BECKHOFF New Automation Technology

Documentation | EN

EL600x, EL602x

Serial Interface Terminals

2020-11-17 | Version: 4.8

Table of contents

1 Foreword ... 5
1.1 Overview Serial Interface Terminals ... 5
1.2 Notes on the documentation ... 6
1.3 Safety instructions ... 7
1.4 Documentation Issue Status ... 8
1.5 Version identification of EtherCAT devices ... 10
1.5.1 Beckhoff Identification Code (BIC) ... 14
2 Product overview ... 16
2.1 EL6001, EL6021 ... 20
2.1.1 Introduction ... 20
2.1.2 Technical data ... 21
2.2 EL6002, EL6022 ... 22
2.2.1 Introduction ... 22
2.2.2 Technical data ... 23
2.3 Start up ... 23
3 Basics communication ... 24
3.1 EtherCAT basics ... 24
3.2 EtherCAT cabling – wire-bound ... 24
3.3 General notes for setting the watchdog ... 25
3.4 EtherCAT State Machine ... 27
3.5 CoE Interface ... 29
3.6 Distributed Clock ... 34
4 Mounting and Wiring ... 35
4.1 Instructions for ESD protection ... 35
4.2 EL6001, EL6021 ... 35
4.2.1 Installation on mounting rails ... 35
4.2.2 Connection ... 38
4.2.3 Positioning of passive Terminals ... 43
4.2.4 LEDs and terminal connector assignments ... 44
4.3 EL6002, EL6022 ... 46
4.3.1 Mounting and demounting - terminals with front unlocking ... 46
4.3.2 Recommended mounting rails ... 48
4.3.3 LEDs and pin assignment ... 49
4.4 Positioning of passive Terminals ... 52
4.5 Installation instructions for enhanced mechanical load capacity ... 53
4.6 Installation positions ... 54
4.7 UL notice ... 56
4.8 ATEX - Special conditions (extended temperature range) ... 57
4.9 Continuative documentation about explosion protection ... 58
5 Commissioning ... 59
5.1 TwinCAT Quick Start ... 59
5.1.1 TwinCAT 2 ... 62
5.1.2 TwinCAT 3 ... 72

5.2 TwinCAT Development Environment ... 85
5.2.1 Installation of the TwinCAT real-time driver ... 86
5.2.2 Notes regarding ESI device description ... 91
5.2.3 TwinCAT ESI Updater ... 95
5.2.4 Distinction between Online and Offline ... 95
5.2.5 OFFLINE configuration creation ... 96
5.2.6 ONLINE configuration creation ... 101
5.2.7 EtherCAT subscriber configuration ... 109
5.3 General Notes - EtherCAT Slave Application ... 118
5.4 Operating modes and process data ... 126
5.5 Hints regarding TcVirtualComDriver ... 133
5.6 Communication features ... 135
5.7 LIN Master Feature EL6001 ... 136
5.8 Example programs ... 140
5.8.1 Sample program 1 ... 140
5.8.2 Sample program 2 ... 143
5.8.3 Sample program 3 (LIN) ... 145

6 Overview of CoE objects EL6001, EL6021 ... 148
6.1 Object description and parameterization ... 148
6.1.1 Objects for commissioning ... 148
6.1.2 Standard objects (0x1000-0x1FFF) ... 150
6.1.3 Profile-specific objects (0x6000-0xFFFF) [from hardware version 03] ... 166
6.2 Control and status word ... 169

7 Overview CoE objects EL6002, EL6022 ... 172
7.1 Object description and parameterization ... 172
7.1.1 Objects for commissioning ... 172
7.1.2 Standard objects (0x1000-0x1FFF) ... 173
7.1.3 Profile-specific objects (0x6000-0xFFFF) [from hardware version 03] ... 185
7.2 Control and status data ... 188

8 Appendix ... 190
8.1 EtherCAT AL Status Codes ... 190
8.2 Firmware compatibility ... 190
8.3 Firmware Update EL/ES/EM/ELM/EPxxxx ... 191
8.3.1 Device description ESI file/XML ... 192
8.3.2 Firmware explanation ... 195
8.3.3 Updating controller firmware *.efw ... 196
8.3.4 FPGA firmware *.rbf ... 198
8.3.5 Simultaneous updating of several EtherCAT devices ... 202
8.4 Restoring the delivery state ... 203
8.5 Support and Service ... 204

1 Foreword

1.1 Overview Serial Interface Terminals

EL6001 [▶ 20] (1 channel Serial Interface Terminal, RS232C)
EL6021 [▶ 20] (1 channel Serial Interface Terminal, RS422/RS485)
EL6002 [▶ 22] (2 channel Serial Interface Terminal, RS232C)
EL6022 [▶ 22] (2 channel Serial Interface Terminal, RS422/RS485)

1.2 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.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

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®, TwinCAT®, 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. 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.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents: EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702 with corresponding applications or registrations in various other countries.

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.

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.

Description of instructions

In this documentation the following instructions are used.
These instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow this safety instruction directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow this safety instruction endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow this safety instruction can lead to injuries to persons.

NOTE

Damage to environment/equipment or data loss

Failure to follow this instruction can lead to environmental damage, equipment damage or data loss.

Tip or pointer

This symbol indicates information that contributes to better understanding.

1.4 Documentation Issue Status

Version	Comment
4.8	• Update chapter "Basic principles" • Update structure • Update revision status
4.7	• Update chapter "Object description and parameterization" • Update structure • Update revision status
4.6	• Correction RS232 level • Update chapter "Technical data" • Update structure • Update revision status
4.5	• Update chapter "Commissioning" • Update revision status
4.4	• Update chapter "Technical data" • Update chapter "Operating modes and process data" • Update chapter " Communication features" • Update structure • Update revision status
4.3	• Update chapter "Technical data" • Addenda chapter "Instructions for ESD protection" • Chapter "ATEX - Special conditions" replaced with chapter "ATEX - Special conditions (extended temperature range)" • Addenda chapter "TwinCAT Quickstart" • Update revision status
4.2	• Update in section ”LEDs and terminal connector assignments”
4.1	• Addenda in section “LEDs and pin assignment”
4.0	• Migration and revision • Section "Mounting and demounting" in section "EL6002, EL6022" in "Mounting and wiring" complemented with "Front unlocking" • Section "Installation instructions for enhanced mechanical load capacity" moved from subsection "EL6001, EL6021" to section "Mounting and wiring" • Section "Installation positions" removed from subsection "EL6001, EL6021" (since already present in higher-level section) • Section "Configuration with the TwinCAT System Manager" moved from section "Commissioning" to subsection "TwinCAT 2.1x" • Section "LIN Feature EL6001" moved to section "Commissioning" • Sections "Sample program 1" and "Sample program 2" consolidated into new section "Sample programs"; new section "Sample programs" integrated in section "Commissioning“ • Section "Sample program 3 (LIN)" added to section "Sample programs"
3.8	• "Technical data" section updated • "Installation instructions for enhanced mechanical load capacity" section supplemented • Structural update • Revision version updated

Version	Comment
3.7	• Update LED description • Update revision status
3.6	• Update revision status • Update structure
3.5	• Update chapter "Technical data" • Update chapter "Object description and parameterization" • Update chapter "Communication features" • Update chapter "Technology" • Update chapter "Process data" • Update structure
3.4	• Update chapter "Technology"
3.3	• Update Technical data
3.2	• Update Technical data
3.1	• Addenda of notes and description of command mode
3.0	• Update chapter "Object description"
2.9	• Addenda chapter "Communication features and TcVirtualComDriver"
2.8	• Update Technical data
2.7	• Update Technical data
2.6	• Update chapter "Technology" and "Process data"
2.5	• Update chapter "Technology"
2.4	• Object description and Technical notes added
2.3	• Firmware compatibility notice, Technical notes added
2.2	• Addenda
2.1	• Addenda
2.0	• First public issue
0.3	• Addenda
0.2	• Corrections and addenda
0.1	• Preliminary documentation for EL60xx

1.5 Version identification of EtherCAT devices

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. *“EL5021 EL terminal, standard IP20 IO device with batch number and revision ID (since 2014/01)”*.
- The type, version and revision are read as decimal numbers, even if they are technically saved in hexadecimal.

Identification number

Beckhoff EtherCAT devices from the different lines have different kinds of identification numbers:

Production lot/batch number/serial number/date code/D number

The serial number 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
Ser. no.: 12063A02: 12 - production week 12 06 - production year 2006 3A - firmware version 3A 02 - hardware version 02

Exceptions can occur in the **IP67 area**, where the following syntax can be used (see respective device documentation):

Syntax: D ww yy x y z u

D - prefix designation
ww - calendar week
yy - year
x - firmware version of the bus PCB
y - hardware version of the bus PCB
z - firmware version of the I/O PCB
u - hardware version of the I/O PCB

Example: D.22081501 calendar week 22 of the year 2008 firmware version of bus PCB: 1 hardware version of bus PCB: 5 firmware version of I/O PCB: 0 (no firmware necessary for this PCB) hardware version of I/O PCB: 1

Unique serial number/ID, ID number

In addition, in some series each individual module has its own unique serial number.

See also the further documentation in the area

- IP67: EtherCAT Box
- Safety: TwinSafe
- Terminals with factory calibration certificate and other measuring terminals

Examples of markings

Fig. 1: EL5021 EL terminal, standard IP20 IO device with serial/ batch number and revision ID (since 2014/01)

Fig. 2: EK1100 EtherCAT coupler, standard IP20 IO device with serial/ batch number

Fig. 3: CU2016 switch with serial/ batch number

Fig. 4: EL3202-0020 with serial/ batch number 26131006 and unique ID-number 204418

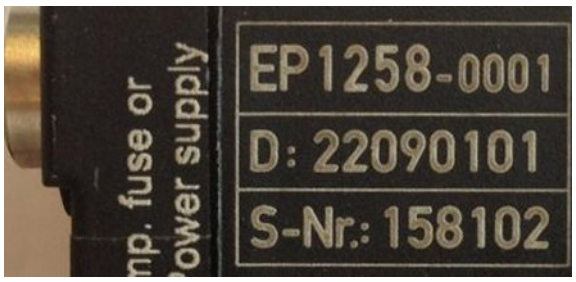


Fig. 5: EP1258-00001 IP67 EtherCAT Box with batch number/ date code 22090101 and unique serial number 158102

Fig. 6: EP1908-0002 IP67 EtherCAT Safety Box with batch number/ date code 071201FF and unique serial number 00346070

Fig. 7: EL2904 IP20 safety terminal with batch number/ date code 50110302 and unique serial number 00331701

Fig. 8: ELM3604-0002 terminal with unique ID number (QR code) 100001051 and serial/ batch number 44160201

1.5.1 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.

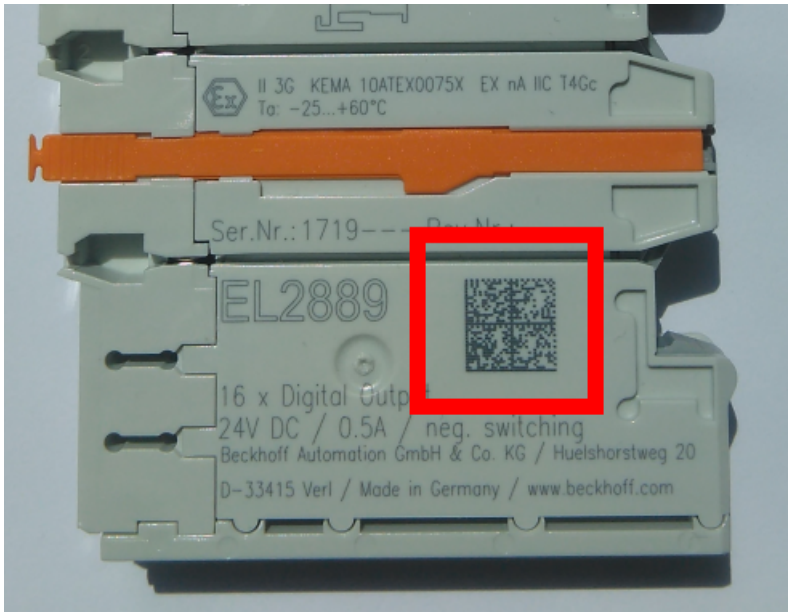


Fig. 9: BIC as data matrix code (DMC, code scheme ECC200)

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. The data under positions 1 to 4 are always available.

The following information is contained:

Item no.	Type of information	Explanation	Data identifier	Number of digits incl. data identifier	Example
1	Beckhoff order number	**Beckhoff order number**	1P	8	1P072222
2	Beckhoff Traceability Number (BTN)	**Unique serial number, see note below**	S	12	SBTNk4p562d7
3	Article description	**Beckhoff article description, e.g. EL1008**	1K	32	1KEL1809
4	Quantity	**Quantity in packaging unit, e.g. 1, 10, etc.**	Q	6	Q1
5	Batch number	Optional: Year and week of production	2P	14	2P401503180016
6	ID/serial number	Optional: Present-day serial number system, e.g. with safety products	51S	12	51S678294104
7	Variant number	Optional: Product variant number on the basis of standard products	30P	32	30PF971, 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 item 1 to 4 and 6. The data identifiers are marked in red for better display:

BTN

An important component of the BIC is the Beckhoff Traceability Number (BTN, item no. 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.

NOTE

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 information.

2 Product overview

Technology

The EL600x and EL602x serial interface terminals enable the connection of devices with an RS232 (or RS485 / RS422) interface. In the case of the EL600x, the data is exchanged with the controller in full duplex mode; in the case of the EL602x, half duplex mode is additionally possible. The terminal has one receive buffer an one transmit buffer per channel, see technical data. Data transfer between the terminal and the controller takes place via a handshake.

The factory setting of the terminals is:

- 9600 baud
- 8N1: 8 data bits, 1 stop bit, no parity
- in the EL600x the RTS/CTS control is active
- the EL602x operates in full duplex mode with deactivated point-to-point connection.

Basic principles

During transfer of several bytes of data, the data (x bytes or 8*x bits in total) are sent in individual telegrams containing 7 or 8 bits, based on the coding specification (e.g. 7E2 or 8N1). A telegram consists of:

- Start bit
- Data bits (7 or 8, starting with the LSB [least significant bit])
- Optional: parity bit
  - "E" EVEN: The parity bit is set by the sender such that the parity is even
  - "O" ODD: The parity bit is set by the sender such that the parity is odd
  - "N" NOT: no parity bit
  - "M" MARK: The parity bit is set to 1 by the sender
  - "S" SPACE: The parity bit is set to 0 by the sender
- Stop bit (1 or 2)

Accordingly, the coding specification 8N1 means: 8 data bits, no parity bit, 1 stop bit.

If 7-bit coding is selected, of each data byte that is transferred from the PLC to the terminal via the cyclic process data, only the lower 7 bits are sent. In other words, if 10 bytes of data (consisting of 8 bits) are sent to the EL60xx, 10 telegrams of 7 bits each are sent.

If 9-bit coding is selected, the 16-bit process data interface must be used. The terminal then expects the 9 useful bits in the lower 9 bits of the 16-bit word.

Frequency

The frequency of the data transfer must be known in the sender and receiver and match within a few percent, in order to ensure that the receiver can correctly detect any changes in level on the line.

Handshake

An additional handshake between the sender and the receiver can be used so that the receiver can indicate that it is ready to receive. The EL60xx supports two types of handshake:

- via special RTS/CTS data cables
  - This features must be activated in the CoE.
  - Only possible with EL6001/EL6002.
- via special data telegrams
  - This features must be activated in the CoE.

RS Standards

The term RS232 (RS-232) is used in this document briefly for the ANSI/EIA/TIA-232-F standard.

The term RS422 (RS-422) is used briefly in this document for the ITU-T V.11 standard according to ANSI/TIA/EIA-422-B-1994.

The term RS485 (RS-485) is used in this document briefly for the ANSI/TIA/EIA-485-A-98 standard.

Level interfaces

The EL6001/6002 devices operate at an RS232 level with reference to GND, the EL6021/6022 devices with a differential RS485/422 level.

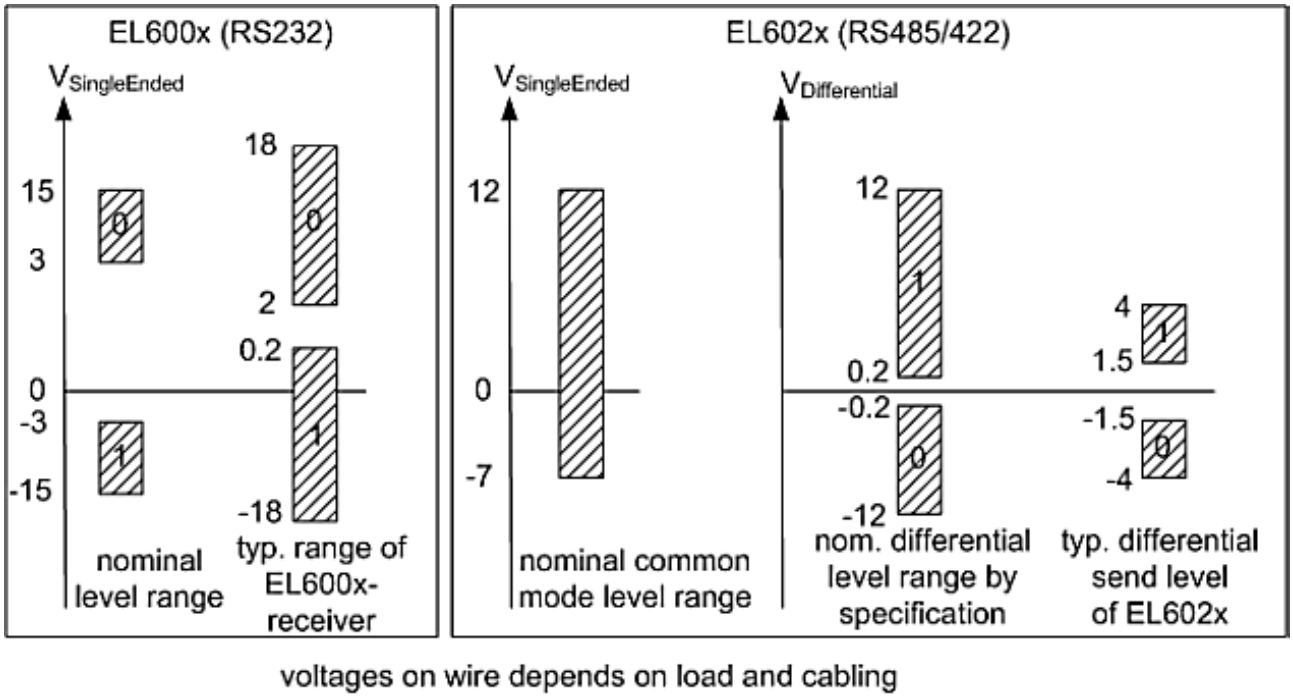


Fig. 10: Level interfaces RS232, RS485/422

Termination and topology

The serial RS422 and RS485 communication technologies operate with voltage levels on a 2-wire line. Reflections at high-resistance line ends can lead to signal distortion. For this reason termination resistors are required at the receiver. For RS422/485 these are 120 Ω resistors, which together with the line resistance result in a voltage drop over the transmission link.

Permitted cable length

The line resistance together with the termination resistor results in an overall voltage drop over the transmission link. An unacceptably high number of termination resistors would result in excessive attenuation of the signal.

The system design should ensure that the voltage does not drop below 200 mV at the receiver (see Fig.), which is the minimum voltage required.

In RS422 mode each line must be terminated with 120 Ω at the receiver.

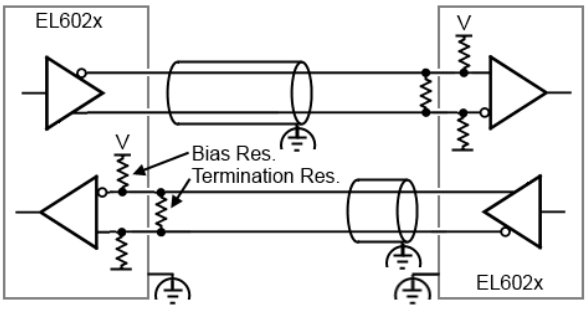


Fig. 11: RS422 termination

In RS485 mode with several devices, termination resistors are only used at the two end devices.

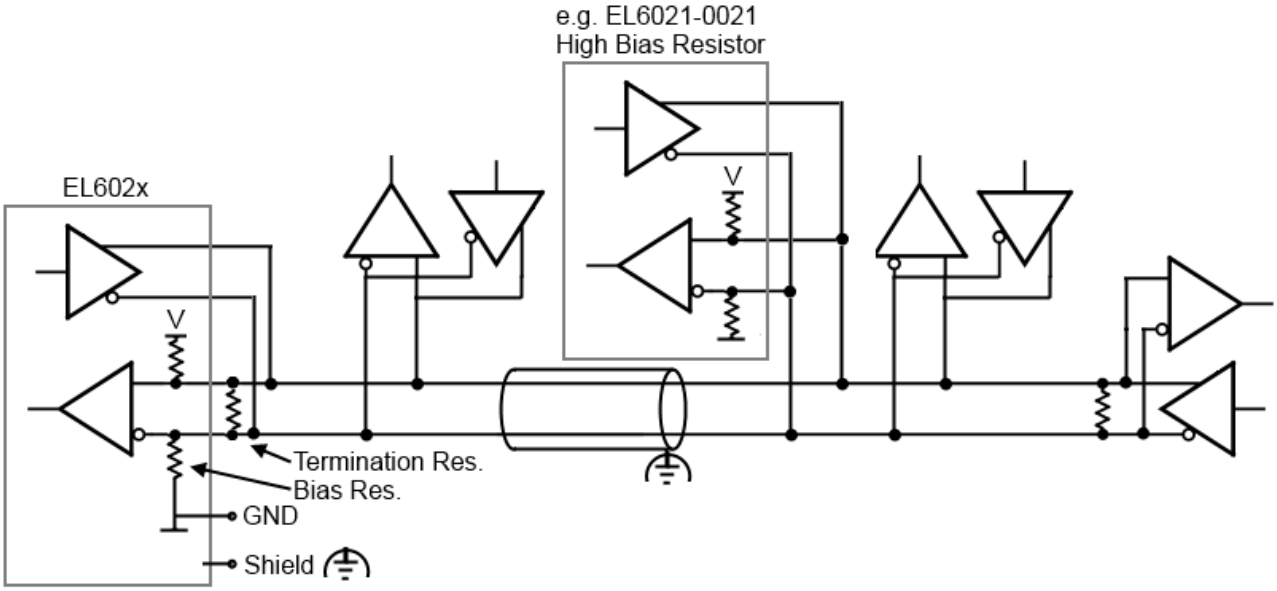


Fig. 12: RS485 termination

The background is the different design of RS422/EIA-422 and RS485/EIA-485:

- RS422: 1 Rx → Tx n (maximum 10 receivers)
- RS485: n Rx → Tx m (maximum 32/128 devices, depending on the resulting bus loading)

Components for RS485 usually have a higher input impedance, resulting in lower bus load.

Termination with EL602x and BIAS resistors

The EL602x devices do not have integrated termination resistors, in order to enable operation in bus mode. Any termination that may be required must be connected outside the terminal.
The EL602x devices feature integrated bias resistors < 1 kOhm, which bring the bus lines to defined levels, even if the line is disconnected.
If several EL602x devices are connected in a bus, the parallel bias resistors may hamper the data communication. In this case the EL6021-0021, which has significantly higher bias resistors, should be used as an intermediate device.

Topology

The termination and the bias resistors generate a load on the bus. However, they are essential for unambiguous bus levels and therefore have to be positioned with diligence. Ideally the RS422/485 bus should be configured as a daisy chain or a simple chain, see Fig. [▶ 19] The following topologies may be problematic:

- Star topologies: each end point should ideally be terminated, but this can lead to excessive bus loading and ambiguous signal levels. Other potential issues are reflections and runtime variations.

- Intermeshed topologies: no clear end points, which means reflections and circulating currents are possible.

Shielding/shield

NOTE

Do not use functional earth for discharge of residual currents or potential differences!

The EL60xx units offer a shielded connection for discharging EMC interference via the cable shield (FE, functional earth). The shield must not be misused for discharging residual currents or potential differences.

The EL60xx units offer a shielded connection for discharging EMC interference via the cable shield (FE, functional earth). The shield must not be misused for discharging residual currents or potential differences.

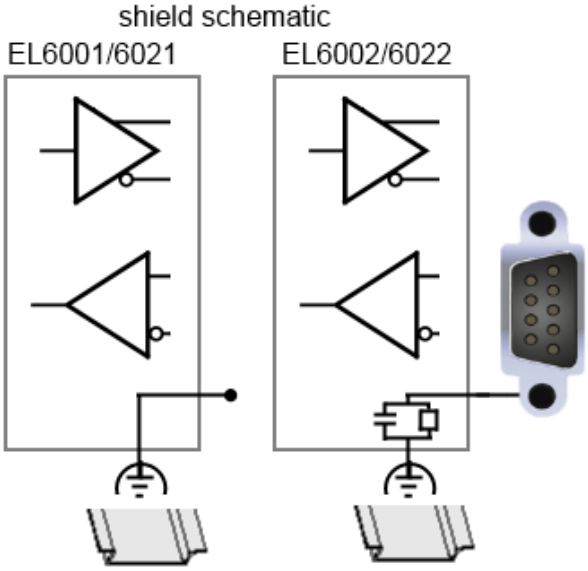
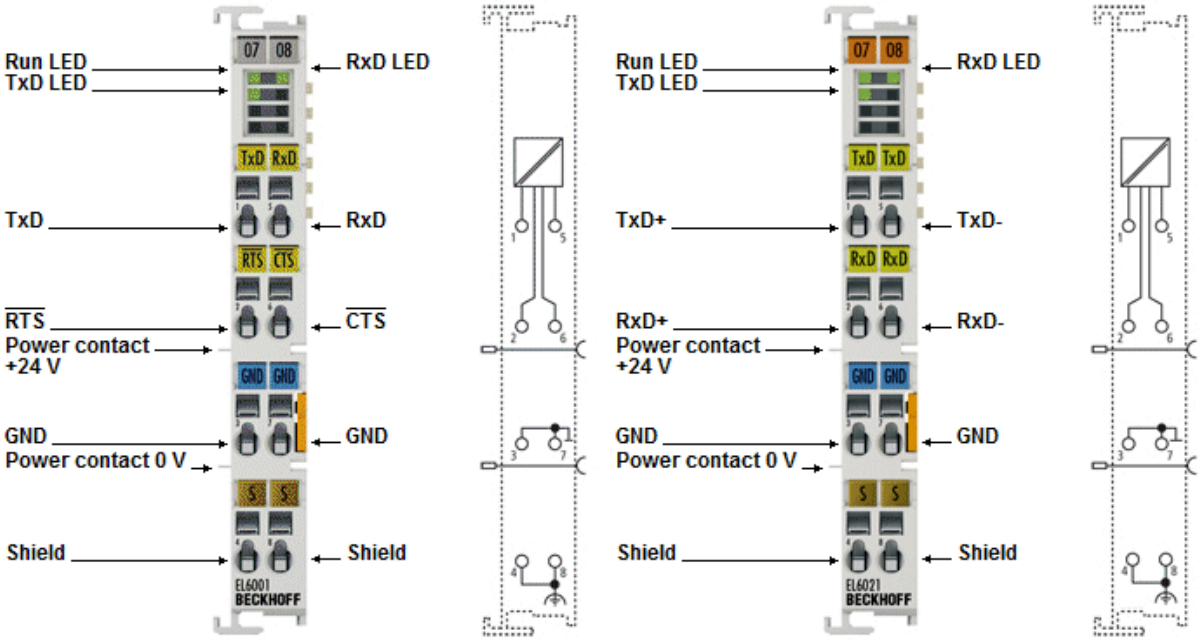


Fig. 13: EL60xx shield connection

In the 2-channel versions the D-Sub 9 shield is connected with the mounting rail via a high-resistance RC combination.

2.1 EL6001, EL6021

2.1.1 Introduction



Serial Interface Terminal (RS232C/RS422/RS485), 1 channel

The EL6001 and EL6021 serial interfaces enable the connection of devices with RS-232 or RS422/RS485 interface. The EL6001 operates in conformity with the CCITT V.28/DIN 66 259-1 standards.
The device connected to the EL6001 EtherCAT Terminal communicates with the automation device via the coupler. The active communication channel operates independently of the higher-level bus system in full duplex mode or selectable half duplex mode (EL6021) at up to 115.2 kbaud.
The RS232 interface guarantees high immunity to interference through electrically isolated signals, which is additionally guaranteed for the EL6021 through differential signal transmission.

In conjunction with the TwinCAT Virtual Serial COM Driver (see TwinCAT Supplements – Communication) the EL6001/EL6021 can be used as a normal Windows COM interface.

Quick links

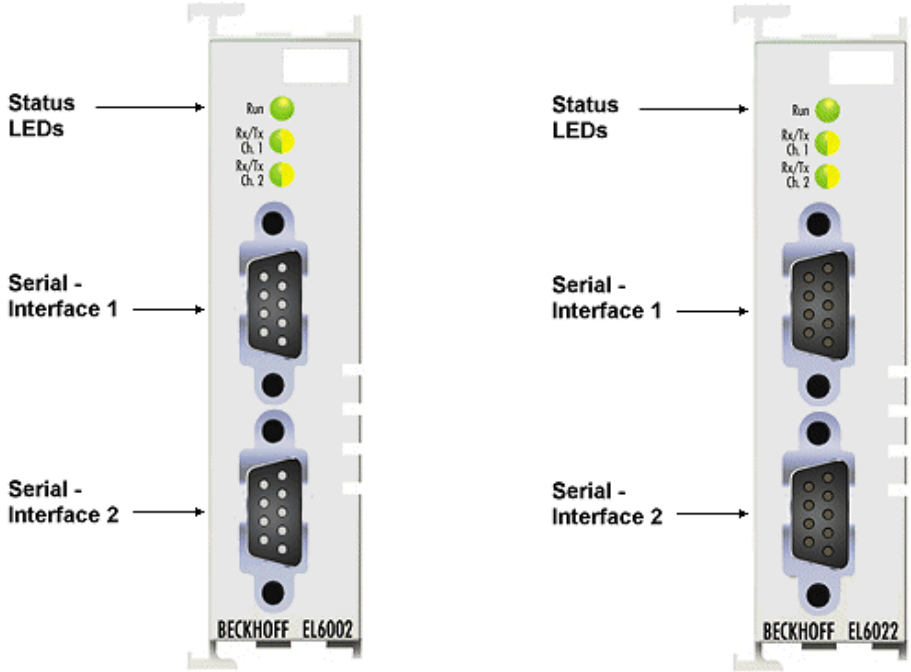
- EtherCAT basics [▶ 24]
- Technology Serial Interface Terminals [▶ 16]
- Commissioning [▶ 59]
- Process data, general notes [▶ 126]
- CoE object description and parameterization EL60x1 [▶ 148]
- Control and status data EL60x1 [▶ 169]

2.1.2 Technical data

Technical data	EL6001	EL6021
Data transfer channels	TxD and RxD, full duplex	TxD and RxD, full/half duplex
Data transfer rate	2400...115200 baud, default: 9600 baud, 8 data bits, no parity, 1 stop bit from firmware 07 [▶ 190]: also 12000 baud and 14400 baud from firmware 11 [▶ 190]: any integer baud rate 1000… 115200 Baud	2400...115200 baud, default: 9600 baud, 8 data bits, no parity, 1 stop bit
Data buffer	864 byte receive buffer, 128 byte transmit buffer EL6001 from FW08: 250 byte transmit buffer	
Bit transfer	-	with differential signal
Level interface	RS232	RS485/422
Bit distortion	< 3 %	-
Cable length	max. 15 m	max. 1000 m (Twisted Pair)
Line impedance	-	120 Ω
Providing external supply	-	-
Diagnosis	Status LEDs	
Power supply	via the E-Bus	
Current consumption via E-bus	typ. 120 mA	typ. 170 mA (in case of short circuit: typ. 250 mA)
Electrical isolation	500 V (E-bus/RS232C)	500 V (E-bus/RS422, E-bus/RS485)
Bit width in process image	1 x 8 bit Control/Status, Inputs/Outputs: 3 x 8 bit user data or 1 x 8 bit Control/Status, Inputs/Outputs: 5 x 8 bit user data or 1 x 16 bit Control/Status, Inputs/Outputs: 22 x 8 bit user data (configurable)	
Configuration	no address setting required configuration via TwinCAT System Manager	
Weight	approx. 55 g	
Permissible ambient temperature range during operation	-25°C ... +60°C (extended temperature range)	
Permissible ambient temperature range during storage	-40°C ... +85°C	
Permissible relative humidity	95%, no condensation	
Dimensions (W x H x D)	approx. 15 mm x 100 mm x 70 mm (width aligned: 12 mm)	
Mounting	on 35 mm mounting rail conforms to EN 60715	
Vibration/shock resistance	conforms to EN 60068-2-6 / EN 60068-2-27, see also installation instructions [▶ 53] for enhanced mechanical load capacity	
EMC immunity/emission	conforms to EN 61000-6-2 / EN 61000-6-4	
Protection class	IP20	
Installation position	variable	
Approval	CE ATEX [▶ 57] cULus [▶ 56] EAC	CE ATEX [▶ 57] cULus [▶ 56] IECEx EAC

2.2 EL6002, EL6022

2.2.1 Introduction



Serial Interface Terminal (RS232C/RS422/RS485), 2 channel

The EL6002 and EL6022 serial interfaces enable the connection of devices with two RS232 or two RS422/RS485 interfaces each with one D-Sub connector (9 pin). The interfaces are electrically isolated from each other and from the EtherCAT.
The devices connected to the EL6002/EL6022 EtherCAT Terminals communicate with the automation device via the Coupler. The active communication channel operates independently of the higher-level EtherCAT system in full duplex mode with 300 baud up to 115.2 kbaud.
The RS232/RS422/RS485 interfaces guarantee high interference immunity through electrically isolated signals. The EL6022 can provide 2 x 5 V/20 mA from the E-bus supply (electrically isolated, short-circuit-proof) as supply for external devices.

In conjunction with the TwinCAT Virtual Serial COM Driver, the EL60xx can be used as a normal Windows COM interface.

Quick links

- EtherCAT basics [▶ 24]
- Technology Serial Interface Terminals [▶ 16]
- Commissioning [▶ 59]
- Process data, general notes [▶ 126]
- CoE object description and parameterization EL60x1 [▶ 172]
- Control and status data EL60x1 [▶ 188]

2.2.2 Technical data

Technical data	EL6002	EL6022
Data transfer channels	2, TxD and RxD, full duplex	2, TXD and RXD, full/half duplex
Connection	2 x D-sub connector (DE9), 9-pin	2 x D-sub connector (DE9), 9-pin
Data transfer rate	300...115200 baud default: 9600 baud, 8 data bits, no parity, 1 stop bit	
Data buffer	864 byte receive buffer, 128 byte transmit buffer per channel	
Level interface	RS232	RS485/422
Cable length	max. 15 m	max. 1000 m (Twisted Pair)
Providing external supply	-	2x typ. 5V (± 20%), from E-bus supply (electrically isolated), max. 20 mA, short-circuit-proof
Diagnosis	Status LEDs	
Power supply	via the E-Bus	
Current consumption via E-bus	typ. 170 mA	typ. 250 mA (in case of short circuit: typ. 250 mA)
Electrical isolation	500 V (E-bus/RS232C)	500 V (E-bus/RS422, E-bus/RS485)
Bit width in process image	1 x 16 bit Control/Status, Inputs/Outputs: 22 x 8 bit user data	
Configuration	no address setting required configuration via TwinCAT System Manager	
Permissible ambient temperature range during operation	-25°C ... +60°C (extended temperature range)	
Permissible ambient temperature range during storage	-40°C ... +85°C	
Permissible relative humidity	95%, no condensation	
Weight	approx. 70 g	
Dimensions (W x H x D)	approx. 26 mm x 100 mm x 52 mm (width aligned: 23 mm)	
Mounting [▶ 46]	on 35 mm mounting rail conforms to EN 60715	
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 class	IP20	
Installation position	variable	
Approval	CE ATEX [▶ 57] cULus [▶ 56] EAC	

2.3 Start up

Start

For commissioning:

- mount the EL600x / EL602x as described in the chapter Mounting and wiring [▶ 35]
- configure the EL600x / EL602x in TwinCAT as described in the chapter Commissioning [▶ 59]

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

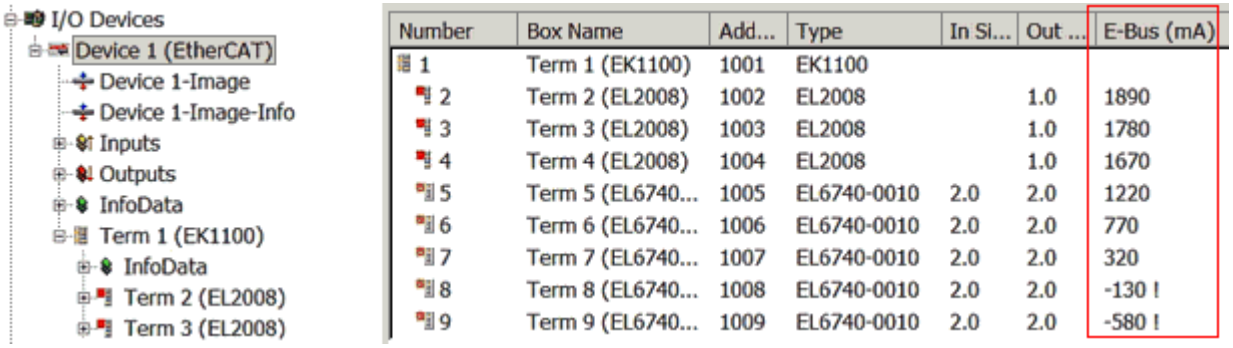
It is recommended to use the appropriate Beckhoff components e.g.
- cable sets ZK1090-9191-xxxx respectively
- RJ45 connector, field assembly ZS1090-0005
- EtherCAT cable, field assembly ZB9010, ZB9020

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.



Number	Box Name	Add...	Type	In Si...	Out ...	E-Bus (mA)
1	Term 1 (EK1100)	1001	EK1100			
2	Term 2 (EL2008)	1002	EL2008		1.0	1890
3	Term 3 (EL2008)	1003	EL2008		1.0	1780
4	Term 4 (EL2008)	1004	EL2008		1.0	1670
5	Term 5 (EL6740...	1005	EL6740-0010	2.0	2.0	1220
6	Term 6 (EL6740...	1006	EL6740-0010	2.0	2.0	770
7	Term 7 (EL6740...	1007	EL6740-0010	2.0	2.0	320
8	Term 8 (EL6740...	1008	EL6740-0010	2.0	2.0	-130 !
9	Term 9 (EL6740...	1009	EL6740-0010	2.0	2.0	-580 !

Fig. 14: System manager current calculation

NOTE

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

ELxxxx terminals are equipped with a safety feature (watchdog) that switches off the outputs after a specifiable time e.g. in the event of an interruption of the process data traffic, depending on the device and settings, e.g. in OFF state.

The EtherCAT slave controller (ESC) in the EL2xxx terminals features two watchdogs:

- SM watchdog (default: 100 ms)
- PDI watchdog (default: 100 ms)

SM watchdog (SyncManager Watchdog)

The SyncManager watchdog is reset after each successful EtherCAT process data communication with the terminal. If no EtherCAT process data communication takes place with the terminal for longer than the set and activated SM watchdog time, e.g. in the event of a line interruption, the watchdog is triggered and the outputs are set to FALSE. The OP state of the terminal is unaffected. The watchdog is only reset after a successful EtherCAT process data access. Set the monitoring time as described below.

The SyncManager watchdog monitors correct and timely process data communication with the ESC from the EtherCAT side.

PDI watchdog (Process Data Watchdog)

If no PDI communication with the EtherCAT slave controller (ESC) takes place for longer than the set and activated PDI watchdog time, this watchdog is triggered.
PDI (Process Data Interface) is the internal interface between the ESC and local processors in the EtherCAT slave, for example. The PDI watchdog can be used to monitor this communication for failure.

The PDI watchdog monitors correct and timely process data communication with the ESC from the application side.

The settings of the SM- and PDI-watchdog must be done for each slave separately in the TwinCAT System Manager.

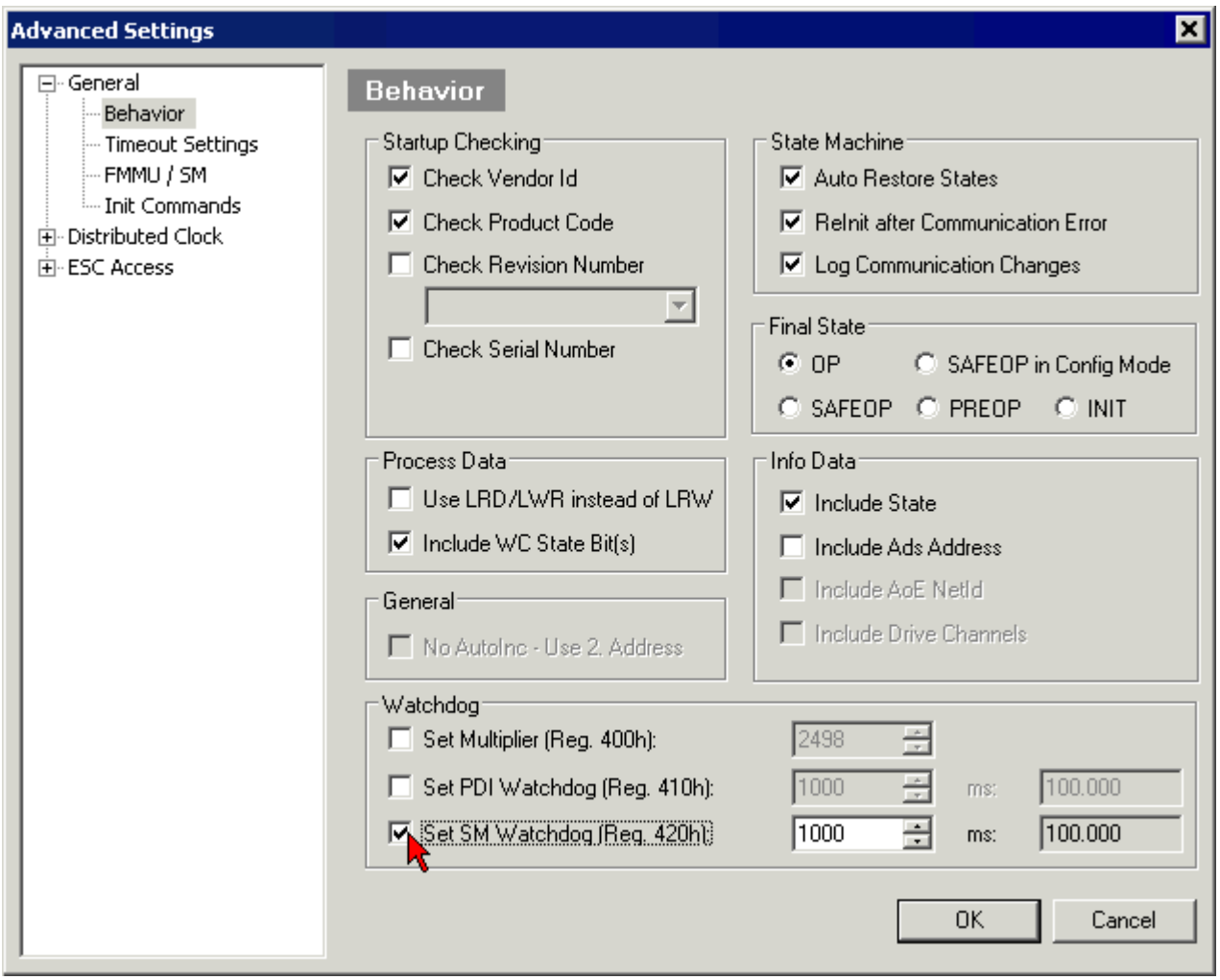


Fig. 15: EtherCAT tab -> Advanced Settings -> Behavior -> Watchdog

Notes:

- the multiplier is valid for both watchdogs.
- each watchdog has its own timer setting, the outcome of this in summary with the multiplier is a resulting time.
- Important: the multiplier/timer setting is only loaded into the slave at the start up, if the checkbox is activated.
  If the checkbox is not activated, nothing is downloaded and the ESC settings remain unchanged.

Multiplier

Multiplier

Both watchdogs receive their pulses from the local terminal cycle, divided by the watchdog multiplier:

1/25 MHz * (watchdog multiplier + 2) = 100 µs (for default setting of 2498 for the multiplier)

The standard setting of 1000 for the SM watchdog corresponds to a release time of 100 ms.

The value in multiplier + 2 corresponds to the number of basic 40 ns ticks representing a watchdog tick.
The multiplier can be modified in order to adjust the watchdog time over a larger range.

Example “Set SM watchdog”

This checkbox enables manual setting of the watchdog times. If the outputs are set and the EtherCAT communication is interrupted, the SM watchdog is triggered after the set time and the outputs are erased. This setting can be used for adapting a terminal to a slower EtherCAT master or long cycle times. The default SM watchdog setting is 100 ms. The setting range is 0...65535. Together with a multiplier with a range of 1...65535 this covers a watchdog period between 0...~170 seconds.

Calculation

Multiplier = 2498 → watchdog base time = 1 / 25 MHz * (2498 + 2) = 0.0001 seconds = 100 µs
SM watchdog = 10000 → 10000 * 100 µs = 1 second watchdog monitoring time

⚠ CAUTION

Undefined state possible!

The function for switching off of the SM watchdog via SM watchdog = 0 is only implemented in terminals from version -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 and
- Operational
- Boot

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

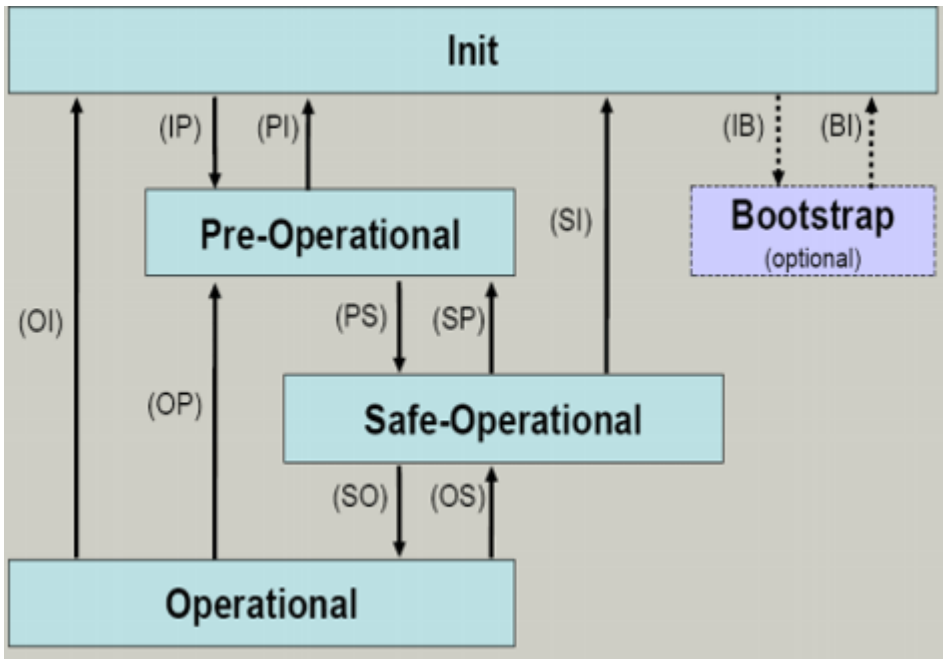


Fig. 16: States of the EtherCAT State Machine

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 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 DP-RAM areas of the EtherCAT slave controller (ECSC).

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

The default set watchdog [▶ 25] monitoring sets the outputs of the 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 watchdog 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)) 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 read 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 parameter 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_{dez})
- SubIndex: 0x00…0xFF (0...255_{dez})

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 (“input” from the perspective of the EtherCAT master)
- 0x7000: Output PDOs (“output” 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:

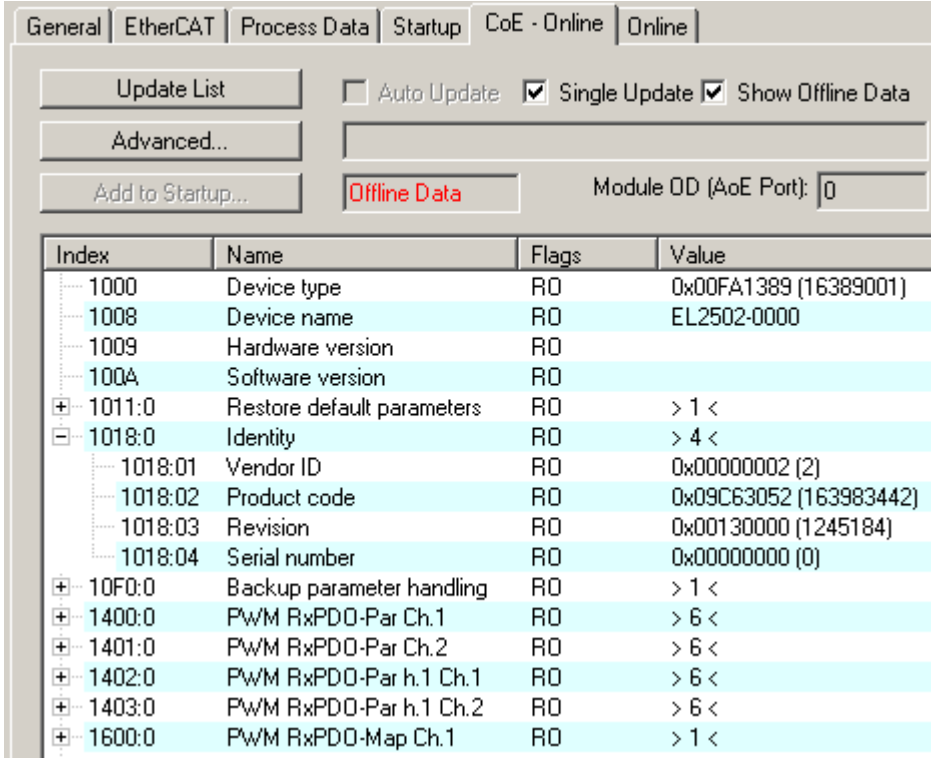


Fig. 17: “CoE Online” tab

The figure above shows the CoE objects available in device “EL2502”, ranging from 0x1000 to 0x1600. The subindices for 0x1018 are expanded.

Data management and function “NoCoeStorage”

Some parameters, particularly the setting parameters of the slave, are configurable and writeable. This can be done in write or read mode

- via the System Manager (Fig. “CoE Online” tab) by clicking
  This is useful for commissioning of the system/slaves. Click on the row of the index to be parameterized and enter a value in the “SetValue” dialog.
- from the control system/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.

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 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.
- Function is not supported: continuous changing of CoE values is not permissible in view of the lifetime limit.

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.

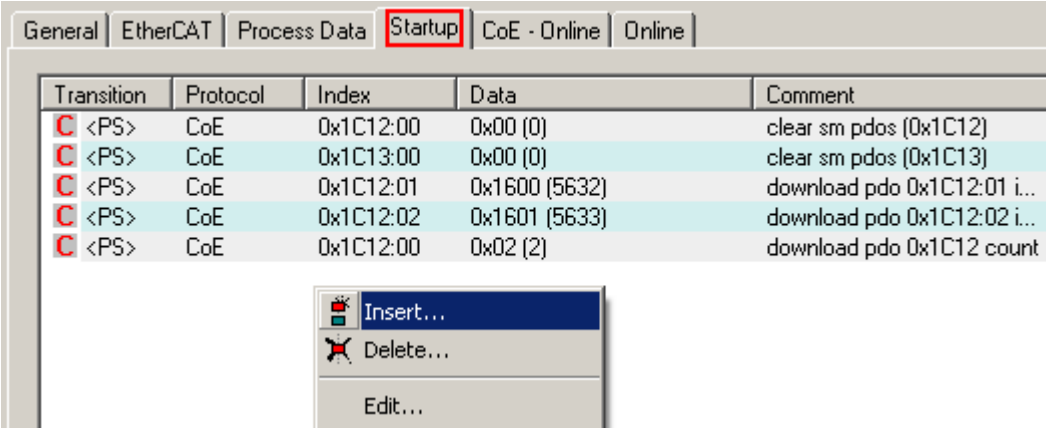


Fig. 18: Startup list in the TwinCAT System Manager

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

Online/offline list

While working with the TwinCAT System Manager, a distinction has to be made whether the EtherCAT device is “available”, i.e. switched on and linked via EtherCAT and therefore **online**, or whether a configuration is created **offline** without connected slaves.

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** is shown in red.

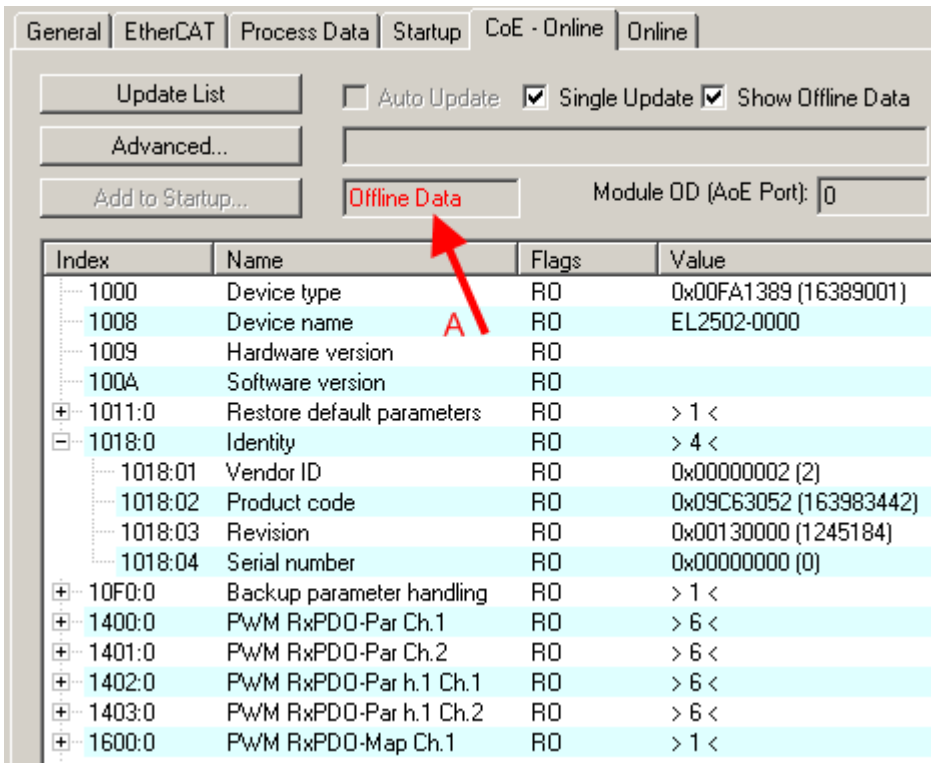


Fig. 19: Offline list

- 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 version of the equipment according to the electronic information is displayed
  - **Online** is shown in green.

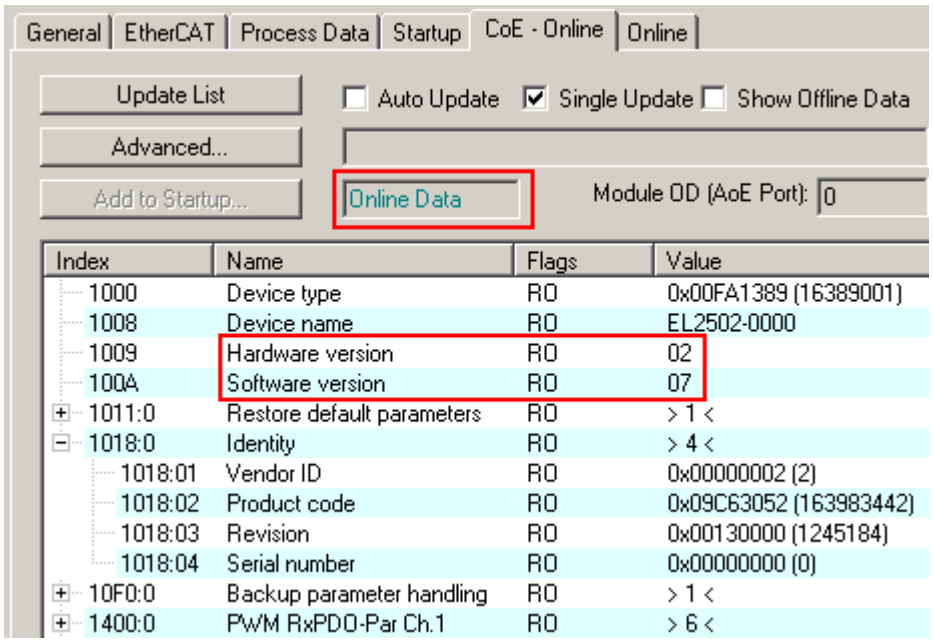


Fig. 20: Online list

Channel-based order

The CoE list is available in EtherCAT devices that usually feature several functionally equivalent channels. For example, a 4-channel analog 0...10 V 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 $16_{dec}/10_{hex}$ 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

NOTE

Destruction of the devices by electrostatic discharge possible!

