentation](#) is available.

## 7.3 Restoring the delivery state

To restore the delivery state (factory settings) of CoE objects for EtherCAT devices ("slaves"), the CoE object *Restore default parameters*, SubIndex 001 can be used via EtherCAT master (e.g. TwinCAT) (see Fig. *Selecting the Restore default parameters PDO*).

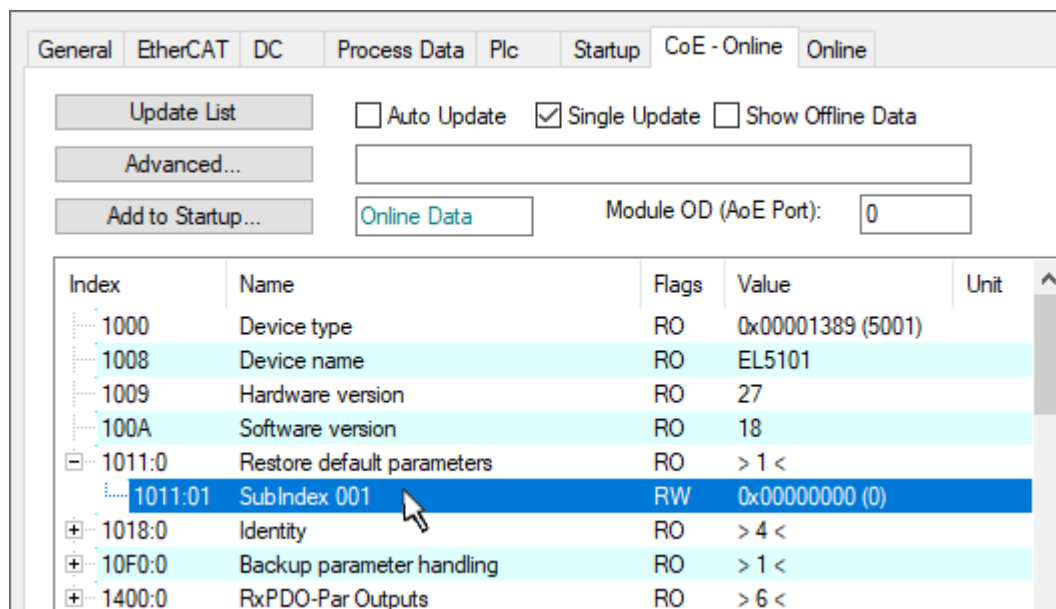


Fig. 185: Selecting the *Restore default parameters* PDO

The image shows a 'Set Value Dialog' window with the following fields and values:

- Dec:** 1684107116
- Hex:** 0x64616F6C
- Float:** 1.6634185e+22
- Bool:** 0 and 1 buttons
- Binary:** 6C 6F 61 64
- Bit Size:** Radio buttons for 1, 8, 16, 32 (selected), 64, and ?

Buttons: OK, Cancel, Hex Edit...

Fig. 186: Entering a restore value in the Set Value dialog

Double-click on *SubIndex 001* to enter the Set Value dialog. Enter the reset value **1684107116** in field *Dec* or the value **0x64616F6C** in field *Hex* (ASCII: "load") and confirm with **OK** (Fig. *Entering a restore value in the Set Value dialog*).

- All changeable entries in the slave are reset to the default values.
- The values can only be successfully restored if the reset is directly applied to the online CoE, i.e. to the slave. No values can be changed in the offline CoE.
- TwinCAT must be in the RUN or CONFIG/Freerun state for this; that means EtherCAT data exchange takes place. Ensure error-free EtherCAT transmission.
- No separate confirmation takes place due to the reset. A changeable object can be manipulated beforehand for the purposes of checking.
- This reset procedure can also be adopted as the first entry in the startup list of the slave, e.g. in the state transition PREOP->SAFEOP or, as in Fig. *CoE reset as a startup entry*, in SAFEOP->OP.

All backup objects are reset to the delivery state.

### **i Alternative restore value**

In some older terminals (FW creation approx. before 2007) the backup objects can be switched with an alternative restore value: Decimal value: 1819238756, Hexadecimal value: 0x6C6F6164.

An incorrect entry for the restore value has no effect.

## 7.4 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

### Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on her internet pages: [www.beckhoff.com](http://www.beckhoff.com)

You will also find further documentation for Beckhoff components there.

### Support

The Beckhoff Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline: +49 5246 963 157  
e-mail: [support@beckhoff.com](mailto:support@beckhoff.com)  
web: [www.beckhoff.com/support](http://www.beckhoff.com/support)

### Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline: +49 5246 963 460  
e-mail: [service@beckhoff.com](mailto:service@beckhoff.com)  
web: [www.beckhoff.com/service](http://www.beckhoff.com/service)

### Headquarters Germany

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20  
33415 Verl  
Germany

Phone: +49 5246 963 0  
e-mail: [info@beckhoff.com](mailto:info@beckhoff.com)  
web: [www.beckhoff.com](http://www.beckhoff.com)

## **Trademark statements**

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

## **Third-party trademark statements**

DeviceNet and EtherNet/IP are trademarks of ODVA, Inc.

DSP System Toolbox, Embedded Coder, MATLAB, MATLAB Coder, MATLAB Compiler, MathWorks, Predictive Maintenance Toolbox, Simscape, Simscape™ Multibody™, Simulink, Simulink Coder, Stateflow and ThingSpeak are registered trademarks of The MathWorks, Inc.

EnDat is a trademark of Dr. Johannes Heidenhain GmbH.

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

More Information:  
**[www.beckhoff.com/EL2574](http://www.beckhoff.com/EL2574)**

Beckhoff Automation GmbH & Co. KG  
Hülshorstweg 20  
33415 Verl  
Germany  
Phone: +49 5246 9630  
[info@beckhoff.com](mailto:info@beckhoff.com)  
[www.beckhoff.com](http://www.beckhoff.com)