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

- Please ensure you are electrostatically discharged and avoid touching the contacts of the device directly.
- Avoid contact with highly insulating materials (synthetic fibers, plastic film etc.).
- Surroundings (working place, packaging and personnel) should by grounded probably, when handling with the devices.
- Each assembly must be terminated at the right hand end with an EL9011 or EL9012 bus end cap, to ensure the protection class and ESD protection.

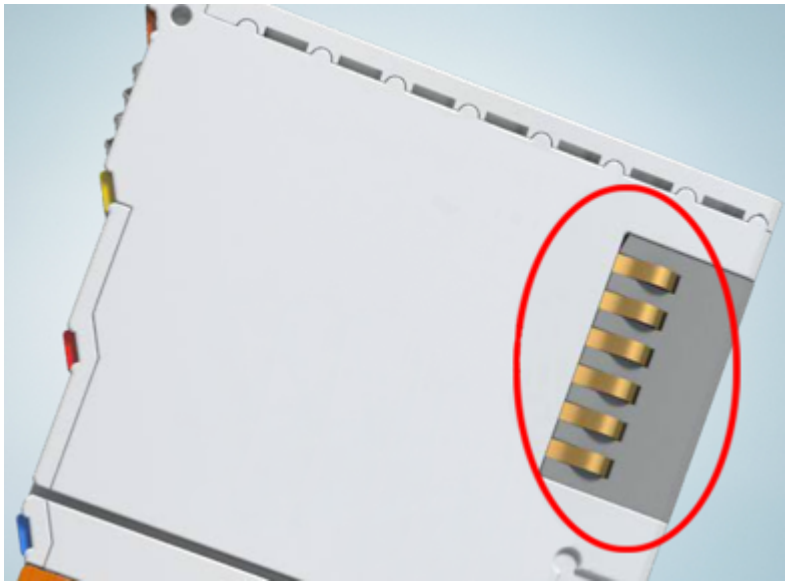


Fig. 21: Spring contacts of the Beckhoff I/O components

4.2 EL6001, EL6021

4.2.1 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!

Assembly

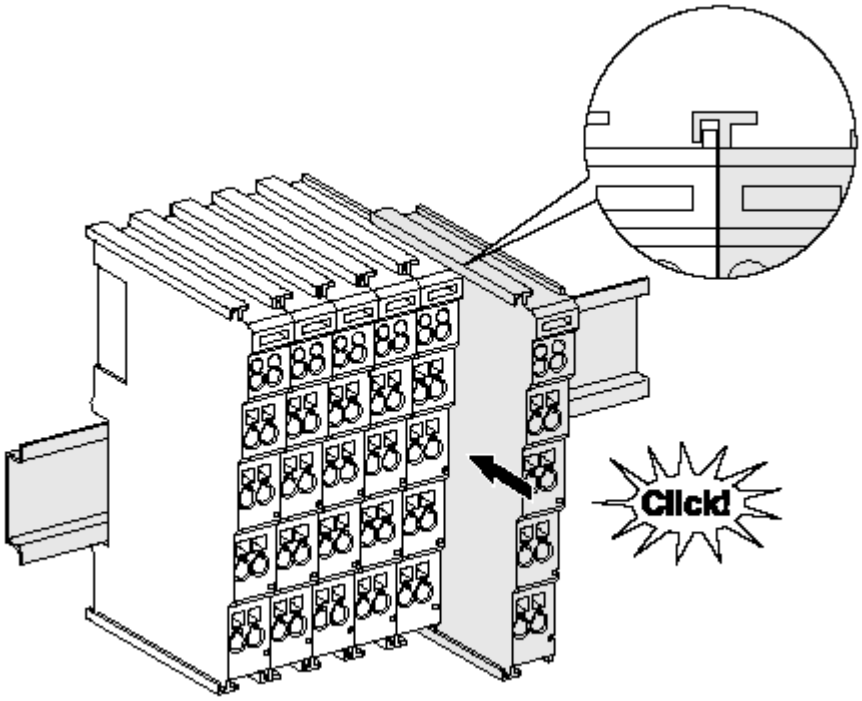


Fig. 22: Attaching on mounting rail

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

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

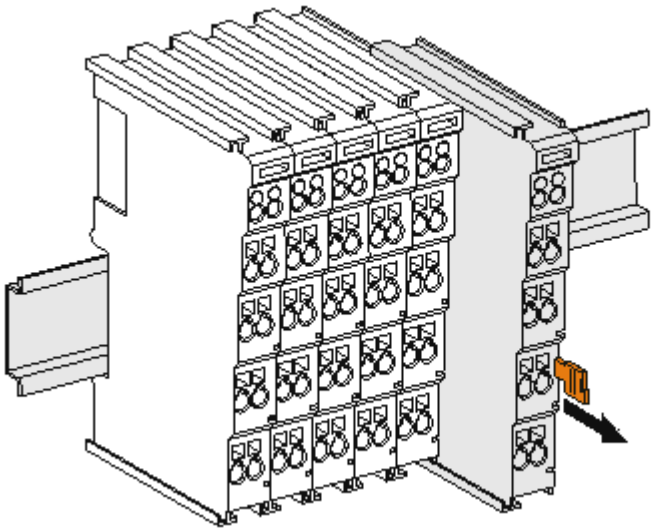


Fig. 23: Disassembling of terminal

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 on the Bus Coupler (up to 24 V) or for higher voltages via power feed terminals.

Power Contacts

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.

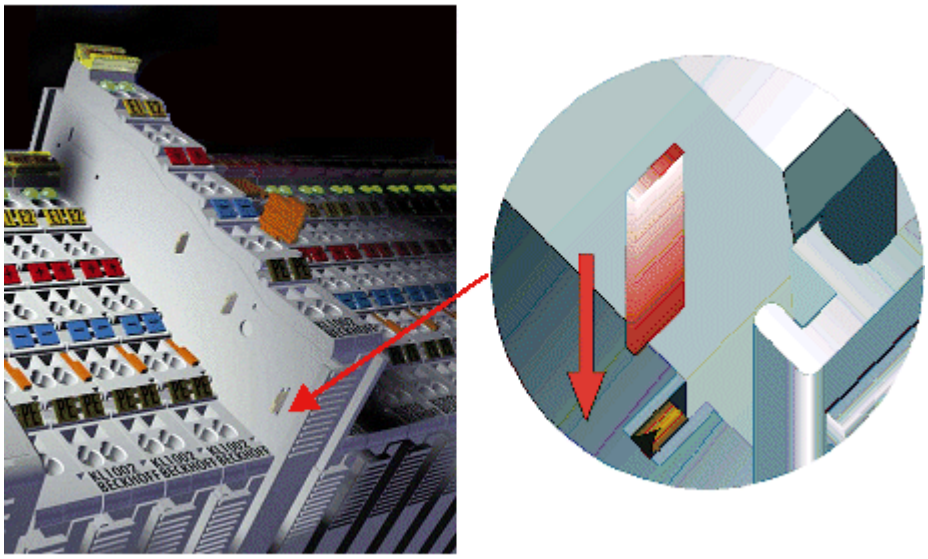


Fig. 24: Power contact on left side

NOTE

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!

4.2.2 Connection

4.2.2.1 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. 25: Standard wiring

The terminals of ELxxxx and KLxxxx series have been tried and tested for years.
They feature integrated screwless spring force technology for fast and simple assembly.

Pluggable wiring (ESxxxx / KSxxxx)

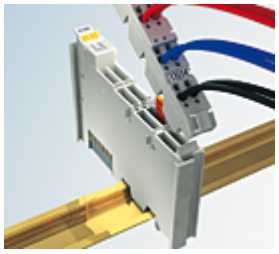


Fig. 26: Pluggable wiring

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^2 and 2.5 mm^2 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. 27: High Density Terminals

The terminals from these series with 16 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 “bonded” (ultrasonically welded) conductors

Ultrasonically “bonded” conductors

It is also possible to connect the Standard and High Density Terminals with ultrasonically “bonded” (ultrasonically welded) conductors. In this case, please note the tables concerning the wire-size width!

4.2.2.2 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

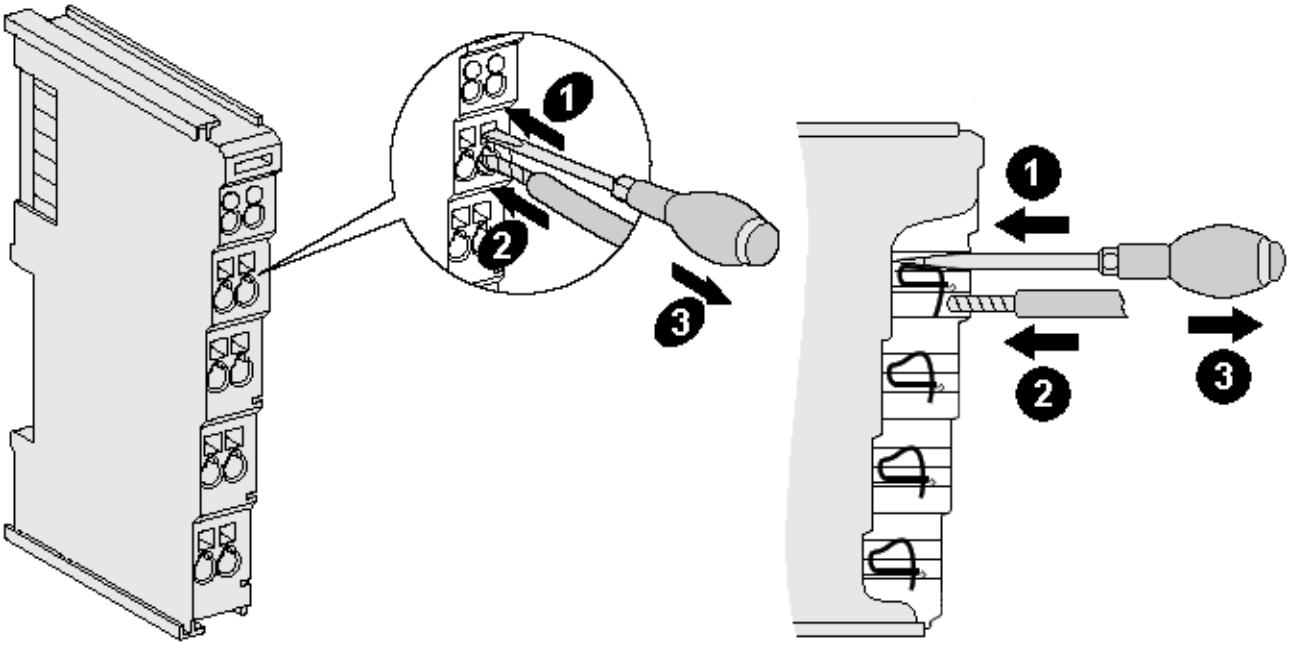


Fig. 28: Connecting a cable on a terminal point

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:

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. The terminal point closes automatically when the pressure is released, holding the wire securely and permanently.

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^2	0.08 ... 2.5 mm^2
Wire size width (fine-wire conductors)	0.08 ... 2.5 mm^2	0,08 ... 2.5 mm^2
Wire size width (conductors with a wire end sleeve)	0.14 ... 1.5 mm^2	0.14 ... 1.5 mm^2
Wire stripping length	8 ... 9 mm	9 ... 10 mm

High Density Terminals (HD Terminals [▶ 39]) with 16 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^2
Wire size width (fine-wire conductors)	0.25 ... 1.5 mm^2
Wire size width (conductors with a wire end sleeve)	0.14 ... 0.75 mm^2
Wire size width (ultrasonically “bonded" conductors)	only 1.5 mm^2
Wire stripping length	8 ... 9 mm

4.2.2.3 Shielding

Shielding

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

4.2.3 Positioning of passive Terminals

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)

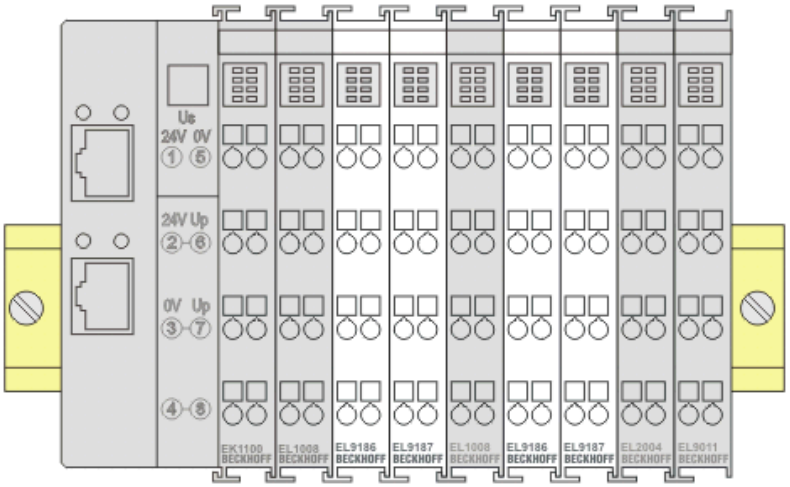


Fig. 29: Correct positioning

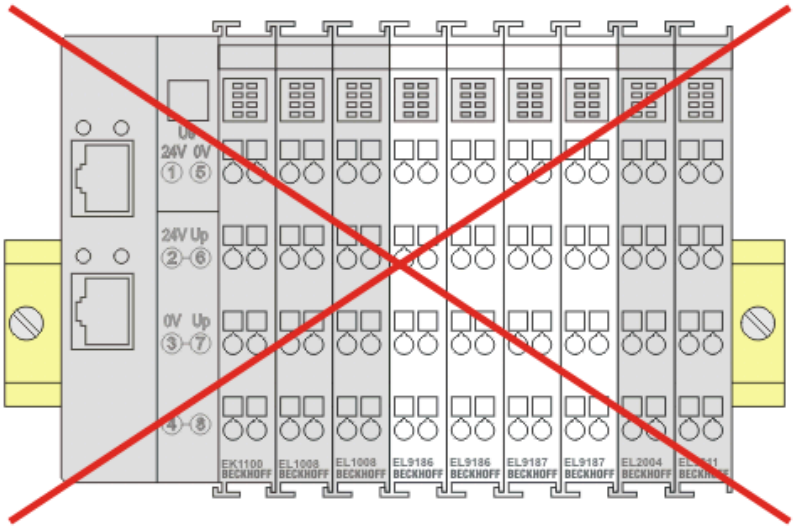


Fig. 30: Incorrect positioning

4.2.4 LEDs and terminal connector assignments

LEDs

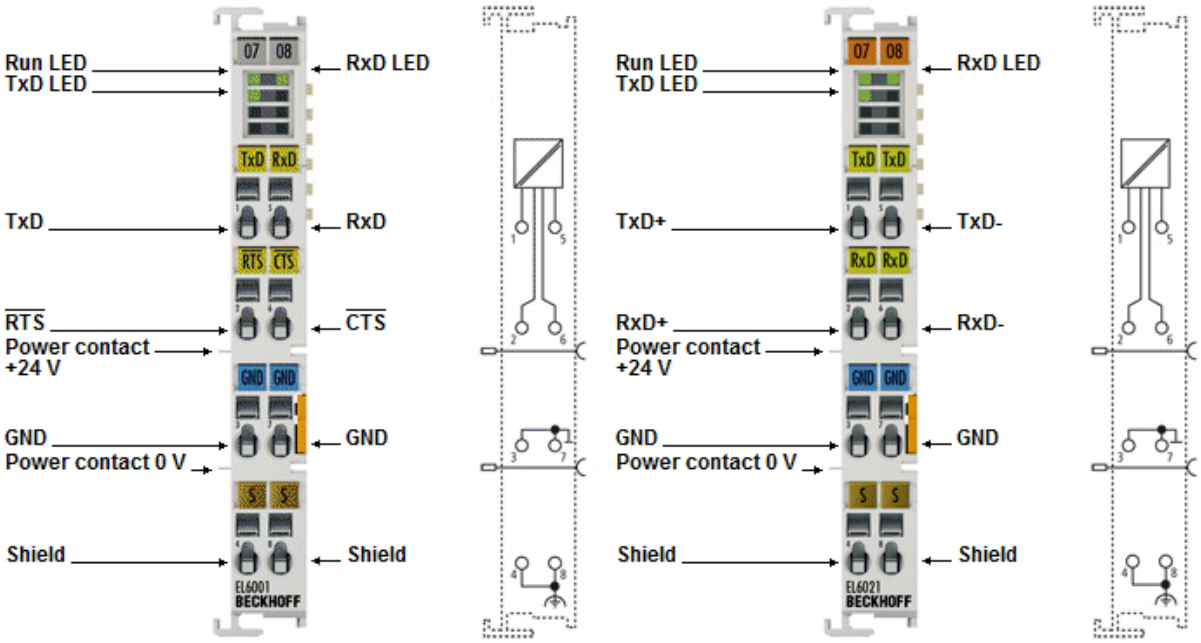


Fig. 31: EL6001, EL6021 - LEDs and Connection

LED	Color	Meaning	
RUN	Green	This LED indicates the terminal's operating state:	
		Off	State of the EtherCAT State Machine [▶ 27]: INIT = initialization of the terminal or BOOTSTRAP = function for firmware updates [▶ 191] of the terminal
		flashing	State of the EtherCAT state machine: PREOP = function for mailbox communication and variant standard settings
		Single flash	State of the EtherCAT state machine: SAFEOP = verification of the Sync Manager [▶ 110] channels and the distributed clocks. Outputs remain in safe state
		On	State of the EtherCAT State Machine: OP = normal operating state; mailbox and process data communication is possible
TxD	Green		State of the transmit signal line (on: HI signal level on transmit line)
RxD	Green		State of the receive signal line (on: HI signal level on receive line)

Connection

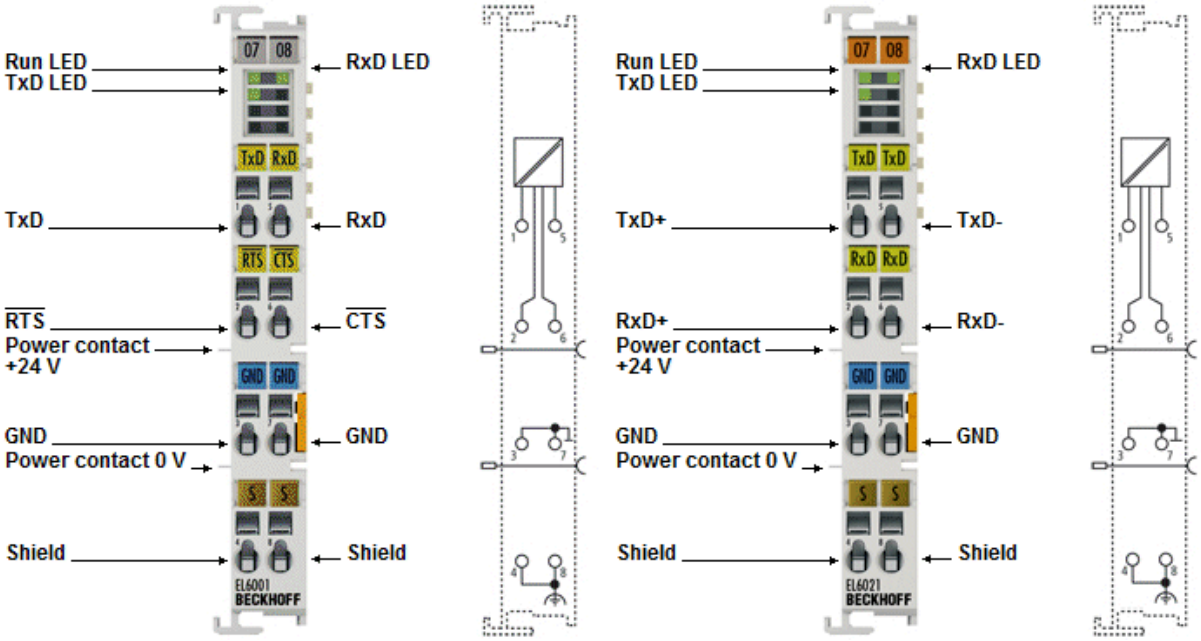


Fig. 32: EL6001, EL6021 - LEDs and Connection

EL6001 terminal connector assignments

Terminal point	Name	Signal
1	TxD	Signal line (Transmit Data)
5	RxD	Signal line (Receive Data)
2	RTS	Control line (Request To Send)
6	CTS	Control line (Clear To Send)
3	GND	Ground (internally bridged with terminal 7)
7	GND	Ground (internally bridged with terminal 3)
4	Shield	Shield (internally bridged with terminal 8)
8	Shield	Shield (internally bridged with terminal 4)

EL6021 terminal connector assignments

Terminal point	Name	Signal
1	TxD+	Signal line + (Transmit Data)
5	TxD-	Signal line - (Transmit Data)
2	RxD+	Signal line + (Receive Data)
6	RxD-	Signal line - (Receive Data)
3	GND	Ground (internally bridged with terminal 7)
7	GND	Ground (internally bridged with terminal 3)
4	Shield	Shield (internally bridged with terminal 8)
8	Shield	Shield (internally bridged with terminal 4)

Connection for RS422 transfer

In RS422 mode, data can be transferred in full duplex mode. Only point to point connections can be established.

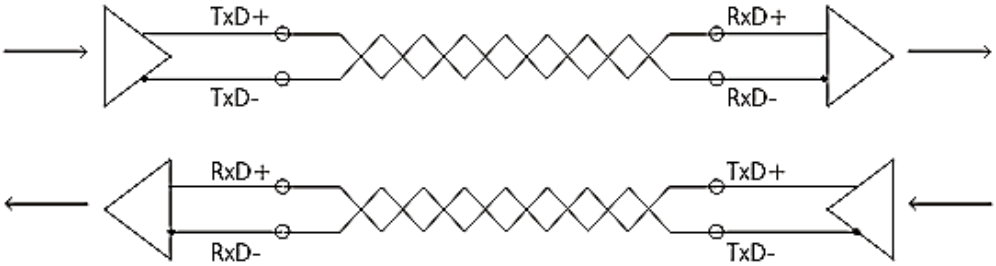


Fig. 33: Connection for RS422 transfer

Connection for RS485 transfer

In RS485 mode, data are exchanged in half duplex mode. A bus structure can be created in this mode of operation.

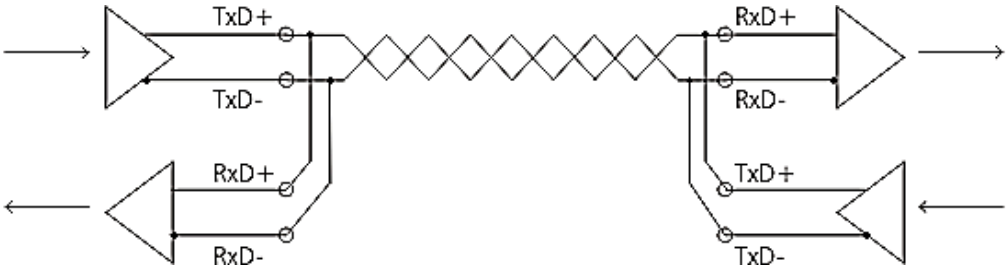


Fig. 34: Connection for RS485 transfer

The transmit and receive lines are connected to one another in RS485 operating mode. As a result, the terminal receives not only the data from other devices, but also its own transmitted data. This can be suppressed with the index 0x8000:06 “Enable half duplex” in the Settings object.

In operating mode RS485, the reception of new data is only possible if transmission is complete.

“Enable half duplex”	“Enable point to point connection (RS422)”	Mode
0	0	RS485: The terminal receives its own data and the data from other devices
0	1	RS422: Normal operating mode; the terminal operates in full duplex mode.
1	0	RS485: The terminal only receives data from other devices
1	1	RS422: The receiver is only enabled after the last data has been transmitted.

4.3 EL6002, EL6022

4.3.1 Mounting and demounting - terminals with front unlocking

The terminal modules are fastened to the assembly surface with the aid of a 35 mm mounting rail (e.g. mounting rail TH 35-15).

Fixing of mounting rails

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 recommended mounting rails under the terminals and couplers, you should use flat mounting connections (e.g. countersunk screws or blind rivets).

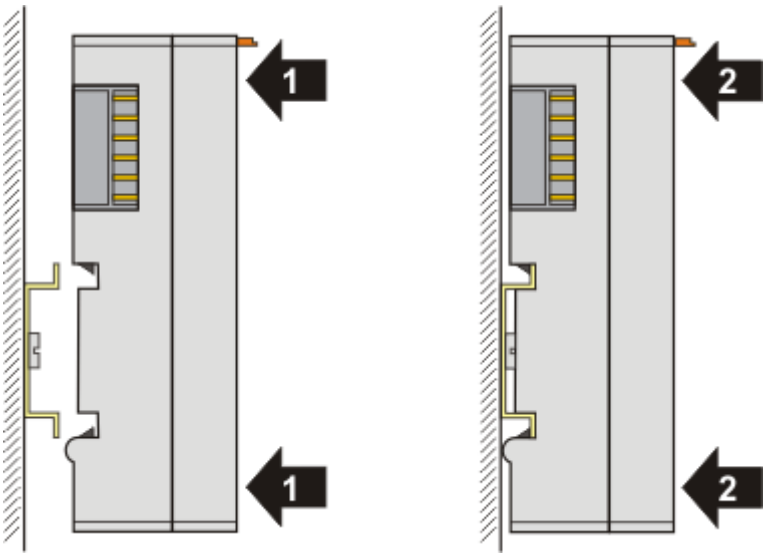
⚠ WARNING

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!

Mounting

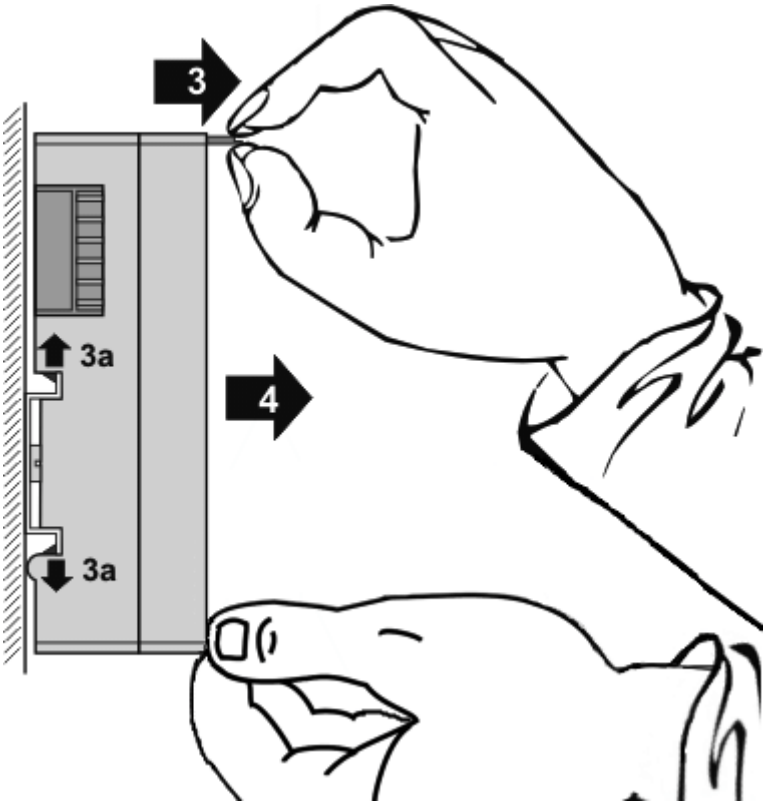
- Fit the mounting rail to the planned assembly location.



  and press (1) the terminal module against the mounting rail until it latches in place on the mounting rail (2).
- Attach the cables.

Demounting

- Remove all the cables.
- Lever the unlatching hook back with thumb and forefinger (3). An internal mechanism pulls the two latching lugs (3a) from the top hat rail back into the terminal module.



- Pull (4) the terminal module away from the mounting surface.
  Avoid canting of the module; you should stabilize the module with the other hand, if required.

4.3.2 Recommended mounting rails

Terminal Modules und EtherCAT Modules of KMxxxx and EMxxxx series, same as the terminals of the EL66xx and EL67xx series can be snapped onto the following recommended mounting rails:

DIN Rail TH 35-7.5 with 1 mm material thickness (according to EN 60715)

DIN Rail TH 35-15 with 1,5 mm material thickness

Pay attention to the material thickness of the DIN Rail

Terminal Modules und EtherCAT Modules of KMxxxx and EMxxxx series, same as the terminals of the EL66xx and EL67xx series does not fit to the DIN Rail TH 35-15 with 2,2 to 2,5 mm material thickness (according to EN 60715)!

4.3.3 LEDs and pin assignment

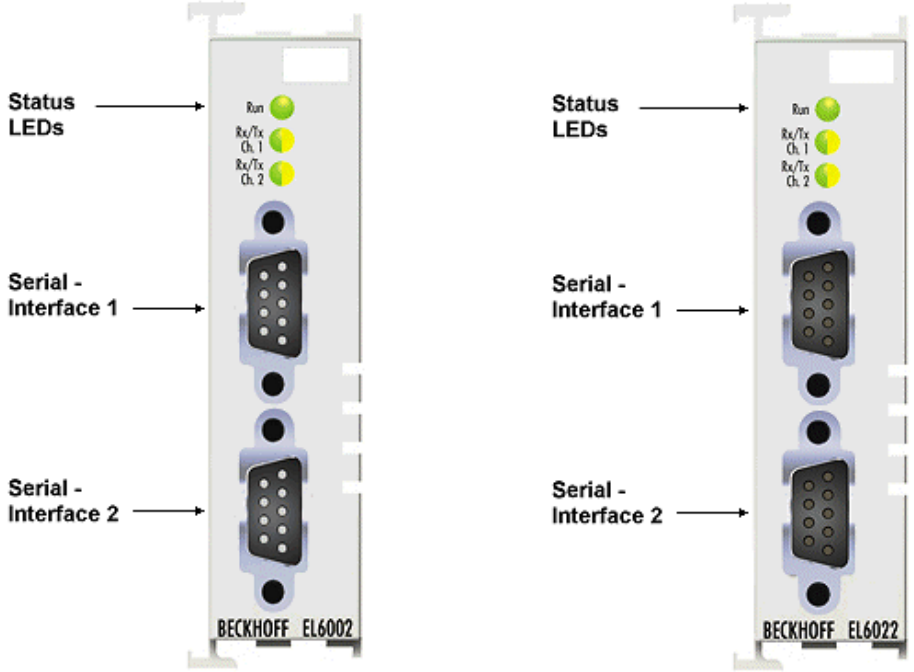


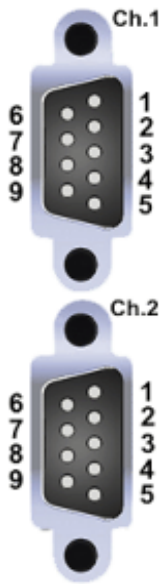
Fig. 35: EL6002, EL6022 - LEDs

LEDs

LED	Color	Meaning	
RUN	Green	This LED indicates the terminal's operating state:	
		Off	State of the EtherCAT State Machine [▶ 27]: INIT = initialization of the terminal or BOOTSTRAP = function for firmware updates [▶ 191] of the terminal
		flashing	State of the EtherCAT State Machine: PREOP = function for mailbox communication and variant standard settings
		Single flash	State of the EtherCAT State Machine: SAFEOP = verification of the Sync Manager [▶ 110] channels and the distributed clocks. Outputs remain in safe state
		On	State of the EtherCAT State Machine: OP = normal operating state; mailbox and process data communication is possible
TxCh. 1	Orange		Serial port at this connection sends data (channel 1)
RxCh. 1	Green		Serial port at this connection receives data (channel 1)
TxCh. 2	Orange		Serial port at this connection sends data (channel 2)
RxCh. 2	Green		Serial port at this connection receives data (channel 2)

EL6002 pin assignment

2 x D-Sub connector, male; 9-pin

D-Sub connector, male (plan view)	Pin assignment channel 1		Pin assignment channel 2	
	**Pin**	**RS232**	**Pin**	**RS232**
	1	DCD (internally bridged with pins 4 and 6)	1	DCD (internally bridged with pins 4 and 6)
	2	RxCH1	2	RxCH2
	3	TxCH1	3	TxCH2
	4	DTR (internally bridged with pins 1 and 6)	4	DTR (internally bridged with pins 1 and 6)
	5	GND	5	GND
	6	DSR (internally bridged with pins 1 and 4)	6	DSR (internally bridged with pins 1 and 4)
	7	RTS CH1	7	RTS CH2
	8	CTS CH1	8	CTS CH2
	9	-	9	-

GND connections

GND for both channels is internally connected via a high-resistance RC combination

EL6022 pin assignment

2 x D-Sub connection socket, 9-pin

D-Sub connector, female (plan view)	Pin assignment channel 1		Pin assignment channel 2	
	Pin	**RS485/RS422**	**Pin**	**RS485/RS422**
	1	-	1	-
	2	Tx+ CH1	2	Tx+ CH2
	3	Rx+ CH1	3	Rx+ CH2
	4	-	4	-
	5	GND	5	GND
	6	+5 V	6	+5 V
	7	Tx- CH1	7	Tx- CH2
	8	Rx- CH1	8	Rx- CH2
	9	-	9	-

GND connections

GND for both channels is internally connected via a high-resistance RC combination

Connection for RS422 transfer

In RS422 mode, data can be transferred in full duplex mode. Only point to point connections can be established.

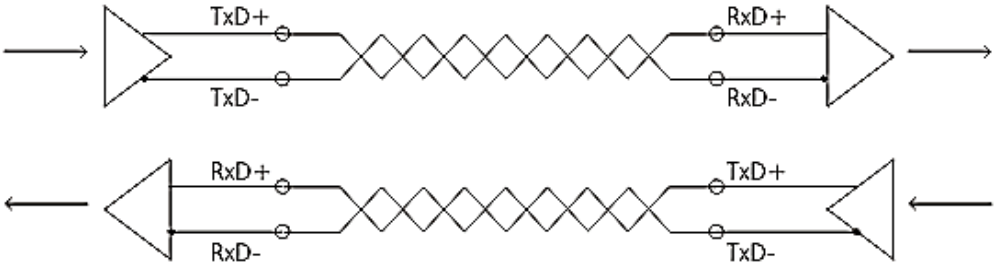


Fig. 36: Connection for RS422 transfer

Connection for RS485 transfer

In RS485 mode, data are exchanged in half duplex mode. A bus structure can be created in this mode of operation.

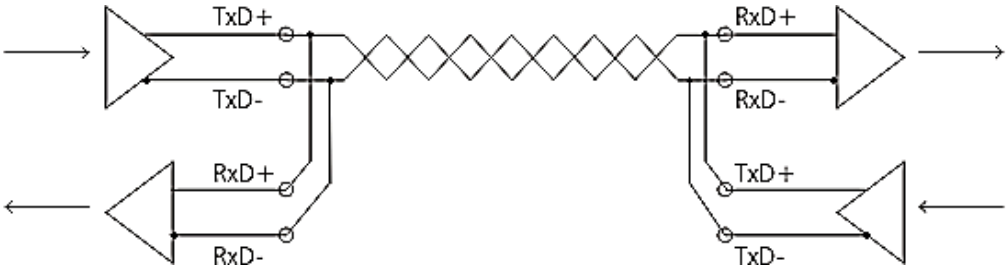


Fig. 37: Connection for RS485 transfer

The transmit and receive lines are connected to one another in RS485 operating mode. As a result, the terminal receives not only the data from other devices, but also its own transmitted data. This can be suppressed with the index 0x8000:06 “Enable half duplex” in the Settings object.

In operating mode RS485, the reception of new data is only possible if transmission is complete.

“Enable half duplex”	“Enable point to point connection (RS422)”	Mode
0	0	RS485: The terminal receives its own data and the data from other devices
0	1	RS422: Normal operating mode; the terminal operates in full duplex mode.
1	0	RS485: The terminal only receives data from other devices
1	1	RS422: The receiver is only enabled after the last data has been transmitted.

4.4 Positioning of passive Terminals

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)

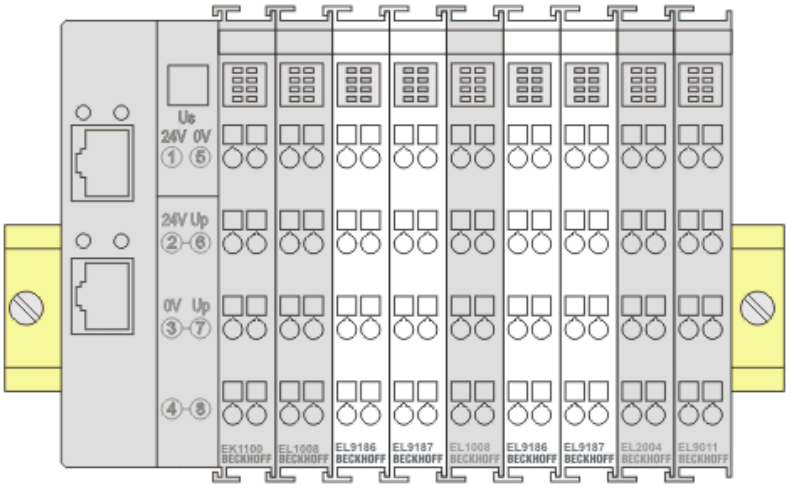


Fig. 38: Correct positioning

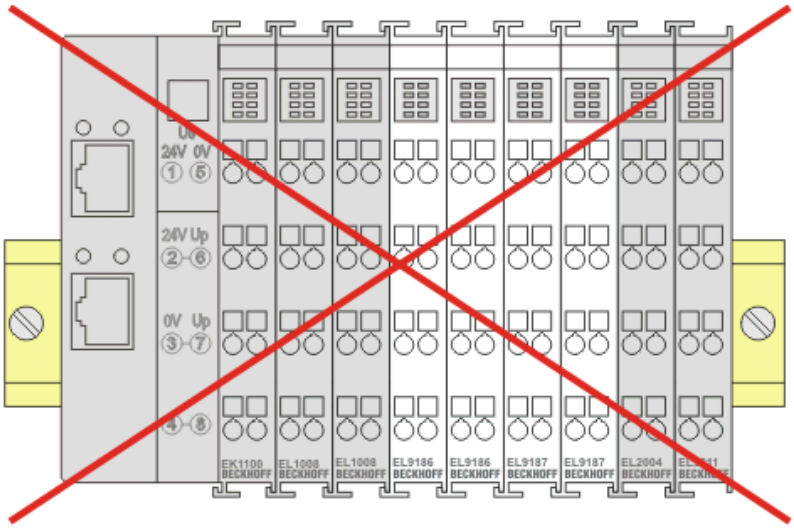


Fig. 39: Incorrect positioning

4.5 Installation instructions for enhanced mechanical load capacity

⚠ WARNING

Risk of injury through electric shock and damage to the device!

Bring the Bus Terminal system into a safe, de-energized state before starting mounting, disassembly or wiring of the Bus Terminals!

Additional checks

The terminals have undergone the following additional tests:

Verification	Explanation
Vibration	10 frequency runs in 3 axes
	6 Hz < f < 60 Hz displacement 0.35 mm, constant amplitude
	60.1 Hz < f < 500 Hz acceleration 5 *g*, constant amplitude
Shocks	1000 shocks in each direction, in 3 axes
	25 *g*, 6 ms

Additional installation instructions

For terminals with enhanced mechanical load capacity, the following additional installation instructions apply:

- The enhanced mechanical load capacity is valid for all permissible installation positions
- Use a mounting rail according to EN 60715 TH35-15
- Fix the terminal segment on both sides of the mounting rail with a mechanical fixture, e.g. an earth terminal or reinforced end clamp
- The maximum total extension of the terminal segment (without coupler) is:
  64 terminals (12 mm mounting with) or 32 terminals (24 mm mounting with)
- Avoid deformation, twisting, crushing and bending of the mounting rail during edging and installation of the rail
- The mounting points of the mounting rail must be set at 5 cm intervals
- Use countersunk head screws to fasten the mounting rail
- The free length between the strain relief and the wire connection should be kept as short as possible. A distance of approx. 10 cm should be maintained to the cable duct.

4.6 Installation positions

NOTE

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.

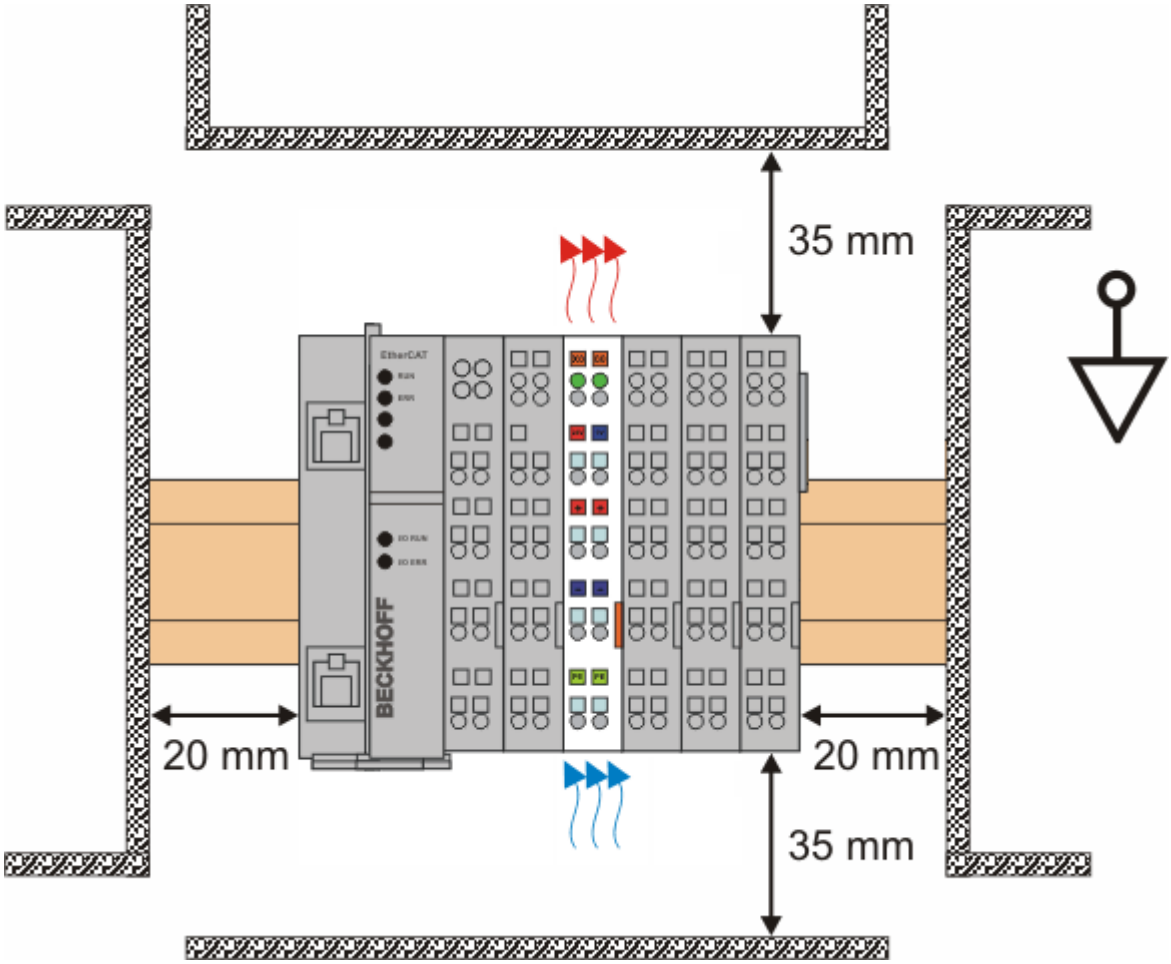


Fig. 40: Recommended distances for standard installation position

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.

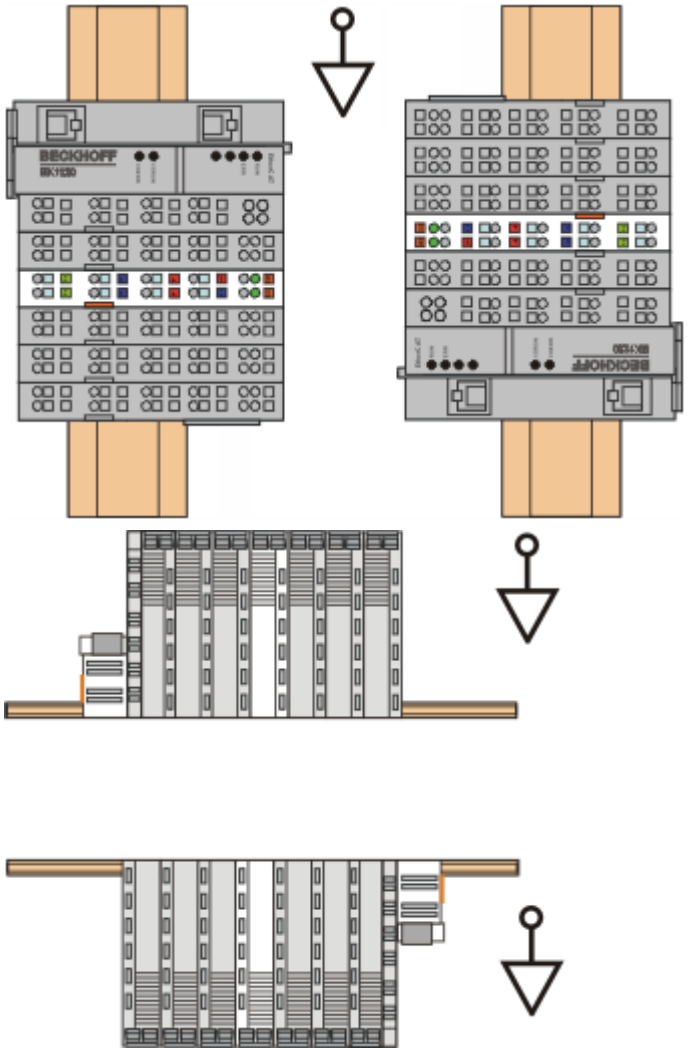


Fig. 41: Other installation positions

4.7 UL notice

	Application Beckhoff EtherCAT modules are intended for use with Beckhoff's UL Listed EtherCAT System only.
	Examination For cULus examination, the Beckhoff I/O System has only been investigated for risk of fire and electrical shock (in accordance with UL508 and CSA C22.2 No. 142).
	For devices with Ethernet connectors Not for connection to telecommunication circuits.

Basic principles

UL certification according to UL508. Devices with this kind of certification are marked by this sign:

4.8 ATEX - Special conditions (extended temperature range)

⚠ WARNING

Observe the special conditions for the intended use of Beckhoff fieldbus components with extended temperature range (ET) in potentially explosive areas (directive 2014/34/EU)!

- The certified components are to be installed in a suitable housing that guarantees a protection class of at least IP54 in accordance with EN 60079-15! The environmental conditions during use are thereby to be taken into account!
- For dust (only the fieldbus components of certificate no. KEMA 10ATEX0075 X Issue 9): The equipment shall be installed in a suitable enclosure providing a degree of protection of IP54 according to EN 60079-0 for group IIIA or IIIB and IP6X for group IIIC, taking into account the environmental conditions under which the equipment is used.
- If the temperatures during rated operation are higher than 70°C at the feed-in points of cables, lines or pipes, or higher than 80°C at the wire branching points, then cables must be selected whose temperature data correspond to the actual measured temperature values!
- Observe the permissible ambient temperature range of -25 to 60°C for the use of Beckhoff fieldbus components with extended temperature range (ET) in potentially explosive areas!
- Measures must be taken to protect against the rated operating voltage being exceeded by more than 40% due to short-term interference voltages!
- The individual terminals may only be unplugged or removed from the Bus Terminal system if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- The connections of the certified components may only be connected or disconnected if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- The fuses of the KL92xx/EL92xx power feed terminals may only be exchanged if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- Address selectors and ID switches may only be adjusted if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!

Standards

The fundamental health and safety requirements are fulfilled by compliance with the following standards:

- EN 60079-0:2012+A11:2013
- EN 60079-15:2010
- EN 60079-31:2013 (only for certificate no. KEMA 10ATEX0075 X Issue 9)

Marking

The Beckhoff fieldbus components with extended temperature range (ET) certified according to the ATEX directive for potentially explosive areas bear the following marking:

II 3G KEMA 10ATEX0075 X Ex nA IIC T4 Gc Ta: -25 … +60°C

II 3D KEMA 10ATEX0075 X Ex tc IIC T135°C Dc Ta: -25 ... +60°C
(only for fieldbus components of certificate no. KEMA 10ATEX0075 X Issue 9)

or

II 3G KEMA 10ATEX0075 X Ex nC IIC T4 Gc Ta: -25 … +60°C

II 3D KEMA 10ATEX0075 X Ex tc IIC T135°C Dc Ta: -25 ... +60°C
(only for fieldbus components of certificate no. KEMA 10ATEX0075 X Issue 9)

4.9 Continuative documentation about explosion protection

Explosion protection for terminal systems

Pay also attention to the continuative documentation

Notes on the use of the Beckhoff terminal systems in hazardous areas according to ATEX and IECEx

that is available for download on the Beckhoff homepage https:\\www.beckhoff.com!

5 Commissioning

5.1 TwinCAT Quick Start

TwinCAT is a development environment for real-time control including 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 Systemmanual:**
  Fieldbus Components → EtherCAT Terminals → EtherCAT System Documentation → Setup in the TwinCAT System Manager
- **TwinCAT 2** → TwinCAT System Manager → I/O - Configuration
- In particular, TwinCAT driver installation:
  Fieldbus components → Fieldbus Cards and Switches → FC900x – PCI Cards for Ethernet → Installation

Devices contain the 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 offline mode can be found under http://infosys.beckhoff.com:
    TwinCAT 2 → TwinCAT System Manager → IO - Configuration → Adding 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 from user PC to the individual control elements:

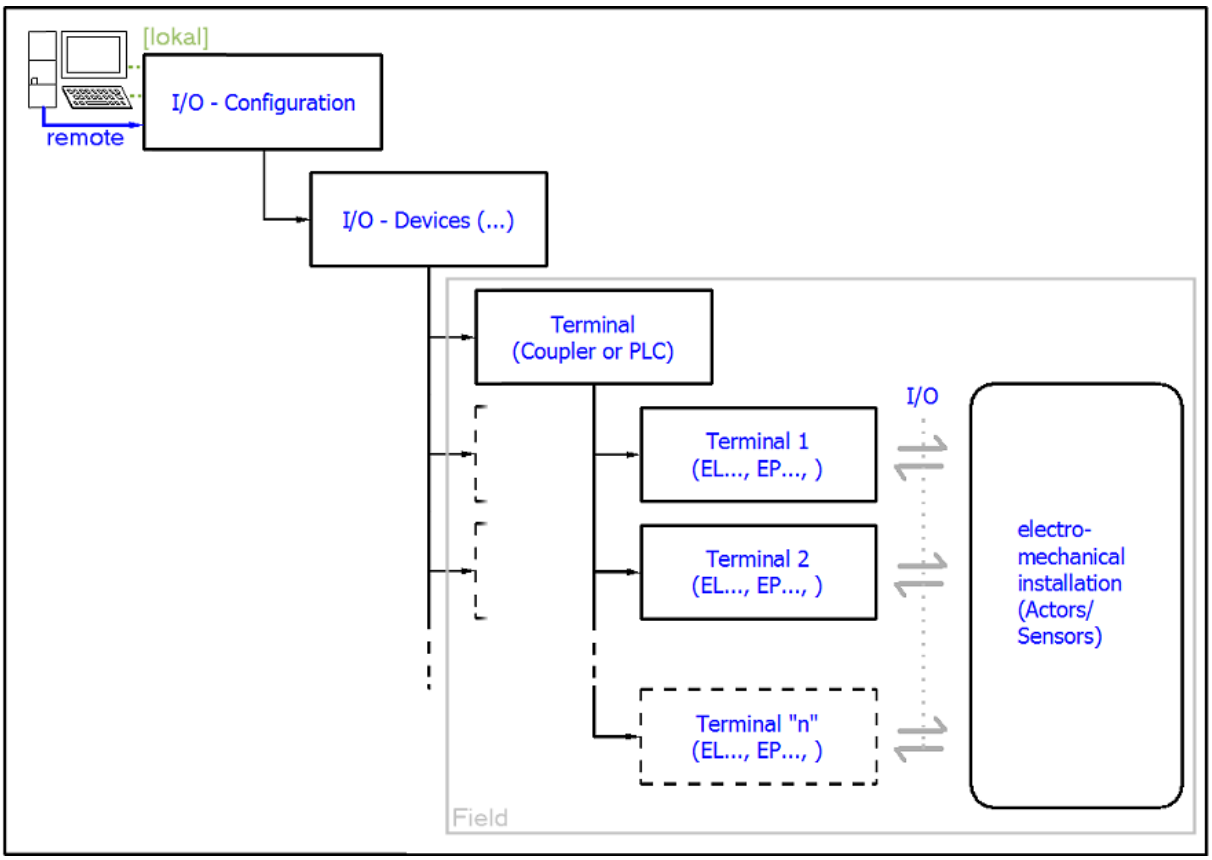


Fig. 42: Relationship between user side (commissioning) and installation

The user inserting of certain components (I/O device, terminal, box...) is the same in TwinCAT 2 and TwinCAT 3. The descriptions below relate to the online procedure.

Sample configuration (actual configuration)

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

- Control system (PLC) **CX2040** including **CX2100-0004** power supply unit
- Connected to the CX2040 on the right (E-bus):
  EL1004 (4-channel digital input terminal 24 V_{DC})
- 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_{DC}; 0.5 A)
- (Optional via X000: a link to an external PC for the user interface)

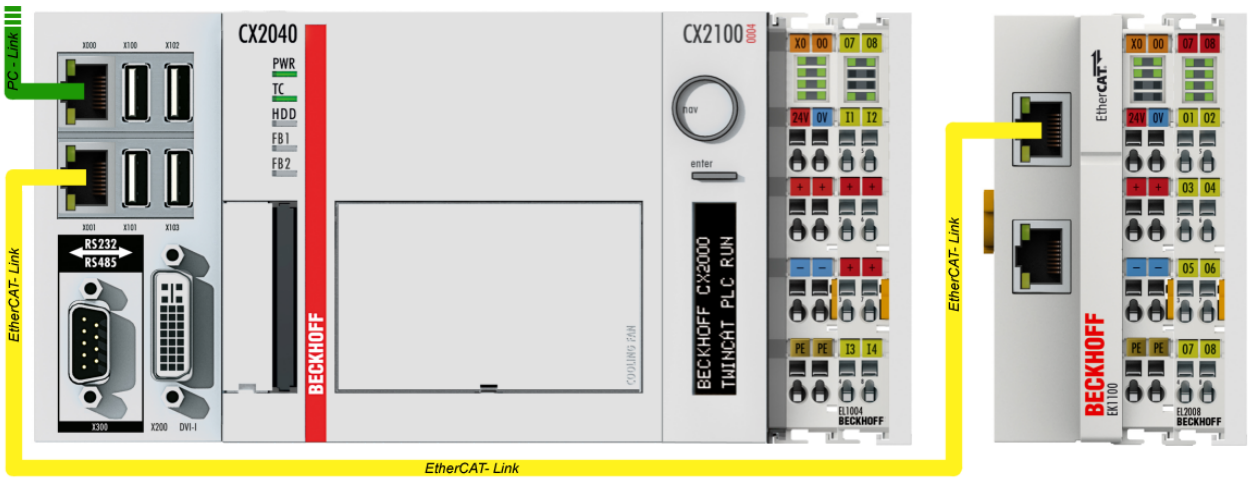


Fig. 43: Control configuration with Embedded PC, input (EL1004) and output (EL2008)

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.

5.1.1 TwinCAT 2

Startup

TwinCAT 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:

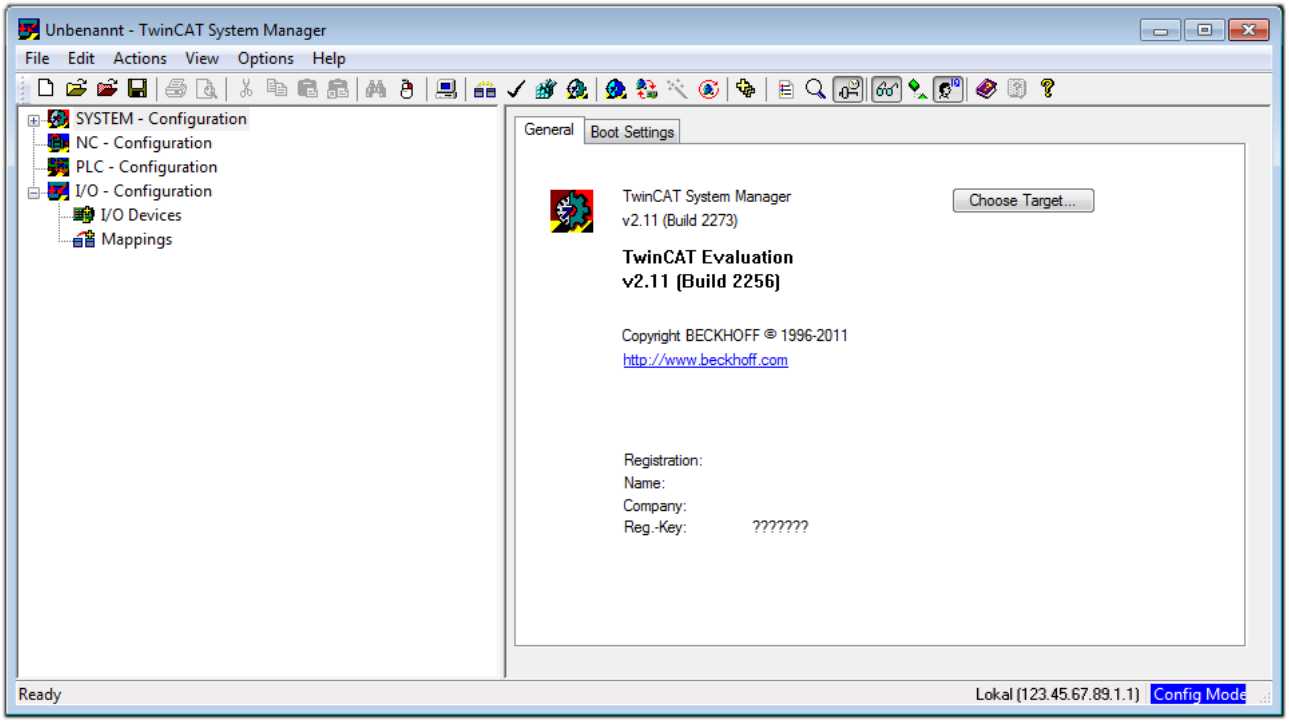


Fig. 44: Initial TwinCAT 2 user interface

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 thereby the next step is "Insert Device [▶ 64]".

If the intention is to address the TwinCAT runtime environment installed on a PLC as development environment remotely from another system, the target system must be made known first. In the menu under "Actions" → "Choose Target System...", via the symbol " " or the "F8" key, open the following window:

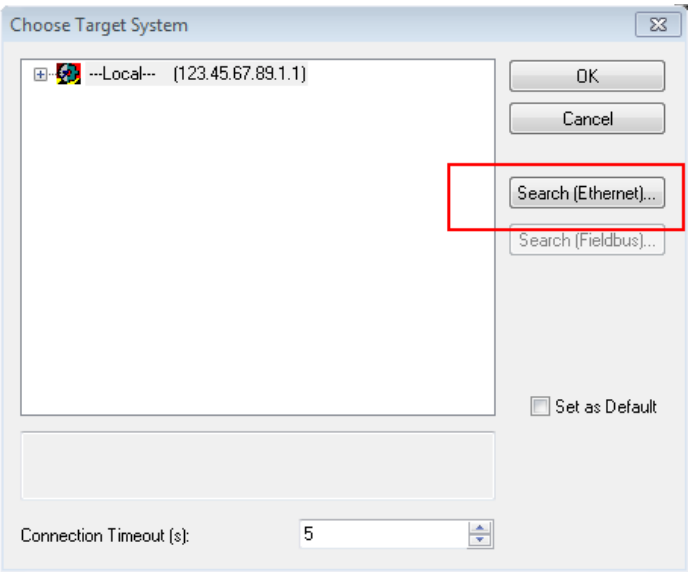


Fig. 45: Selection of the target system

Use “Search (Ethernet)...” to enter the target system. Thus a next 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.

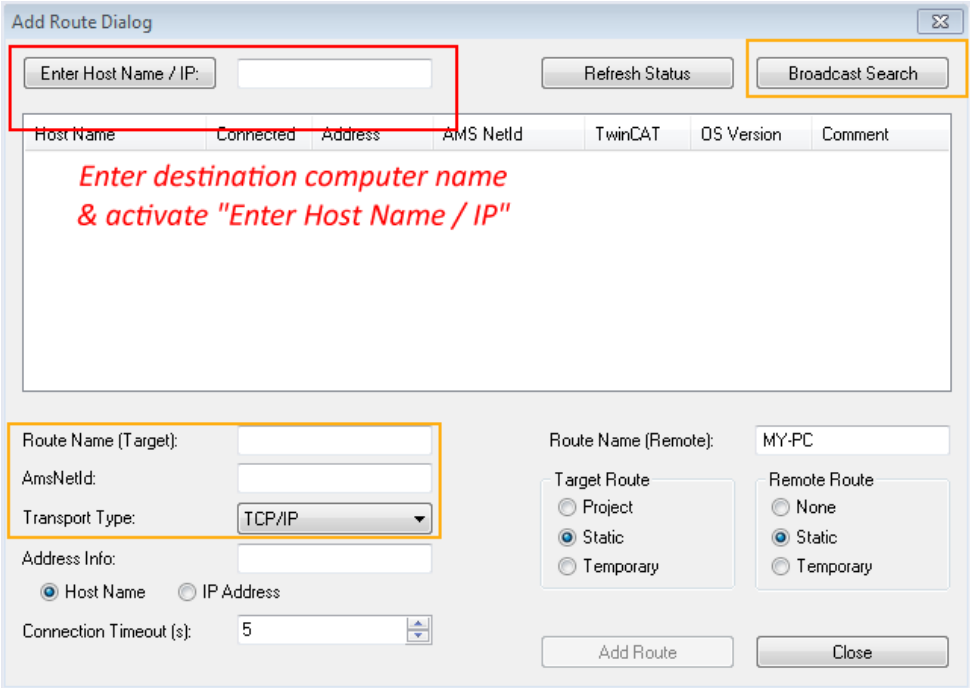
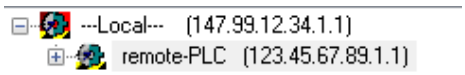


Fig. 46: Specify the PLC for access by the TwinCAT System Manager: selection of the target system

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



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 menu “Actions” → “Set/Reset TwinCAT to Config Mode…” (Shift + F4).

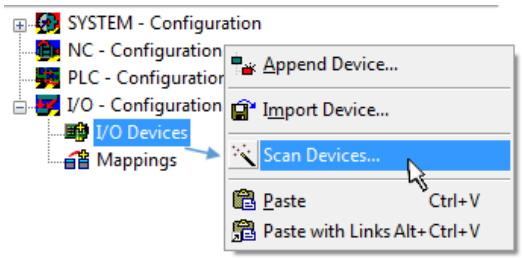


Fig. 47: Select “Scan Devices...”

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

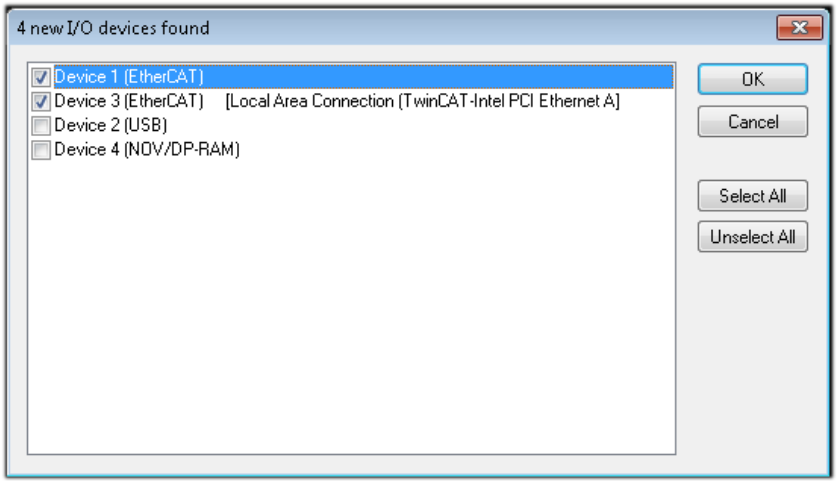


Fig. 48: Automatic detection of I/O devices: selection the devices to be integrated

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 sample configuration [▶ 60] described at the beginning of this section, the result is as follows:

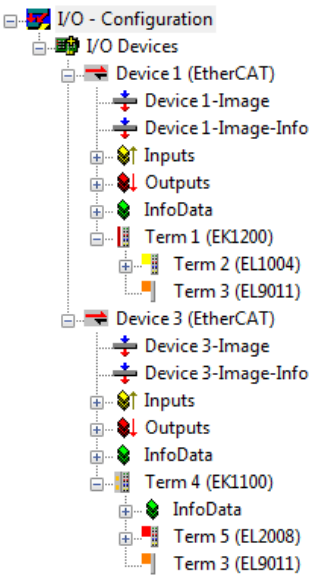


Fig. 49: Mapping of the configuration in the TwinCAT 2 System Manager

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

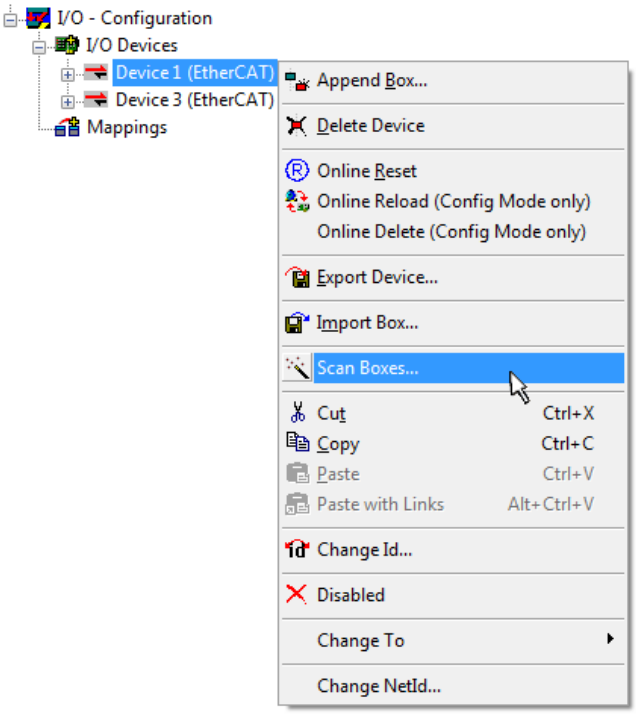


Fig. 50: Reading of individual terminals connected to a device

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 the creation of 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 to Structured Text (ST).

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

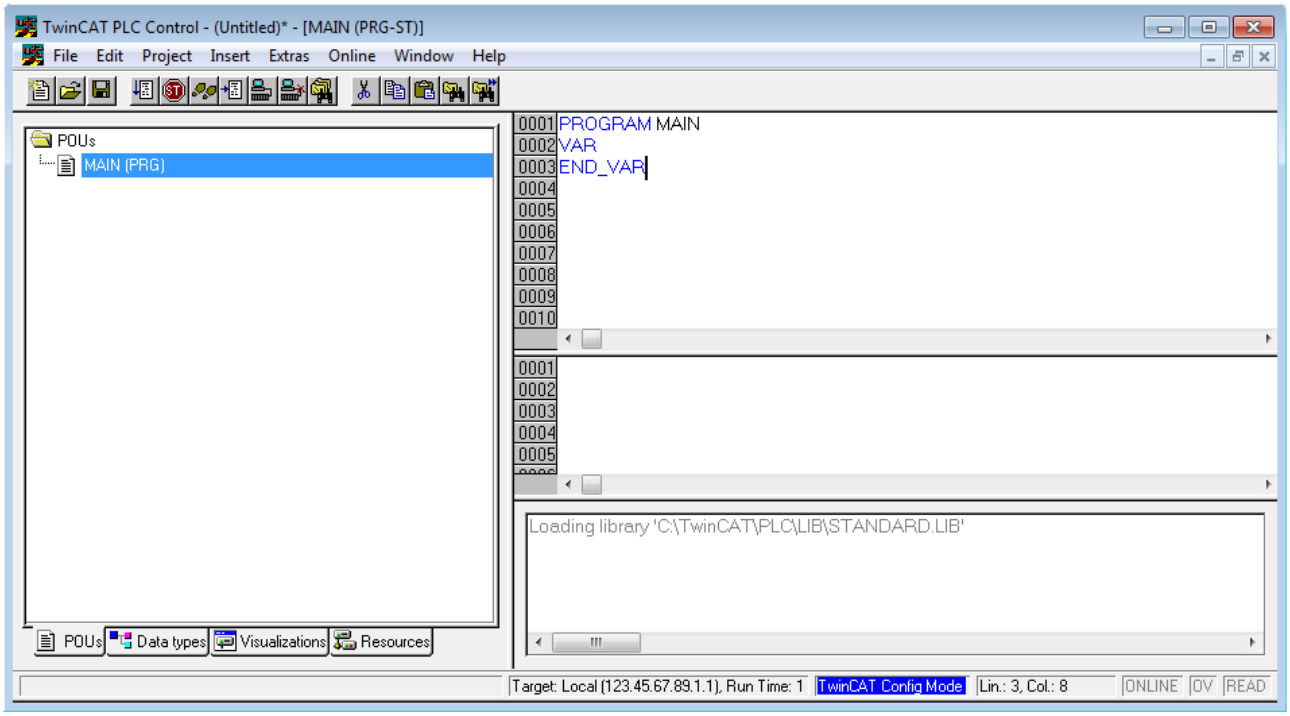


Fig. 51: TwinCAT PLC Control after startup

Sample variables and a sample program have been created and stored under the name “PLC_example.pro”:

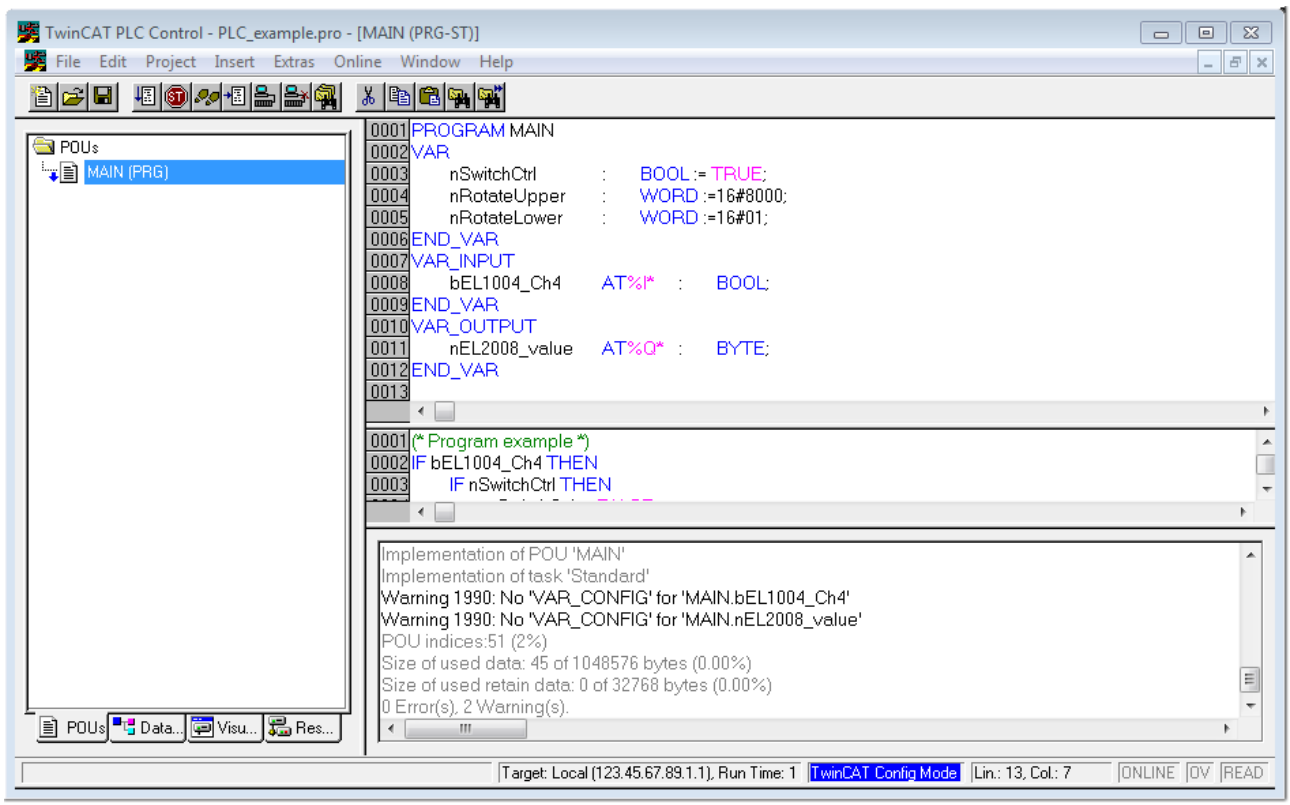


Fig. 52: Sample program with variables after a compile process (without variable integration)

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** via the context menu of the PLC configuration; right-click and select “Append PLC Project…”:

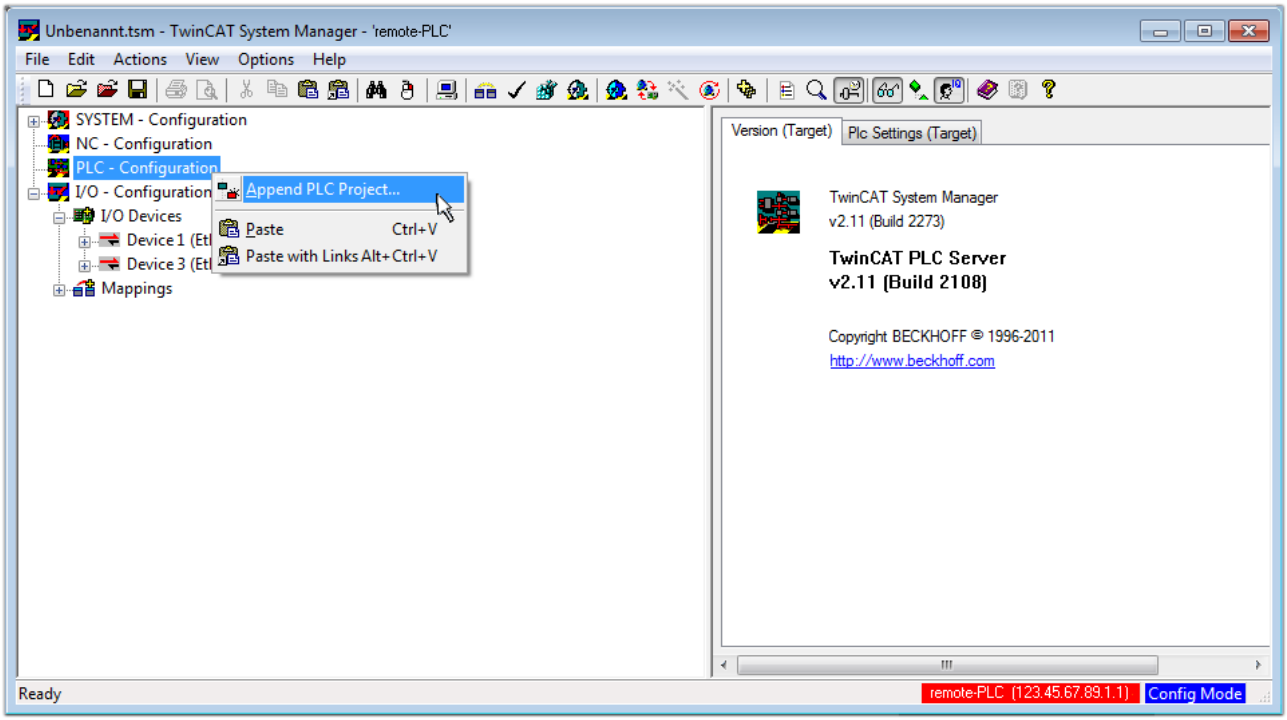


Fig. 53: Appending the TwinCAT PLC Control project

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:

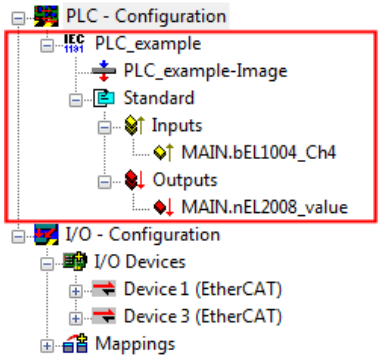


Fig. 54: PLC project integrated in the PLC configuration of the System Manager

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”:

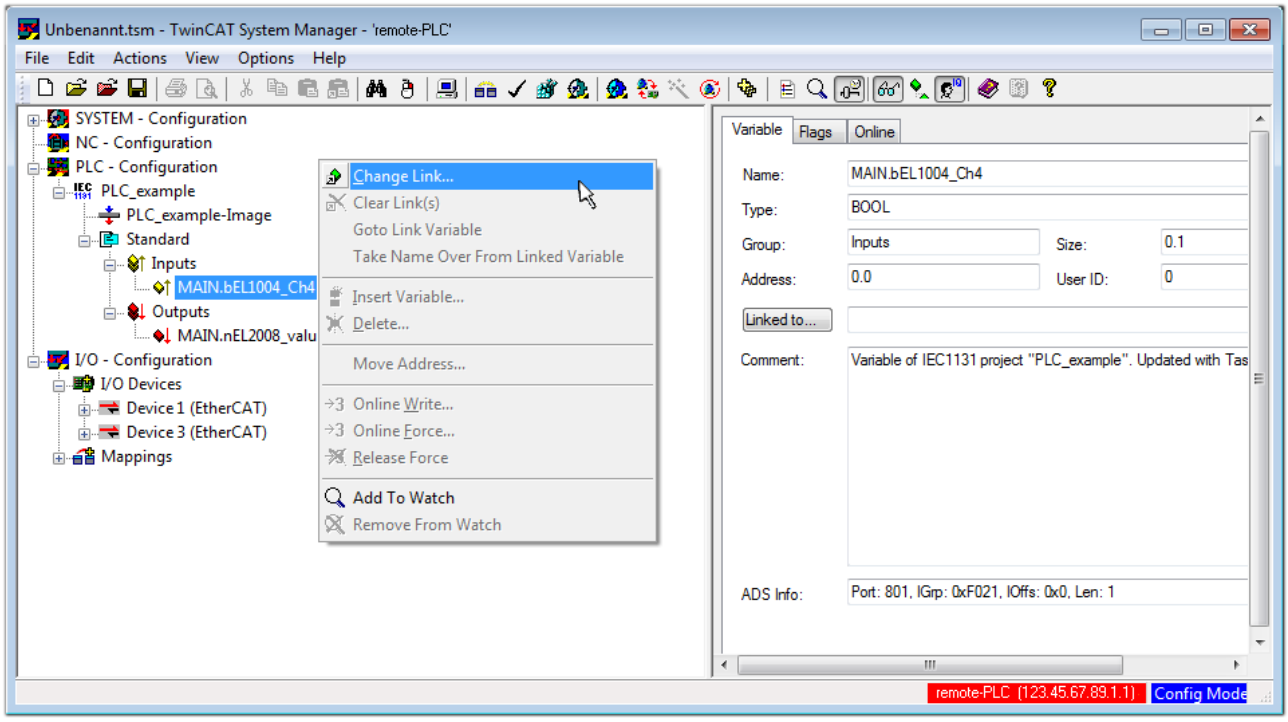


Fig. 55: Creating the links between PLC variables and process objects

In the window that opens, the process object for the variable “bEL1004_Ch4” of type BOOL can be selected from the PLC configuration tree:

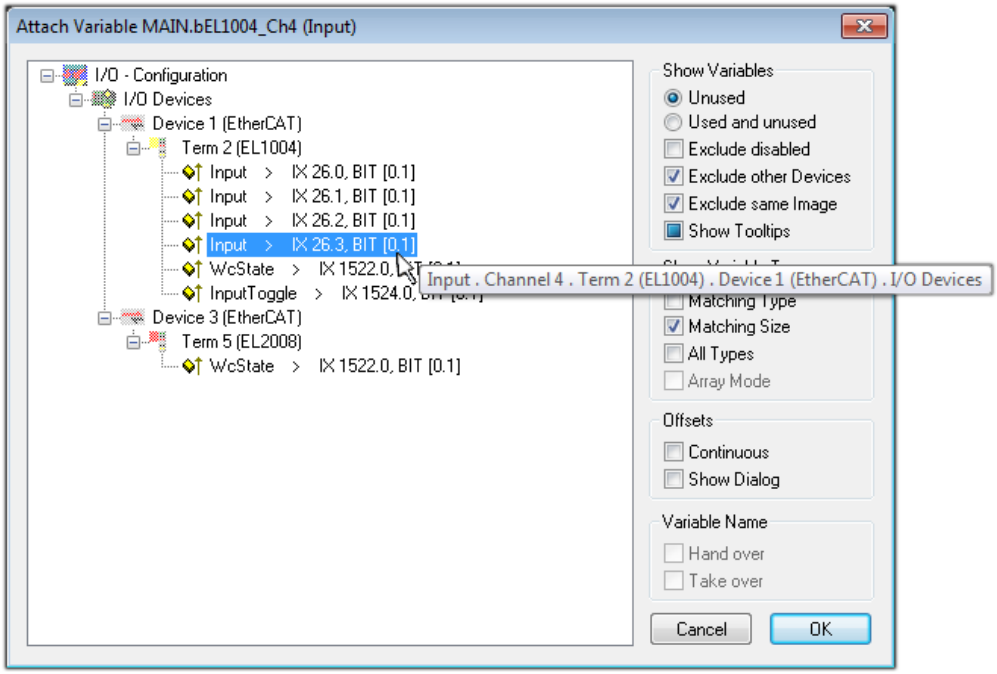


Fig. 56: Selecting PDO of type BOOL

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

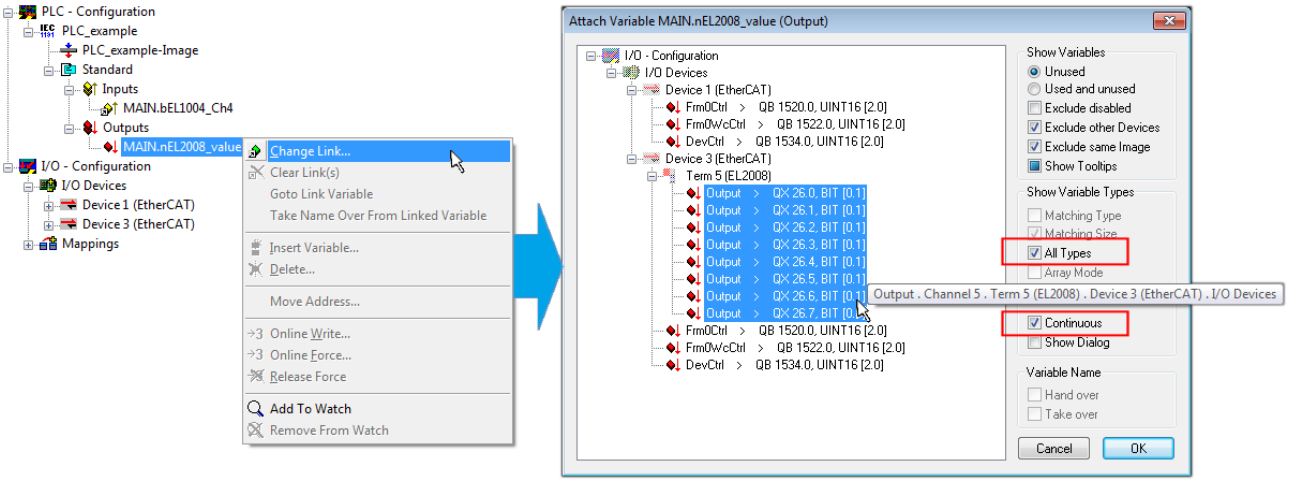


Fig. 57: Selecting several PDOs simultaneously: activate “Continuous” and “All types”

Note that the “Continuous” checkbox was also activated. This is designed to allocate the bits contained in the byte of the variable “nEL2008_value” sequentially to all eight selected output bits of the EL2008 terminal. In this way it is 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 ( ) at the yellow or red object of the variable indicates that a link exists. The links can also be checked by selecting a “Goto Link Variable” from the context menu of a variable. The object opposite, in this case the PDO, is automatically selected:

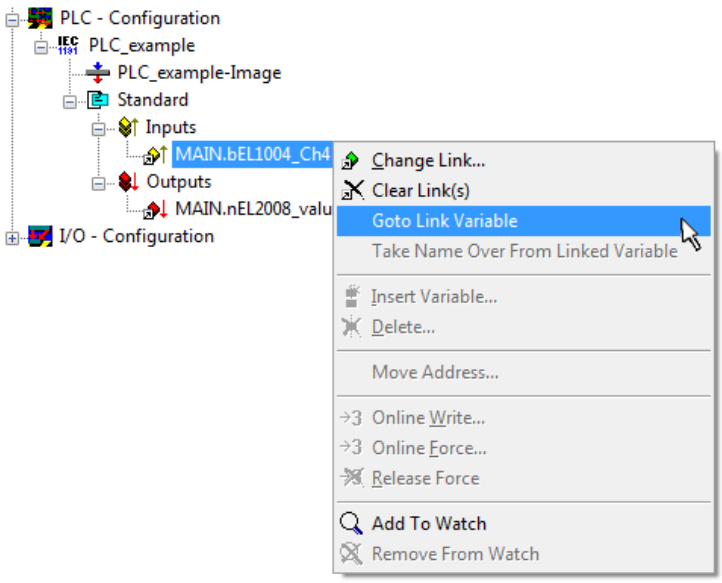
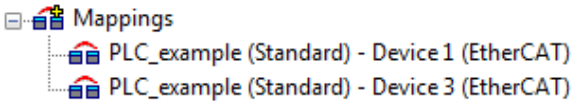


Fig. 58: Application of a “Goto Link” variable, using “MAIN.bEL1004_Ch4” as a sample

The process of assigning variables to the PDO is completed via the menu selection “Actions” → “Generate Mappings”, key Ctrl+M or by clicking on the symbol in the menu.

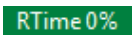
This can be visualized in the configuration:



The process of creating links can also take place in the opposite direction, i.e. starting with individual PDOs to variable. However, in this example it would then 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 possible to allocate this a set of bit-standardized variables (type “BOOL”). Here, too, a “Goto Link Variable” from the context menu of a PDO 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 are overwritten!” and “Restart TwinCAT system in Run mode” with “OK”.

A few seconds later the real-time status 

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 Ethernet via “Online” → “Choose Run-Time System…”:

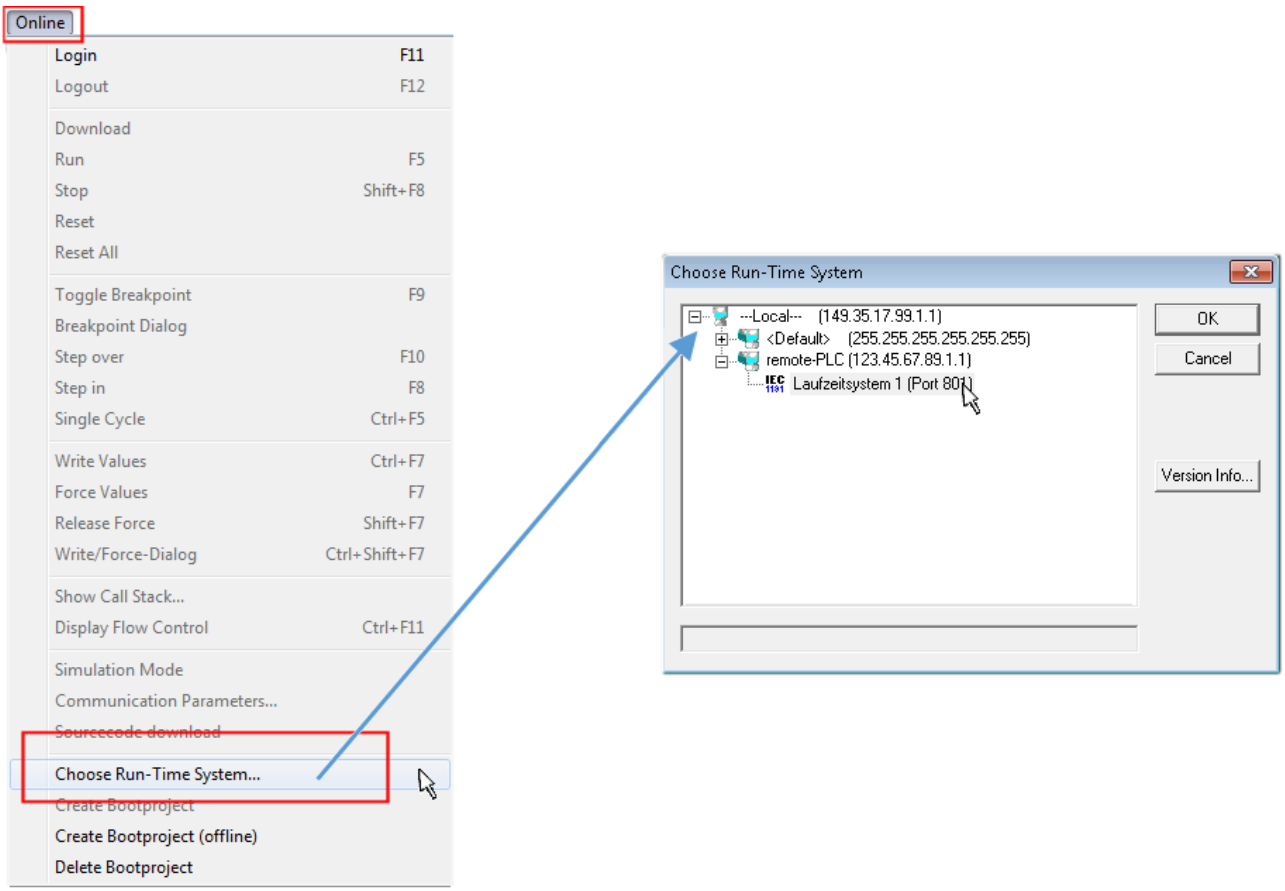


Fig. 59: Choose target system (remote)

In this sample “Runtime system 1 (port 801)” is selected and confirmed. Link the PLC with the real-time system via 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 acknowledged with “Yes”. The runtime environment is ready for the program start:

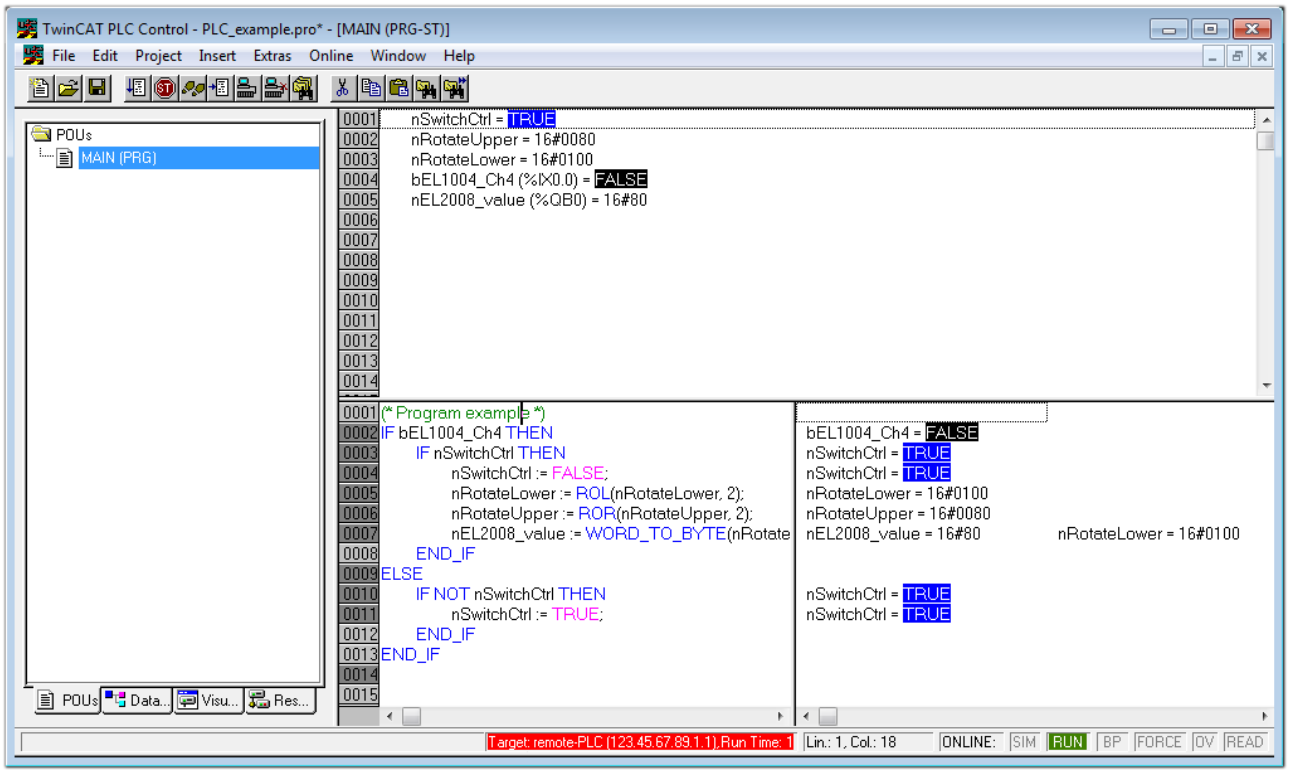


Fig. 60: PLC Control logged in, ready for program startup

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

5.1.2 TwinCAT 3

Startup

TwinCAT makes the development environment areas available together with Microsoft Visual Studio: after startup, the project folder explorer appears on the left in the general window area (cf. “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:

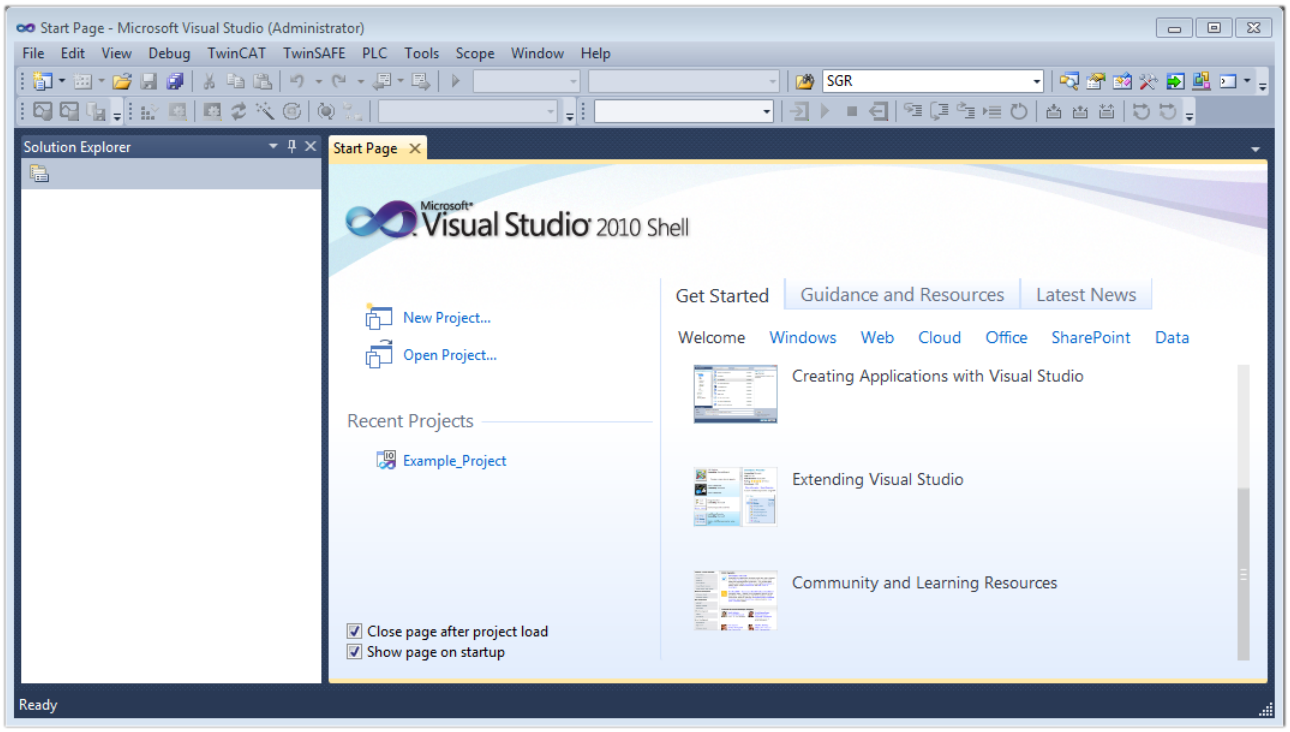


Fig. 61: Initial TwinCAT 3 user interface

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):

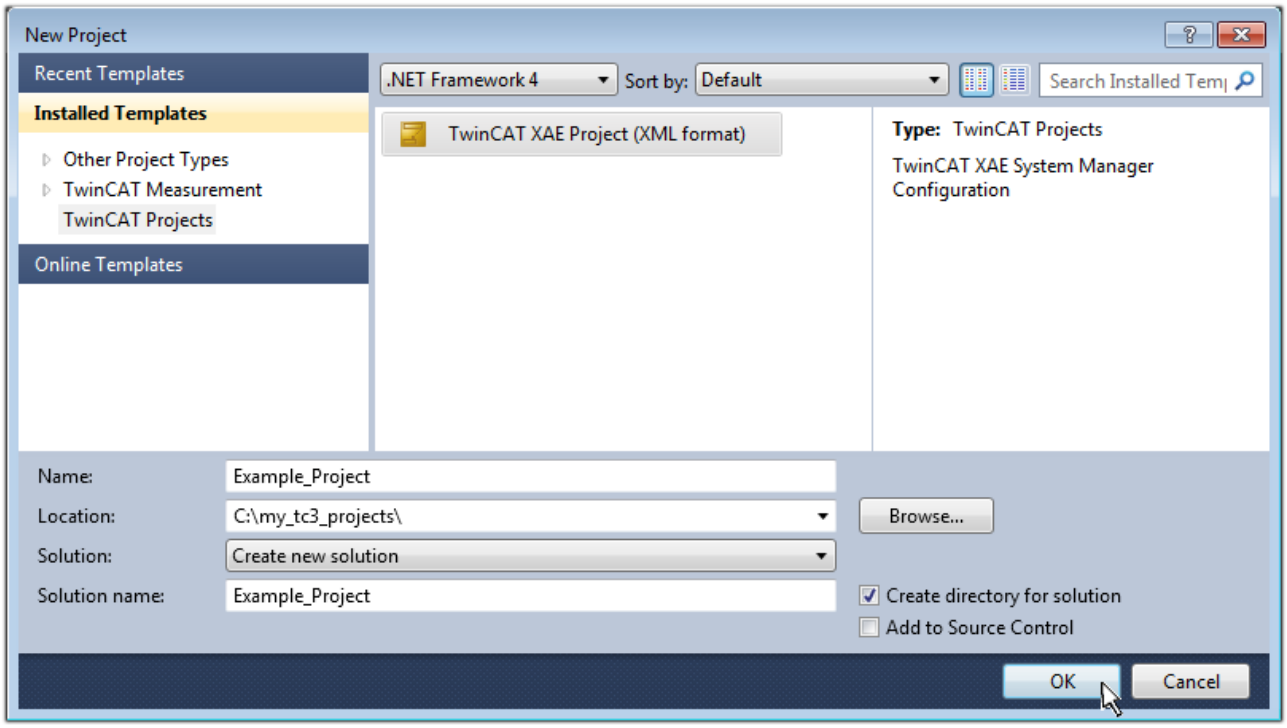


Fig. 62: Create new TwinCAT project

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

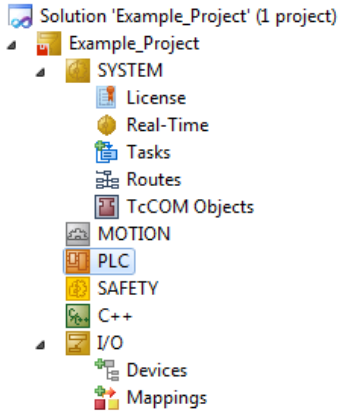
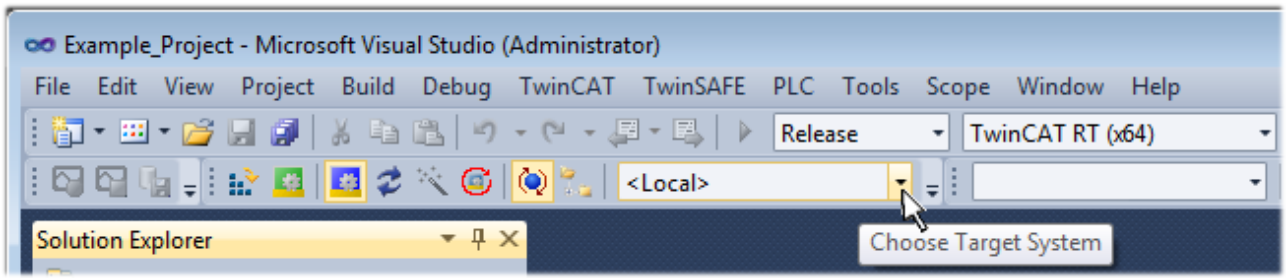


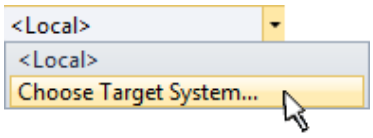
Fig. 63: New TwinCAT3 project in the project folder explorer

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, TwinCAT can be used in local mode and thereby the next step is "Insert Device [▶ 75]".

If the intention is to address the TwinCAT runtime environment installed on a PLC as development environment remotely from another system, 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:

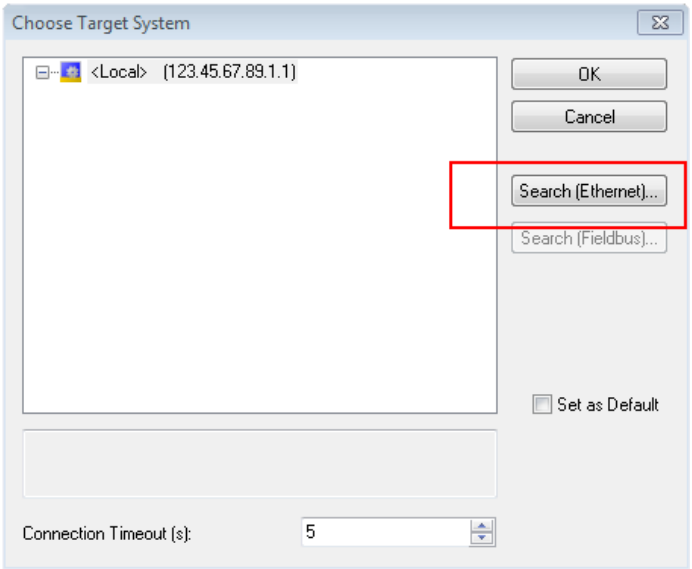


Fig. 64: Selection dialog: Choose the target system

Use “Search (Ethernet)...” to enter the target system. Thus a next 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.

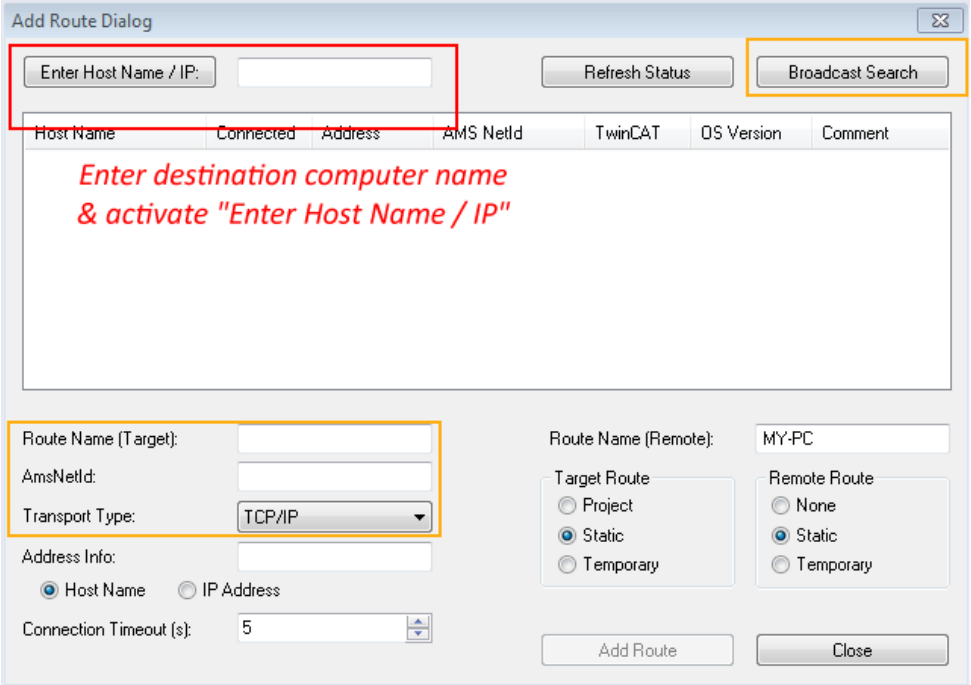
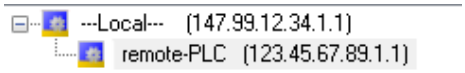


Fig. 65: Specify the PLC for access by the TwinCAT System Manager: selection of the target system

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



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

Adding devices

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

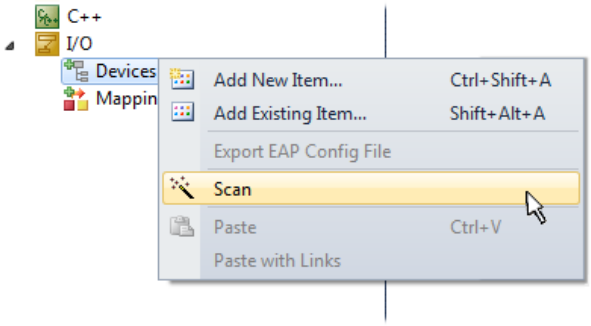


Fig. 66: Select “Scan”

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

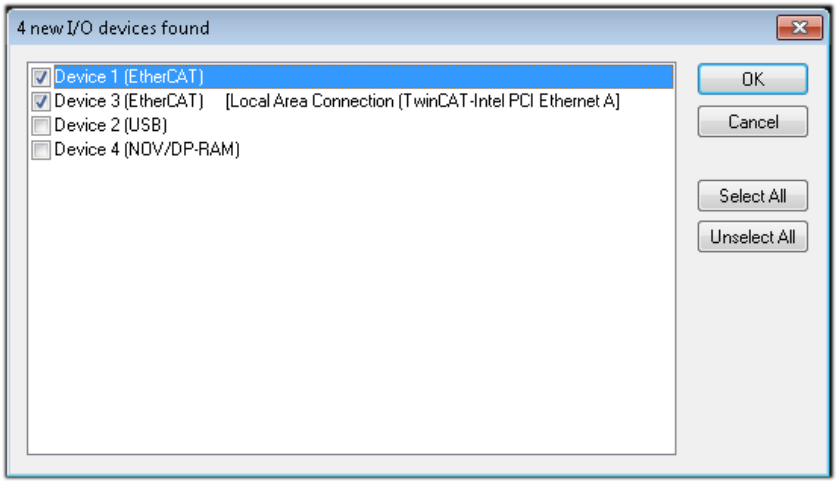


Fig. 67: Automatic detection of I/O devices: selection the devices to be integrated

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 sample configuration [▶ 60] described at the beginning of this section, the result is as follows:

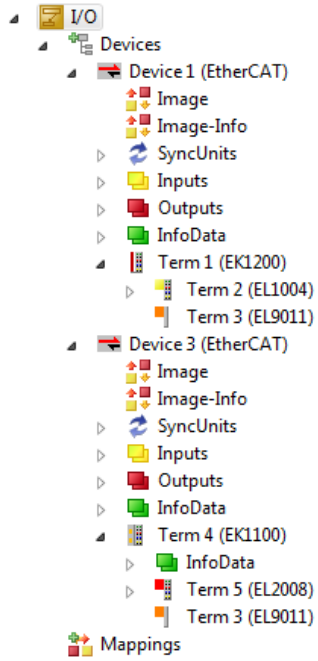


Fig. 68: Mapping of the configuration in VS shell of the TwinCAT3 environment

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

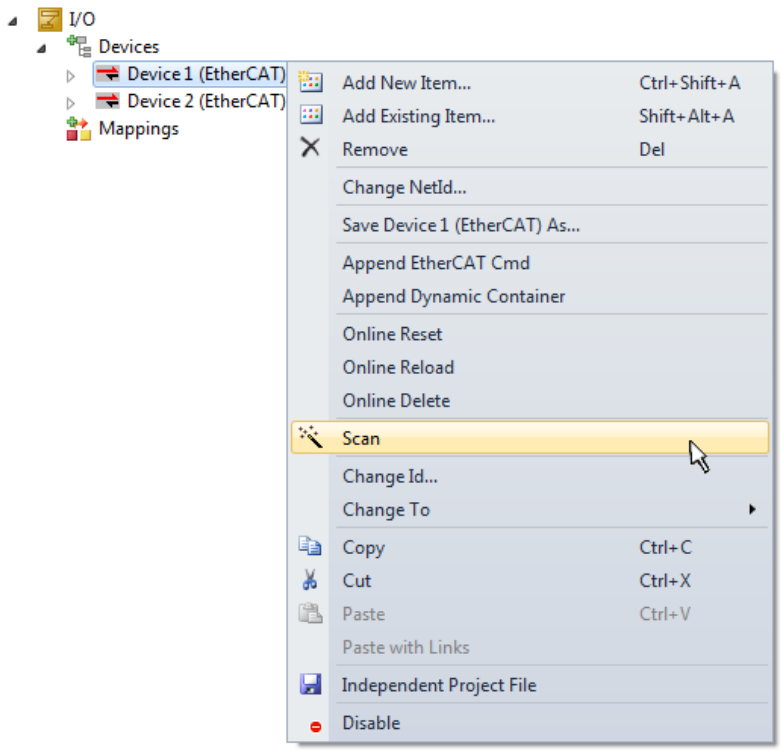


Fig. 69: Reading of individual terminals connected to a device

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

Programming the PLC

TwinCAT PLC Control is the development environment for the creation of 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 to Structured Text (ST).

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

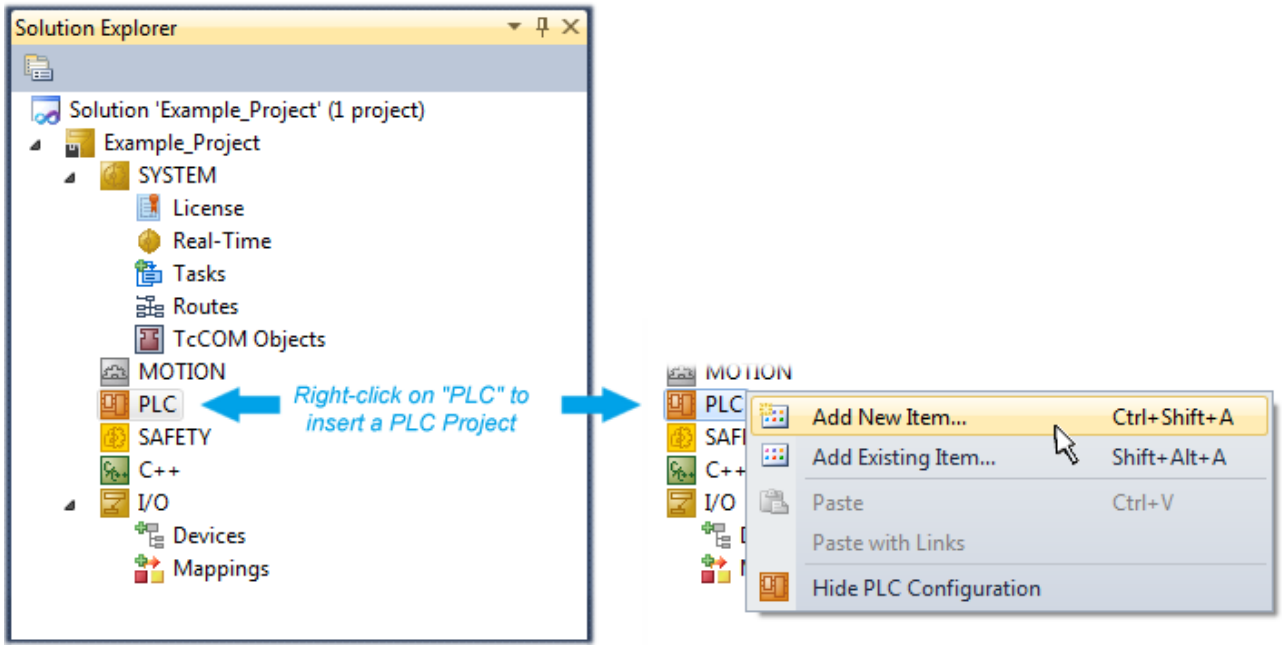


Fig. 70: Adding the programming environment in “PLC”

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

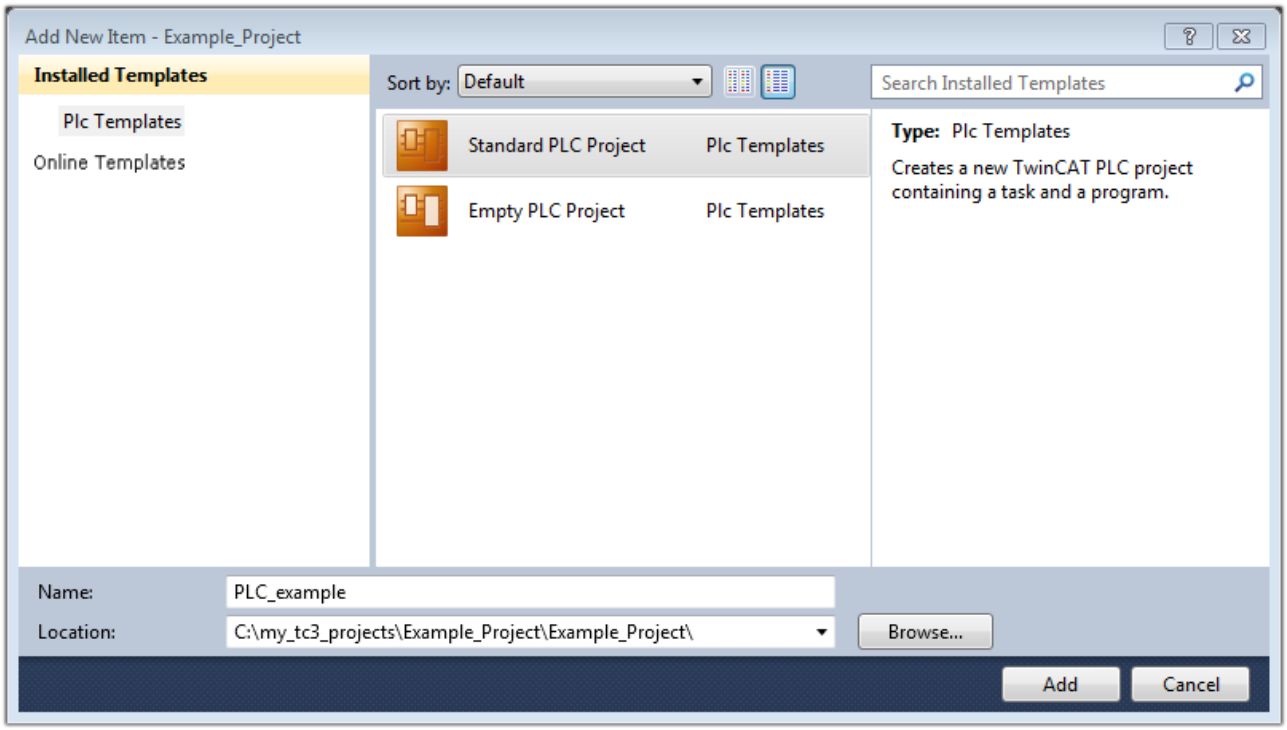


Fig. 71: Specifying the name and directory for the PLC programming environment

The “Main” program, which already exists by 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:

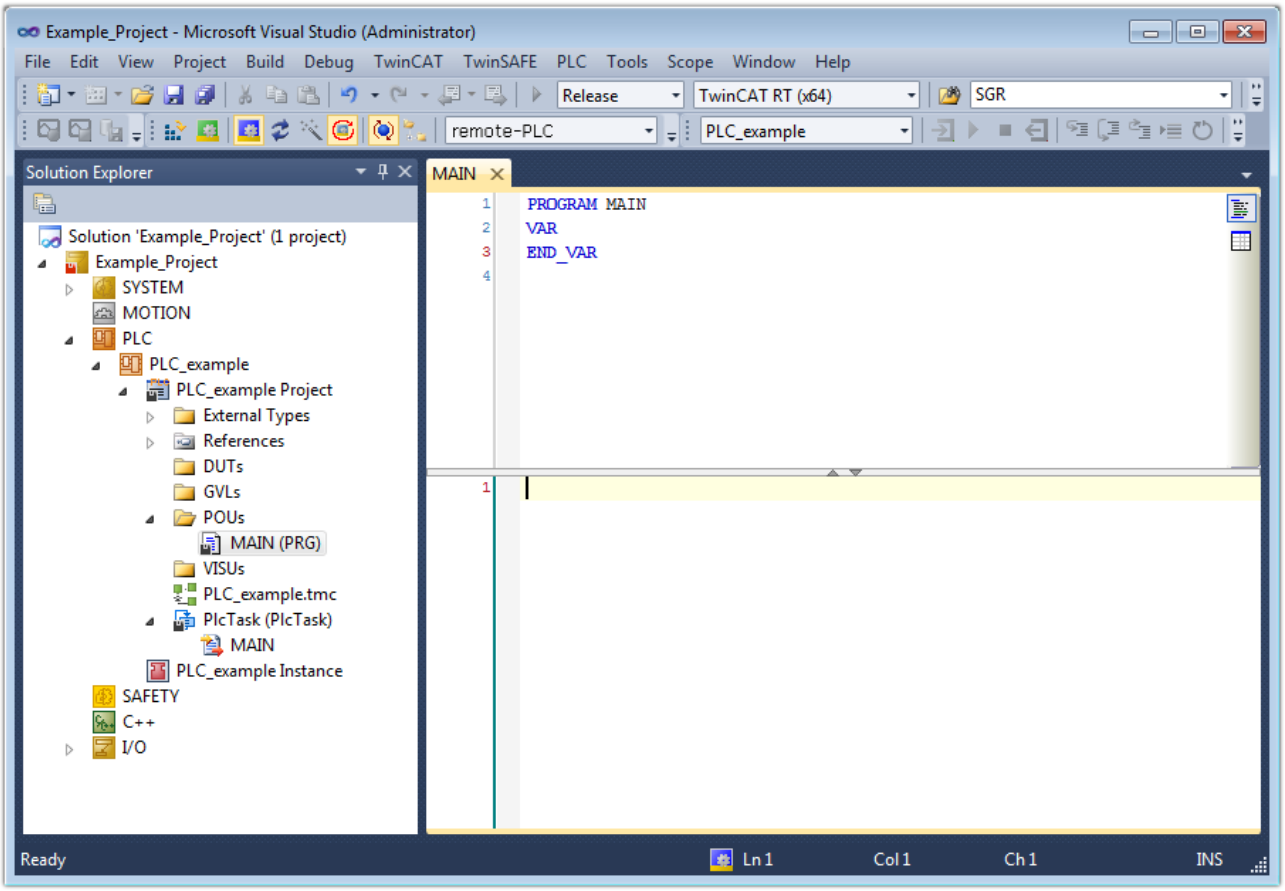


Fig. 72: Initial “Main” program of the standard PLC project

To continue, sample variables and a sample program have now been created:

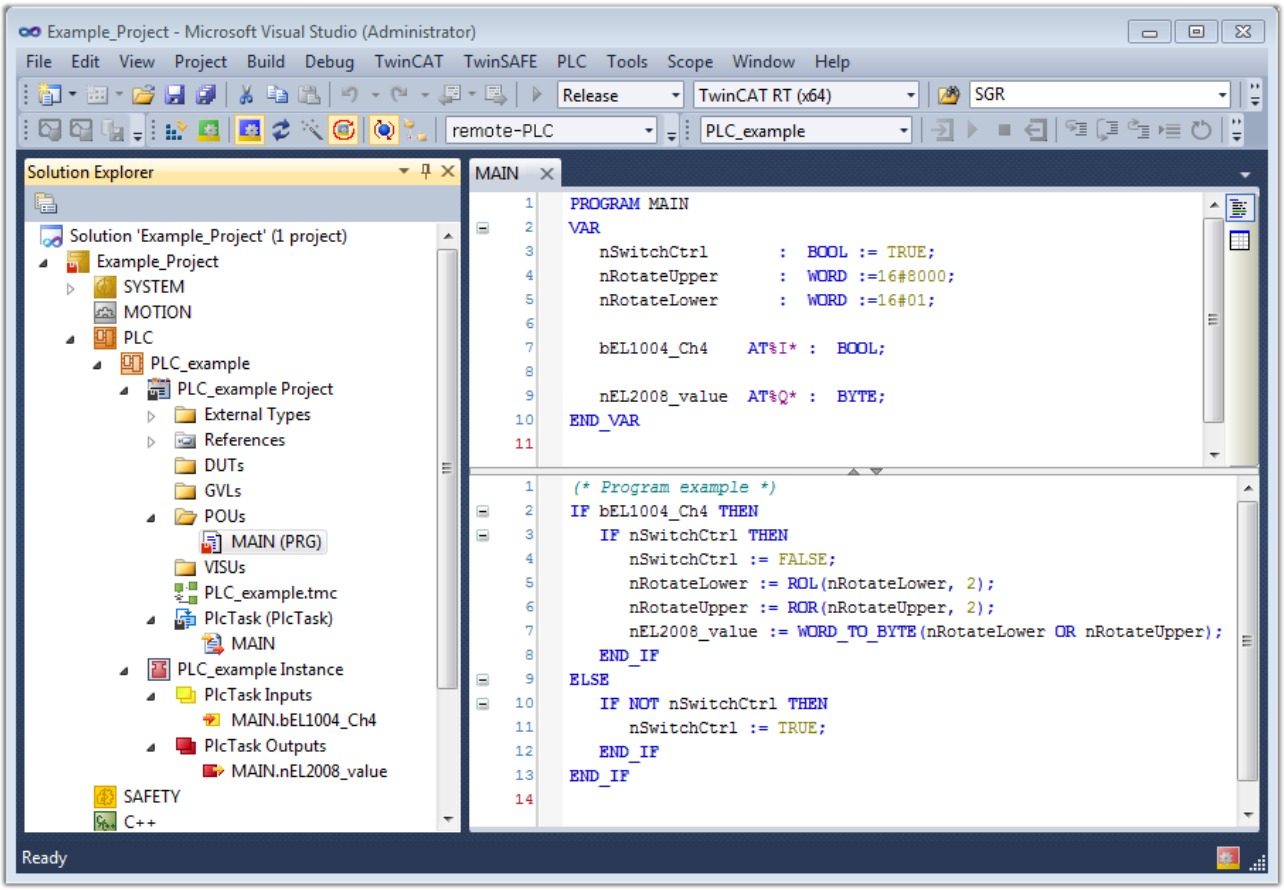


Fig. 73: Sample program with variables after a compile process (without variable integration)

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

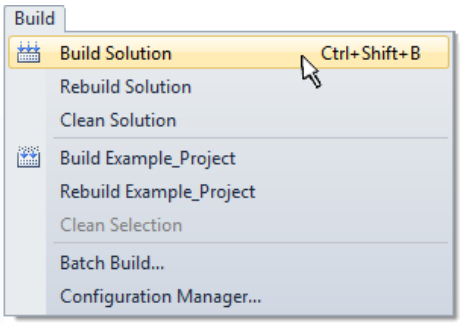
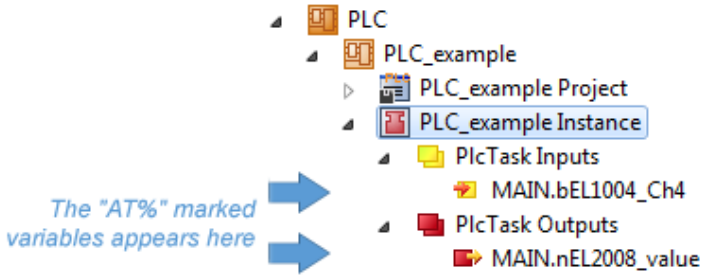


Fig. 74: Start program compilation

The following variables, identified in the ST/ PLC program with “AT%”, are then available in 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 for selecting a suitable process object (PDO) for linking:

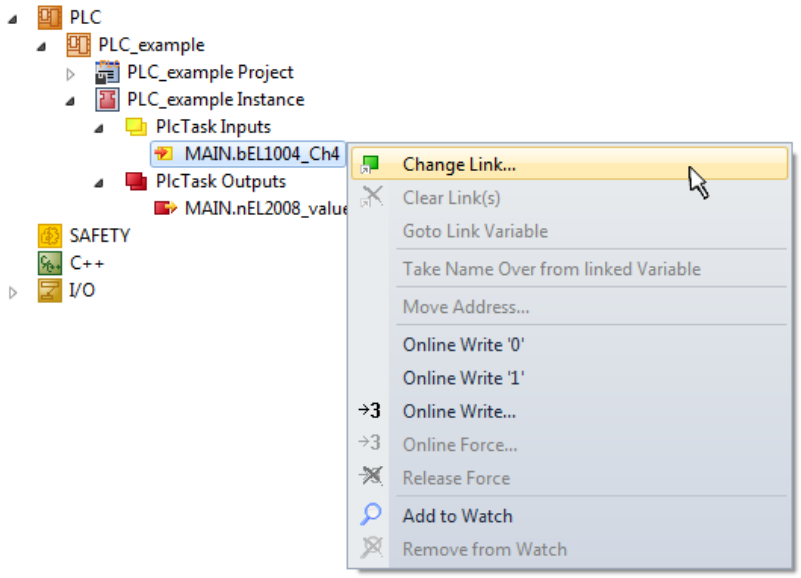


Fig. 75: Creating the links between PLC variables and process objects

In the window that opens, the process object for the variable “bEL1004_Ch4” of type BOOL can be selected from the PLC configuration tree:

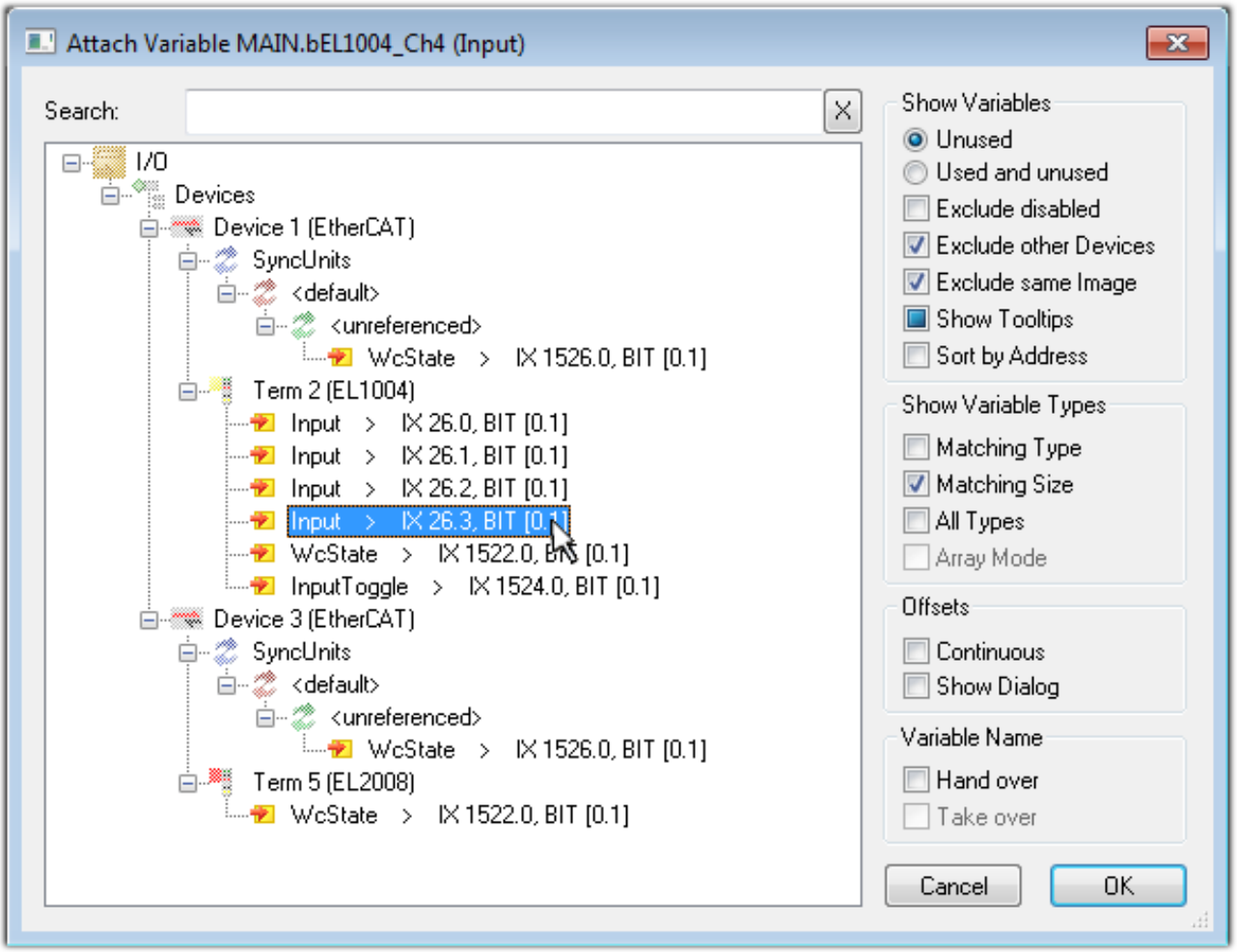


Fig. 76: Selecting PDO of type BOOL

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

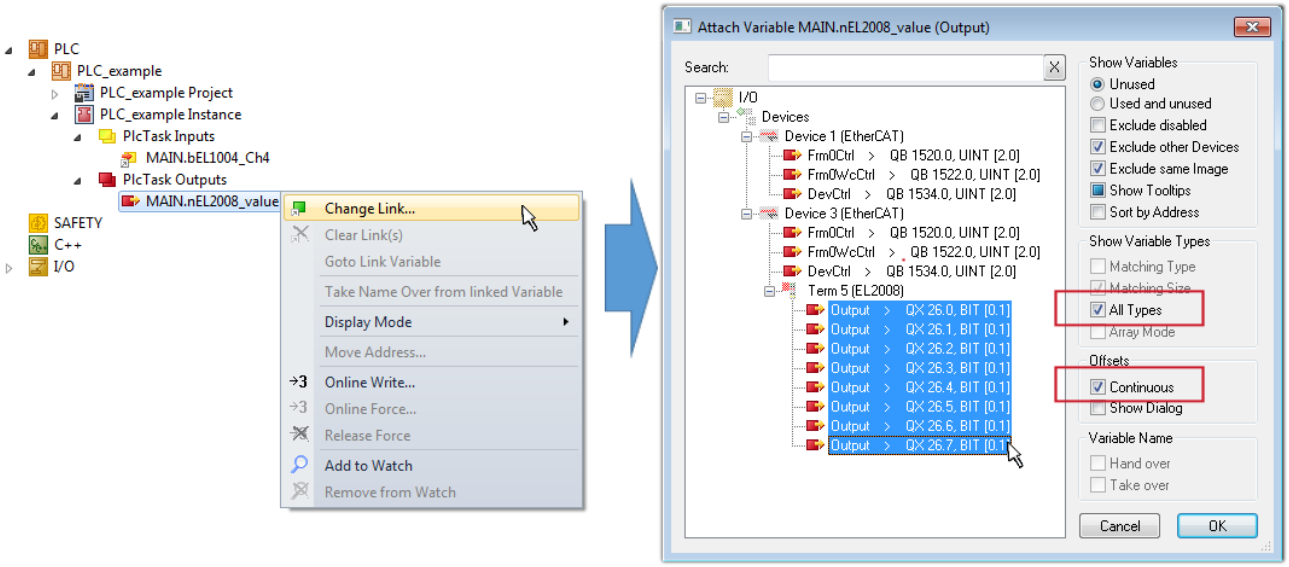


Fig. 77: Selecting several PDOs simultaneously: activate “Continuous” and “All types”

Note that the “Continuous” checkbox was also activated. This is designed to allocate the bits contained in the byte of the variable “nEL2008_value” sequentially to all eight selected output bits of the EL2008 terminal. In this way it is 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 ( ) at the yellow or red object of the variable indicates that a link exists. The links can also be checked by selecting a “Goto Link Variable” from the context menu of a variable. The object opposite, in this case the PDO, is automatically selected:

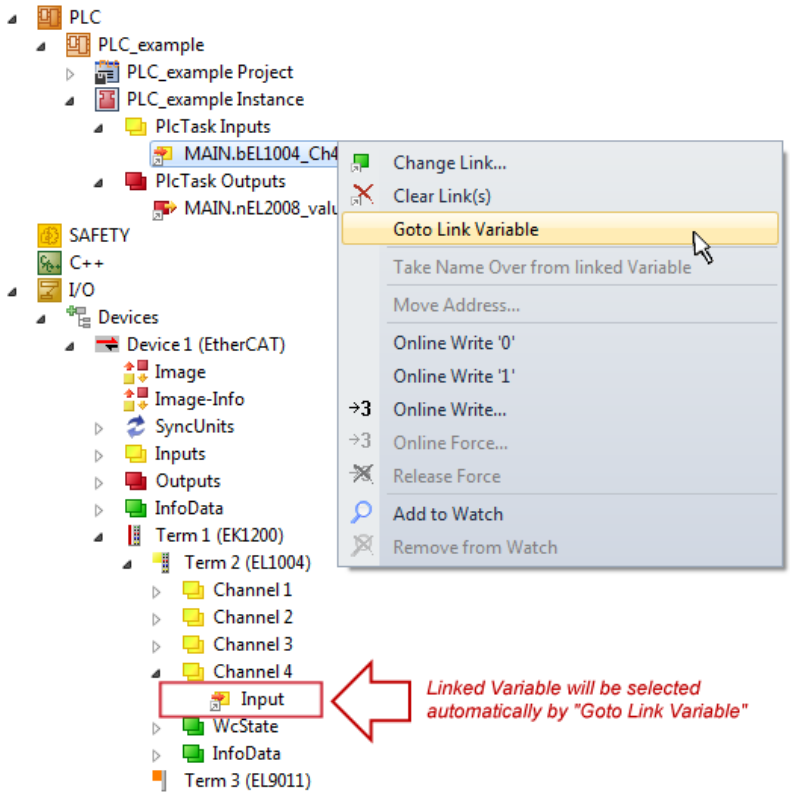


Fig. 78: Application of a “Goto Link” variable, using “MAIN.bEL1004_Ch4” as a sample

The process of creating links can also take place in the opposite direction, i.e. starting with individual PDOs to variable. However, in this example it would then 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 possible to allocate this a set of bit-standardized variables (type “BOOL”). Here, too, a “Goto Link Variable” from the context menu of a PDO can be executed in the other direction, so that the respective PLC instance can then be selected.

Note on the type of variable assignment

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 it is possible to create a structure 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 tab “PLC” via the check box.
3. The data type in the “Data Type” field can then be copied using the “Copy” button.

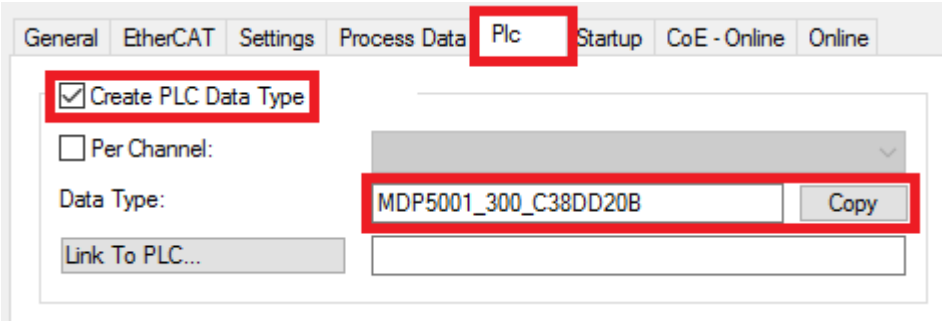


Fig. 79: Creating a PLC data type

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

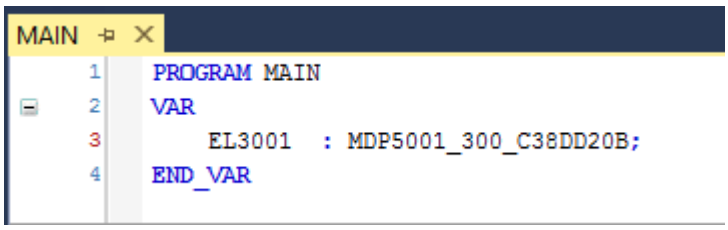


Fig. 80: Instance_of_struct

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.

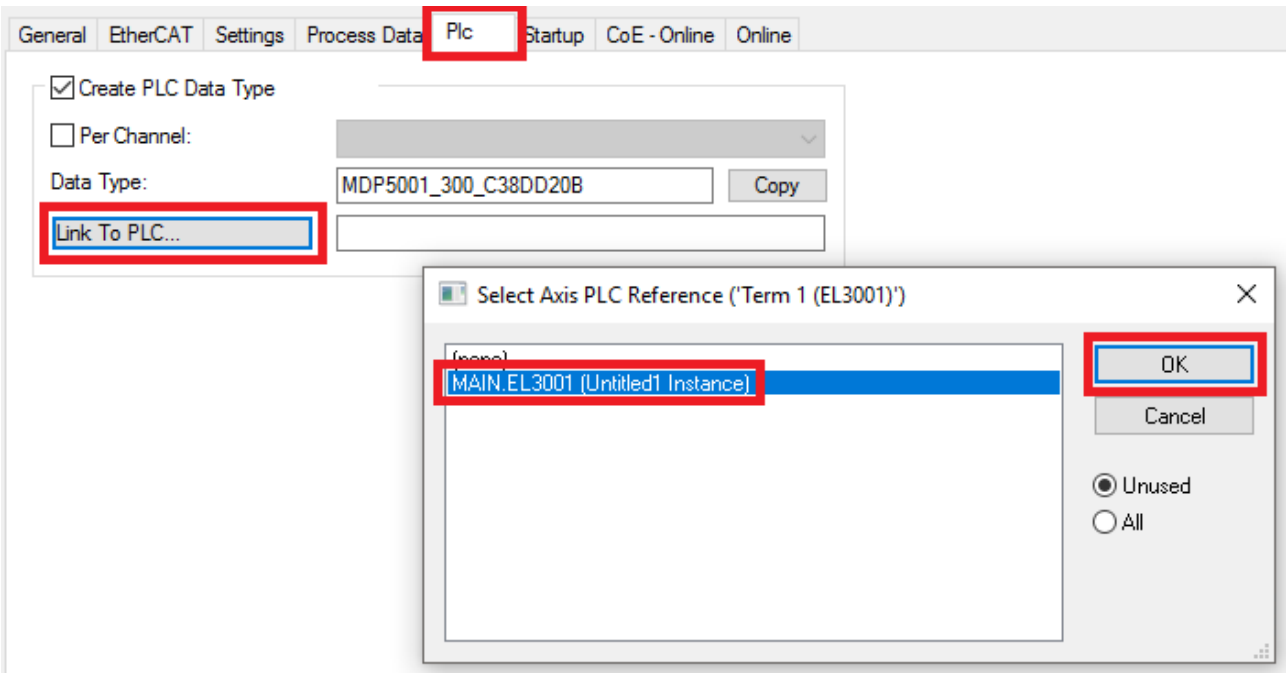


Fig. 81: Linking the structure

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

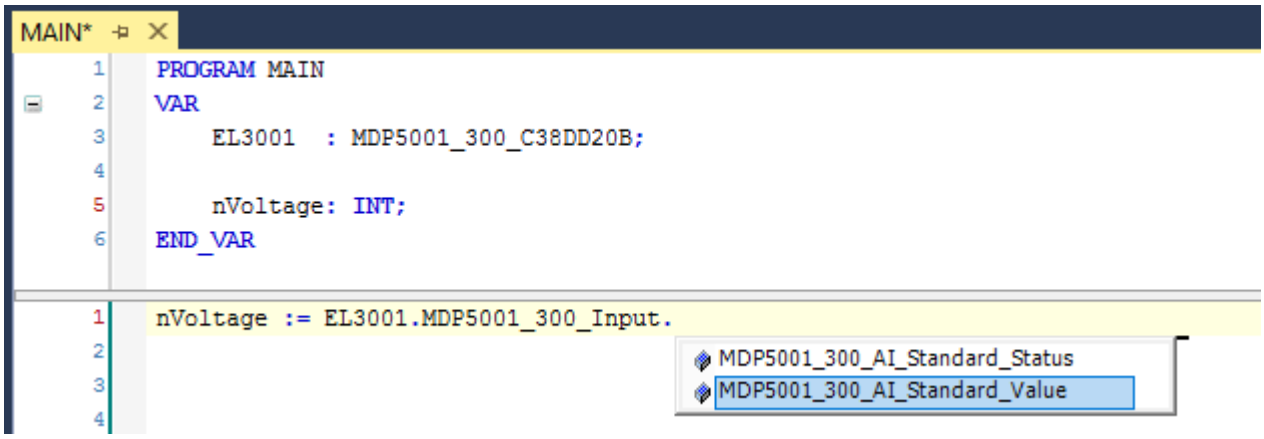
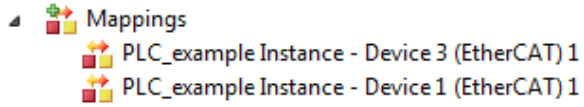


Fig. 82: Reading a variable from the structure of the process data

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 settings of the development environment to the runtime system. Confirm the messages “Old configurations are 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

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

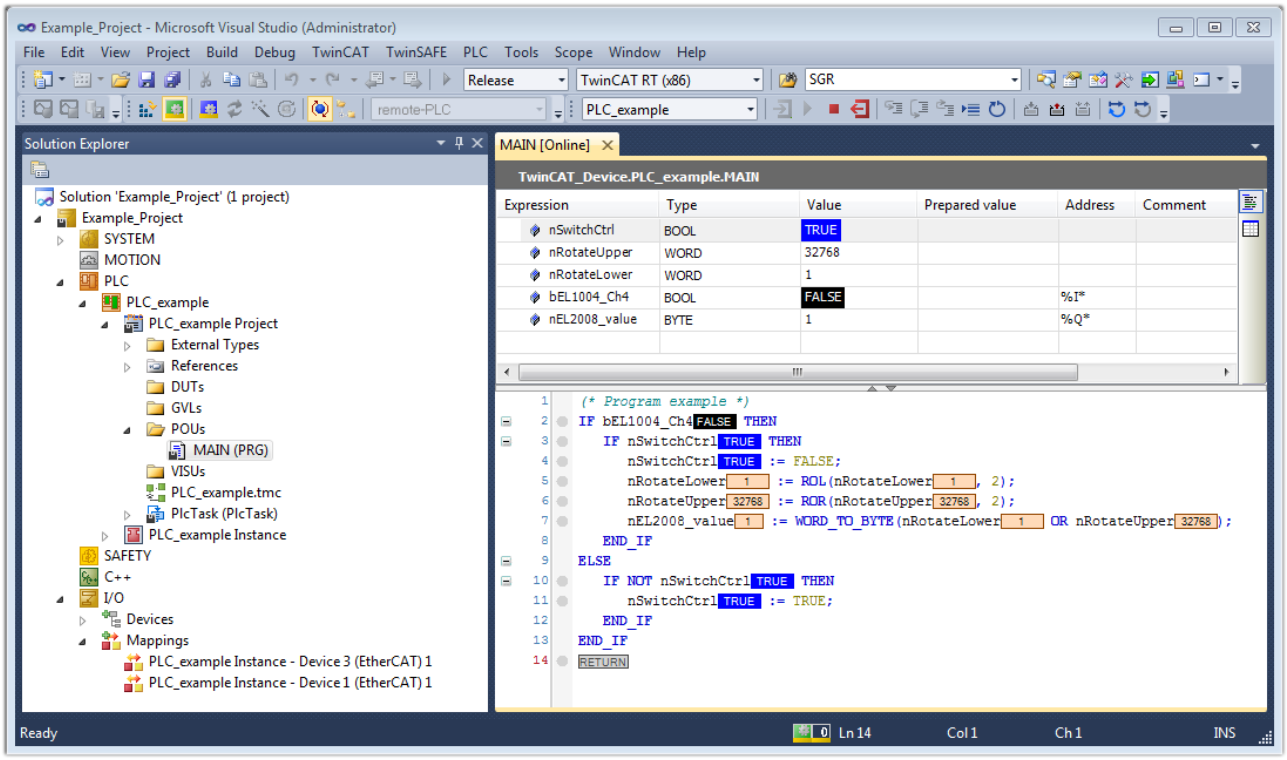


Fig. 83: TwinCAT development environment (VS shell): logged-in, after program startup

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

5.2 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- und 64-Bit-Operatingsystem
  - 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.

5.2.1 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. One option is described here.

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

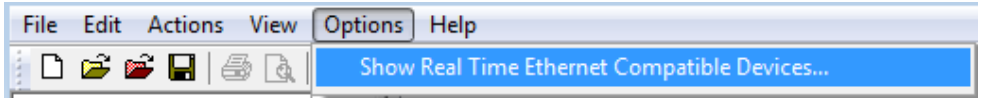


Fig. 84: System Manager “Options” (TwinCAT 2)

This have to be called up by the Menü “TwinCAT” within the TwinCAT 3 environment:

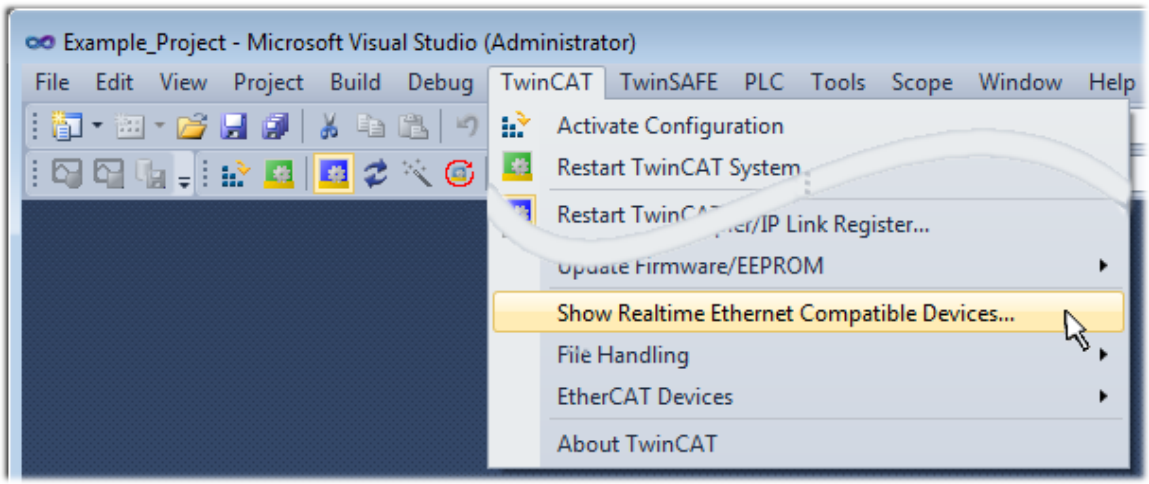


Fig. 85: Call up under VS Shell (TwinCAT 3)

The following dialog appears:

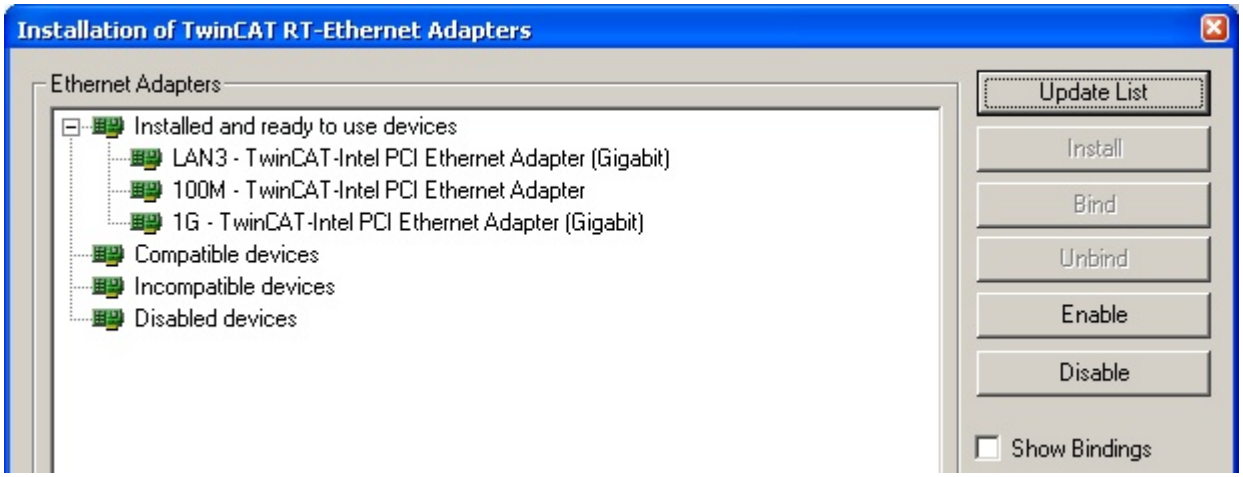


Fig. 86: Overview of network interfaces

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” [▶ 96] in order to view the compatible ethernet ports via its EtherCAT properties (tab “Adapter”, button “Compatible Devices…”):

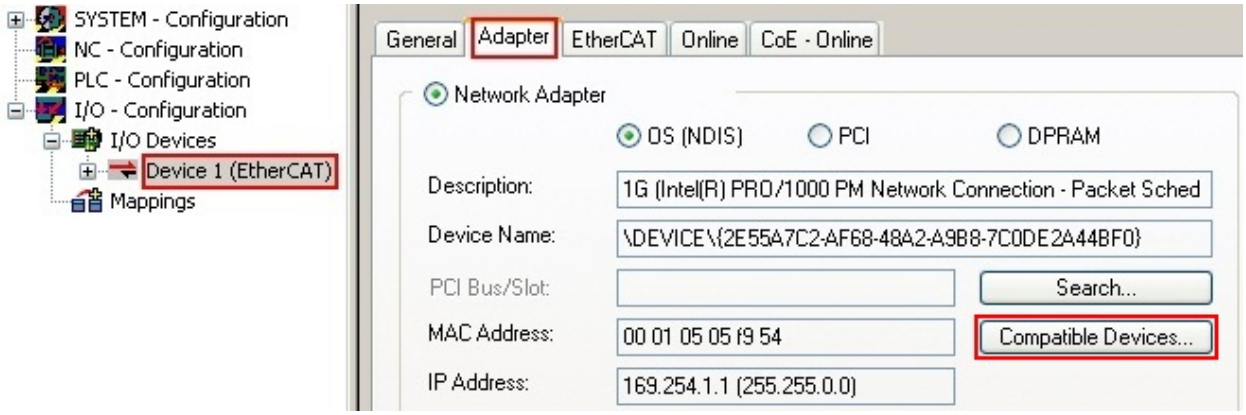


Fig. 87: EtherCAT device properties(TwinCAT 2): click on “Compatible Devices…” of tab “Adapte””

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)

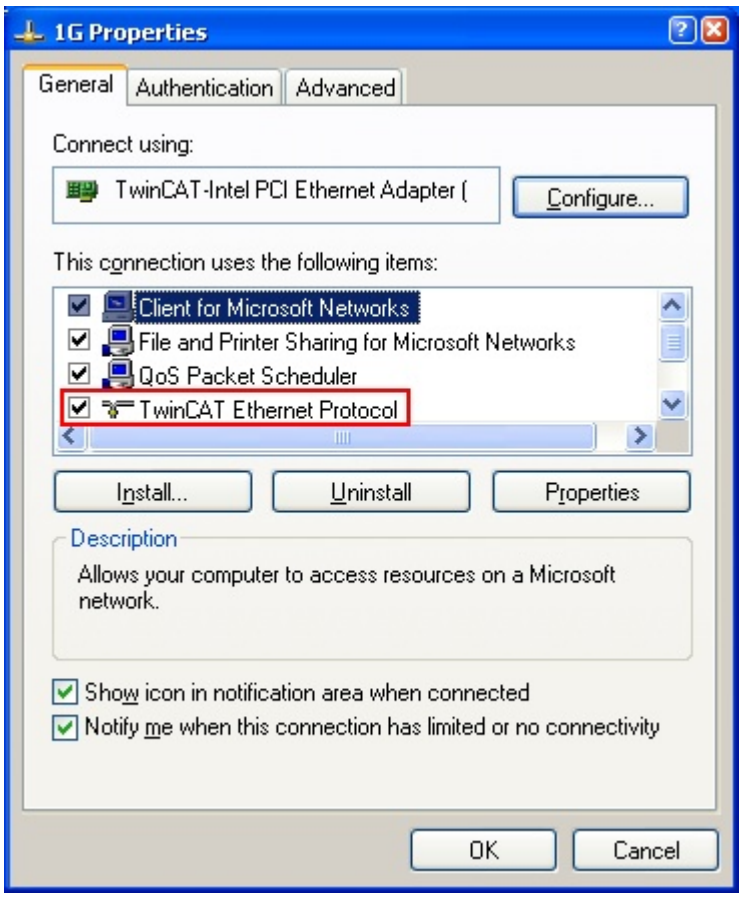


Fig. 88: Windows properties of the network interface

A correct setting of the driver could be:

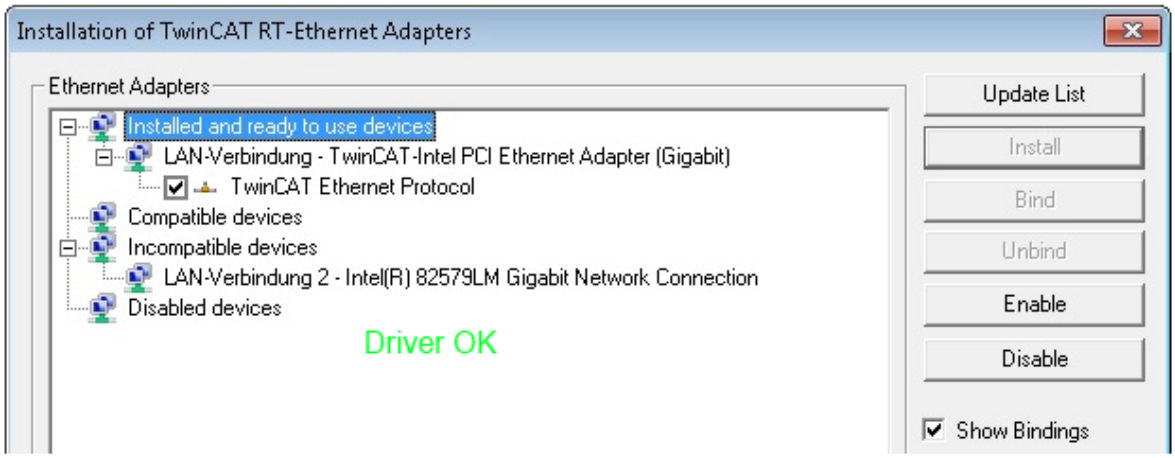


Fig. 89: Exemplary correct driver setting for the Ethernet port

Other possible settings have to be avoided:

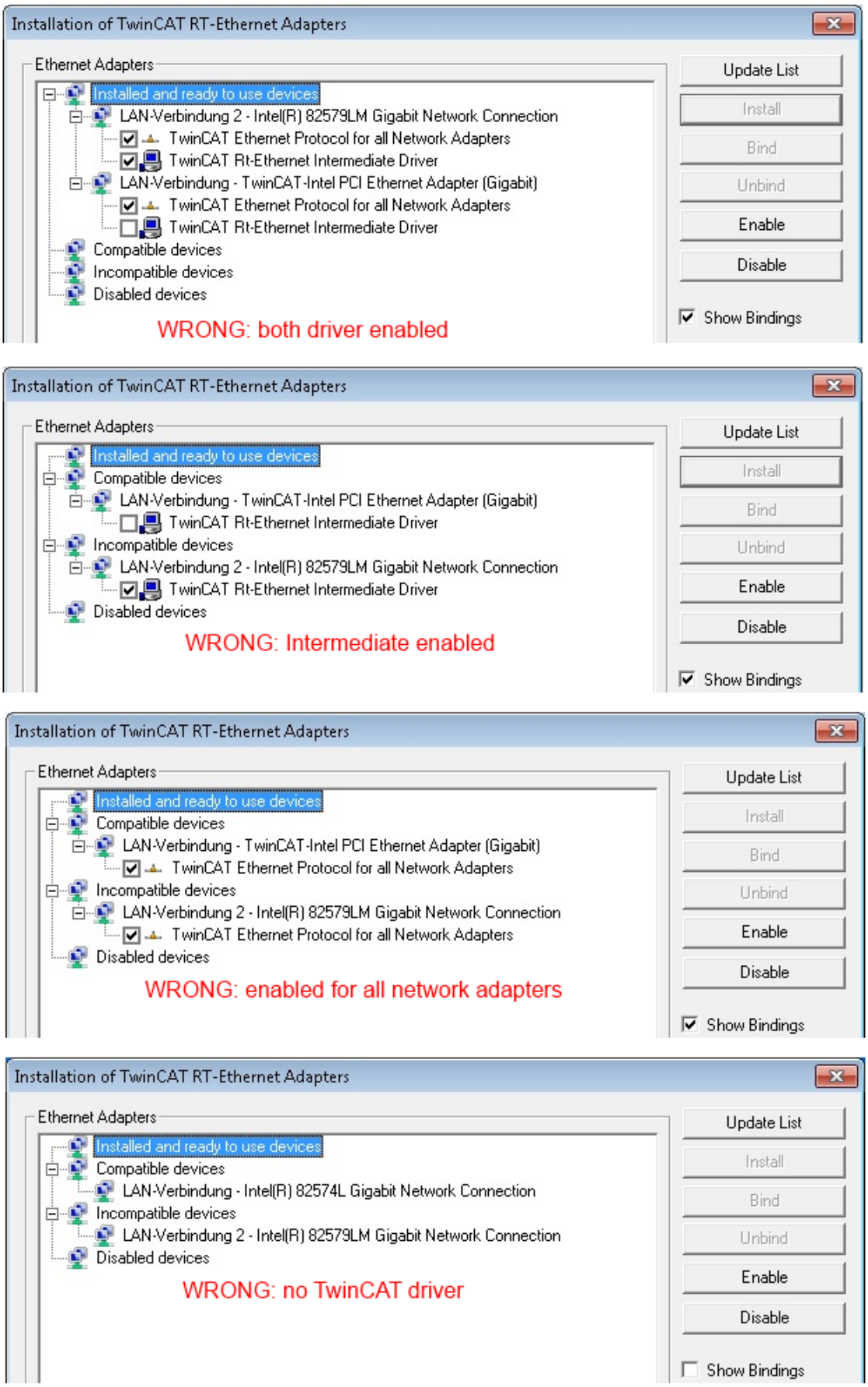


Fig. 90: Incorrect driver settings for the Ethernet port

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.

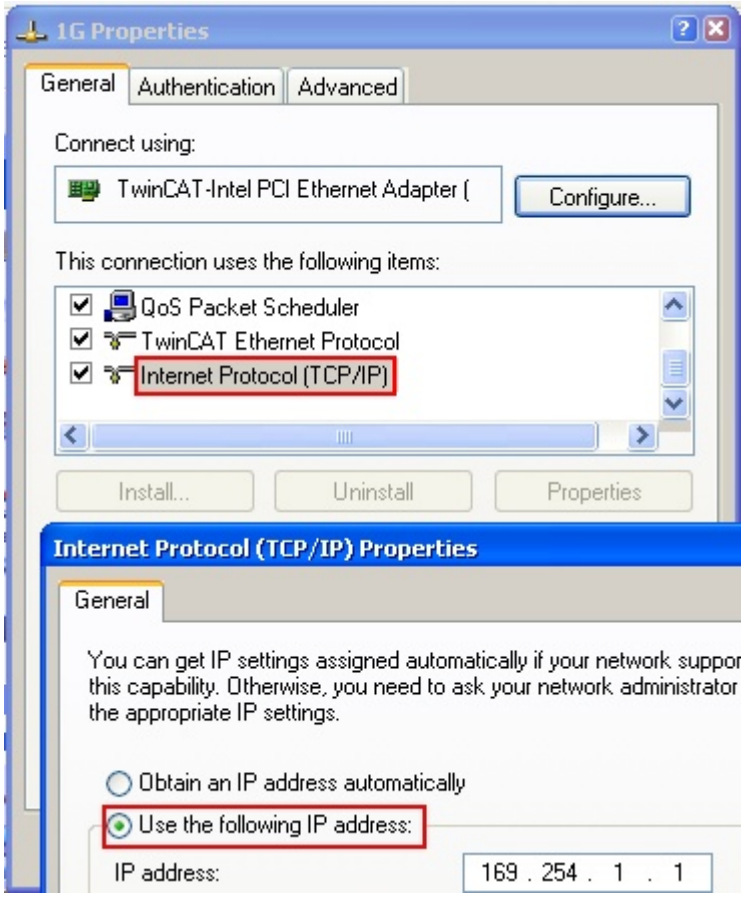


Fig. 91: TCP/IP setting for the Ethernet port

5.2.2 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 [▶ 95] 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”

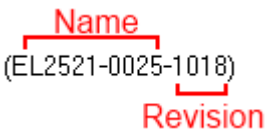


Fig. 92: Identifier structure

The order identifier consisting of name + type (here: EL2521-0010) 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 [▶ 10].

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.

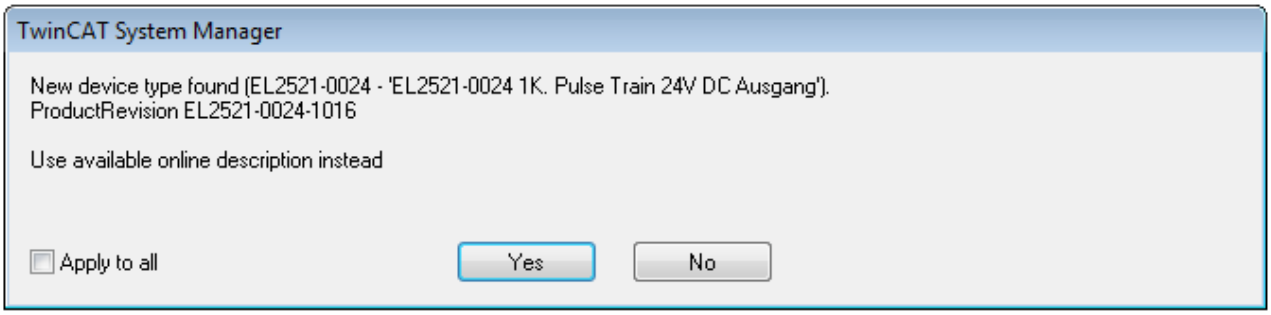


Fig. 93: OnlineDescription information window (TwinCAT 2)

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

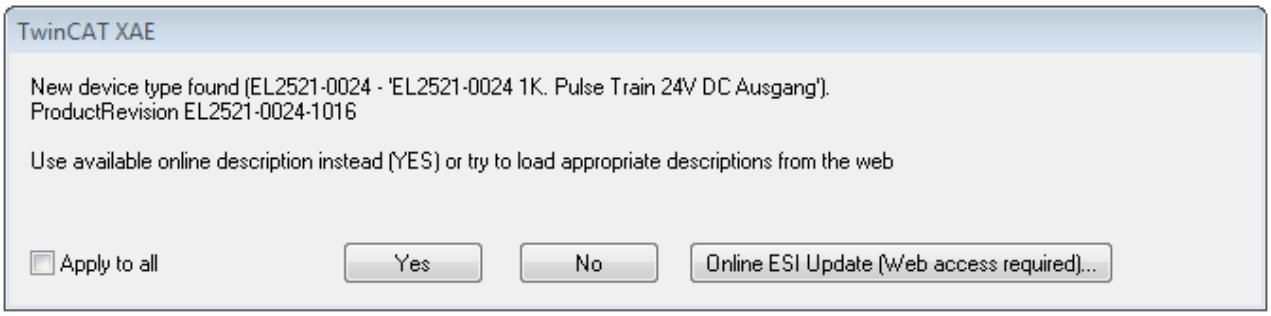


Fig. 94: Information window OnlineDescription (TwinCAT 3)

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.

NOTE

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 [▶ 96]".

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.

OnlineDescriptionCache00000002.xml

Fig. 95: File OnlineDescription.xml created by the System Manager

Is a slave 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)*.

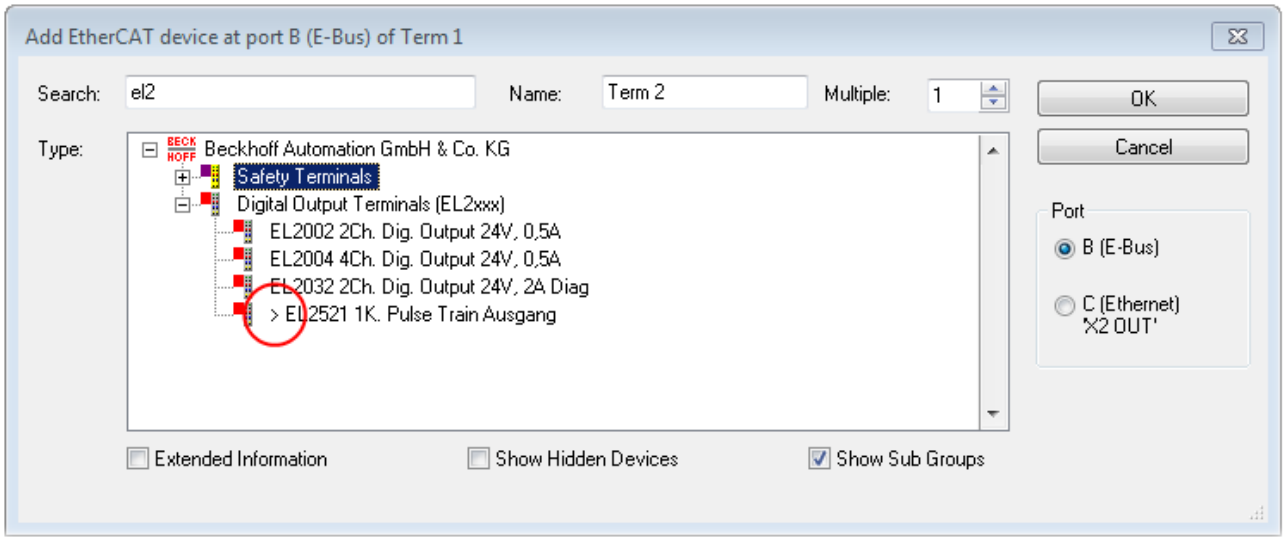


Fig. 96: Indication of an online recorded ESI of EL2521 as an example

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.

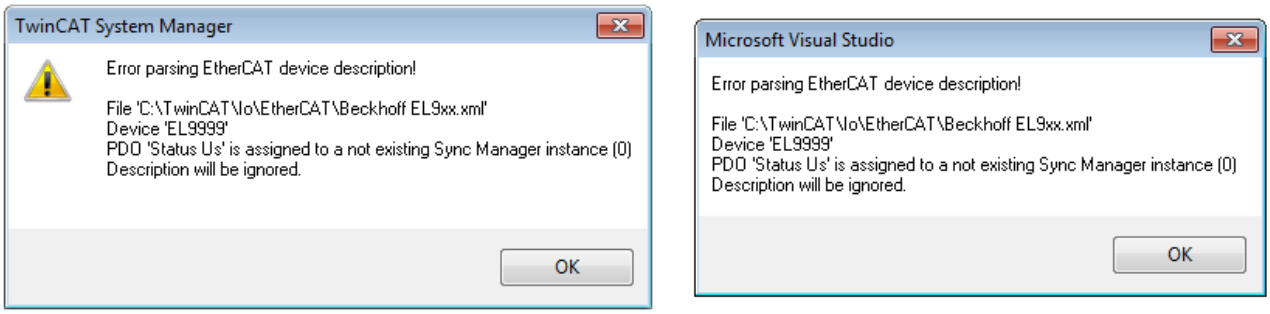


Fig. 97: Information window for faulty ESI file (left: TwinCAT 2; right: TwinCAT 3)

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

5.2.3 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:

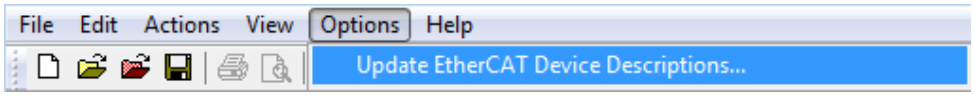


Fig. 98: Using the ESI Updater (>= TwinCAT 2.11)

The call up takes place under:
“Options” → “Update EtherCAT Device Descriptions”

Selection under TwinCAT 3:

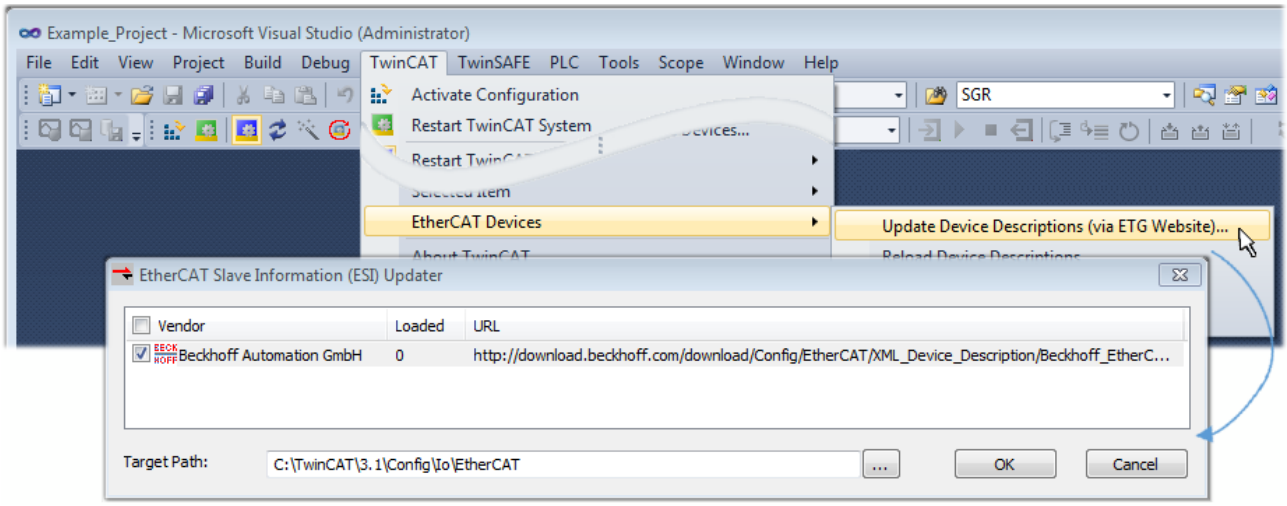


Fig. 99: Using the ESI Updater (TwinCAT 3)

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)…”.

5.2.4 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” [▶ 91].

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 [▶ 101] (Ethernet port at the IPC)
- detecting the connected EtherCAT devices [▶ 102]. This step can be carried out independent of the preceding step
- troubleshooting [▶ 105]

The scan with existing configuration [▶ 106] can also be carried out for comparison.

5.2.5 OFFLINE configuration creation

Creating the EtherCAT device

Create an EtherCAT device in an empty System Manager window.

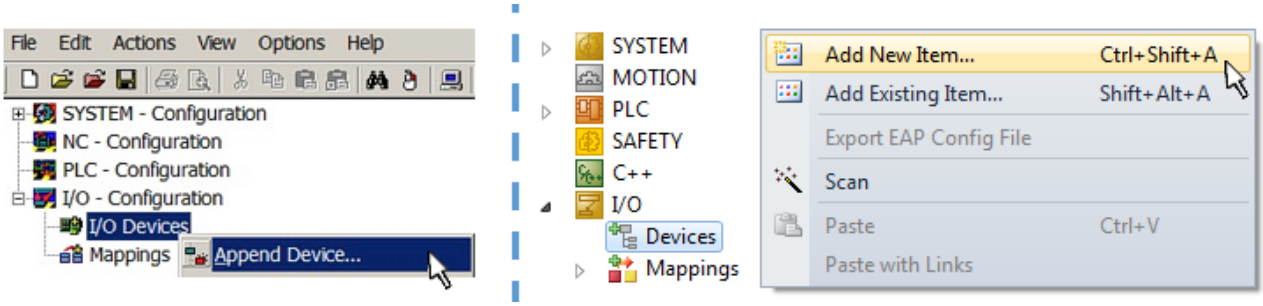


Fig. 100: Append EtherCAT device (left: TwinCAT 2; right: TwinCAT 3)

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”.

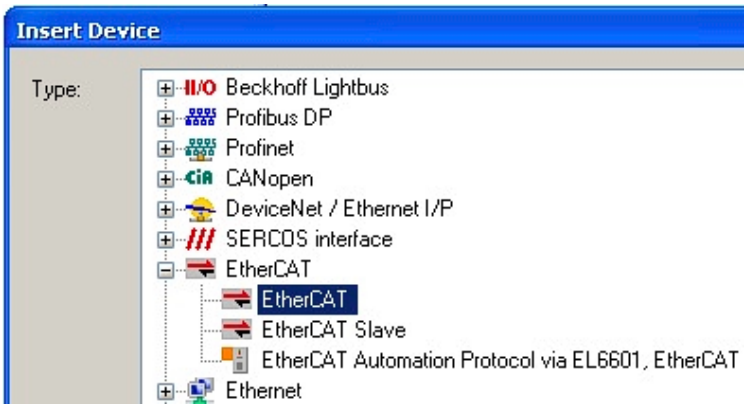


Fig. 101: Selecting the EtherCAT connection (TwinCAT 2.11, TwinCAT 3)

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

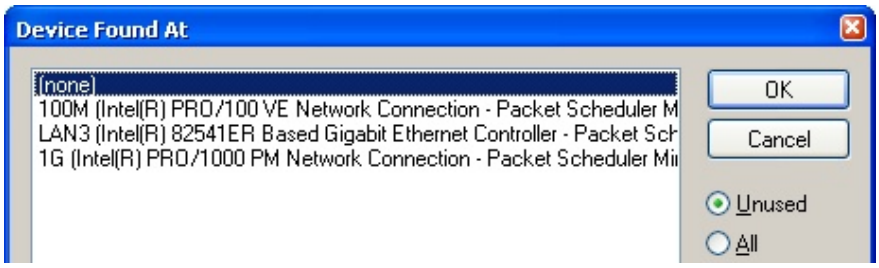


Fig. 102: Selecting the Ethernet port

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)”.

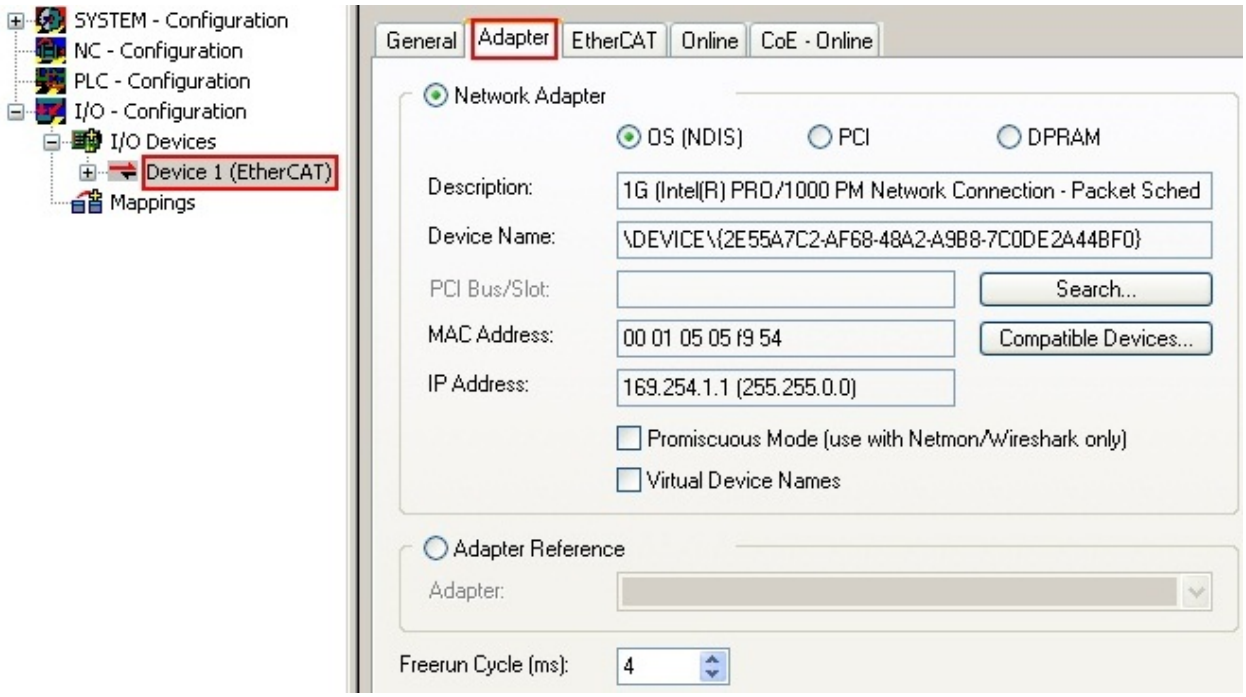
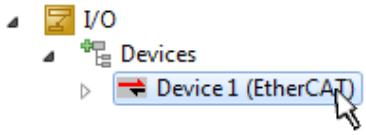


Fig. 103: EtherCAT device properties (TwinCAT 2)

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

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 [▶ 86].

Defining EtherCAT slaves

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

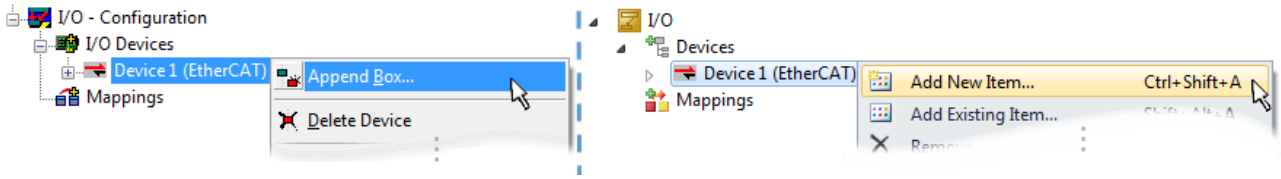


Fig. 104: Appending EtherCAT devices (left: TwinCAT 2; right: TwinCAT 3)

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: EK couplers, EP boxes, devices with RJ45/M8/M12 connector

- “E-Bus”: LVDS “terminal bus”, “EJ-module”: EL/ES terminals, various modular modules

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

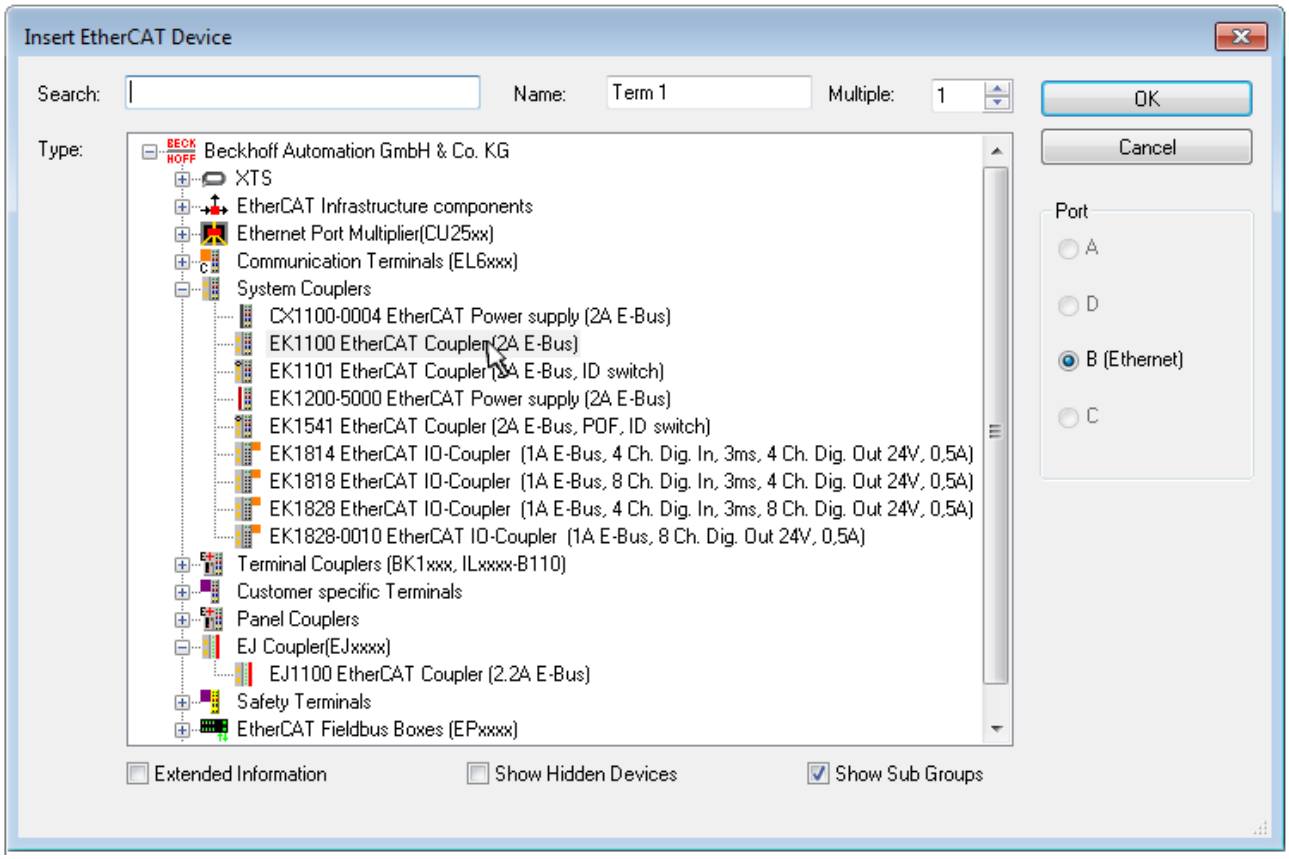


Fig. 105: Selection dialog for new EtherCAT device

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”.

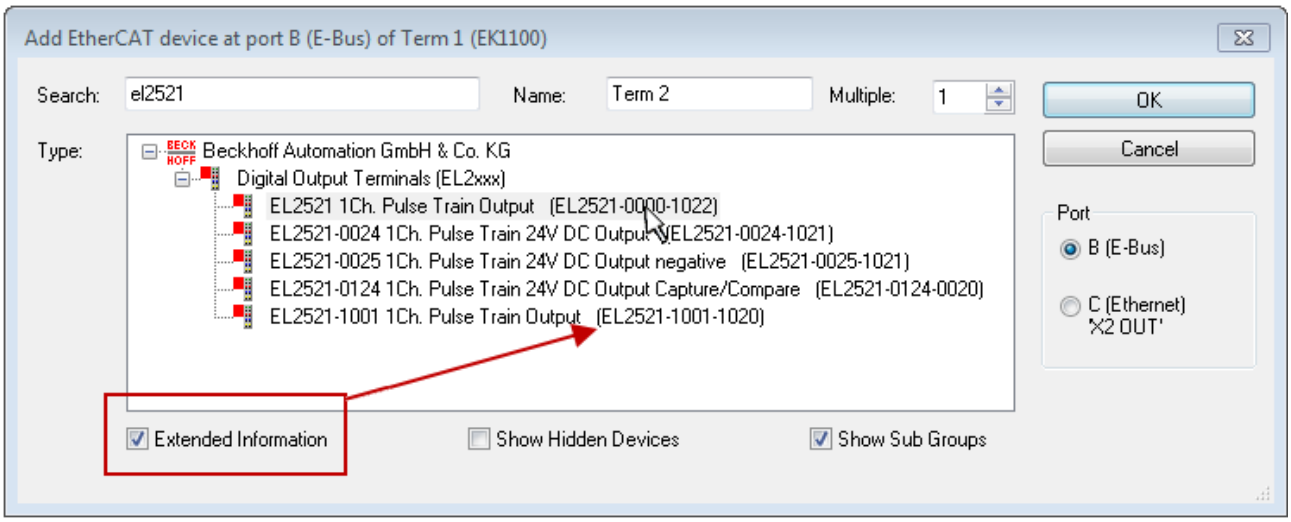


Fig. 106: Display of device revision

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”.

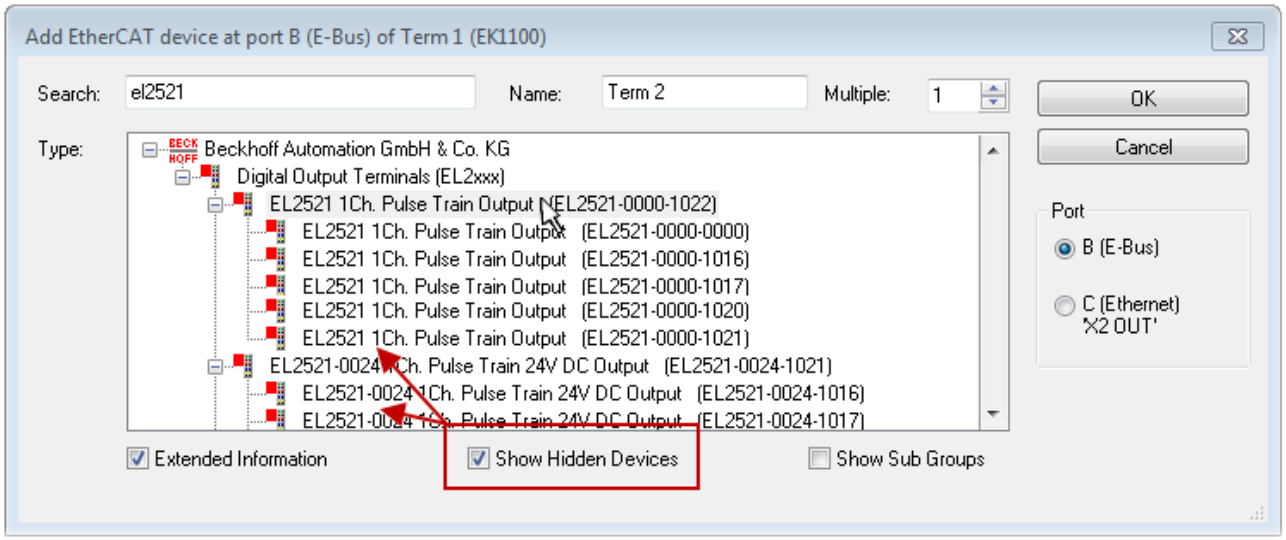


Fig. 107: Display of previous revisions

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 >= 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.

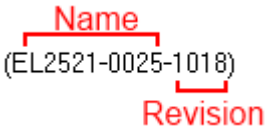


Fig. 108: Name/revision of the terminal

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, ...

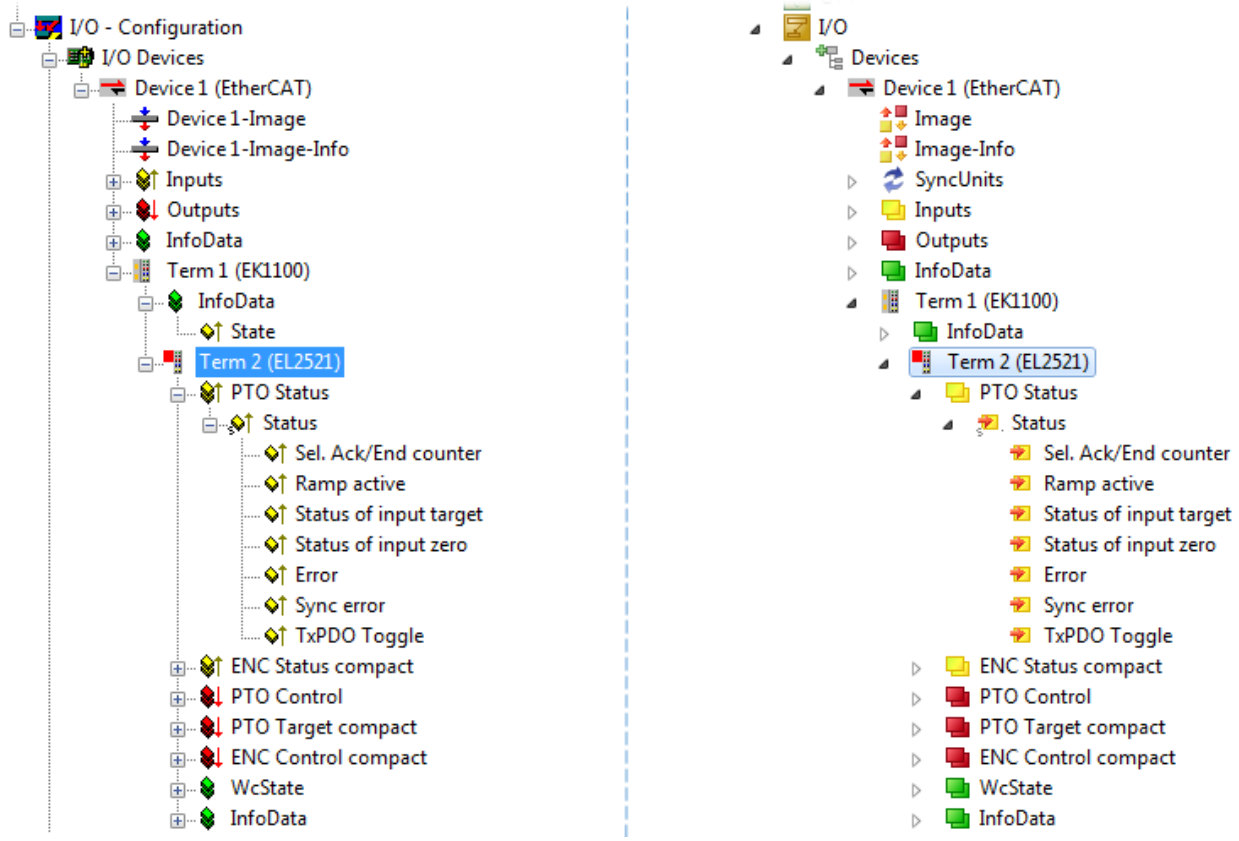


Fig. 109: EtherCAT terminal in the TwinCAT tree (left: TwinCAT 2; right: TwinCAT 3)

5.2.6 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: Config Mode .
- 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)”

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.

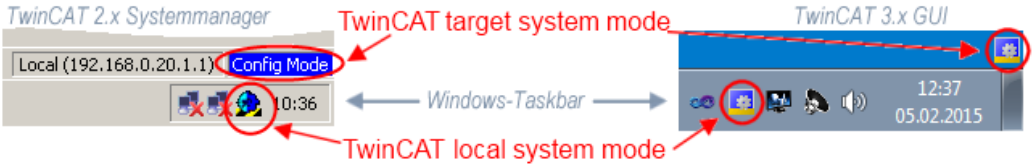


Fig. 110: Differentiation local/target system (left: TwinCAT 2; right: TwinCAT 3)

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

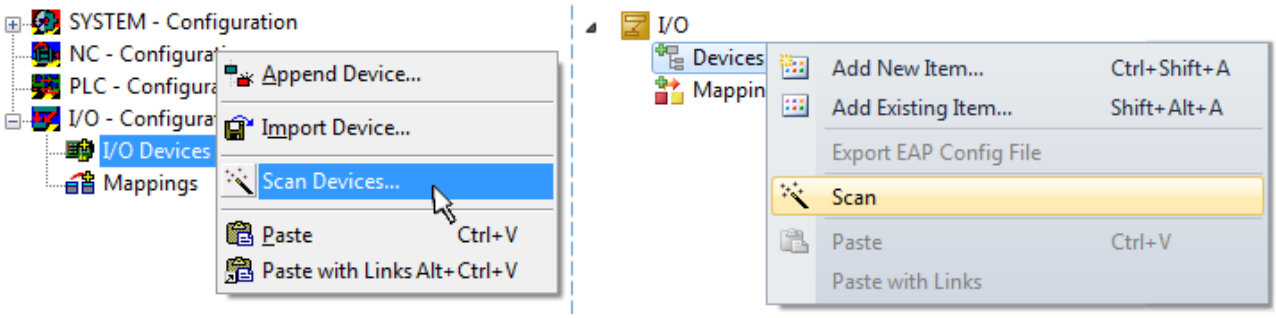


Fig. 111: Scan Devices (left: TwinCAT 2; right: TwinCAT 3)

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

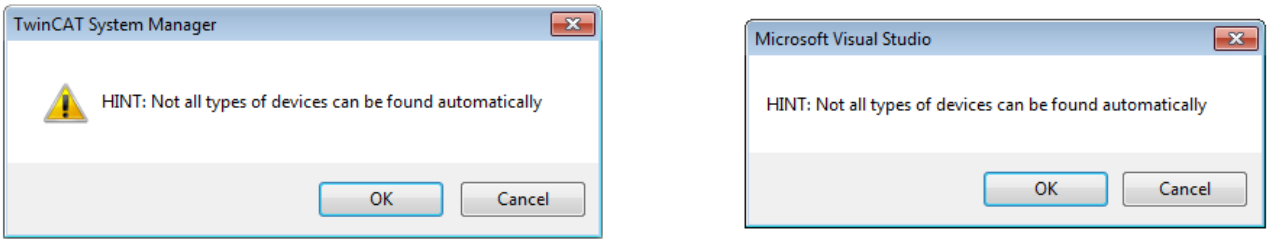


Fig. 112: Note for automatic device scan (left: TwinCAT 2; right: TwinCAT 3)

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” .

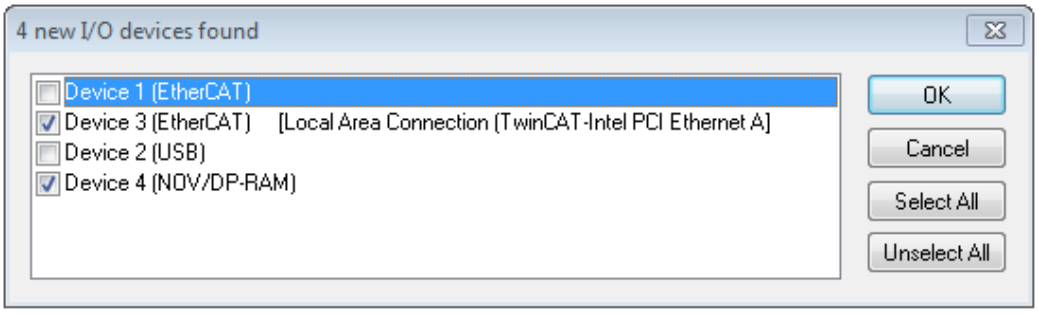


Fig. 113: Detected Ethernet devices

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 [▶ 86].

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.

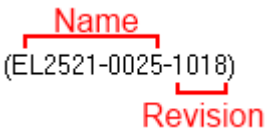


Fig. 114: Example default state

NOTE

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 [▶ 106] 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:

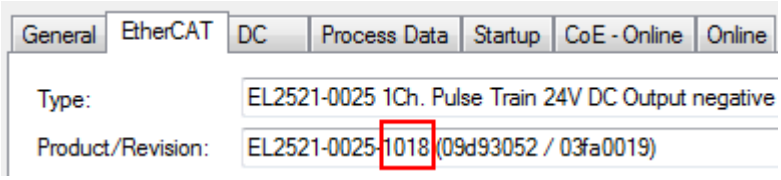


Fig. 115: Installing EthetCAT terminal with revision -1018

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 [▶ 106] 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:

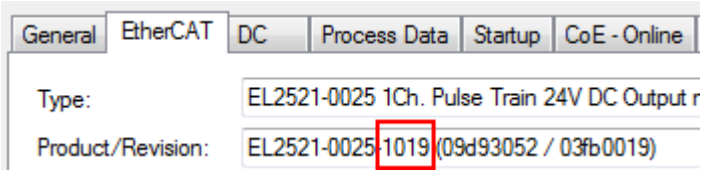


Fig. 116: Detection of EtherCAT terminal with revision -1019

This is usually not noticed by the commissioning engineers. TwinCAT cannot signal anything either, since virtually a new configuration is 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. Þ 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. 117: Scan query after automatic creation of an EtherCAT device (left: TwinCAT 2; right: TwinCAT 3)

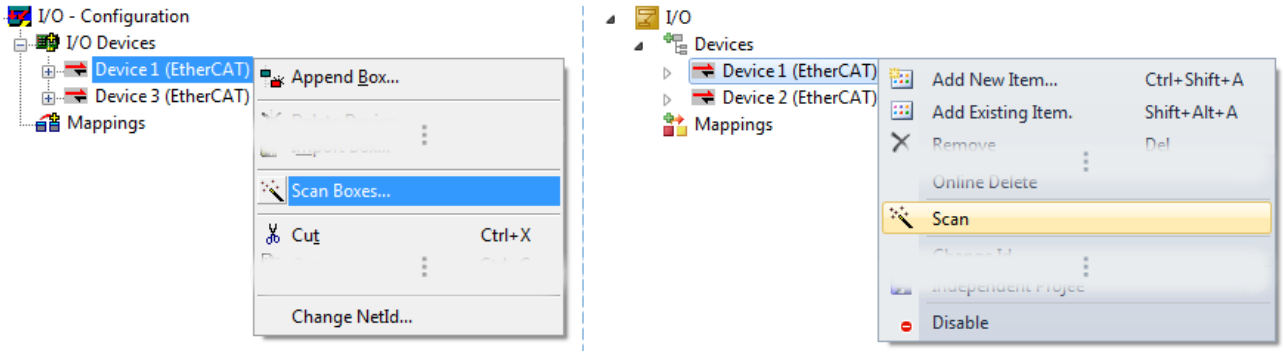


Fig. 118: Manual triggering of a device scan on a specified EtherCAT device (left: TwinCAT 2; right: TwinCAT 3)

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.

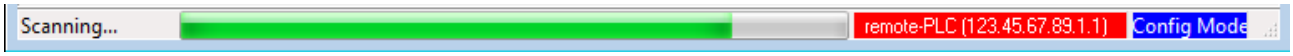


Fig. 119: *Scan progressexemplary by TwinCAT 2*

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

Fig. 120: 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).

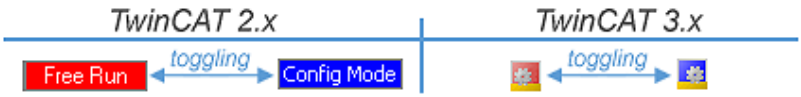


Fig. 121: Displaying of “Free Run” and “Config Mode” toggling right below in the status bar

Fig. 122: 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*.

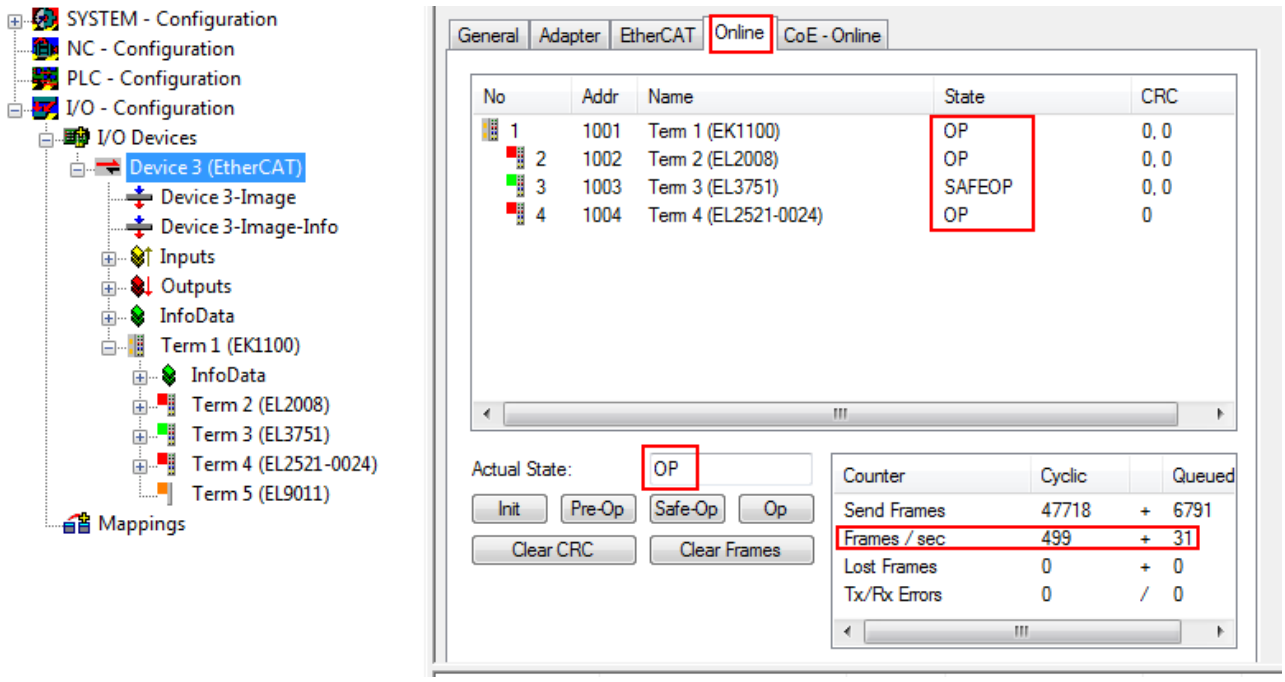


Fig. 123: Online display example

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 [▶ 96].

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.

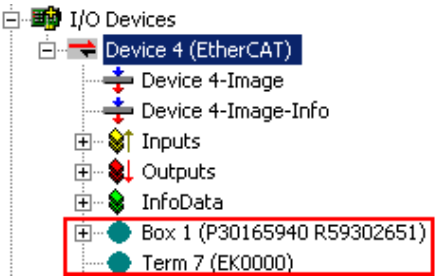


Fig. 124: Faulty identification

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

NOTE

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. 125: 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.

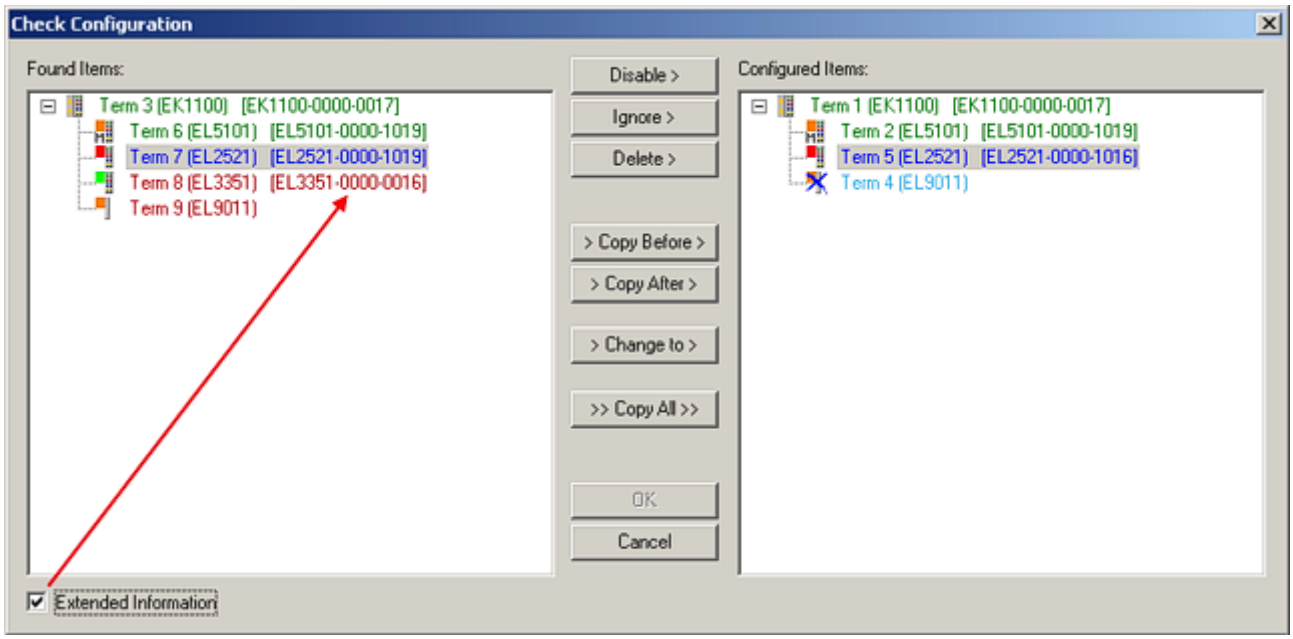


Fig. 126: Correction dialog

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	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. If the found revision is higher than the configured revision, the slave may be used provided compatibility issues are taken into account. 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.
light blue	This EtherCAT slave is ignored (“Ignore” button)
red	• This EtherCAT slave is not present on the other side. • 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.

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 >= 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.

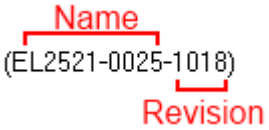


Fig. 127: Name/revision of the terminal

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, ...

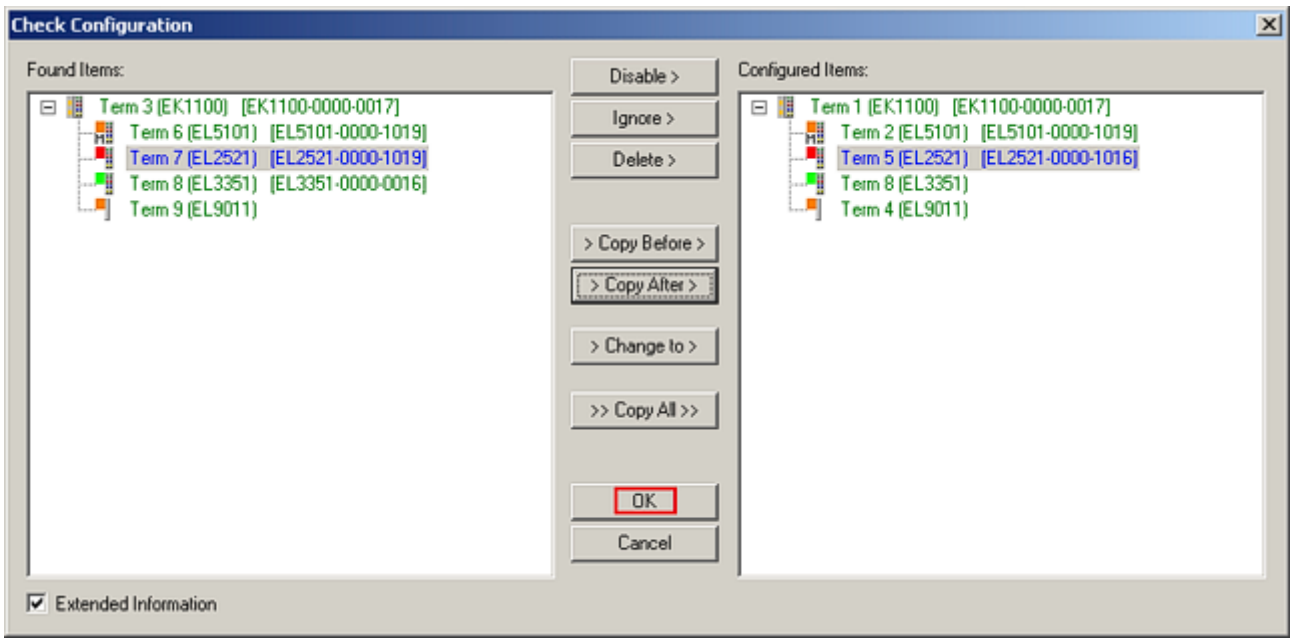


Fig. 128: Correction dialog with modifications

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.

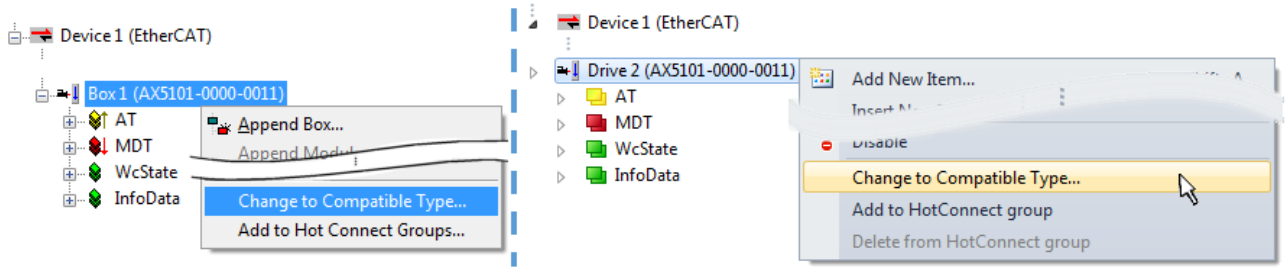


Fig. 129: Dialog “Change to Compatible Type…” (left: TwinCAT 2; right: TwinCAT 3)

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

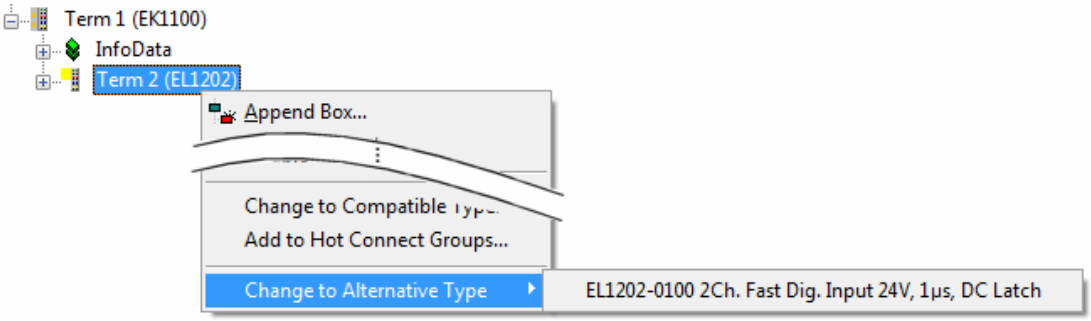


Fig. 130: TwinCAT 2 Dialog Change to Alternative Type

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).

5.2.7 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).

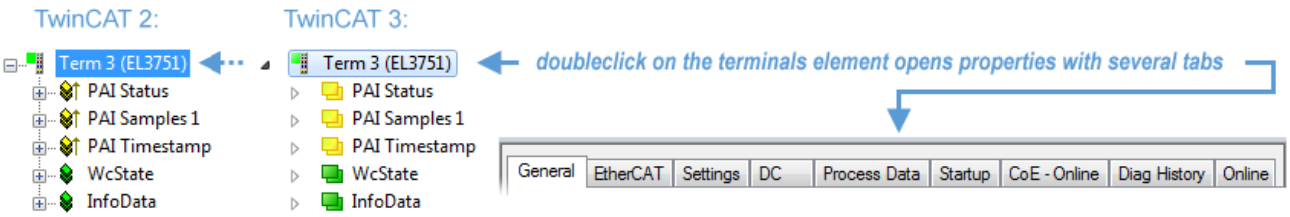


Fig. 131: Branch element as terminal EL3751

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

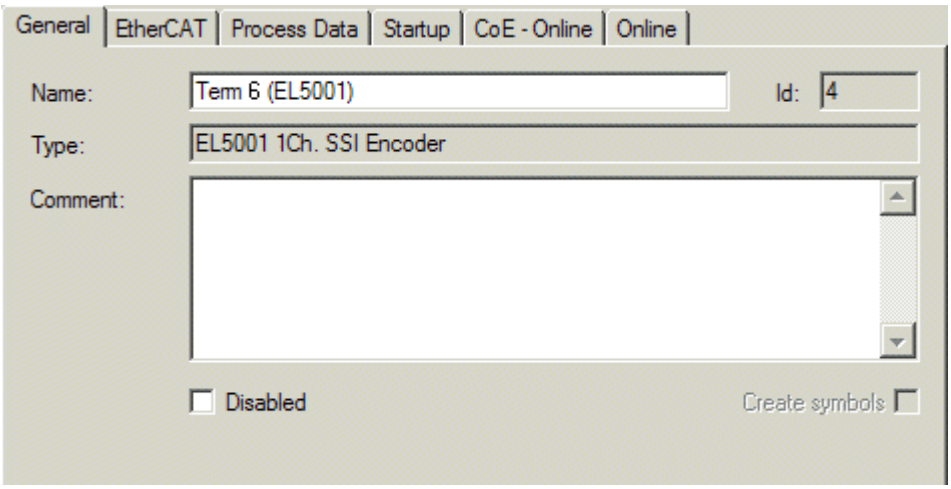


Fig. 132: “General” tab

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

“EtherCAT” tab

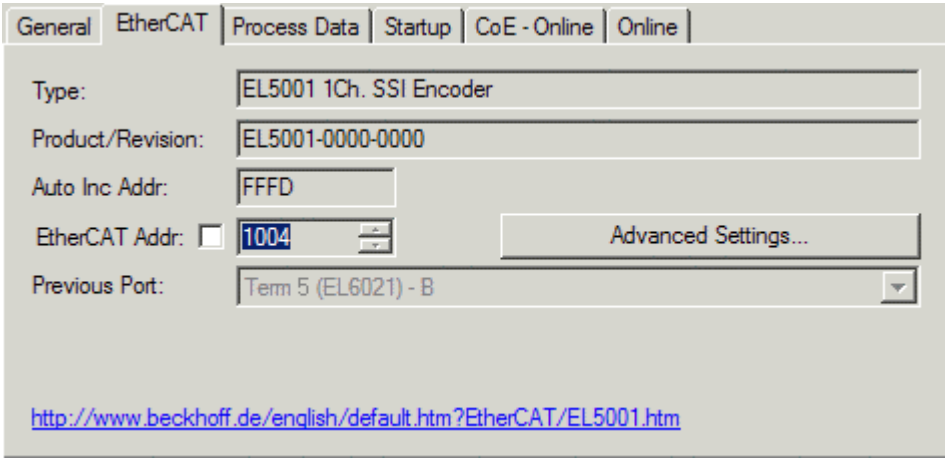


Fig. 133: “EtherCAT” tab

Type	EtherCAT device type
Product/Revision	Product and revision number of the EtherCAT device
Auto Inc Addr.	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_{hex}. For each further slave the address is decremented by 1 ($FFFF_{hex}$, $FFFE_{hex}$ etc.).
EtherCAT Addr.	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.
Previous Port	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.
Advanced Settings	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 (**P**rocess **D**ata **O**bjects, 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.

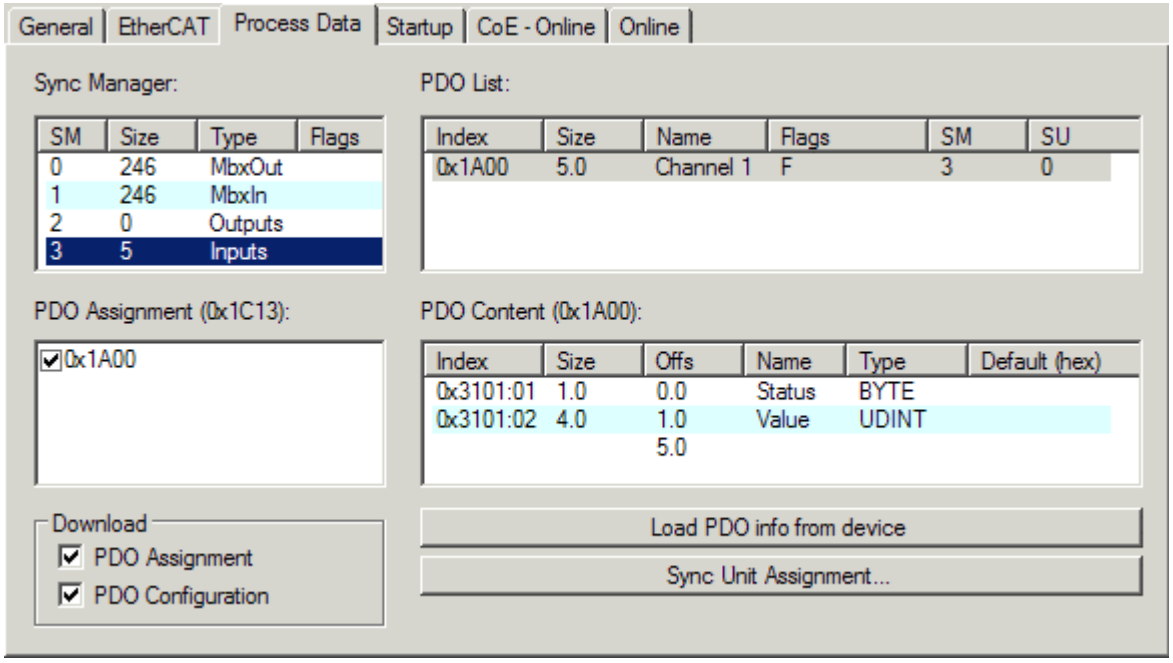


Fig. 134: “Process Data” tab

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”).

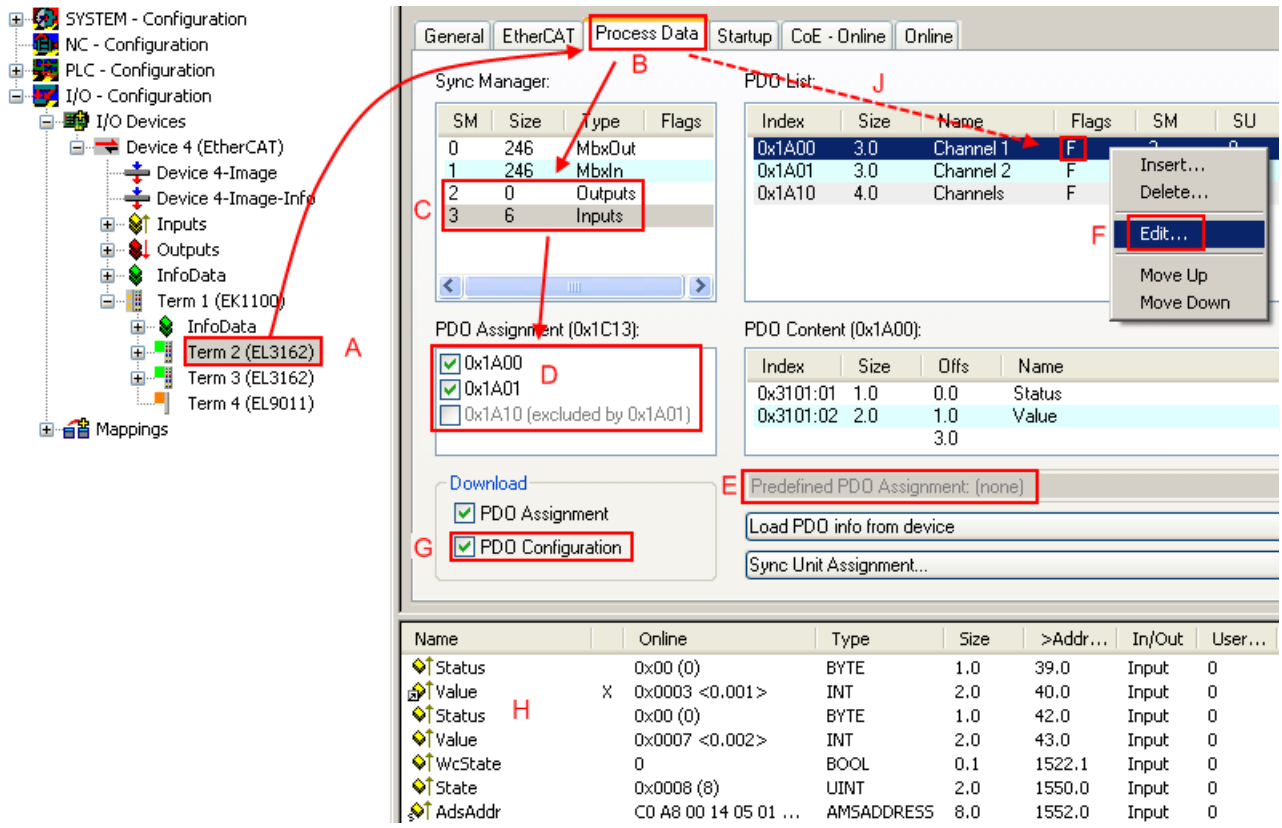


Fig. 135: Configuring the process data

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 [▶ 117] 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.

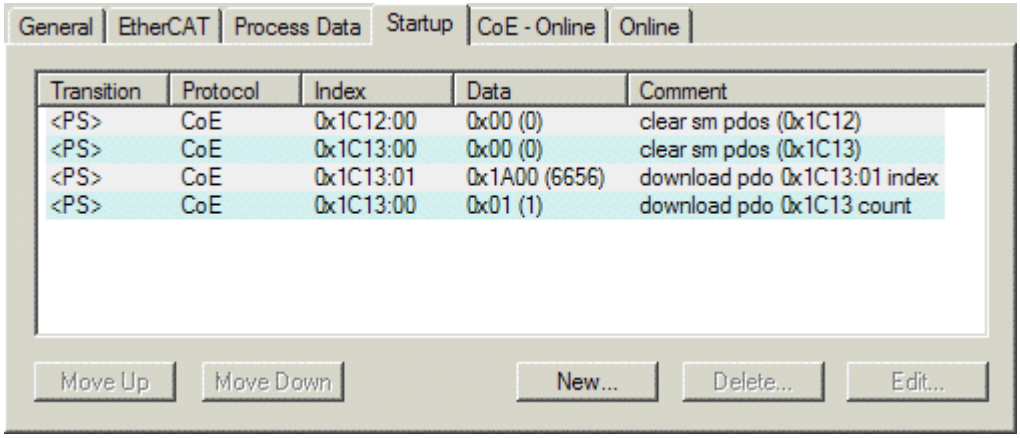


Fig. 136: “Startup” tab

Column	Description
Transition	Transition to which the request is sent. This can either be • the transition from pre-operational to safe-operational (PS), or • the transition from safe-operational to operational (SO). 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

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

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

Update List | Auto Update

Advanced... | All Objects

Index	Name	Flags	Value
1000	Device type	RO	0x00000000 (0)
1008	Device name	RO	EL5001-0000
1009	Hardware version	RO	V00.01
100A	Software version	RO	V00.08
1011:0	Restore default parameter	RW	> 1 <
1011:01	SubIndex 001	RW	0
1018:0	Identity object	RO	> 4 <
1018:01	Vendor id	RO	0x00000002 (2)
1018:02	Product code	RO	0x13893052 (327757906)
1018:03	Revision number	RO	0x00000000 (0)
1018:04	Serial number	RO	0x00000001 (1)
1A00:0	TxPDO 001 mapping	RO	> 2 <
1A00:01	Subindex 001	RO	0x3101:01, 8
1A00:02	Subindex 002	RO	0x3101:02, 32
1C00:0	SM type	RO	> 4 <
1C00:01	SubIndex 001	RO	0x01 (1)
1C00:02	SubIndex 002	RO	0x02 (2)
1C00:03	SubIndex 003	RO	0x03 (3)
1C00:04	SubIndex 004	RO	0x04 (4)
1C13:0	SM 3 PDO assign (inputs)	RW	> 1 <
1C13:01	SubIndex 001	RW	0x1A00 (6656)
3101:0	Inputs	RO P	> 2 <
3101:01	Status	RO P	0x41 (65)
3101:02	Value	RO P	0x00000000 (0)
4061:0	Feature bits	RW	> 4 <
4061:01	disable frame error	RW	FALSE
4061:02	enbale power failure Bit	RW	FALSE
4061:03	enable inhibit time	RW	FALSE
4061:04	enable test mode	RW	FALSE
4066	SSI-coding	RW	Gray code (1)
4067	SSI-baudrate	RW	500 kBaud (3)
4068	SSI-frame type	RW	Multitum 25 bit (0)
4069	SSI-frame size	RW	0x0019 (25)
406A	Data length	RW	0x0018 (24)
406B	Min. inhibit time[µs]	RW	0x0000 (0)

Fig. 137: “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.

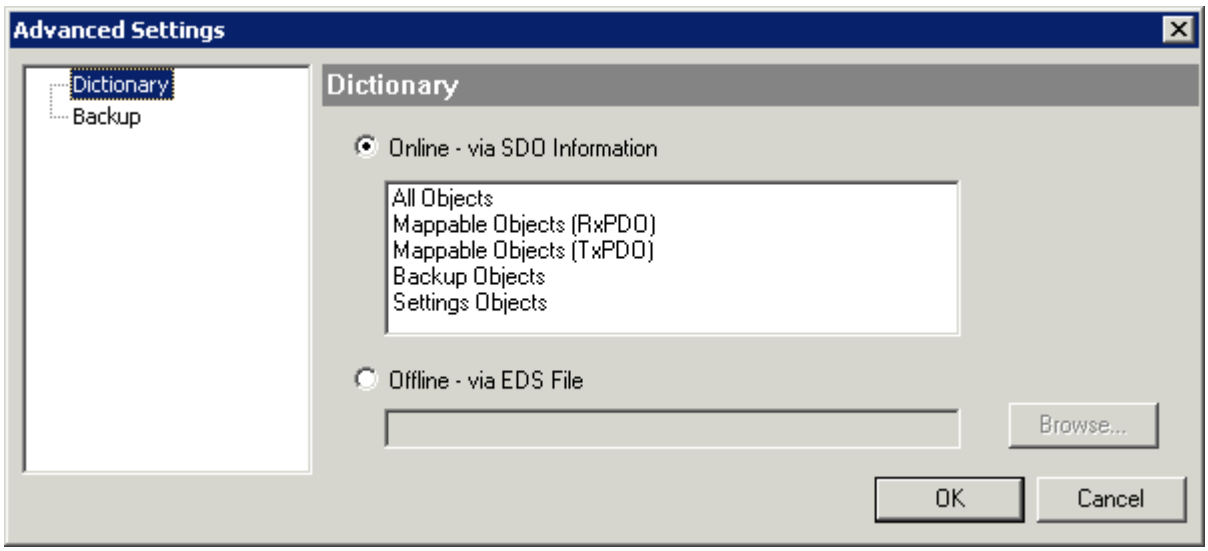


Fig. 138: Dialog “Advanced settings”

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

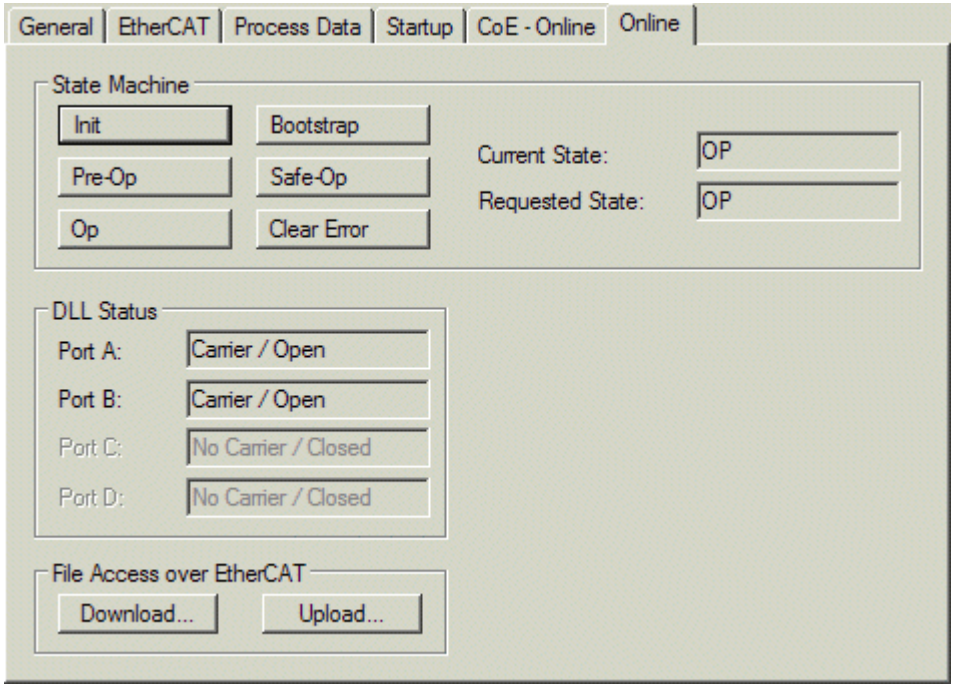


Fig. 139: “Online” tab

State Machine

Init	This button attempts to set the EtherCAT device to the *Init* state.
Pre-Op	This button attempts to set the EtherCAT device to the *pre-operational* state.
Op	This button attempts to set the EtherCAT device to the *operational* state.
Bootstrap	This button attempts to set the EtherCAT device to the *Bootstrap* state.
Safe-Op	This button attempts to set the EtherCAT device to the *safe-operational* state.
Clear Error	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 *Clear Error* button is pressed the error flag is cleared, and the current state is displayed as PREOP again.
Current State	Indicates the current state of the EtherCAT device.
Requested State	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

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

"DC" tab (Distributed Clocks)

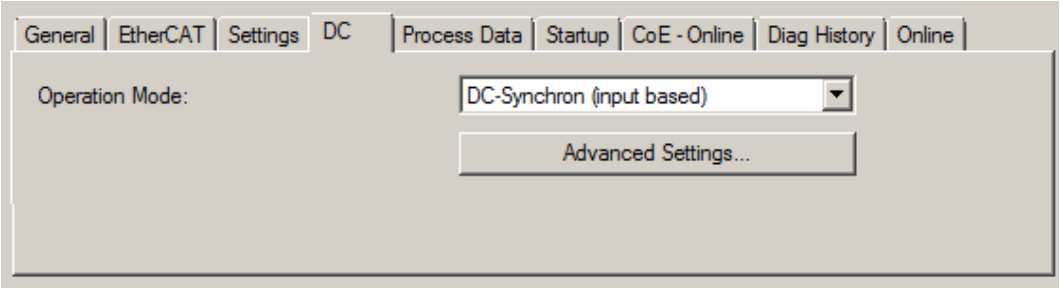


Fig. 140: "DC" tab (Distributed Clocks)

Operation Mode	Options (optional): • FreeRun • SM-Synchron • DC-Synchron (Input based) • DC-Synchron
Advanced Settings…	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

5.2.7.1 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.

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 [▶ 115]),

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 *PDO Assignment* 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 [▶ 112] 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.

5.3 General Notes - EtherCAT Slave Application

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.

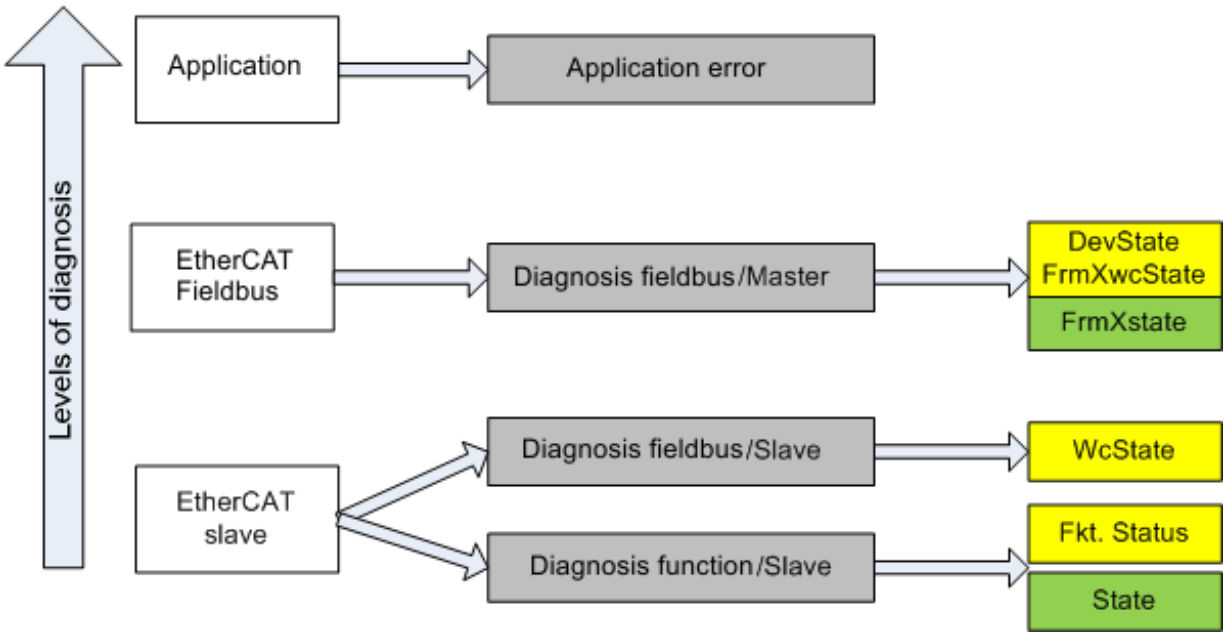


Fig. 141: Selection of the diagnostic information of an EtherCAT Slave

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.

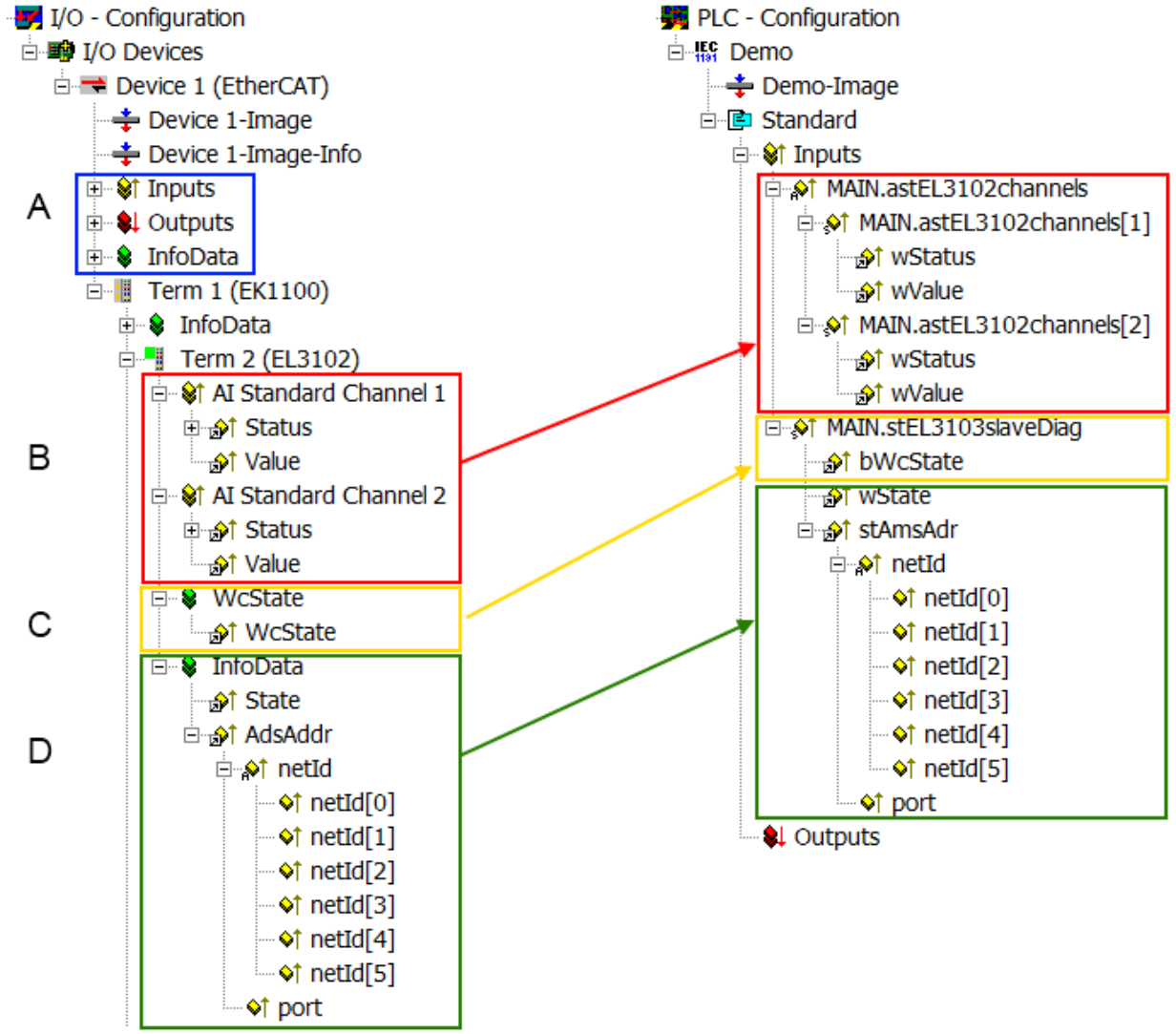


Fig. 142: Basic EtherCAT Slave Diagnosis in the PLC

The following aspects are covered here:

Code	Function	Implementation	Application/evaluation
A	The EtherCAT Master's diagnostic information updated acyclically (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: • CoE in the Master for communication with/through the Slaves • Functions from *TcEtherCAT.lib* • Perform an OnlineScan
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 • the bit significations may be found in the device documentation • other devices may supply more information, or none that is typical of a slave	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 • is only rarely/never changed, except when the system starts up • is itself determined acyclically (e.g. EtherCAT Status)	State current Status (INIT..OP) of the Slave. The Slave must be in OP (=8) when operating normally. *AdsAddr* 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 *port* (= 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.

NOTE

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*:

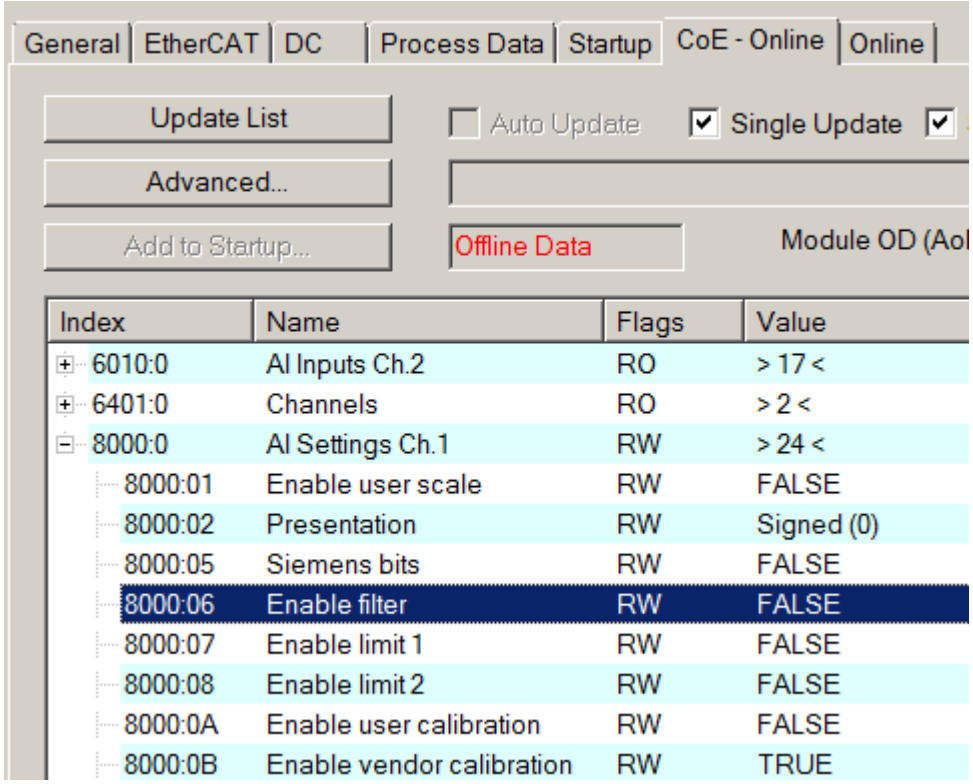


Fig. 143: EL3102, CoE directory

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.

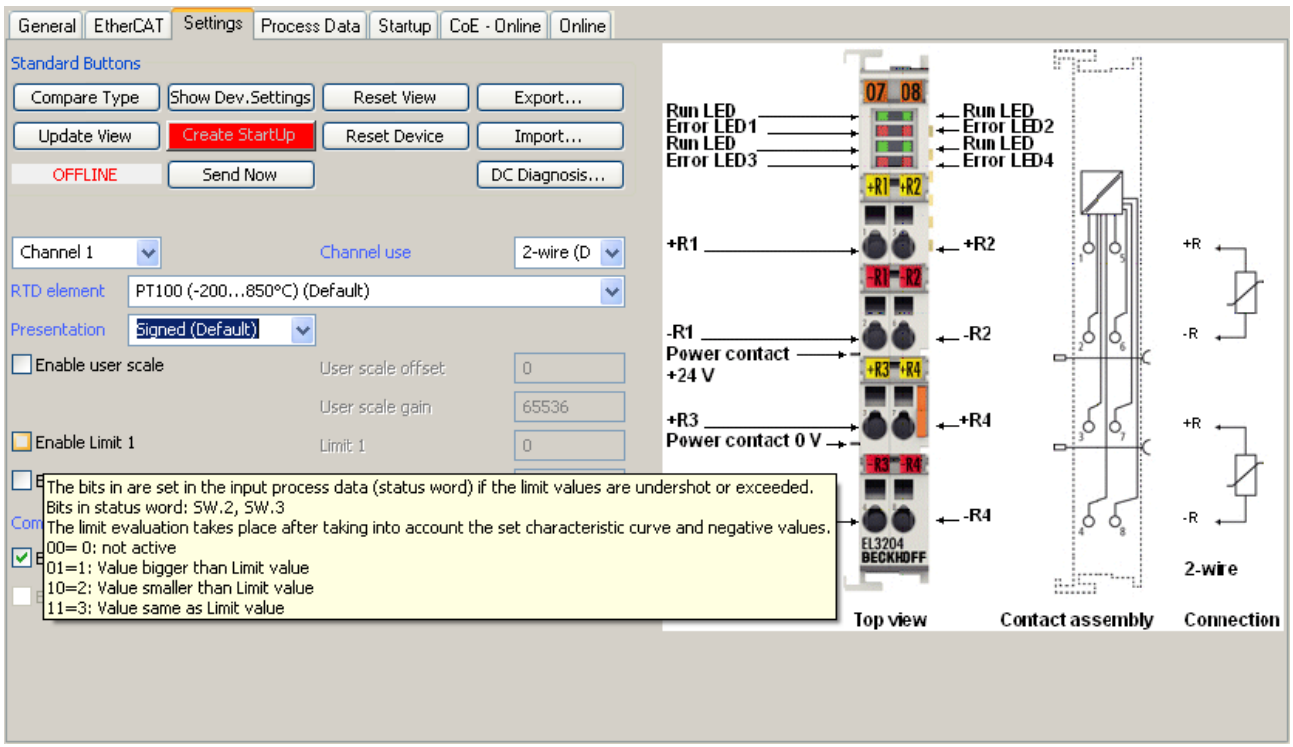


Fig. 144: Example of commissioning aid for a EL3204

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 [▶ 27]" 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.

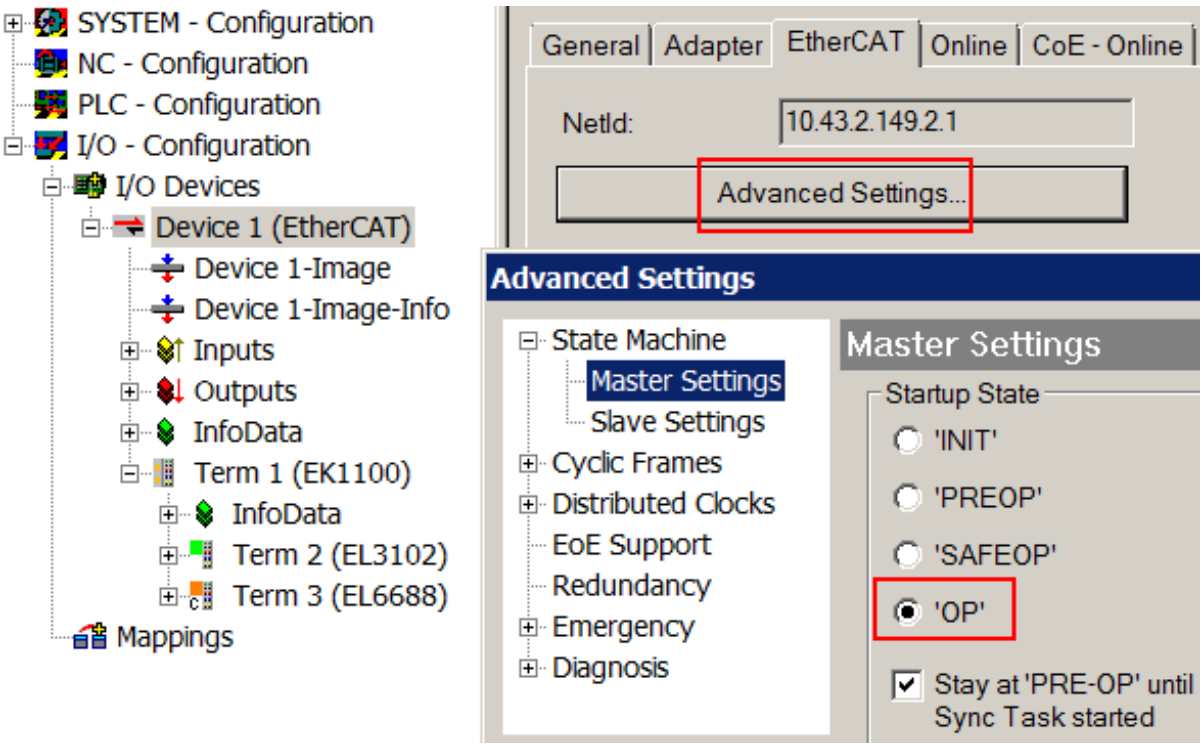


Fig. 145: Default behaviour of the System Manager

In addition, the target state of any particular Slave can be set in the “Advanced Settings” dialogue; the standard setting is again OP.

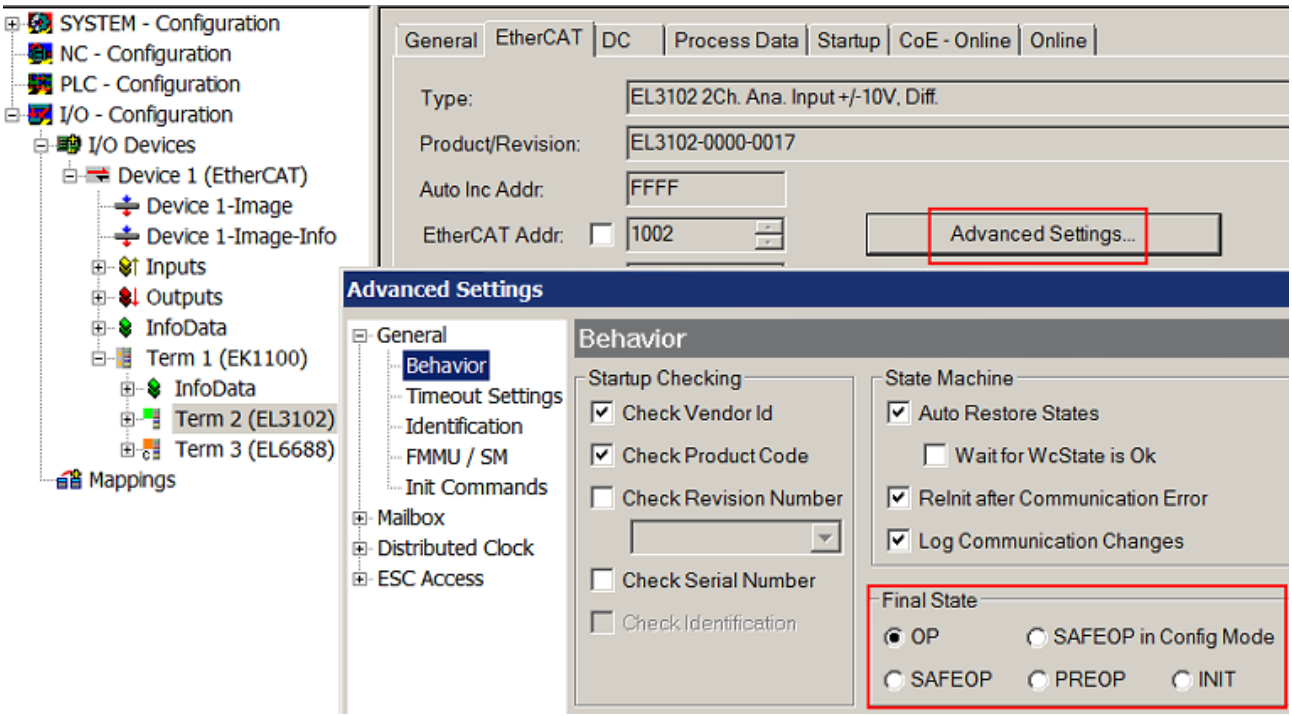


Fig. 146: Default target state in the Slave

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.

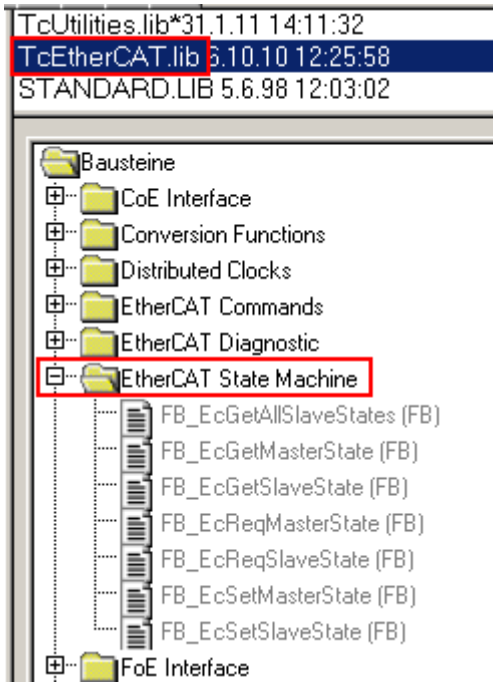


Fig. 147: PLC function blocks

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 | EtherCAT | 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. 148: 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:

Message
E-Bus Power of Terminal 'Term 3 (EL6688)' may to low (-240 mA) - please check!

Fig. 149: Warning message for exceeding E-Bus current

NOTE

Caution! Malfunction possible!

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

5.4 Operating modes and process data

Version notes

The single-channel EL60x1 has been developed and enhanced over a number of years. Through further development of the EL6001/EL6021 the following functional extensions have been implemented:

- from firmware (FW) 05 / hardware (HW) 03, (EL6001); firmware (FW) 04 / hardware (HW) 03, (EL6021) the objects for status monitoring and parameterization are also available from index 0x6000 (profile-specific objects) and can be parameterized in the TwinCAT System Manager, depending on the hardware.
- from firmware (FW) 06 / hardware (HW) 03, (EL6021) the command mode [▶ 131] is supported
- from firmware 08 (EL6001) the 16-bit process data interface for sending/receiving of > 8 bit is supported
- from firmware 11 (EL6001) all integer baud rates 1000…115200 baud can be used.
-

Older versions of the EL6001/EL6021 do not support this functionality!

from FW/HW	ESI	Control/status/parameterizing objects area
EL6001 01/01 EL6021 01/01	from EL6001-0000-0000 from EL6021-0000-0000	Index 0x300n:01 (Control-Word) Index 0x310n:01 (Status-Word) Index 0x4070 (Data bytes in send buffer) Index 0x4071 (Data bytes in receive buffer) Index 0x4072 (Diagnosis) Index 0x4073 (Baud rate) Index 0x4074 (Data frame) Index 0x4075 (Feature bits)
EL6001 05/03 EL6021 04/03	from EL6001-0000-0016 from EL6021-0000-0016	in addition to the above described objects: Index 0x6000 (COM inputs) Index 0x7000 (COM outputs) Index 0xA000 (COM Diag data Index 0x8000 (COM settings)
EL6021 06/03	from EL6021-0000-0018	in addition to the above described objects: Index 0xB000 (Command)
EL6001 08/03	from EL6001-0000-0019	in addition to the above described objects: Objects for 16-bit PDO
EL6001 11/11	from EL6001-0000-0020	All baud rates 1000…115200 via explicitBaudrate implemented

The EL6002/EL6022 devices with revision -0016 already feature objects in the 6000, 7000 and 8000 range from the first release.

Compatibility in the case of service

Example: An EL6001/EL6021 employed and projected with Hardware 03 or higher cannot be replaced by an EL6001/EL6021 with an older hardware version (<03)! The reverse case can be implemented without problem!

Process data monitoring

- **WcState**: if ≠ 0, this EtherCAT device does not take part in the process data traffic
- **State**: if ≠ 8, the EtherCAT device is not in OP (operational) status
- **TxPDO state, SyncError:** if ≠ 0, then no valid process data are available, e.g. caused by broken wire
- **TxPDO Toggle**: if this bit is toggling, a new set of process data is available

StartUp entries (hardware < 03)

StartUp list entry

For the EL6001/EL6021 with hardware < 03, the startUp entries can only be set in the transition from SafeOP to OP (S -> O). The default setting is PreOP -> SafeOP (P -> S). When creating StartUp entries, ensure that the checkbox "S -> O" is ticked (see Fig.)!

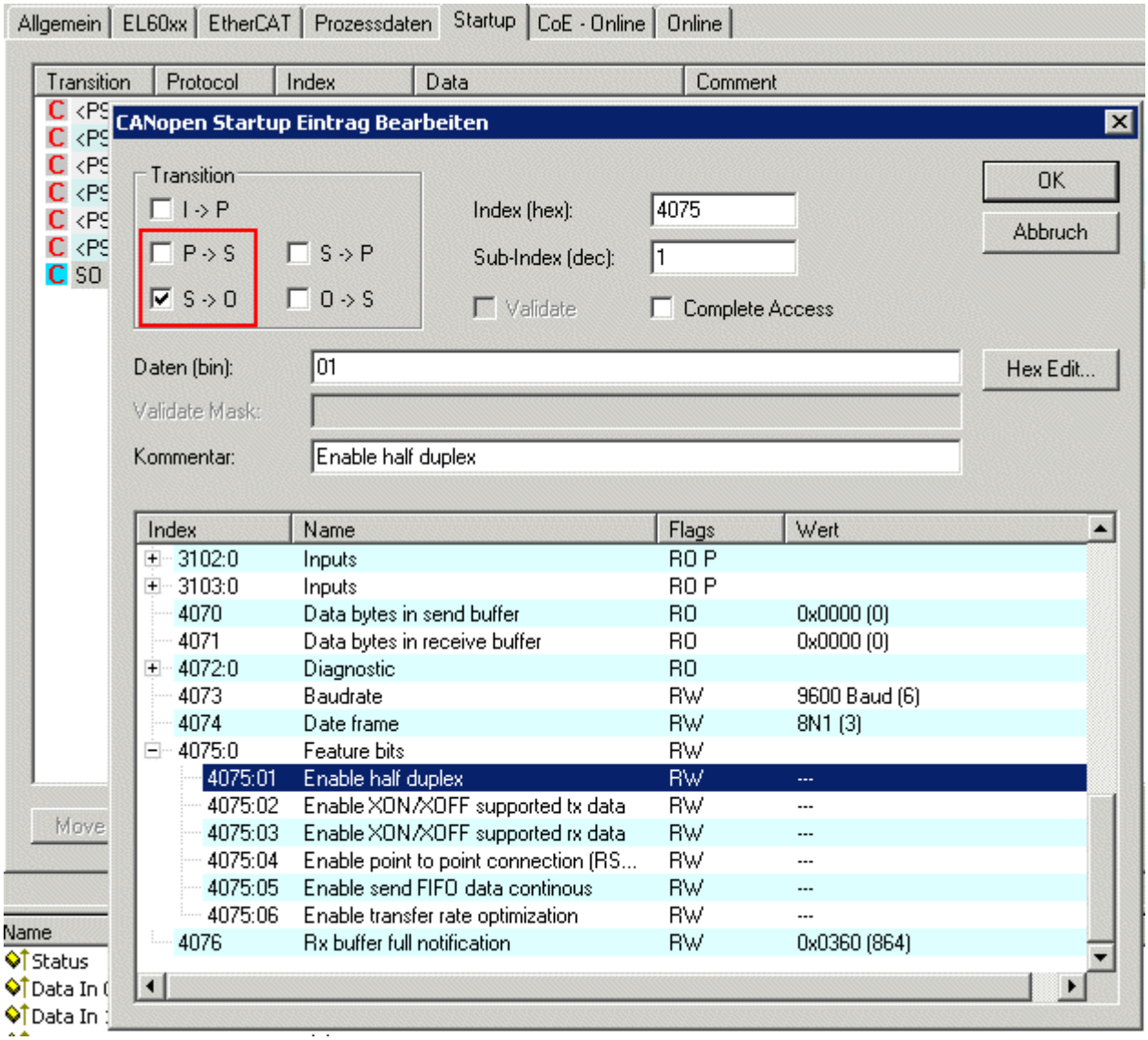


Fig. 150: *StartUp entry with transition S -> O*

Process data

As delivered, 22 bytes of user data and 1 control/status word are transferred.

The process data are generated from CoE objects 0x6000 (Inputs) and 0x7000 (Outputs) and are described in chapter Object description and parameterization [▶ 172] in detail.

Enlarged process image 50x 16-bit

An enlarged process data interface is necessary for communication with >8 data bits. If the terminal supports this (see Communication features [▶ 135]) a 50-word interface can be set as an alternative to the 22-byte interface (PDOs 0x1605 and 0x1A05). This can be used with every encryption regulation (7xx, 8xx), but only makes sense with a regulation >8 bits, e.g. 9N1. In each case the least significant bits must be occupied by the data bits, i.e. in the case of 9N1 the 9 least significant bits in the data words.

Process data of the EL60x2 from revision -0016

EL60x2 from revision -0016

Sync Manager (SM) – PDO Assignment

SM2, PDO assignment 0x1C12				
Index	**Index of excluded PDOs**	**Size (byte.bit)**	**Name**	**PDO content**
0x1600	-	24.0	COM Outputs Channel1	Index 0x7000:01 - Ctrl__Transmit request
				Index 0x7000:02 - Ctrl__Receive accepted
				Index 0x7000:03 - Ctrl__Init request
				Index 0x7000:04 - Ctrl__Send continous
				Index 0x7000:09 - Ctrl__Output length
				Index 0x7000:11 - Data out 0
				--
				Index 0x7000:26
0x1601	-	24.0	COM Outputs Channel2	Index 0x7010:01
				Index 0x7010:02
				Index 0x7010:03
				Index 0x7010:04
				Index 0x7010:09
				Index 0x7010:11
				--
				Index 0x7010:26
0x1604 (default)	-	24.0	COM Outputs Channel1	Index 0x7001:01 - Ctrl
				Index 0x7000:11 - Data out 0
				--
				Index 0x7000:26
0x1605 (default)	-	24.0	COM Outputs Channel2	Index 0x7011:01 - Ctrl
				Index 0x7010:11 - Data out 0
				--
				Index 0x7010:26

SM2, PDO assignment 0x1C13				
Index	**Index of excluded PDOs**	**Size (byte.bit)**	**Name**	**PDO content**
0x1A00	-	24.0	COM Inputs Channel1	Index 0x6000:01 - Status__Transmit accepted
				Index 0x6000:02 - Status__Receive request
				Index 0x6000:03 - Status__Init accepted
				Index 0x6000:04 - Status__Buffer full
				Index 0x6000:05 - Status__Input length
				Index 0x6000:06 - Status__Framing error
				Index 0x6000:07 - Status__Overrun error
				Index 0x6000:09 - Status__Input length
				Index 0x6000:11 - Data in 0
				--
				Index 0x6000:26
0x1A01	-	24.0	COM Inputs Channel2	Index 0x6010:01
				Index 0x6010:02
				Index 0x6010:03
				Index 0x6010:04 - Status__Buffer full
				Index 0x6010:05
				Index 0x6010:06
				Index 0x6010:07
				Index 0x6010:09
				Index 0x6010:11
				--
				Index 0x6010:26
0x1A04(default)	-	24.0	COM Inputs Channel1	Index 0x6001:01
				Index 0x6000:11
				--
				Index 0x6000:26
0x1A05(default)	-	24.0	COM Inputs Channel2	Index 0x6011:01
				Index 0x6010:11
				--
				Index 0x6010:26

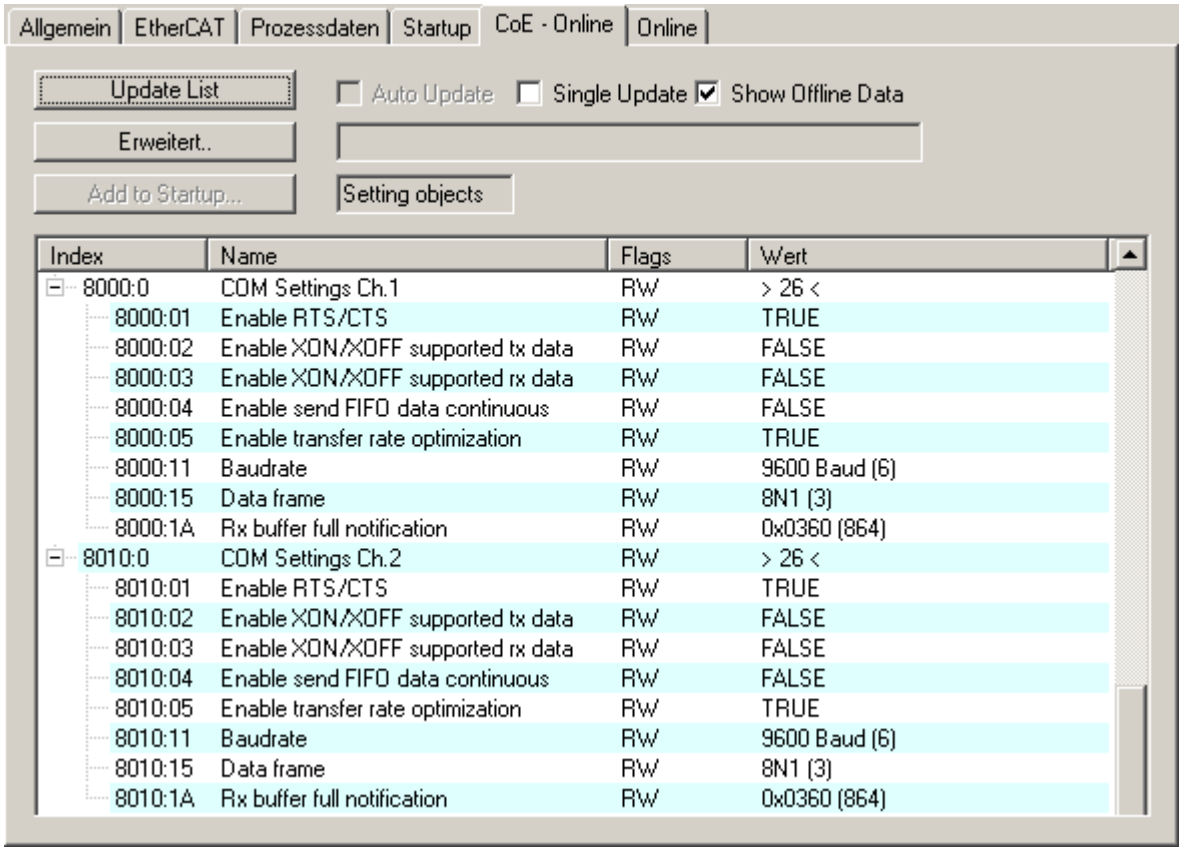
Features and application notes

Optional features are generally set in the CoE directory (CAN over EtherCAT, index 0x80n0).

Parameterization via the CoE list (CAN over EtherCAT)

Please note the following general CoE information when using/manipulating the CoE parameters: - Keep a startup list if components have to be replaced - Differentiate between online/offline dictionary, ensure existence of current XML description - use “CoE reload” for resetting changes Please note especially for the EL60xx, if any changes are made to the communication settings (baud rate, data frame, feature bits), an InitRequest via the control word is required to apply the changes.

The following CoE settings are possible from object 0x8000 and are shown below in their default settings:



Index	Name	Flags	Wert
8000:0	COM Settings Ch.1	RW	> 26 <
8000:01	Enable RTS/CTS	RW	TRUE
8000:02	Enable XON/XOFF supported tx data	RW	FALSE
8000:03	Enable XON/XOFF supported rx data	RW	FALSE
8000:04	Enable send FIFO data continuous	RW	FALSE
8000:05	Enable transfer rate optimization	RW	TRUE
8000:11	Baudrate	RW	9600 Baud (6)
8000:15	Data frame	RW	8N1 (3)
8000:1A	Rx buffer full notification	RW	0x0360 (864)
8010:0	COM Settings Ch.2	RW	> 26 <
8010:01	Enable RTS/CTS	RW	TRUE
8010:02	Enable XON/XOFF supported tx data	RW	FALSE
8010:03	Enable XON/XOFF supported rx data	RW	FALSE
8010:04	Enable send FIFO data continuous	RW	FALSE
8010:05	Enable transfer rate optimization	RW	TRUE
8010:11	Baudrate	RW	9600 Baud (6)
8010:15	Data frame	RW	8N1 (3)
8010:1A	Rx buffer full notification	RW	0x0360 (864)

Fig. 151: *“CoE - Online, EL60x4 terminals” tab*

Transfer rates

The terminal has a process image of 22 bytes of user data. It possible to transmit or receive these 22 bytes every second cycle at the most.
The data is transferred from the terminal to the controller in the first cycle. In the second cycle, the controller must acknowledge that it has accepted the data.
Therefore, if the cycle time is 10 ms, 50 times 22 bytes can be transmitted per second.

With a set data frame of 8N1, each transmitted byte consists of a start bit, eight data bits and a stop bit. This is equivalent to 10 bits per user byte.

With the above mentioned settings, a continuous data transfer rate of:
- 50[1/s]*22[Byte]*10[Bit] = 11000 bps

can be achieved.

The next lower baud rate is 9600 baud. Accordingly, continuous transfer at a maximum baud rate of 9600 can be secured with a cycle time of 10 ms.

From firmware 11 the CoE 0x8000:1B supports with *explicitBaudrate* all baud rates 1000…115200 baud. When the baud rate is selected, the compatibility of the remote station to the set baud rate must be taken into account.
If only low quantities of data are to be transmitted or received sporadically (e.g. bar code scanner) the baud rate can also be set higher, or the cycle time can be enlarged.
If the controller cannot fetch the data quickly enough from the terminal, the data will be stored intermediately in the internal buffer of the terminal. The buffer for received data has a size of 864 bytes. If this is exhausted, all further data will be lost.

Another scenario: the controller transfers significantly more data to the terminal than the latter can transmit. For a ‘baud rate’ setting of 300 and a ‘data frame’ setting of 8N1, the terminal can only transmit 30 bytes per second. However, if more than these 30 bytes per second are received, a 128 byte transmit buffer will be written to first in this case also. Once this is full, all further data will be lost.

Optimization of transfer rates

In normal operating mode the data received will be adopted immediately into the process image. In order to enable a continuous flow of data, the 'Enable transfer rate optimization' option in the Settings object is activated as standard. Due to this switch, the data will first be stored intermediately in the receive buffer (864 bytes).

The data will only be copied into the process image if no further character is received for 16 bit periods or if the buffer is full.

Continuous transmission of data

Usually the EL60xx terminal automatically decides when to send the data bytes contained in the buffer. For many applications it is helpful to have a continuous data stream. For this purpose, the Beckhoff EL60xx terminals feature the 'Enable send FIFO data continuous' setting in the Settings object. If this switch is set.

- The internal send buffer (128 bytes) must be filled first. To this end, data will be sent from the controller to the terminal as in a normal data transfer.
- Data transfer from the buffer commences with a rising edge of the bit "Send continuous"
- If the data has been transferred, the terminal informs the controller by setting the "InitAccepted" bit. "Init accepted" is cleared with "Send continuous".

This setting enables up to 128 bytes to be transferred without long delays, even with slower EtherCAT cycle times and high baud rates.

The terminal tries to leave as little distance as possible between the telegrams by seamlessly following each stop bit with the next start bit. Nevertheless, in two-channel mode high baud rates may result in short pauses between telegrams. A 20-byte data block may be sent in two blocks of 5 and 15 or 7 and 13 telegrams, for example, or with a different distribution. In a two-channel terminal channel 1 is given priority.

Prioritization

Since received data normally cannot be repeated, it has a higher priority than data to be transmitted. Furthermore, the priority decreases as the channel number increases. Hence, the reception of data on channel 1 has the highest priority.

Command mode

From firmware 06 / revision -0018 the EL6021 supports the so-called command mode. Certain terminal functions can be used and controlled through optional combination and/or sequence of commands. The following functions are currently supported:

- Multi-data frame feature: (from firmware 06) change of coding during ongoing data communication (sending)
  To this end the send buffer of the terminal should be filled with the bytes to be sent (note the maximum buffer size). As soon as the send process is started via the control word, the first n bytes are sent based on coding A, the remaining bytes in the buffer are sent with coding B. The buffer can then be refilled and sent accordingly. Example: The first byte is sent with mark parity, the remaining bytes with space parity.
  Sequence:
  - After each startup/restart of the EL60xx EtherCAT slave the parameterization has to be repeated. The command function is not stored in the event of a power failure.
  - Activation of SendContinuous mode through writing of 0x1 after 0x8000:04
  - Activation of the multi-data frame feature through writing of 0x2001 after request 0xB000:01
    Check that 0xB000:02 = 0
  - Specify coding A through writing of 0x2100 + [date frame code] after 0xB000:01
    Example: 8E2 = 0x12 --> write value 0x2112
    Check that 0xB000:02 = 0
  - Specify n bytes to be sent in coding A through writing of 0x2200 + [n] after 0xB000:01
    Check that 0xB000:02 = 0

- Specify coding B through writing of 0x2300 + [date frame code] after 0xB000:01
  Check that 0xB000:02 = 0
- Start sending through rising edge of Ctrl *SendContinuous*
- Terminal signals completed transfer through *InitAccepted* = 1. *InitAccepted* is reset with *SendContinuous*.

  CmdNameMeaning0x2000 + ControlByteCmdControlByteBit 0: Enable MultiDataFrame feature Bit 1..7: do not use0x2100 + Valuefirst data frameselect preferred value from data frame table0x2200 + NoOfBytesNo of bytesNo of bytes transfered with first data frame0x2300 + Valuesecond data frameselect prefered value from data frame table

  Note: In this mode coding 8M1 and 8S1 is also supported, although this coding cannot be selected via 0x8000:15.

- Further features can be implemented on request

Data transfer examples

Initialization

Initialization is performed prior to the first transmission/reception. The terminal is thereby parameterized with the data from the corresponding Settings object.

Procedure:

1. Set “Init request” to 1
2. The terminal confirms successful initialization by setting “Init accepted”
3. Reset “Init request”
4. The terminal sets “Init accepted” to 0

The terminal is now ready for data exchange.

Data transmission from the controller to the terminal (send 2 characters)

1. Set “Output length” to 2
2. Fill “Data Out 0” and “Data Out 1” with user data
3. Change the state of “Transmit request”
4. The terminal acknowledges receipt of the data by changing the state of the “Transmit accepted” bit.

Data transmission from the terminal to the controller (receive characters)

1. The terminal indicates that there is new data in the process image by changing the state of the “Receive request” bit.
2. The number of bytes received is written in “Input length”
3. The controller acknowledges acceptance of the bytes by changing the state of “Receive request”

5.5 Hints regarding TcVirtualComDriver

In case the EL60xx is to cooperate with the TwinCAT Virtual Serial COM Driver, here are a few hints:

Usually the customer-specific higher-level Windows application sets the desired feature of the COM interface according to the application, e.g. 2400 baud and 7N2 coding. Therefore, the customer-specific entries made *before* parameterization in the CoE StartUp entries, the Settings dialogue, the VirtualComPort or the device CoE are usually irrelevant.

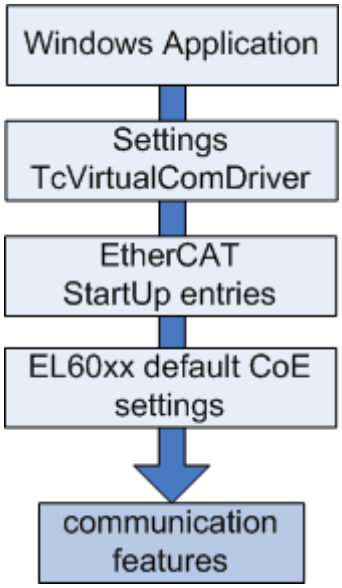


Fig. 152: *Each higher level (if available or activated) dictates the communication features to the level below it.*

NOTE

TcVirtualComDriver version

Check whether the desired Baud rate/coding combination of the TcVirtualComDriver version and EL/EP terminal used by you is also supported. If this is not the case:

- it may not be possible to open the COM port
- an ADS error message may appear in the ADS logger in the System Manager
- a CoE error message may appear in the ADS logger

The TcVirtualComDriver supports ExplicitBaudrate (CoE 0x80n0:1B) from version 1.18.

If the COM application has then set the features (such as baud rate) during the runtime, the correct setting can be checked against it in the Online CoE, see fig.

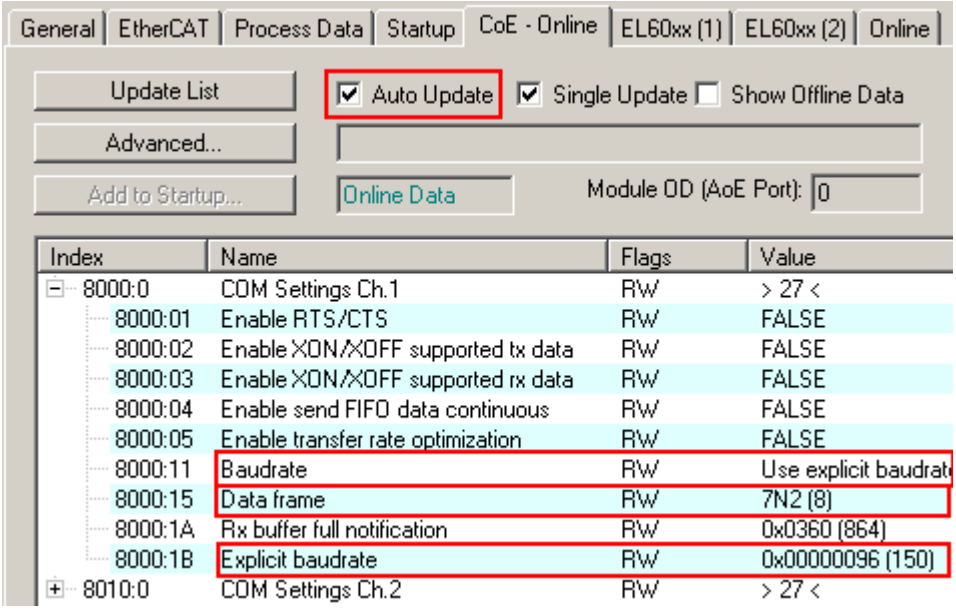


Fig. 153: *Checking the settings desired by the COM application in the CoE*

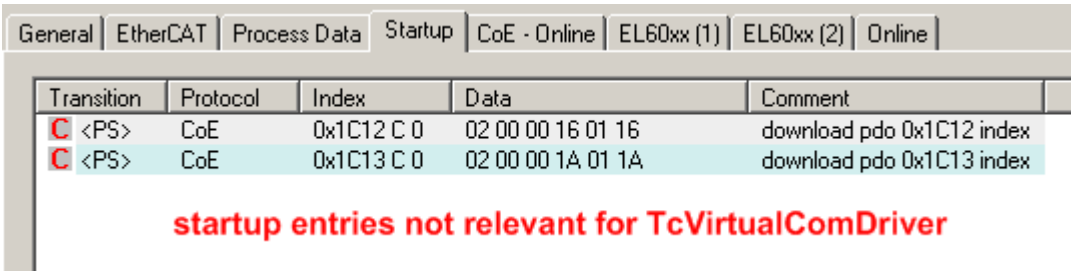


Fig. 154: *Default startup entries of a EL6002 (example) – only the default entries (in this case 0x1C12 and 0x1C13) are required*

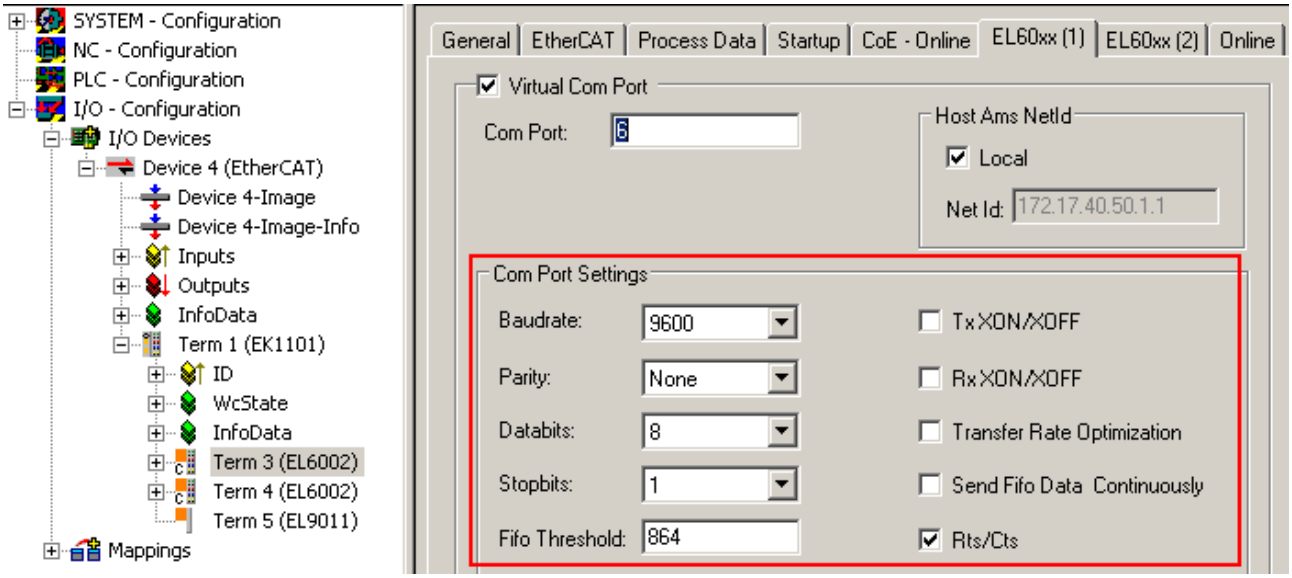


Fig. 155: *VirtualComDriver Settings (example)*

The settings are relevant only if no specification of the COM features is made or can be made from the application. The COM port is opened with these default settings.

Exception: *TransferRateOptimization, SendFifoDataContinously* and *FiFoThreshold* are to be set only from TwinCAT.

5.6 Communication features

Central features of the EL60xx devices for serial communication are baud rate and data telegram structure/ coding on the field side. The devices differ in the available features. For a clearer overview, the devices are listed here with the parameters available according to the firmware/hardware revision.

Check on the Beckhoff web page whether more up-to-date documentation is available.

Device overview						
	Available baud rate			Available coding		
Device	Baud rate [baud]	CoE Value [0x80n0:11]	from FW/HW	Coding	CoE Value [0x80n0:15]	from FW/HW
EL6001	2400	4		7E1	1	
	4800	5		7O1	2	
	9600	6		8N1	3	
	12000	14	from FW07	8E1	4	
	14400	15	from FW07	8O1	5	
	19200	7		7E2	9	
	38400	8		7O2	10	
	57600	9		8N2	11	
	115200	10		8E2	12	
	104167	Explicit Baudrate [0x8000:1B]	from FW08	8O2	13	
				9N1	32 Extended data frame [▶ 172] [0x8000:1C]	from FW08
	1000…115200 *)	Explicit Baudrate [0x8000:1B]	from FW11			
EL6002	110	Explicit Baudrate [▶ 133] [0x80n0:1B]	from FW03	7E1	1	
	150	Explicit Baudrate [▶ 133] [0x80n0:1B]	from FW03	7O1	2	
	300	1		8N1	3	
	600	2		8E1	4	
	1200	3		8O1	5	
	2400	4		7N2	8	from FW03
	4800	5		7E2	9	
	9600	6		7O2	10	
	19200	7		8N2	11	
	38400	8		8E2	12	
	57600	9		8O2	13	
	115200	10				

Device overview						
	Available baud rate			Available coding		
Device	Baud rate [baud]	CoE Value [x80n0:11]	from FW/HW	Coding	CoE Value [x80n0:15]	from FW/HW
EL6021	2400	4		7E1	1	
	4800	5		7O1	2	
	9600	6		8N1	3	
	19200	7		8E1	4	
	38400	8		8O1	5	
	57600	9		7E2	9	
	115200	10		7O2	10	
				8N2	11	
				8E2	12	
				8O2	13	
				8S1	18	Command mode from FW06
				8M1	19	Command mode from FW06
EL6022	300	1		7E1	1	
	600	2		7O1	2	
	1200	3		8N1	3	
	2400	4		8E1	4	
	4800	5		8O1	5	
	9600	6		7E2	9	
	19200	7		7O2	10	
	38400	8		8N2	11	
	57600	9		8E2	12	
	115200	10		8O2	13	
EP6022	300	1		7E1	1	
	600	2		7O1	2	
	1200	3		8N1	3	
	2400	4		8E1	4	
	4800	5		8O1	5	
	9600	6		7E2	9	
	19200	7		7O2	10	
	38400	8		8N2	11	
	57600	9		8E2	12	
	115200	10		8O2	13	

*) any integer baud rate can be set as an int32 value. The frequency error across all baud rates is < 0,6 %.

5.7 LIN Master Feature EL6001

Description of “LIN master support”

The EL6001 realizes a bus interface for RS232 levels. From software version 10, the EL6001 features an auxiliary function for sending and receiving LIN telegrams. This function does not cover the formal treatment of the protocol layers for LIN (“protocol stack”) or a physical LIN connection. Instead, the EL6001 can deal with the prolonged start/end mark of a LIN telegram in write and read direction.

”Above” the terminal a LIN stack is required that is implemented in the PLC, for example, below the terminal an RS232 <-> 5/12 V level converter is required.

Before using this function, please check the suitability in your system, since the EL6001 does not have a fully comprehensive LIN implementation!

A complete LIN frame results from a master query with directly following slave response. As master operating, the terminal sends the master frame and receives the slave data. With activated LIN functionality of the EL6001 (since FW10) a “sync break” and a “sync byte” is always output before the “protected identifier” (PID). In fact this terminal is therefore able to operate as a data receiving slave, too, but not as a data sending slave node.

Activation

The addition of a “sync break” and “sync field” for sending, which is required for LIN, can be activated via the “Command” object. To this end, enter the value 0x3000 in the CoE object 0xB000:01 [▶ 168]. The response can be read in CoE object 0xB000:03 with the values 0x01 0x00 0x00 0x4C 0x49 0x4E (4C 49 4E = ASCII “LIN”) as confirmation of the activation of this function. The terminal is then ready for receiving frames on the LIN bus containing the 13-bit “sync break” and the “sync field” (0x55), and to make only the following information in the receive data and to output these IDs during sending.

This setting is not permanently stored in the terminal and has to be written again after each new startup. The startup [▶ 112] list can be used to circumvent this.

User-specific baud rate

Any already available baud rate and or the baud rate 10417 can be entered in the object “Explicit baudrate” (0x8000:1B).

Further parameters

The following settings within CoE object 0x8000 [▶ 150] are necessary to ensure correct function in LIN mode:

Index:Subindex	Name	Value
0x8000:01	Enable RTS/CTS	FALSE
0x8000:02	Enable XON/XOFF supported tx data	FALSE
0x8000:03	Enable XON/XOFF supported rx data	FALSE
0x8000:04	Enable send FIFO data continuous	FALSE
0x8000:05	Enable transfer rate optimization	TRUE
0x8000:15	Dataframe	3 (8N1)
0x8000:1A	Rx buffer full notification	See objects for commissioning [▶ 150].

Operation

The application of the control and status words remains unchanged in LIN mode.

In LIN mode, the EL6001 automatically precedes the user data with the “Break field” and the “Sync byte” field. During receiving these two fields are automatically removed.

The process image should be used as follows:

Process data	Contents
Ctrl.Output length	Number of user data bytes (n) + 2
Data Out 0	Protected ID field (PID = Protected identifier field)
Data Out 1	Data byte 1
Data Out n	Data byte n
Data Out n+1	Checksum

EL6001 LIN example

The following LIN communication example is intended to illustrate that participation of a PLC controller via the EL6001 terminal in a LIN cluster is possible by activating the complementary LIN functionality via the 0xB000 CoE “command” object in combination with a physical conversion of RS-232 to the LIN bus.

Hardware connection based on a UART-LIN converter

The implementation of the physical layer as a basic prerequisite for LIN communication includes the provision of one-wire bus with corresponding signal levels 0V / 12V. This is best done with an RS232-LIN converter, which is connected to a sub-D connector and on the other side provides three poles for ground, power supply and the electrical connection to the one-wire LIN-bus.

The use of an RS232-LIN converter requires a sub-D 9-pin connector to be connected to the EL6001 terminal, as shown in the diagram:

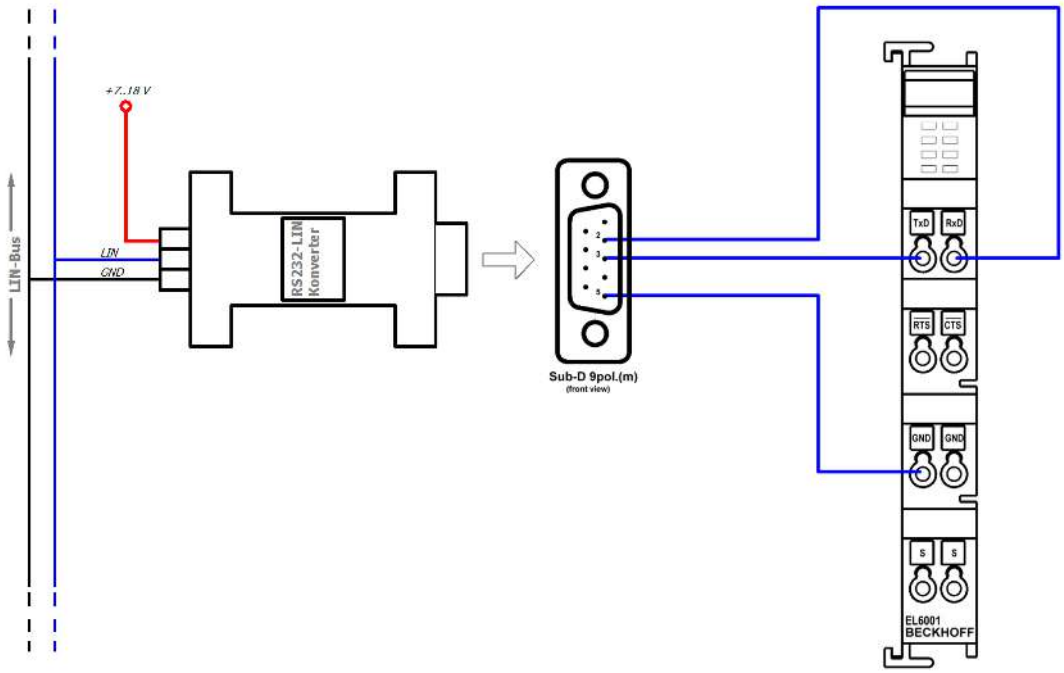


Fig. 156: RS232-LIN sub-D connector connected to the EL6001 terminal

Electrical connections between the EL6001 terminal and the sub-D 9-pin connector:

- RxD →pin 2
- TxD →pin 3
- GND → pin 5

Programming in ST

A LIN communication can be represented by a PLC “master task”, whereby a master sends an “unconditional frame”, as usual with a LIN bus, and only outputs the PID (protocol identifier) of a known slave node on the LIN bus, thereby generally requesting data from the respective slave. The chapter on commissioning contains a relevant programming example [▶ 140].

In this example the slave has the ID 0x07; the master therefore sends the ID 0x07 on the bus, including the calculated parity, resulting in the PID field 0x47. The message on the LIN bus then looks as follows:

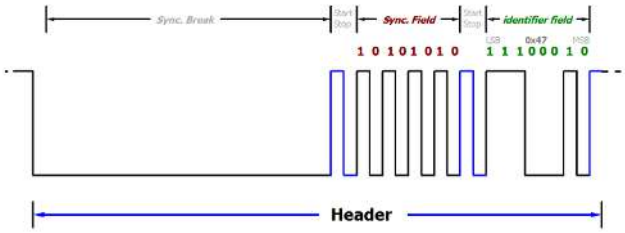


Fig. 157: LIN frame example: Query from master to node with ID 0x07

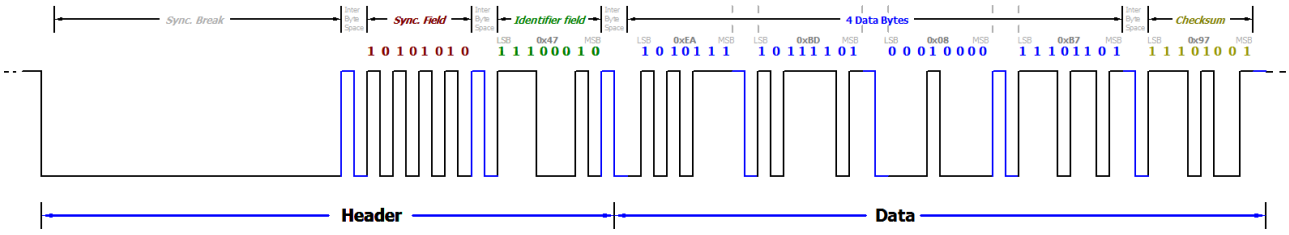


Fig. 158: LIN frame example: ID0x07 with data 0xEA,0xBD,0x08,0xB7 + checksum 0x97

Oscilloscope recordings LIN frame with ID 0x07:

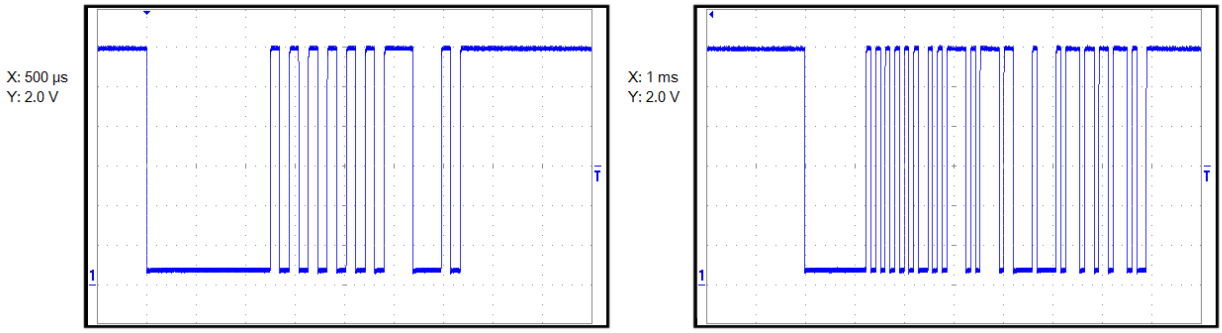


Fig. 159: Left: Query on the LIN bus with PID 0x47, right: LIN frame with the same PID and data, including checksum

5.8 Example programs

Using the sample programs

This document contains sample applications of our products for certain areas of application. The application notes provided here are based on typical features of our products and only serve as examples. The notes contained in this document explicitly do not refer to specific applications. The customer is therefore responsible for assessing and deciding whether the product is suitable for a particular application. We accept no responsibility for the completeness and correctness of the source code contained in this document. We reserve the right to modify the content of this document at any time and accept no responsibility for errors and missing information.

5.8.1 Sample program 1

https://infosys.beckhoff.com/content/1033/el600x_el602x/Resources/zip/1719190795.zip

Connection of a serial bar code scanner

In this example a bar code reader is connected to the EL6001. Characters will be read by the reader until the ASCII character 0x0D (13_{dec}, CR) is received.

Data:

- Quick task for executing the serial communication: 1 ms cycle time
- Standard PLC task: 10 ms cycle time
- Barcode scanner on channel 1
- TwinCAT 2.11 required
- “TwinCAT PLC Serial Communication” supplement is required

A detailed description for the use of the serial communication library is stored in the Beckhoff Information System.

Beckhoff Information System -> TwinCAT -> TwinCAT PLC -> TwinCAT libraries for PC-based systems -> TwinCAT PLC Library : Serial communication

Starting the example program

The application examples have been tested with a test configuration and are described accordingly.

Certain deviations when setting up actual applications are possible.

The following hardware and software were used for the test configuration:

- TwinCAT master PC with Windows XP Professional SP 3, TwinCAT version 2.11 (Build 1528) and INTEL PRO/100 VE Ethernet adapter
- Beckhoff EK1100 EtherCAT coupler, EL6001 terminals
- Serial barcode scanner with 9-pole sub-D connector

Procedure for starting the program

- After clicking the Download button, save the zip file locally on your hard disk, and unzip the *.TSM (configuration) and the *.PRO (PLC program) files into a temporary working folder
- Run the *.TSM file and the *.PRO file; the TwinCAT System Manager and TwinCAT PLC will open
- Connect the hardware in accordance with fig. 1 and connect the Ethernet adapter of your PC to the EtherCAT coupler (further information on this can be found in the corresponding coupler manuals)
- Select the local Ethernet adapter (with real-time driver, if one) under System configuration, I/O configuration, I/O devices, Device (EtherCAT); on the “Adapter” tab choose “Search...”, select the appropriate adapter and confirm (see Fig. 2a + 2b)

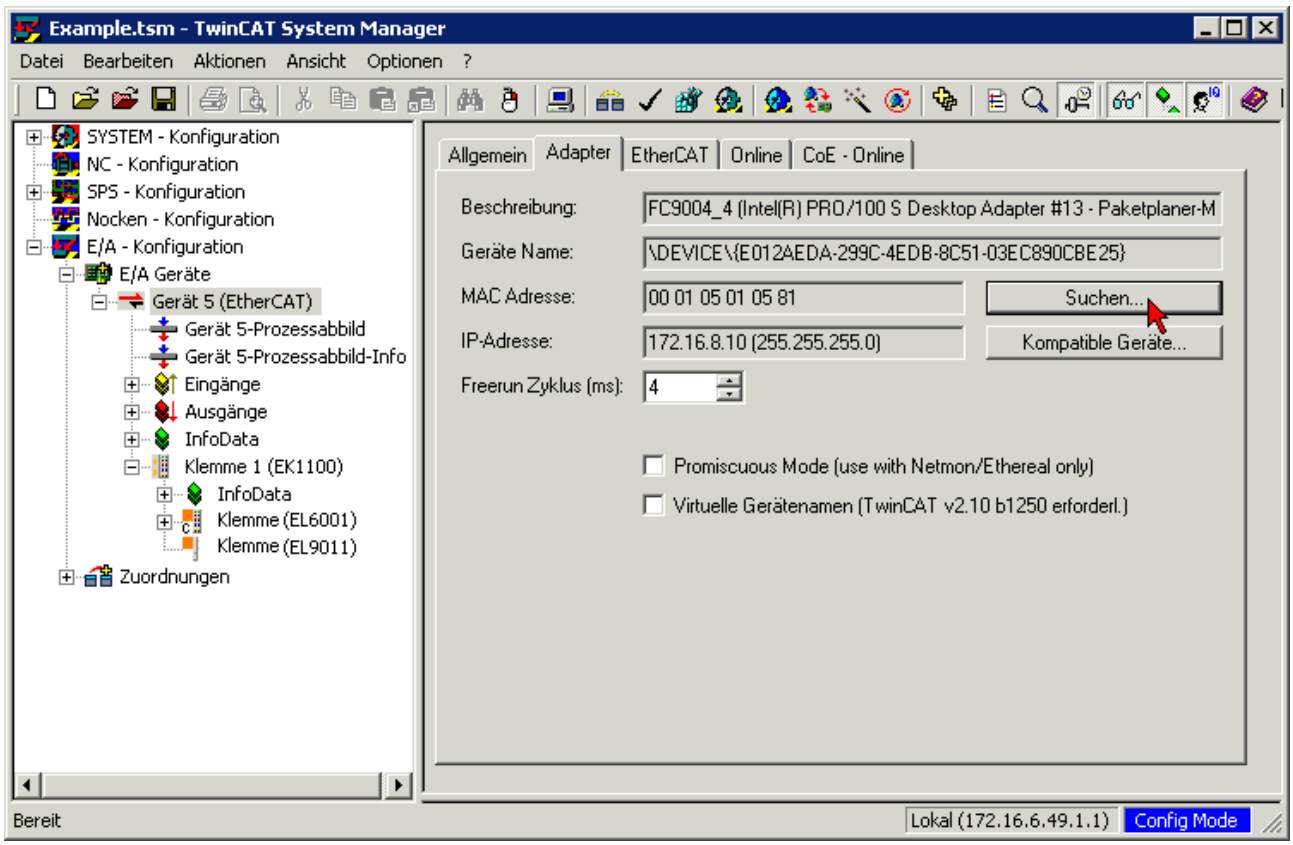


Fig. 160: *Searching the Ethernet adapter*

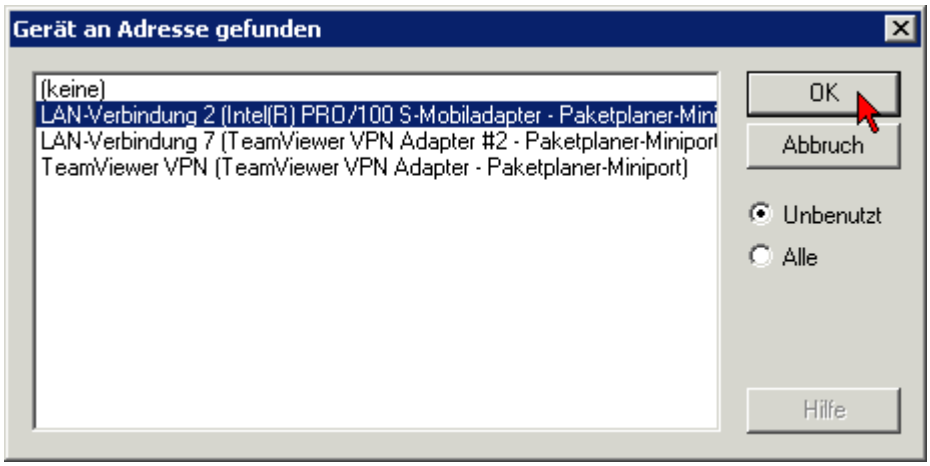


Fig. 161: *Selection and confirmation of the Ethernet adapter*

Activation of the configuration and confirmation (Fig. 3a +3b)

Fig. 162: *Activation of the configuration*

Fig. 163: *Confirming the activation of the configuration*

- Confirming new variable mapping, restart in RUN mode (Fig. 4a + 4b)

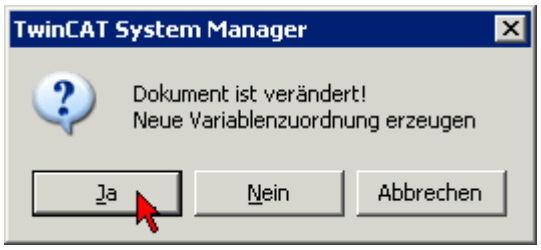


Fig. 164: *Generating variable mapping*

Fig. 165: *Restarting TwinCAT in RUN mode*

- In TwinCAT PLC, under the “Project” menu, select “Rebuild all” to compile the project (Fig. 5)

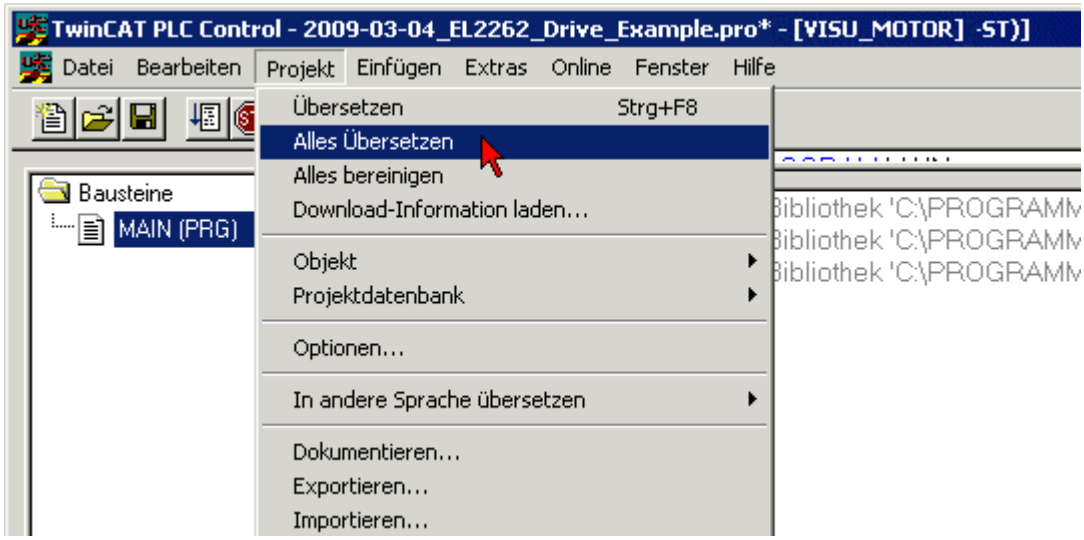


Fig. 166: *Compile project*

- In TwinCAT PLC: log in with the “F11” button, confirm loading the program (Fig. 6), run the program with the “F5” button

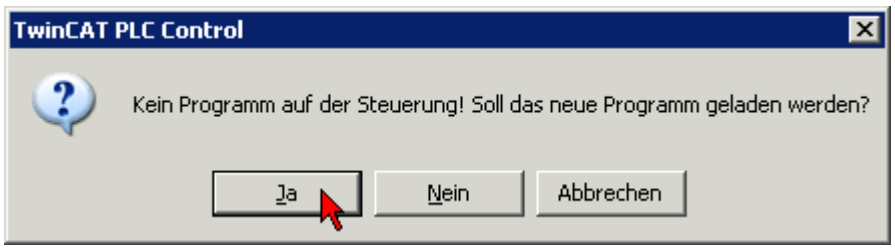


Fig. 167: *Confirming program start*

- After the character 0x13 has been received, the barcode is stored in “LastBarcode”

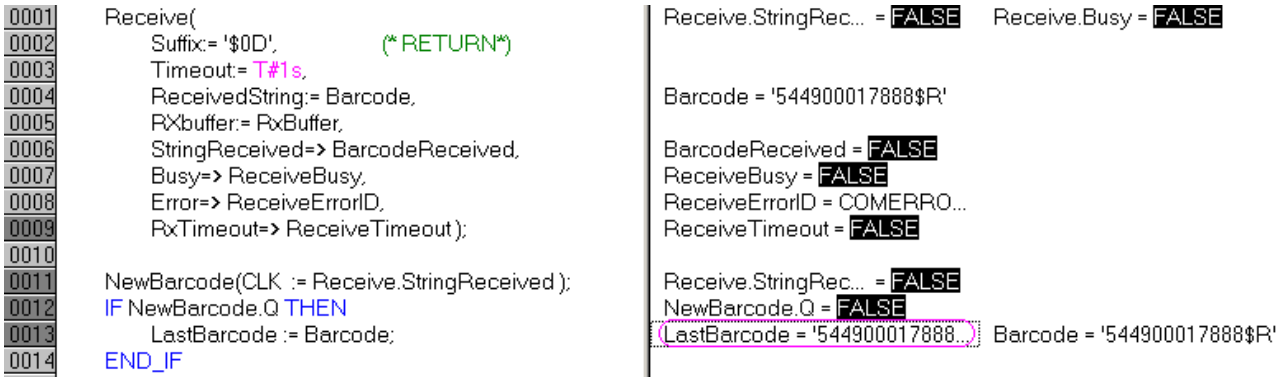


Fig. 168: *Received barcode*

5.8.2 Sample program 2

https://infosys.beckhoff.com/content/1033/el600x_el602x/Resources/zip/1719192971.zip

Reading and interpreting time telegrams

This example shows a way to process and interpret the most diverse serial time protocols in the PLC. To this end, IEC61131-PLC blocks will be presented that interpret the bit stream arriving at the PLC and, if necessary, extract the time/place information. This information can be used, for example, to synchronize controllers or record movements.

In this example, it is assumed that the data is delivered via a 22-byte interface by a Beckhoff EL/KL60xx serial data exchange terminal.

Background information

Not only in maritime applications is time and place information transported via serial buses: In the majority of cases an existing source distributes its information to lower level listeners/listeners cyclically or after a trigger via RS232 / RS485, USB or Ethernet.
A very large number of telegram formats exists worldwide for the distribution of time and place information; these are also known as ‘sentences’. Such a telegram consists of n bytes of data and is characterized by:

- Start and end characters STX, ETX for telegram recognition, possibly more than one end character
- a defined and fixed length
- a defined structure
- checksum (not necessary)
- type designations in the sentence if necessary

The most diverse organizations and companies have developed open or proprietary formats for specific purposes of use. Therefore, two sample implementations will be presented in this example that can easily be adapted to other specific protocols. The telegram formats in the example are the Meinberg Standard and NMEA0183 v2.3 type RMC.

Sources of information

GPS or radio controlled clock gateways are used as data transmitters in the serial sector. These devices receive the respective time signal (GPS via satellite or radio controlled clock via long wave) and convert it to the serial, wire-bound transmission e.g. RS232 with 8N1. The gateway often contains a local clock source in order to be able to continue distributing reliable time information for a certain time in the event of a short-term failure of the reference signal (GPS, radio transmitter). In Central Europe, the German DCF77 transmitter can be used.
If necessary, further information from the reference signal can be used:

- GPS: location information (W/N and height), upcoming clock change DCF77: weather information, major incidents
- DCF77: weather information, major incidents

Synchronization of lower level time slaves

In general, lower level slaves should be adjusted to the time gateway, i.e. they should synchronize their time. The following are necessary for this:

- Offset: the absolute deviation of the slave's own clock from the gateway reference time – this information can be transmitted, for example, acyclically and serially if it is known when the time information is to be considered valid. Gaining these offset-information is possible via the serial transport route from this example.
- the frequency ratio: a high-precision cyclic signal from the gateway to the slaves allows drift processes to be compensated and might state the time when the above absolute offset is valid. One example of such a signal is the widespread PPS (pulses per second).

If the serial telegram from the gateway is placed cyclically to the bus, the time of the first bit can often be interpreted as a PPS signal. In the block in this example, this information would be lost; only the absolute time information is evaluated.

Time formats

Time telegrams conforming to the NMEA0183 standard are widespread. Please note:

- there are currently (as of 2009) 8 versions of NMEA0183 1.5 to 4.00 within the NMEA0183 standard – the structure of telegrams may have changed between the versions.
- 70 different formats are defined in NMEA0183 v2.30 alone; device manufacturers can create their own formats in addition.
- The telegram is called a sentence
- A TalkerID (2 characters) and a TypeID (3 characters) at the start define the type of sentence used.
- A checksum is calculated for the telegram.
- Information can be found online at www.nmea.org or elsewhere.

Furthermore, many proprietary formats exist, such as Meinberg Standard, Siemens SINEC H1 and SAT 1703, or military formats, such as the IRIG codes (USA).

Using the sample program

The PLC project contains 2 function blocks (FBs), which must be linked exclusively with an EL/KL600x. Then collect the received bytes from the terminal and interpret the contents as far as possible. The FBs cover:

- Meinberg Standard
- NMEA0183 v2.3 type RMC

For other time formats, you can create your own interpretation FBs on the basis of a known telegram structure; contact your device manufacturer regarding this.

Hardware used in the example: EK1100 and EL6001 (also EL600x, KL600x)
The FB to be tested must be linked with its process data to the terminal in the System Manager (22-byte process image and control/status word).
The NMEA block is linked in the example. The baud rate of the terminal must be set to match your transmitter.

References

- www.beuth.de, IEC61162: based on NMEA2000
- http://gpsinformation.net private, via NMEA, many formats: http://www.gpsinformation.org/dale/nmea.htm
  http://www.nmea.de/nmea0183datensaetze.html
- http://www.meinberg.de/german/info/irig.htm, IRIG codes

5.8.3 Sample program 3 (LIN)

Download (TwinCAT 3 example program):

https://infosys.beckhoff.com/content/1033/el600x_el602x/Resources/zip/1805853195.zip

Global variables for LIN master example program

```
VAR_GLOBAL
nSetBaudrate : UINT := 10417; // Für Rx-Delay-Berechnung
(* I/O variables for EL6001 terminal acting as Master*)
COMin_EL6001_MASTER AT %I* : EL6inData22B;(* linked to the EL6001 in the TwinCAT System Manager *)
COMout_EL6001_MASTER AT %Q*: EL6outData22B;(* linked to the EL6001 in the TwinCAT System Manager *)
RxBuffer_MASTER : ComBuffer;(* Receive data buffer; used with all receive function blocks *)
TxBuffer_MASTER : ComBuffer;(* Transmit data buffer; used with all receive function blocks *)
END_VAR
```

Data types for LIN master-slave example program

```
TYPE tDataFrame : ARRAY[0..8] OF BYTE; END_TYPE // Datentyp für ein LIN-Frame
```

Function for adding data length information and parity bits:

Declaration part:

```
FUNCTION F_ADD_LIN_NODE_PARITY : BYTE
VAR_INPUT
nNodeID:BYTE; // Inputvariable: node-Id
nReqLen:BYTE; // Inputvariable: length-identification: 2,4,8 Byte-Frame
END_VAR
VAR
bParity0:BYTE; // Internal intermediate value parity 0
bParity1:BYTE; // Internal intermediate value parity 1
nPrepId:BYTE; // Intermediate value for PID
END_VAR
```

Execution part:

```
nPrepId := nNodeID OR SHL(nReqLen,4);

bParity0 :=
(nNodeID AND 2#0001)
XOR (SHR((nNodeID AND 2#0010), 1))
XOR (SHR((nNodeID AND 2#0100), 2))
XOR (SHR((nNodeID AND 2#0001_0000), 4));

bParity1 := 16#01 AND (NOT(
SHR((nNodeID AND 2#0010), 1)
XOR (SHR((nNodeID AND 2#1000),3))
XOR (SHR((nNodeID AND 2#0001_0000), 4))
XOR (SHR((nNodeID AND 2#0010_0000), 5))));

F_ADD_LIN_NODE_PARITY := nPrepId OR SHL(bParity0,6) OR SHL(bParity1,7);
```

Function for calculating the checksum (conventional method)

Declaration part:

```
FUNCTION F_CALC_LIN_CHKSUM : BYTE
VAR_INPUT
pData:POINTER TO ARRAY[0..10] OF BYTE; // Pointer to source datafield
nLen:BYTE; // Number of bytes used for calculating the checksum
END_VAR
VAR
```

```
i: BYTE; // Counter variable
wResult:WORD; // Resulting output value
END_VAR
```

Execution part:

```
wResult := BYTE_TO_WORD(pData^[0]);
FOR i := 1 TO (nLen-1) DO
wResult := wResult + BYTE_TO_WORD(pData^[i]);
IF wResult > 255 THEN
wResult := wResult - 255;
END_IF
END_FOR
F_CALC_LIN_CHKSUM := WORD_TO_BYTE(NOT(wResult));
```

This master program part should be called from a corresponding separate task. Here, a node in the LIN bus with the ID 0x07 is queried every 200 ms.

Declaration part:

```
PROGRAM EL6001_MASTER
VAR
Timer: TON; // Timer for periodical requests by the master
Send: SendData; // Functionblock of TC2_SerialCom
SendBusy: BOOL; // Flag copy of SendData.Busy
SendErrorID: ComError_t; // Flag-Copy of Error-ID
aDataTX:tDataFrame; // Data frame being send
Receive: ReceiveData; // Functionblock of TC2_SerialCom
LastReceivedDataBytes: tDataFrame; // Copy (Latch) of received data
DataReceived: BOOL; // Flag copy of receive confirmation
ReceiveBusy: BOOL; // Flag copy of reception not ready
ReceiveError: BOOL; // Flag copy of receive error
ReceiveErrorID: ComError_t;// Error-ID copy
ReceiveTimeout: BOOL; // Flag copy of receive-timeout
ReceiveCounter: UDINT := 0; // Number of received frames
aDataRX:tDataFrame; // Receiving data frame buffer
nDataLen:BYTE := 4; // Fixed data length
nState:BYTE := 0; // Initial state (start)
bNodeId_SL1: BYTE := 16#07; // ID of slave node
bReqLen_SL1: BYTE := 0; // Optional entry for 4 Data bytes Value 2
nRxChecksum: BYTE; // Storage of received checksum
nCalcChecksum:BYTE; // Storage of calculated checksum
T_ReceiveDelay:TIME; // Storage of calculated delay time
END_VAR
```

Execution part:

```
(*==================================================================
Receive data
*)
CASE nState OF
0:
Timer(IN:=TRUE, PT:=T#0.5S); // Call timer for periodical master requests
IF Timer.Q THEN
// Put ID into Tranceive data:
aDataTX[0] := F_ADD_LIN_NODE_PARITY(bNodeId_SL1, bReqLen_SL1);
LastReceivedDataBytes[0] := aDataTX[0];
// Send request to Slave 1 (get Data)
Send(pSendData:= ADR(aDataTX), Length:= 1,
TXbuffer:= TxBuffer_MASTER,
Busy => SendBusy, Error => SendErrorID);
Timer(IN:=FALSE); (* reset timer *)
IF NOT SendBusy THEN // Wait until sending ends
nState := nState + 1;
END_IF
END_IF
1:
// Delaytime by 1/Tbaud * Number of Bytes * (8 Databits + 2 Bit:start-Stop) * 1000ms
T_ReceiveDelay := REAL_TO_TIME((1/DINT_TO_REAL(nSetBaudrate)) * 33 * 1000); // .. for 1 Byte
Timer(IN:=TRUE, PT:=T_ReceiveDelay);
IF Timer.Q THEN
// Wait until ID is send
nState := nState + 1;
Timer(IN:=FALSE); (* reset timer *)
END_IF
2:
Receive(
pReceiveData:= ADR(aDataRX),
SizeReceiveData:= (nDataLen + 1),
RXbuffer:= RxBuffer_MASTER,
```

```
Timeout:= T#1S,
DataReceived=> DataReceived,
busy=> ReceiveBusy,
Error=> ReceiveErrorID,
RxTimeout=> ReceiveTimeout );
IF DataReceived THEN
// DataReceived := FALSE;
ReceiveCounter := ReceiveCounter + 1;
IF NOT ReceiveBusy THEN
// Compare checksum
nRxChecksum := aDataRX[nDataLen];
nCalcChecksum := F_CALC_LIN_CHKSUM(pData := ADR(aDataRX), nLen := nDataLen);
IF(nRxChecksum = nCalcChecksum) THEN
// Response received - clear databuffer:
memset(ADR(LastReceivedDataBytes[1]), 0, (SIZEOF(aDataRX)-1));
// Take-over data when checksum OK:
memcpy(ADR(LastReceivedDataBytes[1]), ADR(aDataRX), (nDataLen +1));
END_IF
nState := 0;
END_IF
ELSE
Timer(IN:=TRUE, PT:=T#0.1S); // Receive-Timeout 100 ms: no data
IF Timer.Q THEN
nState := 0;
END_IF
END_IF
END_CASE
(*=================================================================
*)
```

Program fast task / RS232 background communication

In this example, third task should be created with as few “cycle ticks” as possible, which deals with the background communication with the EL6001 terminal (as master).

Declaration part:

```
PROGRAM FAST
VAR
(* background communication with the EL6001 as Master device *)
COMportControl_MASTER: SerialLineControl;
COMportControlError_MASTER: BOOL;
COMportControlErrorID_MASTER: ComError_t;
END_VAR
```

Execution part:

```
COMportControl_MASTER(
Mode:= SERIALLINEMODE_EL6_22B,
pComIn:= ADR(COMin_EL6001_MASTER),(* I/O data; see global variables *)
pComOut:= ADR(COMout_EL6001_MASTER),(* I/O data; see global variables *)
SizeComIn:= SIZEOF(COMin_EL6001_MASTER),(* I/O data; see global variables *)
TxBuffer:= TxBuffer_MASTER,(* transmit buffer; see global variables *)
RxBuffer:= RxBuffer_MASTER,(* receive buffer; see global variables *)
Error=> COMportControlError_MASTER,(* indicating an error *)
ErrorID=> COMportControlErrorID_MASTER );(* contains the error-ID *)
```

6 Overview of CoE objects EL6001, EL6021

6.1 Object description and parameterization

EtherCAT XML Device Description

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

Parameterization

The terminal is parameterized via the CoE - Online tab [▶ 113] (with a double-click on the respective object, see below).

Introduction

The CoE overview contains objects for different intended applications:

- Objects required for parameterization [▶ 148] during commissioning
- Objects for indicating internal settings [▶ 150] (may be fixed)
- Profile specific objects [▶ 166], which represent the status displays of the inputs and outputs (from hardware version 03 [▶ 190])

The following section first describes the objects required for normal operation, followed by a complete overview of missing objects.

6.1.1 Objects for commissioning

Index 0x1011 Restore default parameters

Index (hex)	Name	Meaning	Data type	Flags	Default
1011:0	Restore default parameters [▶ 203]	Restore default parameters	UINT8	RO	0x01 (1_{dec})
1011:01	SubIndex 0x001	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_{dec})

Index 0x4073 Baud rate

Index (hex)	Name	Meaning	Data type	Flags	Default
4073:0	Baud rate	Detailed information can be found in chapter Communication features [▶ 135]	UINT16	RW	0x0006 (6_{dec})

Index 0x4074 Data frame

Index (hex)	Name	Meaning	Data type	Flags	Default
4074:0	Data frame	Detailed information can be found in chapter Communication features [▶ 135]	UINT16	RW	0x0003 (3_{dec})

Index 0x4075 Feature bits

Index (hex)	Name	Meaning	Data type	Flags	Default
4075:0	Feature bits	Length of this object	UINT8	RO	0x06 (6_{dec})
4075:01	**EL6001:** Enable RTS/CTS	0: RTS/CTS not enabled 1: RTS/CTS enabled	BOOLEAN	RW	0x00 (0_{dec})
	EL6021: Enable half duplex	0: Full-duplex mode 1: Half-duplex mode			
4075:02	Enable XON/ XOFF supported tx data	0: XON/XOFF is not supported for send data 1: XON/XOFF is supported for send data	BOOLEAN	RW	0x00 (0_{dec})
4075:03	Enable XON/ XOFF supported rx data	0: XON/XOFF is not supported for receive data 1: XON/XOFF is supported for receive data	BOOLEAN	RW	0x00 (0_{dec})
4075:04	**EL6001:** Enable send FIFO data continous	0: No continuous sending of data from the FIFO 1: The send buffer is filled (up to 128 bytes) by the controller. The buffer content is sent with rising edge of bit CW.3 [▶ 169] in the control word. The terminal acknowledges the data transfer to the controller through setting of bit SW.2 [▶ 170] in the status word. SW.2 is reset with CW.3.	BOOLEAN	RO	0x00 (0_{dec})
	EL6021: Enable point to point connection (RS422)	0: Point-to-point connection disabled 1: Point-to-point connection enabled	BOOLEAN	RO	0x00 (0_{dec})
4075:05	**EL6001:** Enable transfer rate optimization	0: Transfer rate optimization switched off 1: Transfer rate optimization switched on: The content of the input buffer is automatically transferred into the process image if • no further byte was received for approx. 16 bit times (i.e. the time it would have taken to receive 2 bytes) after data were received; • the input buffer is full	BOOLEAN	RO	0x01 (1_{dec})
	EL6021: Enable send FIFO data continuous	0: No continuous sending of data from the FIFO 1: The send buffer is filled (up to 128 bytes) by the controller. The buffer content is sent with rising edge of bit CW.3 [▶ 169] in the control word. The terminal acknowledges the data transfer to the controller through setting of bit SW.2 [▶ 170] in the status word. SW.2 is reset with CW.3.	BOOLEAN	RO	0x00 (0_{dec})
4075:06	**EL6021 only:** Enable transfer rate optimization	0: Transfer rate optimization switched off 1: Transfer rate optimization switched on: The content of the input buffer is automatically transferred into the process image if • no further byte was received for approx. 16 bit times (i.e. the time it would have taken to receive 2 bytes) after data were received; • the process image is filled	BOOLEAN	RW	0x01 (1_{dec})

Index 0x4076 Rx buffer full notification

Index (hex)	Name	Meaning	Data type	Flags	Default
4076:0	Rx buffer full notification	The value determines the number of data in the receive FIFO from which bit SW.3 [▶ 170] (BUF_F) is set in the status byte.	UINT16	RW	0x0000 (0_{dec})

Index 0x8000 COM settings [from hardware version 03]

Index (hex)	Name	Meaning		Data type	Flags	Default
8000:0	COM Settings	Max. SubIndex (hex)		UINT8	RO	0x26 (38_{dec})
8000:01**	Enable RTS/CTS	FALSE	RTS/CTS not enabled	BOOLEAN	RW	0x01 (1_{dec})
		TRUE	RTS/CTS enabled			
8000:02	Enable XON/ XOFF supported tx data	FALSE	XON/XOFF is not supported for send data	BOOLEAN	RW	0x00 (0_{dec})
		TRUE	XON/XOFF is supported for send data			
8000:03	Enable XON/ XOFF supported rx data	FALSE	XON/XOFF is not supported for receive data	BOOLEAN	RW	0x00 (0_{dec})
		TRUE	XON/XOFF is supported for receive data			
8000:04	Enable send FIFO data continuous	FALSE	No continuous sending of data from the FIFO	BOOLEAN	RW	0x00 (0_{dec})
		TRUE	Continuous sending of data from the FIFO enabled: The send buffer is filled (up to 128 bytes) by the controller. The filled buffer content is sent on rising edge of bit “SendContinious [▶ 167]”. The terminal acknowledges the data transfer to the controller through setting of bit "InitAccepted". “InitAccepted [▶ 166]” is reset with “SendContinuous”.			
8000:05	Enable transfer rate optimization	FALSE	Transfer rate optimization switched off	BOOLEAN	RW	0x01 (1_{dec})
		TRUE	Transfer rate optimization switched on: The content of the input buffer is automatically transferred into the process image if • no further byte was received for approx. 16 bit times (i.e. the time it would have taken to receive 2 bytes) after data were received; • the process image is filled			
8000:06***	Enable half duplex	FALSE	Full-duplex mode	BOOLEAN	RW	0x00 (0_{dec})
		TRUE	Half-duplex mode			
8000:07***	Enable point to point connection (RS422)	FALSE	Point-to-point connection disabled	BOOLEAN	RW	0x00 (0_{dec})
		TRUE	Point-to-point connection enabled			
8000:11	Baud rate	Detailed information can be found in chapter Communication features [▶ 135]		BIT4	RW	0x06 (6_{dec})
8000:15	Data frame	Detailed information can be found in chapter Communication features [▶ 135]		BIT4	RW	0x03 (3_{dec})
8000:1A	Rx buffer full notification	The value specifies the number of data in the receive FIFO, from which the bit “buffer full [▶ 168]” is set.		UINT16	RW	0x0360 (864_{dec})
8000:1B**	Explicit baudrate	In this object the desired baud rate can be entered directly as a number. Only the baud rates specified in Communication features [▶ 135] are supported. Changes to this object are also adopted into the objects 0x8000:11 and 4073		UINT32	RW	0x00000384 (9600_{dec})
8000:1C**	Extended data frame	In this object special formats can also be selected in addition to the usual data frames (e.g. 9N1). Changes to this object are also adopted in the objects 0x8000:15 and 0x4074.		ENUM16	RW	0x0003 (3_{dec})

**) only EL6001
***) only EL6021

6.1.2 Standard objects (0x1000-0x1FFF)

The standard objects have the same meaning for all EtherCAT slaves.

Index 0x1000 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 CoE profile used (5001). The Hi-Word contains the module profile according to the modular device profile.	UINT32	RO	0x02581389 (39326601_{dec})

Index 0x1008 Device name

Index (hex)	Name	Meaning	Data type	Flags	Default
1008:0	Device name	Device name of the EtherCAT slave	STRING	RO	EL6001 EL6021

Index 0x1009 Hardware version

Index (hex)	Name	Meaning	Data type	Flags	Default
1009:0	Hardware version	Hardware version of the EtherCAT slave	STRING	RO	04

Index 0x100A Software version

Index (hex)	Name	Meaning	Data type	Flags	Default
100A:0	Software version	Firmware version of the EtherCAT slave	STRING	RO	05

Index 0x1018 Identity

Index (hex)	Name	Meaning	Data type	Flags	Default
1018:0	Identity	Information for identifying the slave	UINT8	RO	0x04 (4_{dec})
1018:01	Vendor ID	Vendor ID of the EtherCAT slave	UINT32	RO	0x00000002 (2_{dec})
1018:02	Product code	Product code of the EtherCAT slave	UINT32	RO	0x17853052 (394604626_{dec})
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	0x00100000 (1048576_{dec})
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_{dec})

Index 0x10F0 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_{dec})
10F0:01	Checksum	Checksum across all backup entries of the EtherCAT slave	UINT32	RO	0x00000000 (0_{dec})

Index 0x1400 RxPDO-Par Outputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1400:0	RxPDO-Par Outputs	PDO Parameter RxPDO 1	UINT8	RO	0x06 (6_{dec})
1400:06	Exclude RxPDOs	Specifies the RxPDOs (index 0x of RxPDO mapping objects) that must not be transferred together with RxPDO 1	OCTET-STRING[6]	RO	01 16 02 16 04 16

Index 0x1401 RxPDO-Par Outputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1401:0	RxPDO-Par Outputs	PDO Parameter RxPDO 2	UINT8	RO	0x06 (6_{dec})
1401:06	Exclude RxPDOs	Specifies the RxPDOs (index 0x of RxPDO mapping objects) that must not be transferred together with Rx-PDO 2	OCTET-STRING[6]	RO	00 16 02 16 04 16

Index 0x1402 RxPDO-Par Outputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1402:0	RxPDO-Par Outputs	PDO Parameter RxPDO 3	UINT8	RO	0x06 (6_{dec})
1402:06	Exclude RxPDOs	Specifies the RxPDOs (index 0x of RxPDO mapping objects) that must not be transferred together with Rx-PDO 3	OCTET-STRING[6]	RO	00 16 01 16 04 16

Index 0x1600 RxPDO-Map Outputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1600:0	RxPDO-Map Outputs	PDO Mapping RxPDO 1	UINT8	RO	0x04 (4_{dec})
1600:01	SubIndex 0x001	1. PDO Mapping entry (object 0x3001 (Outputs), entry 0x01 (Ctrl))	UINT32	RO	0x3001:01, 8
1600:02	SubIndex 0x002	2. PDO Mapping entry (object 0x3001 (Outputs), entry 0x02 (Data Out 0))	UINT32	RO	0x3001:02, 8
1600:03	SubIndex 0x003	3. PDO Mapping entry (object 0x3001 (Outputs), entry 0x03 (Data Out 1))	UINT32	RO	0x3001:03, 8
1600:04	SubIndex 0x004	4. PDO Mapping entry (object 0x3001 (Outputs), entry 0x04 (Data Out 2))	UINT32	RO	0x3001:04, 8

Index 0x1601 RxPDO-Map Outputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1601:0	RxPDO-Map Outputs	PDO Mapping RxPDO 2	UINT8	RO	0x06 (6_{dec})
1601:01	SubIndex 0x001	1. PDO Mapping entry (object 0x3002 (Outputs), entry 0x01 (Ctrl))	UINT32	RO	0x3002:01, 8
1601:02	SubIndex 0x002	2. PDO Mapping entry (object 0x3002 (Outputs), entry 0x02 (Data Out 0))	UINT32	RO	0x3002:02, 8
1601:03	SubIndex 0x003	3. PDO Mapping entry (object 0x3002 (Outputs), entry 0x03 (Data Out 1))	UINT32	RO	0x3002:03, 8
1601:04	SubIndex 0x004	4. PDO Mapping entry (object 0x3002 (Outputs), entry 0x04 (Data Out 2))	UINT32	RO	0x3002:04, 8
1601:05	SubIndex 0x005	5. PDO Mapping entry (object 0x3002 (Outputs), entry 0x05 (Data Out 3))	UINT32	RO	0x3002:05, 8
1601:06	SubIndex 0x006	6. PDO Mapping entry (object 0x3002 (Outputs), entry 0x06 (Data Out 4))	UINT32	RO	0x3002:06, 8

Index 0x1602 RxPDO-Map Outputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1602:0	RxPDO-Map Outputs	PDO Mapping RxPDO 3	UINT8	RO	0x17 (23_{dec})
1602:01	SubIndex 0x001	1. PDO Mapping entry (object 0x3003 (Outputs), entry 0x01 (Ctrl))	UINT32	RO	0x3003:01, 16
1602:02	SubIndex 0x002	2. PDO Mapping entry (object 0x3003 (Outputs), entry 0x02 (Data Out 0))	UINT32	RO	0x3003:02, 8
1602:03	SubIndex 0x003	3. PDO Mapping entry (object 0x3003 (Outputs), entry 0x03 (Data Out 1))	UINT32	RO	0x3003:03, 8
1602:04	SubIndex 0x004	4. PDO Mapping entry (object 0x3003 (Outputs), entry 0x04 (Data Out 2))	UINT32	RO	0x3003:04, 8
1602:05	SubIndex 0x005	5. PDO Mapping entry (object 0x3003 (Outputs), entry 0x05 (Data Out 3))	UINT32	RO	0x3003:05, 8
1602:06	SubIndex 0x006	6. PDO Mapping entry (object 0x3003 (Outputs), entry 0x06 (Data Out 4))	UINT32	RO	0x3003:06, 8
1602:07	SubIndex 0x007	7. PDO Mapping entry (object 0x3003 (Outputs), entry 0x07 (Data Out 5))	UINT32	RO	0x3003:07, 8
1602:08	SubIndex 0x008	8. PDO Mapping entry (object 0x3003 (Outputs), entry 0x08 (Data Out 6))	UINT32	RO	0x3003:08, 8
1602:09	SubIndex 0x009	9. PDO Mapping entry (object 0x3003 (Outputs), entry 0x09 (Data Out 7))	UINT32	RO	0x3003:09, 8
1602:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x3003 (Outputs), entry 0x0A (Data Out 8))	UINT32	RO	0x3003:0A, 8
1602:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x3003 (Outputs), entry 0x0B (Data Out 9))	UINT32	RO	0x3003:0B, 8
1602:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x3003 (Outputs), entry 0x0C (Data Out 10))	UINT32	RO	0x3003:0C, 8
1602:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x3003 (Outputs), entry 0x0D (Data Out 11))	UINT32	RO	0x3003:0D, 8
1602:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x3003 (Outputs), entry 0x0E (Data Out 12))	UINT32	RO	0x3003:0E, 8
1602:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x3003 (Outputs), entry 0x0F (Data Out 13))	UINT32	RO	0x3003:0F, 8
1602:10	SubIndex 0x016	16. PDO Mapping entry (object 0x3003 (Outputs), entry 0x10 (Data Out 14))	UINT32	RO	0x3003:10, 8
1602:11	SubIndex 0x017	17. PDO Mapping entry (object 0x3003 (Outputs), entry 0x11 (Data Out 15))	UINT32	RO	0x3003:11, 8
1602:12	SubIndex 0x018	18. PDO Mapping entry (object 0x3003 (Outputs), entry 0x12 (Data Out 16))	UINT32	RO	0x3003:12, 8
1602:13	SubIndex 0x019	19. PDO Mapping entry (object 0x3003 (Outputs), entry 0x13 (Data Out 17))	UINT32	RO	0x3003:13, 8
1602:14	SubIndex 0x020	20. PDO Mapping entry (object 0x3003 (Outputs), entry 0x14 (Data Out 18))	UINT32	RO	0x3003:14, 8
1602:15	SubIndex 0x021	21. PDO Mapping entry (object 0x3003 (Outputs), entry 0x15 (Data Out 19))	UINT32	RO	0x3003:15, 8
1602:16	SubIndex 0x022	22. PDO Mapping entry (object 0x3003 (Outputs), entry 0x16 (Data Out 20))	UINT32	RO	0x3003:16, 8
1602:17	SubIndex 0x023	23. PDO Mapping entry (object 0x3003 (Outputs), entry 0x17 (Data Out 21))	UINT32	RO	0x3003:17, 8

Index 0x1604 COM RxPDO-Map Outputs [ab Hardwarestand 03]

Index (hex)	Name	Meaning	Data type	Flags	Default
1604:0	COM RxPDO-Map Outputs	PDO Mapping RxPDO 5	UINT8	RO	0x1C (28_{dec})
1604:01	SubIndex 0x001	1. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x01 (Transmit request))	UINT32	RO	0x7000:01, 1
1604:02	SubIndex 0x002	2. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x02 (Receive accepted))	UINT32	RO	0x7000:02, 1
1604:03	SubIndex 0x003	3. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x03 (Init request))	UINT32	RO	0x7000:03, 1
1604:04	SubIndex 0x004	4. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x04 (Send continues))	UINT32	RO	0x7000:04, 1
1604:05	SubIndex 0x005	5. PDO Mapping entry (4 bits align)	UINT32	RO	0x0000:00, 4
1604:06	SubIndex 0x006	6. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x09 (Output length))	UINT32	RO	0x7000:09, 8
1604:07	SubIndex 0x007	7. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x11 (Data Out 0))	UINT32	RO	0x7000:11, 8
1604:08	SubIndex 0x008	8. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x12 (Data Out 1))	UINT32	RO	0x7000:12, 8
1604:09	SubIndex 0x009	9. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x13 (Data Out 2))	UINT32	RO	0x7000:13, 8
1604:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x14 (Data Out 3))	UINT32	RO	0x7000:14, 8
1604:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x15 (Data Out 4))	UINT32	RO	0x7000:15, 8
1604:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x16 (Data Out 5))	UINT32	RO	0x7000:16, 8
1604:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x17 (Data Out 6))	UINT32	RO	0x7000:17, 8
1604:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x18 (Data Out 7))	UINT32	RO	0x7000:18, 8
1604:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x19 (Data Out 8))	UINT32	RO	0x7000:19, 8
1604:10	SubIndex 0x016	16. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x1A (Data Out 9))	UINT32	RO	0x7000:1A, 8
1604:11	SubIndex 0x017	17. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x1B (Data Out 10))	UINT32	RO	0x7000:1B, 8
1604:12	SubIndex 0x018	18. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x1C (Data Out 11))	UINT32	RO	0x7000:1C, 8
1604:13	SubIndex 0x019	19. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x1D (Data Out 12))	UINT32	RO	0x7000:1D, 8
1604:14	SubIndex 0x020	20. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x1E (Data Out 13))	UINT32	RO	0x7000:1E, 8
1604:15	SubIndex 0x021	21. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x1F (Data Out 14))	UINT32	RO	0x7000:1F, 8
1604:16	SubIndex 0x022	22. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x20 (Data Out 15))	UINT32	RO	0x7000:20, 8
1604:17	SubIndex 0x023	23. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x21 (Data Out 16))	UINT32	RO	0x7000:21, 8
1604:18	SubIndex 0x024	24. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x22 (Data Out 17))	UINT32	RO	0x7000:22, 8
1604:19	SubIndex 0x025	25. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x23 (Data Out 18))	UINT32	RO	0x7000:23, 8
1604:1A	SubIndex 0x026	26. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x24 (Data Out 19))	UINT32	RO	0x7000:24, 8
1604:1B	SubIndex 0x027	27. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x25 (Data Out 20))	UINT32	RO	0x7000:25, 8
1604:1C	SubIndex 0x028	28. PDO Mapping entry (object 0x7000 (COM Outputs), entry 0x26 (Data Out 21))	UINT32	RO	0x7000:26, 8

Index 0x1605 COM ext. outputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1605:0	COM ext. outputs	PDO Mapping RxPDO 6	UINT8	RO	0x38 (56_{dec})
1605:01	SubIndex 0x001	1. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x01 (Transmit request))	UINT32	RO	0x7001:01, 1
1605:02	SubIndex 0x002	2. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x02 (Receive accepted))	UINT32	RO	0x7001:02, 1
1605:03	SubIndex 0x003	3. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x03 (Init request))	UINT32	RO	0x7001:03, 1
1605:04	SubIndex 0x004	4. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x04 (Send continues))	UINT32	RO	0x7001:04, 1
1605:05	SubIndex 0x005	5. PDO Mapping entry (4 bits align)	UINT32	RO	0x0000:00, 4
1605:06	SubIndex 0x006	6. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x09 (Output length))	UINT32	RO	0x7001:09, 8
1605:07	SubIndex 0x007	7. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x11 (Data Out 0))	UINT32	RO	0x7001:11, 16
1605:08	SubIndex 0x008	8. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x12 (Data Out 1))	UINT32	RO	0x7001:12, 16
1605:09	SubIndex 0x009	9. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x13 (Data Out 2))	UINT32	RO	0x7001:13, 16
1605:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x14 (Data Out 3))	UINT32	RO	0x7001:14, 16
1605:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x15 (Data Out 4))	UINT32	RO	0x7001:15, 16
1605:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x16 (Data Out 5))	UINT32	RO	0x7001:16, 16
1605:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x17 (Data Out 6))	UINT32	RO	0x7001:17, 16
1605:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x18 (Data Out 7))	UINT32	RO	0x7001:18, 16
1605:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x19 (Data Out 8))	UINT32	RO	0x7001:19, 16
1605:10	SubIndex 0x016	16. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x1A (Data Out 9))	UINT32	RO	0x7001:1A, 16
1605:11	SubIndex 0x017	17. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x1B (Data Out 10))	UINT32	RO	0x7001:1B, 16
1605:12	SubIndex 0x018	18. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x1C (Data Out 11))	UINT32	RO	0x7001:1C, 16
1605:13	SubIndex 0x019	19. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x1D (Data Out 12))	UINT32	RO	0x7001:1D, 16
1605:14	SubIndex 0x020	20. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x1E (Data Out 13))	UINT32	RO	0x7001:1E, 16
1605:15	SubIndex 0x021	21. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x1F (Data Out 14))	UINT32	RO	0x7001:1F, 16
1605:16	SubIndex 0x022	22. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x20 (Data Out 15))	UINT32	RO	0x7001:20, 16
1605:17	SubIndex 0x023	23. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x21 (Data Out 16))	UINT32	RO	0x7001:21, 16
1605:18	SubIndex 0x024	24. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x22 (Data Out 17))	UINT32	RO	0x7001:22, 16
1605:19	SubIndex 0x025	25. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x23 (Data Out 18))	UINT32	RO	0x7001:23, 16
1605:1A	SubIndex 0x026	26. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x24 (Data Out 19))	UINT32	RO	0x7001:24, 16
1605:1B	SubIndex 0x027	27. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x25 (Data Out 20))	UINT32	RO	0x7001:25, 16
1605:1C	SubIndex 0x028	28. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x26 (Data Out 21))	UINT32	RO	0x7001:26, 16
1605:1D	SubIndex 0x029	29. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x27 (Data Out 22))	UINT32	RO	0x7001:27, 16
1605:1E	SubIndex 0x030	30. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x28 (Data Out 23))	UINT32	RO	0x7001:28, 16
1605:1F	SubIndex 0x031	31. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x29 (Data Out 24))	UINT32	RO	0x7001:29, 16

Index (hex)	Name	Meaning	Data type	Flags	Default
1605:20	SubIndex 0x032	32. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x2A (Data Out 25))	UINT32	RO	0x7001:2A, 16
1605:21	SubIndex 0x033	33. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x2B (Data Out 26))	UINT32	RO	0x7001:2B, 16
1605:22	SubIndex 0x034	34. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x2C (Data Out 27))	UINT32	RO	0x7001:2C, 16
1605:23	SubIndex 0x035	35. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x2D (Data Out 28))	UINT32	RO	0x7001:2D, 16
1605:24	SubIndex 0x036	36. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x2E (Data Out 29))	UINT32	RO	0x7001:2E, 16
1605:25	SubIndex 0x037	37. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x2F (Data Out 30))	UINT32	RO	0x7001:2F, 16
1605:26	SubIndex 0x038	38. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x30 (Data Out 31))	UINT32	RO	0x7001:30, 16
1605:27	SubIndex 0x039	39. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x31 (Data Out 32))	UINT32	RO	0x7001:31, 16
1605:28	SubIndex 0x040	40. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x32 (Data Out 33))	UINT32	RO	0x7001:32, 16
1605:29	SubIndex 0x041	41. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x33 (Data Out 34))	UINT32	RO	0x7001:33, 16
1605:2A	SubIndex 0x042	42. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x34 (Data Out 35))	UINT32	RO	0x7001:34, 16
1605:2B	SubIndex 0x043	43. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x35 (Data Out 36))	UINT32	RO	0x7001:35, 16
1605:2C	SubIndex 0x044	44. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x36 (Data Out 37))	UINT32	RO	0x7001:36, 16
1605:2D	SubIndex 0x045	45. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x37 (Data Out 38))	UINT32	RO	0x7001:37, 16
1605:2E	SubIndex 0x046	46. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x38 (Data Out 39))	UINT32	RO	0x7001:38, 16
1605:2F	SubIndex 0x047	47. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x39 (Data Out 40))	UINT32	RO	0x7001:39, 16
1605:30	SubIndex 0x048	48. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x3A (Data Out 41))	UINT32	RO	0x7001:3A, 16
1605:31	SubIndex 0x049	49. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x3B (Data Out 42))	UINT32	RO	0x7001:3B, 16
1605:32	SubIndex 0x050	50. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x3C (Data Out 43))	UINT32	RO	0x7001:3C, 16
1605:33	SubIndex 0x051	51. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x3D (Data Out 44))	UINT32	RO	0x7001:3D, 16
1605:34	SubIndex 0x052	52. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x3E (Data Out 45))	UINT32	RO	0x7001:3E, 16
1605:35	SubIndex 0x053	53. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x3F (Data Out 46))	UINT32	RO	0x7001:3F, 16
1605:36	SubIndex 0x054	54. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x40 (Data Out 47))	UINT32	RO	0x7001:40, 16
1605:37	SubIndex 0x055	55. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x41 (Data Out 48))	UINT32	RO	0x7001:41, 16
1605:38	SubIndex 0x056	56. PDO Mapping entry (object 0x7001 (COM ext. outputs), entry 0x42 (Data Out 49))	UINT32	RO	0x7001:42, 16

Index 0x1800 TxPDO-Par Inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1800:0	TxPDO-Par Inputs	PDO Parameter TxPDO 1	UINT8	RO	0x06 (6_{dec})
1800:06	Exclude TxPDOs	Specifies the TxPDOs (index 0x of TxPDO mapping objects) that must not be transferred together with TxPDO 1	OCTET-STRING[6]	RO	01 1A 02 1A 04 1A

Index 0x1801 TxPDO-Par Inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1801:0	TxPDO-Par Inputs	PDO Parameter TxPDO 2	UINT8	RO	0x06 (6_{dec})
1801:06	Exclude TxPDOs	Specifies the TxPDOs (index 0x of TxPDO mapping objects) that must not be transferred together with TxPDO 2	OCTET-STRING[6]	RO	00 1A 02 1A 04 1A

Index 0x1802 TxPDO-Par Inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1802:0	TxPDO-Par Inputs	PDO Parameter TxPDO 3	UINT8	RO	0x06 (6_{dec})
1802:06	Exclude TxPDOs	Specifies the TxPDOs (index 0x of TxPDO mapping objects) that must not be transferred together with TxPDO 3	OCTET-STRING[6]	RO	00 1A 01 1A 04 1A

Index 0x1804 COM TxPDO-Par Inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1804:0	COM TxPDO-Par Inputs	PDO Parameter TxPDO 5	UINT8	RO	0x06 (6_{dec})
1804:06	Exclude TxPDOs	Specifies the TxPDOs (index 0x of TxPDO mapping objects) that must not be transferred together with TxPDO 5	OCTET-STRING[8]	RO	00 1A 01 1A 02 1A 05 1A

Index 0x1805 COM ext. inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1805:0	COM ext. inputs	PDO Parameter TxPDO 6	UINT8	RO	0x06 (6_{dec})
1805:06	Exclude TxPDOs	Specifies the TxPDOs (index 0x of TxPDO mapping objects) that must not be transferred together with TxPDO 6	OCTET-STRING[8]	RO	00 1A 01 1A 02 1A 04 1A

Index 0x1A00 TxPDO-Map Inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1A00:0	TxPDO-Map Inputs	PDO Mapping TxPDO 1	UINT8	RO	0x04 (4_{dec})
1A00:01	SubIndex 0x001	1. PDO Mapping entry (object 0x3101 (Inputs), entry 0x01 (Status))	UINT32	RO	0x3101:01, 8
1A00:02	SubIndex 0x002	2. PDO Mapping entry (object 0x3101 (Inputs), entry 0x02 (Data In 0))	UINT32	RO	0x3101:02, 8
1A00:03	SubIndex 0x003	3. PDO Mapping entry (object 0x3101 (Inputs), entry 0x03 (Data In 1))	UINT32	RO	0x3101:03, 8
1A00:04	SubIndex 0x004	4. PDO Mapping entry (object 0x3101 (Inputs), entry 0x04 (Data In 2))	UINT32	RO	0x3101:04, 8

Index 0x1A01 TxPDO-Map Inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1A01:0	TxPDO-Map Inputs	PDO Mapping TxPDO 2	UINT8	RO	0x06 (6_{dec})
1A01:01	SubIndex 0x001	1. PDO Mapping entry (object 0x3102 (Inputs), entry 0x01 (Status))	UINT32	RO	0x3102:01, 8
1A01:02	SubIndex 0x002	2. PDO Mapping entry (object 0x3102 (Inputs), entry 0x02 (Data In 0))	UINT32	RO	0x3102:02, 8
1A01:03	SubIndex 0x003	3. PDO Mapping entry (object 0x3102 (Inputs), entry 0x03 (Data In 1))	UINT32	RO	0x3102:03, 8
1A01:04	SubIndex 0x004	4. PDO Mapping entry (object 0x3102 (Inputs), entry 0x04 (Data In 2))	UINT32	RO	0x3102:04, 8
1A01:05	SubIndex 0x005	5. PDO Mapping entry (object 0x3102 (Inputs), entry 0x05 (Data In 3))	UINT32	RO	0x3102:05, 8
1A01:06	SubIndex 0x006	6. PDO Mapping entry (object 0x3102 (Inputs), entry 0x06 (Data In 4))	UINT32	RO	0x3102:06, 8

Index 0x1A02 TxPDO-Map Inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1A02:0	TxPDO-Map Inputs	PDO Mapping TxPDO 3	UINT8	RO	0x17 (23_{dec})
1A02:01	SubIndex 0x001	1. PDO Mapping entry (object 0x3103 (Inputs), entry 0x01 (Status))	UINT32	RO	0x3103:01, 16
1A02:02	SubIndex 0x002	2. PDO Mapping entry (object 0x3103 (Inputs), entry 0x02 (Data In 0))	UINT32	RO	0x3103:02, 8
1A02:03	SubIndex 0x003	3. PDO Mapping entry (object 0x3103 (Inputs), entry 0x03 (Data In 1))	UINT32	RO	0x3103:03, 8
1A02:04	SubIndex 0x004	4. PDO Mapping entry (object 0x3103 (Inputs), entry 0x04 (Data In 2))	UINT32	RO	0x3103:04, 8
1A02:05	SubIndex 0x005	5. PDO Mapping entry (object 0x3103 (Inputs), entry 0x05 (Data In 3))	UINT32	RO	0x3103:05, 8
1A02:06	SubIndex 0x006	6. PDO Mapping entry (object 0x3103 (Inputs), entry 0x06 (Data In 4))	UINT32	RO	0x3103:06, 8
1A02:07	SubIndex 0x007	7. PDO Mapping entry (object 0x3103 (Inputs), entry 0x07 (Data In 5))	UINT32	RO	0x3103:07, 8
1A02:08	SubIndex 0x008	8. PDO Mapping entry (object 0x3103 (Inputs), entry 0x08 (Data In 6))	UINT32	RO	0x3103:08, 8
1A02:09	SubIndex 0x009	9. PDO Mapping entry (object 0x3103 (Inputs), entry 0x09 (Data In 7))	UINT32	RO	0x3103:09, 8
1A02:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x3103 (Inputs), entry 0x0A (Data In 8))	UINT32	RO	0x3103:0A, 8
1A02:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x3103 (Inputs), entry 0x0B (Data In 9))	UINT32	RO	0x3103:0B, 8
1A02:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x3103 (Inputs), entry 0x0C (Data In 10))	UINT32	RO	0x3103:0C, 8
1A02:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x3103 (Inputs), entry 0x0D (Data In 11))	UINT32	RO	0x3103:0D, 8
1A02:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x3103 (Inputs), entry 0x0E (Data In 12))	UINT32	RO	0x3103:0E, 8
1A02:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x3103 (Inputs), entry 0x0F (Data In 13))	UINT32	RO	0x3103:0F, 8
1A02:10	SubIndex 0x016	16. PDO Mapping entry (object 0x3103 (Inputs), entry 0x10 (Data In 14))	UINT32	RO	0x3103:10, 8
1A02:11	SubIndex 0x017	17. PDO Mapping entry (object 0x3103 (Inputs), entry 0x11 (Data In 15))	UINT32	RO	0x3103:11, 8
1A02:12	SubIndex 0x018	18. PDO Mapping entry (object 0x3103 (Inputs), entry 0x12 (Data In 16))	UINT32	RO	0x3103:12, 8
1A02:13	SubIndex 0x019	19. PDO Mapping entry (object 0x3103 (Inputs), entry 0x13 (Data In 17))	UINT32	RO	0x3103:13, 8
1A02:14	SubIndex 0x020	20. PDO Mapping entry (object 0x3103 (Inputs), entry 0x14 (Data In 18))	UINT32	RO	0x3103:14, 8
1A02:15	SubIndex 0x021	21. PDO Mapping entry (object 0x3103 (Inputs), entry 0x15 (Data In 19))	UINT32	RO	0x3103:15, 8
1A02:16	SubIndex 0x022	22. PDO Mapping entry (object 0x3103 (Inputs), entry 0x16 (Data In 20))	UINT32	RO	0x3103:16, 8
1A02:17	SubIndex 0x023	23. PDO Mapping entry (object 0x3103 (Inputs), entry 0x17 (Data In 21))	UINT32	RO	0x3103:17, 8

Index 0x1A04 COM TxPDO-Map Inputs [ab Hardwarestand 03]

Index (hex)	Name	Meaning	Data type	Flags	Default
1A04:0	COM TxPDO-Map Inputs	PDO Mapping TxPDO 5	UINT8	RO	0x1F (31_{dec})
1A04:01	SubIndex 0x001	1. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x01 (Transmit accepted))	UINT32	RO	0x6000:01, 1
1A04:02	SubIndex 0x002	2. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x02 (Receive request))	UINT32	RO	0x6000:02, 1
1A04:03	SubIndex 0x003	3. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x03 (Init accepted))	UINT32	RO	0x6000:03, 1
1A04:04	SubIndex 0x004	4. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x04 (Buffer full))	UINT32	RO	0x6000:04, 1
1A04:05	SubIndex 0x005	5. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x05 (Parity error))	UINT32	RO	0x6000:05, 1
1A04:06	SubIndex 0x006	6. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x06 (Framing error))	UINT32	RO	0x6000:06, 1
1A04:07	SubIndex 0x007	7. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x07 (Overrun error))	UINT32	RO	0x6000:07, 1
1A04:08	SubIndex 0x008	8. PDO Mapping entry (1 bits align)	UINT32	RO	0x0000:00, 1
1A04:09	SubIndex 0x009	9. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x09 (Input length))	UINT32	RO	0x6000:09, 8
1A04:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x11 (Data In 0))	UINT32	RO	0x6000:11, 8
1A04:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x12 (Data In 1))	UINT32	RO	0x6000:12, 8
1A04:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x13 (Data In 2))	UINT32	RO	0x6000:13, 8
1A04:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x14 (Data In 3))	UINT32	RO	0x6000:14, 8
1A04:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x15 (Data In 4))	UINT32	RO	0x6000:15, 8
1A04:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x16 (Data In 5))	UINT32	RO	0x6000:16, 8
1A04:10	SubIndex 0x016	16. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x17 (Data In 6))	UINT32	RO	0x6000:17, 8
1A04:11	SubIndex 0x017	17. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x18 (Data In 7))	UINT32	RO	0x6000:18, 8
1A04:12	SubIndex 0x018	18. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x19 (Data In 8))	UINT32	RO	0x6000:19, 8
1A04:13	SubIndex 0x019	19. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x1A (Data In 9))	UINT32	RO	0x6000:1A, 8
1A04:14	SubIndex 0x020	20. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x1B (Data In 10))	UINT32	RO	0x6000:1B, 8
1A04:15	SubIndex 0x021	21. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x1C (Data In 11))	UINT32	RO	0x6000:1C, 8
1A04:16	SubIndex 0x022	22. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x1D (Data In 12))	UINT32	RO	0x6000:1D, 8
1A04:17	SubIndex 0x023	23. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x1E (Data In 13))	UINT32	RO	0x6000:1E, 8
1A04:18	SubIndex 0x024	24. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x1F (Data In 14))	UINT32	RO	0x6000:1F, 8
1A04:19	SubIndex 0x025	25. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x20 (Data In 15))	UINT32	RO	0x6000:20, 8
1A04:1A	SubIndex 0x026	26. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x21 (Data In 16))	UINT32	RO	0x6000:21, 8
1A04:1B	SubIndex 0x027	27. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x22 (Data In 17))	UINT32	RO	0x6000:22, 8
1A04:1C	SubIndex 0x028	28. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x23 (Data In 18))	UINT32	RO	0x6000:23, 8
1A04:1D	SubIndex 0x029	29. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x24 (Data In 19))	UINT32	RO	0x6000:24, 8
1A04:1E	SubIndex 0x030	30. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x25 (Data In 20))	UINT32	RO	0x6000:25, 8
1A04:1F	SubIndex 0x031	31. PDO Mapping entry (object 0x6000 (COM Inputs), entry 0x26 (Data In 21))	UINT32	RO	0x6000:26, 8

Index 0x1A05 COM ext. inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
1A05:0	COM ext. inputs	PDO Mapping TxPDO 6	UINT8	RO	0x3B (59_{dec})
1A05:01	SubIndex 0x001	1. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x01 (Transmit accepted))	UINT32	RO	0x6001:01, 1
1A05:02	SubIndex 0x002	2. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x02 (Receive request))	UINT32	RO	0x6001:02, 1
1A05:03	SubIndex 0x003	3. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x03 (Init accepted))	UINT32	RO	0x6001:03, 1
1A05:04	SubIndex 0x004	4. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x04 (Buffer full))	UINT32	RO	0x6001:04, 1
1A05:05	SubIndex 0x005	5. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x05 (Parity error))	UINT32	RO	0x6001:05, 1
1A05:06	SubIndex 0x006	6. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x06 (Framing error))	UINT32	RO	0x6001:06, 1
1A05:07	SubIndex 0x007	7. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x07 (Overrun error))	UINT32	RO	0x6001:07, 1
1A05:08	SubIndex 0x008	8. PDO Mapping entry (1 bits align)	UINT32	RO	0x0000:00, 1
1A05:09	SubIndex 0x009	9. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x09 (Input length))	UINT32	RO	0x6001:09, 8
1A05:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x11 (Data In 0))	UINT32	RO	0x6001:11, 16
1A05:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x12 (Data In 1))	UINT32	RO	0x6001:12, 16
1A05:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x13 (Data In 2))	UINT32	RO	0x6001:13, 16
1A05:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x14 (Data In 3))	UINT32	RO	0x6001:14, 16
1A05:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x15 (Data In 4))	UINT32	RO	0x6001:15, 16
1A05:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x16 (Data In 5))	UINT32	RO	0x6001:16, 16
1A05:10	SubIndex 0x016	16. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x17 (Data In 6))	UINT32	RO	0x6001:17, 16
1A05:11	SubIndex 0x017	17. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x18 (Data In 7))	UINT32	RO	0x6001:18, 16
1A05:12	SubIndex 0x018	18. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x19 (Data In 8))	UINT32	RO	0x6001:19, 16
1A05:13	SubIndex 0x019	19. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x1A (Data In 9))	UINT32	RO	0x6001:1A, 16
1A05:14	SubIndex 0x020	20. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x1B (Data In 10))	UINT32	RO	0x6001:1B, 16
1A05:15	SubIndex 0x021	21. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x1C (Data In 11))	UINT32	RO	0x6001:1C, 16
1A05:16	SubIndex 0x022	22. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x1D (Data In 12))	UINT32	RO	0x6001:1D, 16
1A05:17	SubIndex 0x023	23. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x1E (Data In 13))	UINT32	RO	0x6001:1E, 16
1A05:18	SubIndex 0x024	24. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x1F (Data In 14))	UINT32	RO	0x6001:1F, 16
1A05:19	SubIndex 0x025	25. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x20 (Data In 15))	UINT32	RO	0x6001:20, 16
1A05:1A	SubIndex 0x026	26. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x21 (Data In 16))	UINT32	RO	0x6001:21, 16
1A05:1B	SubIndex 0x027	27. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x22 (Data In 17))	UINT32	RO	0x6001:22, 16
1A05:1C	SubIndex 0x028	28. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x23 (Data In 18))	UINT32	RO	0x6001:23, 16
1A05:1D	SubIndex 0x029	29. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x24 (Data In 19))	UINT32	RO	0x6001:24, 16
1A05:1E	SubIndex 0x030	30. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x25 (Data In 20))	UINT32	RO	0x6001:25, 16
1A05:1F	SubIndex 0x031	31. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x26 (Data In 21))	UINT32	RO	0x6001:26, 16

Index (hex)	Name	Meaning	Data type	Flags	Default
1A05:20	SubIndex 0x032	32. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x27 (Data In 22))	UINT32	RO	0x6001:27, 16
1A05:21	SubIndex 0x033	33. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x28 (Data In 23))	UINT32	RO	0x6001:28, 16
1A05:22	SubIndex 0x034	34. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x29 (Data In 24))	UINT32	RO	0x6001:29, 16
1A05:23	SubIndex 0x035	35. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x2A (Data In 25))	UINT32	RO	0x6001:2A, 16
1A05:24	SubIndex 0x036	36. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x2B (Data In 26))	UINT32	RO	0x6001:2B, 16
1A05:25	SubIndex 0x037	37. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x2C (Data In 27))	UINT32	RO	0x6001:2C, 16
1A05:26	SubIndex 0x038	38. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x2D (Data In 28))	UINT32	RO	0x6001:2D, 16
1A05:27	SubIndex 0x039	39. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x2E (Data In 29))	UINT32	RO	0x6001:2E, 16
1A05:28	SubIndex 0x040	40. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x2F (Data In 30))	UINT32	RO	0x6001:2F, 16
1A05:29	SubIndex 0x041	41. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x30 (Data In 31))	UINT32	RO	0x6001:30, 16
1A05:2A	SubIndex 0x042	42. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x31 (Data In 32))	UINT32	RO	0x6001:31, 16
1A05:2B	SubIndex 0x043	43. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x32 (Data In 33))	UINT32	RO	0x6001:32, 16
1A05:2C	SubIndex 0x044	44. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x33 (Data In 34))	UINT32	RO	0x6001:33, 16
1A05:2D	SubIndex 0x045	45. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x34 (Data In 35))	UINT32	RO	0x6001:34, 16
1A05:2E	SubIndex 0x046	46. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x35 (Data In 36))	UINT32	RO	0x6001:35, 16
1A05:2F	SubIndex 0x047	47. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x36 (Data In 37))	UINT32	RO	0x6001:36, 16
1A05:30	SubIndex 0x048	48. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x37 (Data In 38))	UINT32	RO	0x6001:37, 16
1A05:31	SubIndex 0x049	49. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x38 (Data In 39))	UINT32	RO	0x6001:38, 16
1A05:32	SubIndex 0x050	50. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x39 (Data In 40))	UINT32	RO	0x6001:39, 16
1A05:33	SubIndex 0x051	51. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x3A (Data In 41))	UINT32	RO	0x6001:3A, 16
1A05:34	SubIndex 0x052	52. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x3B (Data In 42))	UINT32	RO	0x6001:3B, 16
1A05:35	SubIndex 0x053	53. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x3C (Data In 43))	UINT32	RO	0x6001:3C, 16
1A05:36	SubIndex 0x054	54. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x3D (Data In 44))	UINT32	RO	0x6001:3D, 16
1A05:37	SubIndex 0x055	55. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x3E (Data In 45))	UINT32	RO	0x6001:3E, 16
1A05:38	SubIndex 0x056	56. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x3F (Data In 46))	UINT32	RO	0x6001:3F, 16
1A05:39	SubIndex 0x057	57. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x40 (Data In 47))	UINT32	RO	0x6001:40, 16
1A05:3A	SubIndex 0x058	58. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x41 (Data In 48))	UINT32	RO	0x6001:41, 16
1A05:3B	SubIndex 0x059	59. PDO Mapping entry (object 0x6001 (COM ext. inputs), entry 0x42 (Data In 49))	UINT32	RO	0x6001:42, 16

Index 0x1C00 Sync manager type

Index (hex)	Name	Meaning	Data type	Flags	Default
1C00:0	Sync manager type	Using the sync managers	UINT8	RO	0x04 (4_{dec})
1C00:01	SubIndex 0x001	Sync-Manager Type Channel 1: Mailbox Write	UINT8	RO	0x01 (1_{dec})
1C00:02	SubIndex 0x002	Sync-Manager Type Channel 2: Mailbox Read	UINT8	RO	0x02 (2_{dec})
1C00:03	SubIndex 0x003	Sync-Manager Type Channel 3: Process Data Write (Outputs)	UINT8	RO	0x03 (3_{dec})
1C00:04	SubIndex 0x004	Sync-Manager Type Channel 4: Process Data Read (Inputs)	UINT8	RO	0x04 (4_{dec})

Index 0x1C12 RxPDO assign

Index (hex)	Name	Meaning	Data type	Flags	Default
1C12:0	RxPDO assign	PDO Assign Outputs	UINT8	RW	0x01 (1_{dec})
1C12:01	SubIndex 0x001	1st allocated RxPDO (contains the index 0x of the associated RxPDO mapping object)	UINT16	RW	0x1602 (5634_{dec})
1C12:02	SubIndex 0x002	2nd allocated RxPDO (contains the index 0x of the associated RxPDO mapping object)	UINT16	RW	0x0000 (0_{dec})

Index 0x1C13 TxPDO assign

Index (hex)	Name	Meaning	Data type	Flags	Default
1C13:0	TxPDO assign	PDO Assign Inputs	UINT8	RW	0x01 (1_{dec})
1C13:01	SubIndex 0x001	1st allocated TxPDO (contains the index 0x of the associated TxPDO mapping object)	UINT16	RW	0x1A02 (6658_{dec})
1C13:02	SubIndex 0x002	2nd allocated TxPDO (contains the index 0x of the associated TxPDO mapping object)	UINT16	RW	0x0000 (0_{dec})

Index 0x1C32 SM output parameter [from hardware version 03]

Index (hex)	Name	Meaning	Data type	Flags	Default
1C32:0	SM output parameter	Synchronization parameters for the outputs	UINT8	RO	0x20 (32_{dec})
1C32:01	Sync mode	Current synchronization mode: • 0: Free Run • 1: Synchron with SM 2 Event • 2: DC-Mode - Synchron with SYNC0 Event • 3: DC-Mode - Synchron with SYNC1 Event	UINT16	RW	0x0001 (1_{dec})
1C32:02	Cycle time	Cycle time (in ns): • Free Run: Cycle time of the local timer • Synchronous with SM 2 event: Master cycle time • DC-Mode: SYNC0/SYNC1 Cycle Time	UINT32	RW	0x0007A120 (500000_{dec})
1C32:03	Shift time	Time between SYNC0 event and output of the outputs (in ns, DC mode only)	UINT32	RO	0x00000000 (0_{dec})
1C32:04	Sync modes supported	Supported synchronization modes: • Bit 0 = 1: free run is supported • Bit 1 = 1: Synchronous with SM 2 event is supported • Bit 2-3 = 01: DC mode is supported • Bit 4-5 = 10: Output shift with SYNC1 event (only DC mode) • Bit 14 = 1: dynamic times (measurement through writing of 1C32:08 [▶ 162])	UINT16	RO	0xC007 (49159_{dec})
1C32:05	Minimum cycle time	Minimum cycle time (in ns)	UINT32	RO	0x00004E20 (20000_{dec})

Index (hex)	Name	Meaning	Data type	Flags	Default
1C32:06	Calc and copy time	Minimum time between SYNC0 and SYNC1 event (in ns, DC mode only)	UINT32	RO	0x00000000 (0_{dec})
1C32:08	Command	• 0: Measurement of the local cycle time is stopped • 1: Measurement of the local cycle time is started The entries 1C32:03, 1C32:05, 1C32:06, 1C32:09 [▶ 162], 1C33:03, 1C33:06, 1C33:09 [▶ 163] are updated with the maximum measured values. For a subsequent measurement the measured values are reset	UINT16	RW	0x0000 (0_{dec})
1C32:09	Delay time	Time between SYNC1 event and output of the outputs (in ns, DC mode only)	UINT32	RO	0x00000000 (0_{dec})
1C32:0B	SM event missed counter	Number of missed SM events in OPERATIONAL (DC mode only)	UINT16	RO	0x0000 (0_{dec})
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_{dec})
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_{dec})
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_{dec})

Index 0x1C33 SM input parameter [from hardware version 03]

Index (hex)	Name	Meaning	Data type	Flags	Default
1C33:0	SM input parameter	Synchronization parameters for the inputs	UINT8	RO	0x20 (32_{dec})
1C33:01	Sync mode	Current synchronization mode: • 0: Free Run • 1: Synchronous with SM 3 event (no outputs available) • 2: DC - Synchron with SYNC0 Event • 3: DC - Synchron with SYNC1 Event • 34: Synchronous with SM 2 event (outputs available)	UINT16	RW	0x0022 (34_{dec})
1C33:02	Cycle time	as 1C32:02 [▶ 162]	UINT32	RW	0x0007A120 (500000_{dec})
1C33:03	Shift time	Time between SYNC0 event and reading of the inputs (in ns, only DC mode)	UINT32	RO	0x00000000 (0_{dec})
1C33:04	Sync modes supported	Supported synchronization modes: • Bit 0: free run is supported • Bit 1: synchronous with SM 2 event is supported (outputs available) • Bit 1: synchronous with SM 3 event is supported (no outputs available) • Bit 2-3 = 01: DC mode is supported • Bit 4-5 = 01: input shift through local event (outputs available) • Bit 4-5 = 10: input shift with SYNC1 event (no outputs available) • Bit 14 = 1: dynamic times (measurement through writing of 1C32:08 [▶ 162] or 1C33:08 [▶ 163])	UINT16	RO	0xC007 (49159_{dec})
1C33:05	Minimum cycle time	as 1C32:05 [▶ 162]	UINT32	RO	0x00004E20 (20000_{dec})
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_{dec})
1C33:08	Command	as 1C32:08 [▶ 162]	UINT16	RW	0x0000 (0_{dec})

Index (hex)	Name	Meaning	Data type	Flags	Default
1C33:09	Delay time	Time between SYNC1 event and reading of the inputs (in ns, only DC mode)	UINT32	RO	0x00000000 (0_{dec})
1C33:0B	SM event missed counter	as 1C32:11 [▶ 162]	UINT16	RO	0x0000 (0_{dec})
1C33:0C	Cycle exceeded counter	as 1C32:12 [▶ 162]	UINT16	RO	0x0000 (0_{dec})
1C33:0D	Shift too short counter	as 1C32:13 [▶ 162]	UINT16	RO	0x0000 (0_{dec})
1C33:20	Sync error	as 1C32:32 [▶ 162]	BOOLEAN	RO	0x00 (0_{dec})

Index 0x3001 Outputs

Index (hex)	Name	Meaning	Data type	Flags	Default
3001:0	Outputs	Length of this object	UINT8	RO	0x04 (4_{dec})
3001:01	Ctrl	Control-Byte [▶ 169]	UINT8	RO	0x00 (0_{dec})
3001:02	Data Out 0	Output byte 0	UINT8	RO	0x00 (0_{dec})
3001:03	Data Out 1	Output byte 1	UINT8	RO	0x00 (0_{dec})
3001:04	Data Out 2	Output byte 2	UINT8	RO	0x00 (0_{dec})

Index 0x3002 Outputs

Index (hex)	Name	Meaning	Data type	Flags	Default
3002:0	Outputs	Length of this object	UINT8	RO	0x06 (6_{dec})
3002:01	Ctrl	Control-Byte [▶ 169]	UINT8	RO	0x00 (0_{dec})
3002:02	Data Out 0	Output byte 0	UINT8	RO	0x00 (0_{dec})
3002:03	Data Out 1	Output byte 1	UINT8	RO	0x00 (0_{dec})
3002:04	Data Out 2	Output byte 2	UINT8	RO	0x00 (0_{dec})
3002:05	Data Out 3	Output byte 3	UINT8	RO	0x00 (0_{dec})
3002:06	Data Out 4	Output byte 4	UINT8	RO	0x00 (0_{dec})

Index 0x3003 Outputs

Index (hex)	Name	Meaning	Data type	Flags	Default
3003:0	Outputs	Length of this object	UINT8	RO	0x17 (23_{dec})
3003:01	Ctrl	Control word [▶ 169]	UINT16	RO	0x0000 (0_{dec})
3003:02	Data Out 0	Output byte 0	UINT8	RO	0x00 (0_{dec})
3003:03	Data Out 1	Output byte 1	UINT8	RO	0x00 (0_{dec})
3003:04	Data Out 2	Output byte 2	UINT8	RO	0x00 (0_{dec})
3003:05	Data Out 3	Output byte 3	UINT8	RO	0x00 (0_{dec})
3003:06	Data Out 4	Output byte 4	UINT8	RO	0x00 (0_{dec})
3003:07	Data Out 5	Output byte 5	UINT8	RO	0x00 (0_{dec})
3003:08	Data Out 6	Output byte 6	UINT8	RO	0x00 (0_{dec})
3003:09	Data Out 7	Output byte 7	UINT8	RO	0x00 (0_{dec})
3003:0A	Data Out 8	Output byte 8	UINT8	RO	0x00 (0_{dec})
3003:0B	Data Out 9	Output byte 9	UINT8	RO	0x00 (0_{dec})
3003:0C	Data Out 10	Output byte 10	UINT8	RO	0x00 (0_{dec})
3003:0D	Data Out 11	Output byte 11	UINT8	RO	0x00 (0_{dec})
3003:0E	Data Out 12	Output byte 12	UINT8	RO	0x00 (0_{dec})
3003:0F	Data Out 13	Output byte 13	UINT8	RO	0x00 (0_{dec})
3003:10	Data Out 14	Output byte 14	UINT8	RO	0x00 (0_{dec})
3003:11	Data Out 15	Output byte 15	UINT8	RO	0x00 (0_{dec})
3003:12	Data Out 16	Output byte 16	UINT8	RO	0x00 (0_{dec})
3003:13	Data Out 17	Output byte 17	UINT8	RO	0x00 (0_{dec})
3003:14	Data Out 18	Output byte 18	UINT8	RO	0x00 (0_{dec})
3003:15	Data Out 19	Output byte 19	UINT8	RO	0x00 (0_{dec})
3003:16	Data Out 20	Output byte 20	UINT8	RO	0x00 (0_{dec})
3003:17	Data Out 21	Output byte 21	UINT8	RO	0x00 (0_{dec})

Index 0x3101 Inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
3101:0	Inputs	Length of this object	UINT8	RO	0x04 (4_{dec})
3101:01	Status	Status byte [▶ 169]	UINT8	RO	0x00 (0_{dec})
3101:02	Data In 0	Input byte 0	UINT8	RO	0x00 (0_{dec})
3101:03	Data In 1	Input byte 1	UINT8	RO	0x00 (0_{dec})
3101:04	Data In 2	Input byte 2	UINT8	RO	0x00 (0_{dec})

Index 0x3102 Inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
3102:0	Inputs	Length of this object	UINT8	RO	0x06 (6_{dec})
3102:01	Status	Status byte [▶ 169]	UINT8	RO	0x00 (0_{dec})
3102:02	Data In 0	Input byte 0	UINT8	RO	0x00 (0_{dec})
3102:03	Data In 1	Input byte 1	UINT8	RO	0x00 (0_{dec})
3102:04	Data In 2	Input byte 2	UINT8	RO	0x00 (0_{dec})
3102:05	Data In 3	Input byte 3	UINT8	RO	0x00 (0_{dec})
3102:06	Data In 4	Input byte 4	UINT8	RO	0x00 (0_{dec})

Index 0x3103 Inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
3103:0	Inputs	Length of this object	UINT8	RO	0x17 (23_{dec})
3103:01	Status	Status word [▶ 169]	UINT16	RO	0x0000 (0_{dec})
3103:02	Data In 0	Input byte 0	UINT8	RO	0x00 (0_{dec})
3103:03	Data In 1	Input byte 1	UINT8	RO	0x00 (0_{dec})
3103:04	Data In 2	Input byte 2	UINT8	RO	0x00 (0_{dec})
3103:05	Data In 3	Input byte 3	UINT8	RO	0x00 (0_{dec})
3103:06	Data In 4	Input byte 4	UINT8	RO	0x00 (0_{dec})
3103:07	Data In 5	Input byte 5	UINT8	RO	0x00 (0_{dec})
3103:08	Data In 6	Input byte 6	UINT8	RO	0x00 (0_{dec})
3103:09	Data In 7	Input byte 7	UINT8	RO	0x00 (0_{dec})
3103:0A	Data In 8	Input byte 8	UINT8	RO	0x00 (0_{dec})
3103:0B	Data In 9	Input byte 9	UINT8	RO	0x00 (0_{dec})
3103:0C	Data In 10	Input byte 10	UINT8	RO	0x00 (0_{dec})
3103:0D	Data In 11	Input byte 11	UINT8	RO	0x00 (0_{dec})
3103:0E	Data In 12	Input byte 12	UINT8	RO	0x00 (0_{dec})
3103:0F	Data In 13	Input byte 13	UINT8	RO	0x00 (0_{dec})
3103:10	Data In 14	Input byte 14	UINT8	RO	0x00 (0_{dec})
3103:11	Data In 15	Input byte 15	UINT8	RO	0x00 (0_{dec})
3103:12	Data In 16	Input byte 16	UINT8	RO	0x00 (0_{dec})
3103:13	Data In 17	Input byte 17	UINT8	RO	0x00 (0_{dec})
3103:14	Data In 18	Input byte 18	UINT8	RO	0x00 (0_{dec})
3103:15	Data In 19	Input byte 19	UINT8	RO	0x00 (0_{dec})
3103:16	Data In 20	Input byte 20	UINT8	RO	0x00 (0_{dec})
3103:17	Data In 21	Input byte 21	UINT8	RO	0x00 (0_{dec})

Index 0x4070 Data bytes in send buffer

Index (hex)	Name	Meaning	Data type	Flags	Default
4070:0	Data bytes in send buffer	Number of data bytes in the send FIFO	UINT16	RO	0x0000 (0_{dec})

Index 0x4071 Data bytes in receive buffer

Index (hex)	Name	Meaning	Data type	Flags	Default
4071:0	Data bytes in receive buffer	Number of data bytes in the receive FIFO	UINT16	RO	0x0000 (0_{dec})

Index 0x4072 Diagnostic

Index (hex)	Name	Meaning	Data type	Flags	Default
4072:0	Diagnostic	Length of this object	UINT8	RO	0x05 (5_{dec})
4072:01	Buffer overflow	A buffer overflow has occurred.	BOOLEAN	RO	0x00 (0_{dec})
4072:02	Parity error	A parity error has occurred. The affected data item is not loaded into the receive FIFO of the terminal and is lost.	BOOLEAN	RO	0x00 (0_{dec})
4072:03	Framing error	A framing error has occurred. The affected data item is not loaded into the receive FIFO of the terminal and is lost.	BOOLEAN	RO	0x00 (0_{dec})
4072:04	Overrun error	An overrun error has occurred. The affected data item is not loaded into the receive FIFO of the terminal and is lost.	BOOLEAN	RO	0x00 (0_{dec})
4072:05	Buffer full	The reception FIFO is full. All further incoming data will be lost!	BOOLEAN	RO	0x00 (0_{dec})

6.1.3 Profile-specific objects (0x6000-0xFFFF) [from hardware version 03]

The profile-specific objects have the same meaning for all EtherCAT slaves that support the profile 5001.

Index 0x6000 COM Inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
6000:0	COM Inputs	Max. SubIndex (hex)	UINT8	RO	0x26 (38_{dec})
6000:01	Transmit accepted	The terminal acknowledges receipt of data by changing the state of this bit. Only now new data can be transferred from the controller to the terminal.	BOOLEAN	RO	0x00 (0_{dec})
6000:02	Receive request	By changing the state of this bit, the terminal informs the controller that the DataIn bytes contain the number of bytes displayed in “Input length [▶ 166]”. The controller must acknowledge receipt of the data by changing the state of the ReceiveAccepted [▶ 167] bit. Only then new data can be transferred from the terminal to the controller.	BOOLEAN	RO	0x00 (0_{dec})
6000:03	Init accepted	0 The terminal is ready again for serial data exchange. 1 Initialization was completed by the terminal.	BOOLEAN	RO	0x00 (0_{dec})
6000:04	Buffer full	The reception FIFO is full. All incoming data will be lost from this point on!	BOOLEAN	RO	0x00 (0_{dec})
6000:05	Parity error	A parity error has occurred.	BOOLEAN	RO	0x00 (0_{dec})
6000:06	Framing error	A framing error has occurred.	BOOLEAN	RO	0x00 (0_{dec})
6000:07	Overrun error	An overrun error has occurred.	BOOLEAN	RO	0x00 (0_{dec})
6000:09	Input length	Number of input bytes available for transfer from the terminal to the controller.	UINT8	RO	0x00 (0_{dec})
6000:11	Data In 0	Input byte 0	UINT8	RO	0x00 (0_{dec})
...	...	....	...	...	...
6000:26	Data In 21	Input byte 21	UINT8	RO	0x00 (0_{dec})

Index 0x6001 COM ext. inputs

Index (hex)	Name	Meaning	Data type	Flags	Default
6001:0	COM ext. inputs		UINT8	RO	0x00 (0_{dec})
6001:01	Transmit accepted	identical to index 0x6000	BOOLEAN	RO	0x00 (0_{dec})
6001:02	Receive request		BOOLEAN	RO	0x00 (0_{dec})
6001:03	Init accepted		BOOLEAN	RO	0x00 (0_{dec})
6001:04	Buffer full		BOOLEAN	RO	0x00 (0_{dec})
6001:05	Parity error		BOOLEAN	RO	0x00 (0_{dec})
6001:06	Framing error		BOOLEAN	RO	0x00 (0_{dec})
6001:07	Overrun error		BOOLEAN	RO	0x00 (0_{dec})
6001:09	Input length		UINT8	RO	0x00 (0_{dec})
6001:11	Data In 0		UINT16	RO	0x0000 (0_{dec})
	....				
6001:42	Data In 49		UINT16	RO	0x0000 (0_{dec})

Index 0x7000 COM Outputs

Index (hex)	Name	Meaning	Data type	Flags	Default
7000:0	COM Outputs	Max. SubIndex (hex)	UINT8	RO	0x26 (38_{dec})
7000:01	Transmit request	By changing the state of this bit, the controller informs the terminal that the DataOut bytes contain the number of bytes displayed in “Output length [▶ 167]”. The terminal acknowledges receipt of the data by changing the state of the “TransmitAccepted [▶ 166]” bit. Only now new data can be transferred from the controller to the terminal.	BOOLEAN	RO	0x00 (0_{dec})
7000:02	Receive accepted	The controller acknowledges receipt of data by changing the state of this bit. Only then new data can be transferred from the terminal to the controller.	BOOLEAN	RO	0x00 (0_{dec})
7000:03	Init request	0 The controller once again requests the terminal to prepare for serial data exchange. 1 The controller requests terminal for initialization. The transmit and receive functions will be blocked, the FIFO pointer will be reset and the interface will be initialized with the values of the responsible Settings object. The execution of the initialization will be acknowledged by the terminal with the ‘Init accepted [▶ 166]’ bit.	BOOLEAN	RO	0x00 (0_{dec})
7000:04	Send continuous	Continuous sending of data from the FIFO. The send buffer is filled (up to 128 bytes) by the controller. The filled buffer contents will be sent on the rising edge of the bit. If the data has been transmitted, the terminal informs the controller by setting the “Init accepted [▶ 166]” bit. “Init accepted [▶ 166]” is cleared with “SendContinuous [▶ 167]”.	BOOLEAN	RO	0x00 (0_{dec})
7000:09	Output length	Number of output bytes available for transfer from the controller to the terminal.	UINT8	RO	0x00 (0_{dec})
7000:11	Data Out 0	Output byte 0	UINT8	RO	0x00 (0_{dec})
...	...	....	...	...	...
7000:26	Data Out 21	Output byte 21	UINT8	RO	0x00 (0_{dec})

Index 0x7001 COM ext. outputs

Index (hex)	Name	Meaning	Data type	Flags	Default
7001:0	COM ext. outputs		UINT8	RO	0x00 (0_{dec})
7001:01	Transmit request	identical to index 0x7000	BOOLEAN	RO	0x00 (0_{dec})
7001:02	Receive accepted		BOOLEAN	RO	0x00 (0_{dec})
7001:03	Init request		BOOLEAN	RO	0x00 (0_{dec})
7001:04	Send continuous		BOOLEAN	RO	0x00 (0_{dec})
7001:09	Output length		UINT8	RO	0x00 (0_{dec})
7001:11	Data Out 0		UINT16	RO	0x0000 (0_{dec})
	…		<	<	<
7001:42	Data Out 49		UINT16	RO	0x0000 (0_{dec})

Index 0xA000 COM Diag data

Index (hex)	Name	Meaning	Data type	Flags	Default
A000:0	COM Diag data	Max. SubIndex (hex)	UINT8	RO	0x21 (33_{dec})
A000:01	Buffer overflow	A buffer overflow has occurred.	BOOLEAN	RO	0x00 (0_{dec})
A000:02	Parity error	A parity error has occurred.	BOOLEAN	RO	0x00 (0_{dec})
A000:03	Framing error	A framing error has occurred	BOOLEAN	RO	0x00 (0_{dec})
A000:04	Overrun error	An overrun error has occurred.	BOOLEAN	RO	0x00 (0_{dec})
A000:05	Buffer full	The reception FIFO is full. All incoming data will be lost from this point on!	BOOLEAN	RO	0x00 (0_{dec})
A000:11	Data bytes in send buffer	Number of data bytes in the send FIFO	UINT16	RO	0x0000 (0_{dec})
A000:21	Data bytes in receive buffer	Number of data bytes in the receive FIFO	UINT16	RO	0x0000 (0_{dec})

Index 0xB000 Command

Only EL6021 since FW06 and EL6001 since FW08.

Index (hex)	Name	Meaning	Data type	Flags	Default
B000:0	Command	Max. SubIndex (hex)	UINT8	RO	0x03 (3_{dec})
B000:01	Request	Commands can be sent to the terminal via the request object see command mode [▶ 129]	OCTET-STRING[2]	RW	{0}
B000:02	Status	Status of the command currently being executed 0: Command executed without error. 255: Command is being executed	UINT8	RO	0x00 (0_{dec})
B000:03	Response	Optional response value of the command Byte 0: see B000:02 Byte 1: not used 2-n: Service response Data	OCTET-STRING[6]	RO	{0}

Index 0xF000 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_{dec})
F000:01	Module Index 0xdistance	Index (hex) interval of the objects of the individual channels	UINT16	RO	0x0010 (16_{dec})
F000:02	Maximum number of modules	Number of channels	UINT16	RO	0x0002 (2_{dec})

Index 0xF008 Code word

Index (hex)	Name	Meaning	Data type	Flags	Default
F008:0	Code word	reserved	UINT32	RW	0x00000000 (0_{dec})

Index 0xF010 Module list

Index (hex)	Name	Meaning	Data type	Flags	Default
F010:0	Module list	Max. SubIndex (hex)	UINT8	RW	0x02 (2_{dec})
F010:01	SubIndex 0x001	-	UINT32	RW	0x00000000 (0_{dec})
F010:02	SubIndex 0x002	-	UINT32	RW	0x00000258 (600_{dec})

6.2 Control and status word

Control word

The control word (CW) is located in the output process image, and is transmitted from the controller to the terminal.

Bit	CW.15	CW.14	CW.13	CW.12	CW.11	CW.10	CW.9	CW.8	CW.7	CW.6	CW.5	CW.4	CW.3	CW.2	CW.1	CW.0
Name	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	-	OL2*	OL1*	OL0*	SC	IR	RA	TR

Bits CW.15 to CW.8 are only shown if the large process image is used.
If the small or medium process image is used, only bits CW.7 to CW.0 are shown as control bytes! The number of output bytes available for transfer from the controller to the terminal is displayed by bits OL2* ... OL0*.

Legend

Bit	Name	Description	
CW.15 ... CW8	OL7 ... OL0 (OutLenght)	1_{dec}... 22_{dec}	If the large process image is used: Number of output bytes available for transfer from the controller to the terminal.
		-	If the small/medium process image is used: not shown
CW.7	-	0_{bin}	reserved
CW.6 ... CW.4	OL2* ... OL0* (OutLenght*)	0	If the large process image is used: reserved
		1 ... 6	If the medium process image is used: Number of output bytes available for transfer from the controller to the terminal.
		1 ... 4	If the small process image is used: Number of output bytes available for transfer from the controller to the terminal.
CW.3	SC (SendContinuous)	rise	Continuous sending of data from the FIFO. The send buffer is filled (up to 128 bytes) by the controller. The buffer content is sent with rising edge of bit SC. The terminal acknowledges the data transfer to the controller through setting of bit SW.2. SW.2 is cancelled with CW.3.
CW.2	IR (InitRequest)	1_{bin}	The controller requests terminal for initialization. The send and receive functions are blocked, the FIFO pointers are reset, and the interface is initialized with the values of the responsible objects (baud rate 4073 [▶ 148], data frame 4074 [▶ 148], feature bits 4075 [▶ 149]). The terminal acknowledges completion of the initialization via bit SW.2 [▶ 170] (IA).
		0_{bin}	The controller once again requests the terminal to prepare for serial data exchange.
CW.1	RA (ReceiveAccepted)	toggle	The controller acknowledges receipt of data by changing the state of this bit. Only then new data can be transferred from the terminal to the controller.
CW.0	TR (TransmitRequest)	toggle	Via a change of state of this bit the controller notifies the terminal that the DataOut bytes contain the number of bytes indicated via the OL bits. The terminal acknowledges receipt of the data in the status byte via a change of state of bit SW.0 [▶ 170] (TA). Only now new data can be transferred from the controller to the terminal.

Status word

The status word (SW) is located in the input process image, and is transmitted from terminal to the controller.

Bit	SW.15	SW.14	SW.13	SW.12	SW.11	SW.10	SW.9	SW.8	SW.7	SW.6	SW.5	SW.4	SW.3	SW.2	SW.1	SW.0
Name (small and medium process image)	-	-	-	-	-	-	-	-	-	IL2*	IL1*	IL0*	BUF_F	IA	RR	TA
Name (large process image)	IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0	-	OVERRUN ERR	FRAMING ERR	PARITY ERR	BUF_F	IA	RR	TA

If the small or medium process image is used, only bits SW.7 to SW.0 are shown as status bytes! The number of input bytes available for transfer from the terminal to the controller is displayed by bits IL2* ... IL0*.

Legend

Bit	Name	Description	
SW.15 ... SW8	IL7 ... IL0 (InLenght*)	1_{dec} ... 22_{dec}	If the large process image is used: Number of input bytes available for transfer from the terminal to the controller.
		-	If the small/medium process image is used: not shown
SW.7	-	0_{bin}	reserved
SW.6	IL2* (InLenght*) OVERRUN ERR	0 ... 1	If the large process image is used: An overrun error has occurred. The affected data item is not loaded into the receive FIFO of the terminal and is lost.
		1 ... 6	If the medium process image is used: Number of input bytes available for transfer from the terminal to the controller.
		1 ... 4	If the small process image is used: Number of input bytes available for transfer from the terminal to the controller.
SW.5	IL1* (InLenght*) FRAMING ERR	0 ... 1	If the large process image is used: A framing error has occurred. The affected data item is not loaded into the receive FIFO of the terminal and is lost.
		1 ... 6	If the medium process image is used: Number of input bytes available for transfer from the terminal to the controller.
		1 ... 4	If the small process image is used: Number of input bytes available for transfer from the terminal to the controller.
SW.4	IL0* (InLenght*) PARITY ERR	0 ... 1	If the large process image is used: A parity error has occurred. The affected data item is not loaded into the receive FIFO of the terminal and is lost
		1 ... 6	If the medium process image is used: Number of input bytes available for transfer from the terminal to the controller.
		1 ... 4	If the small process image is used: Number of input bytes available for transfer from the terminal to the controller.
SW.3	BUF_F	1_{bin}	The reception FIFO is full. All further incoming data will be lost!
SW.2	IA (InitAccepted-Bit)	1_{bin}	Initialization was completed by the terminal.
		0_{bin}	The terminal is ready again for serial data exchange.
SW.1	RR (ReceiveRequest)	toggle	Via a change of state of this bit the terminal notifies the controller that the DataIn bytes contain the number of bytes indicated via the IL bits. The controller has to acknowledge receipt of the data in the control byte via a change of state of bit CW.1 [▶ 169] (RA). Only then new data can be transferred from the terminal to the controller.
SW.0	TA (TransmitAccepted)	toggle	The terminal acknowledges receipt of data by changing the state of this bit. Only now new data can be transferred from the controller to the terminal.

Data transfer examples

The examples use the large process image.

- Initialization:

Control word		Status word		Comment
CW.15 ... CW.8	CW.7 ... CW.0	SW.15 ... SW.8	SW.7 ... SW.0	
xxxx $xxxx_{bin}$	xxxx $xxxx_{bin}$	xxxx $xxxx_{bin}$	xxxx $xxxx_{bin}$	Start of data transmission
0000 0000_{bin}	0000 0**1**00_{bin}	0000 0000_{bin}	0000 0000_{bin}	The controller requests terminal initialization
0000 0000_{bin}	0000 0100_{bin}	0000 0000_{bin}	0000 0**1**00_{bin}	Command is executed: Terminal initialization is complete
0000 0000_{bin}	0000 0**0**00_{bin}	0000 0000_{bin}	0000 0100_{bin}	The controller requests the terminal to prepare for serial data exchange
0000 0000_{bin}	0000 0000_{bin}	0000 0000_{bin}	0000 0**0**00_{bin}	Command is executed: Terminal is ready for serial data exchange

- Data transfer from the controller to the terminal:

Control word		Status word		Comment
CW.15 ... CW.8	CW.7 ... CW.0	SW.15 ... SW.8	SW.7 ... SW.0	
0000 0000_{bin}	0000 0000_{bin}	xxxx $xxxx_{bin}$	0000 x0x0_{bin}	Start of data transmission
0000 00**10**$_{bin}$	0000 000**1**$_{bin}$	xxxx $xxxx_{bin}$	0000 x0x0_{bin}	The controller requests transmission of **2** bytes by the terminal
0000 0010_{bin}	0000 0001_{bin}	xxxx $xxxx_{bin}$	0000 x0x**1**$_{bin}$	Command is executed: Terminal has loaded **2** bytes into the transmission FIFO
000**1 0000**$_{bin}$	0000 000**0**$_{bin}$	xxxx $xxxx_{bin}$	0000 x0x1_{bin}	The controller requests transmission of **16** bytes by the terminal
0001 0000_{bin}	0000 0000_{bin}	xxxx $xxxx_{bin}$	0000 x0x**0**$_{bin}$	Command is executed: Terminal has loaded 16 bytes into the transmission FIFO

- Data transfer from the terminal to the controller:

Control word		Status word		Comment
CW.15 ... CW.8	CW.7 ... CW.0	SW.15 ... SW.8	SW.7 ... SW.0	
xxxx $xxxx_{bin}$	0000 000x$_{bin}$	0000 0000_{bin}	0000 000x$_{bin}$	Start of data transmission
xxxx $xxxx_{bin}$	0000 000x$_{bin}$	0000 00**11**$_{bin}$	0000 00**1**x$_{bin}$	The terminal requests transfer of **3** bytes from the controller
xxxx $xxxx_{bin}$	0000 00**1**x$_{bin}$	0000 0011_{bin}	0000 001x$_{bin}$	Acknowledgement: Controller has received 3 bytes from the reception FIFO
xxxx $xxxx_{bin}$	0000 001x$_{bin}$	000**1 0110**$_{bin}$	0000 00**0**x$_{bin}$	The terminal requests transfer of **22** bytes from the controller
xxxx $xxxx_{bin}$	0000 00**0**x$_{bin}$	0001 0110_{bin}	0000 000x$_{bin}$	Acknowledgement: Controller has received 22 bytes from the reception FIFO

7 Overview CoE objects EL6002, EL6022

7.1 Object description and parameterization

EtherCAT XML Device Description

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

Parameterization

The terminal is parameterized via the CoE - Online tab [▶ 113] (with a double-click on the respective object, see below).

Introduction

The CoE overview contains objects for different intended applications:

- Objects required for parameterization [▶ 172] during commissioning
- Objects for indicating internal settings [▶ 173] (may be fixed)
- Profile specific objects [▶ 185], which represent the status displays of the inputs and outputs.

The following section first describes the objects required for normal operation, followed by a complete overview of missing objects.

7.1.1 Objects for commissioning

Index 0x1011 Restore default parameters

Index (hex)	Name	Meaning	Data type	Flags	Default
1011:0	Restore default parameters [▶ 203]	Restore default parameters	UINT8	RO	0x01 (1_{dec})
1011:01	SubIndex 0x001	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_{dec})

Index 0x80n0 COM Settings Ch. 1 (n = 0), Ch. 2 (n = 1)

Index (hex)	Name	Meaning		Data type	Flags	Default
80n0:0	COM Settings Ch. 1 + Ch. 2	Max. SubIndex (hex)		UINT8	RO	0x1A (26_{dec})
80n0:01**	Enable RTS/CTS	FALSE	RTS/CTS not enabled	BOOLEAN	RW	0x01 (1_{dec})
		TRUE	RTS/CTS enabled			
80n0:02	Enable XON/XOFF supported tx data	FALSE	XON/XOFF is not supported for send data	BOOLEAN	RW	0x00 (0_{dec})
		TRUE	XON/XOFF is supported for send data			
80n0:03	Enable XON/XOFF supported rx data	FALSE	XON/XOFF is not supported for receive data	BOOLEAN	RW	0x00 (0_{dec})
		TRUE	XON/XOFF is supported for receive data			
80n0:04	Enable send FIFO data continuous	FALSE	No continuous sending of data from the FIFO	BOOLEAN	RW	0x00 (0_{dec})
		TRUE	Continuous sending of data from the FIFO enabled: The send buffer is filled (up to 128 bytes) by the controller. The filled buffer content is sent on rising edge of bit “SendContinious”. The terminal acknowledges the data transfer to the controller through setting of bit “InitAccepted”. “InitAccepted” is reset with “SendContinuous”.			
80n0:05	Enable transfer rate optimization	FALSE	Transfer rate optimization switched off	BOOLEAN	RW	0x01 (1_{dec})
		TRUE	Transfer rate optimization switched on: The content of the input buffer is automatically transferred into the process image if • no further byte was received for approx. 16 bit times (i.e. the time it would have taken to receive 2 bytes) after data were received; • the process image is filled			
80n0:06***	Enable half duplex	FALSE	Full-duplex mode	BOOLEAN	RW	0x00 (0_{dec})
		TRUE	Half-duplex mode			
80n0:07***	Enable point to point connection (RS422)	FALSE	Point-to-point connection disabled	BOOLEAN	RW	0x00 (0_{dec})
		TRUE	Point-to-point connection enabled			
80n0:11	Baud rate	Detailed information can be found in chapter Communication features [▶ 135]		BIT4	RW	0x06 (6_{dec})
80n0:15	Data frame	Detailed information can be found in chapter Communication features [▶ 135]		BIT4	RW	0x03 (3_{dec})
80n0:1A	Rx buffer full notification	The value specifies the number of data in the receive FIFO, from which the bit “buffer full” is set.		UINT16	RW	0x0360 (864_{dec})
80n0:1B****	Explicit baudrate	More detailed information can be found in chapters TcVirtualComDriver [▶ 133] and Communication features [▶ 135]		UINT32	RW	0x00002580 (9600_{dec})

**) only EL6002
***) only EL6022
****) only EL6002 from firmware 03 [▶ 190]

7.1.2 Standard objects (0x1000-0x1FFF)

The standard objects have the same meaning for all EtherCAT slaves.

Index 0x1000 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 CoE profile used (5001). The Hi-Word contains the module profile according to the modular device profile.	UINT32	RO	0x02581389 (39326601_{dec})

Index 0x1008 Device name

Index (hex)	Name	Meaning	Data type	Flags	Default
1008:0	Device name	Device name of the EtherCAT slave	STRING	RO	EL60xx

Index 0x1009 Hardware version

Index (hex)	Name	Meaning	Data type	Flags	Default
1009:0	Hardware version	Hardware version of the EtherCAT slave	STRING	RO	00

Index 0x100A Software version

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

Index 0x1018 Identity

Index (hex)	Name	Meaning	Data type	Flags	Default
1018:0	Identity	Information for identifying the slave	UINT8	RO	0x04 (4_{dec})
1018:01	Vendor ID	Vendor ID of the EtherCAT slave	UINT32	RO	0x00000002 (2_{dec})
1018:02	Product code	Product code of the EtherCAT slave	UINT32	RO	EL6002: 0x17723052 (393359442_{dec}) EL6022: 0x17863052 (394670162_{dec})
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	0x00100000 (1048576_{dec})
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_{dec})

Index 0x10F0 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_{dec})
10F0:01	Checksum	Checksum across all backup entries of the EtherCAT slave	UINT32	RO	0x00000000 (0_{dec})

Index 0x1600 COM RxPDO-Map Outputs Ch.1

Index (hex)	Name	Meaning	Data type	Flags	Default
1600:0	COM RxPDO-Map Outputs Ch.1	PDO Mapping RxPDO 1	UINT8	RO	0x1C (28_{dec})
1600:01	SubIndex 0x001	1. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x01 (Transmit request))	BOOLEAN	RO	0x7000:01, 1
1600:02	SubIndex 0x002	2. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x02 (Receive accepted))	BOOLEAN	RO	0x7000:02, 1
1600:03	SubIndex 0x003	3. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x03 (Init request))	BOOLEAN	RO	0x7000:03, 1
1600:04	SubIndex 0x004	4. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x04 (Send continuous))	BOOLEAN	RO	0x7000:04, 1
1600:05	SubIndex 0x005	5. PDO Mapping entry (4 bits align)	Align4	RO	0x0000:00, 4
1600:06	SubIndex 0x006	6. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x09 (Output length))	UINT8	RO	0x7000:09, 8
1600:07	SubIndex 0x007	7. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x11 (Data Out 0))	UINT8	RO	0x7000:11, 8
1600:08	SubIndex 0x008	8. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x12 (Data Out 1))	UINT8	RO	0x7000:12, 8
1600:09	SubIndex 0x009	9. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x13 (Data Out 2))	UINT8	RO	0x7000:13, 8
1600:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x14 (Data Out 3))	UINT8	RO	0x7000:14, 8
1600:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x15 (Data Out 4))	UINT8	RO	0x7000:15, 8
1600:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x16 (Data Out 5))	UINT8	RO	0x7000:16, 8
1600:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x17 (Data Out 6))	UINT8	RO	0x7000:17, 8
1600:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x18 (Data Out 7))	UINT8	RO	0x7000:18, 8
1600:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x19 (Data Out 8))	UINT8	RO	0x7000:19, 8
1600:10	SubIndex 0x016	16. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x1A (Data Out 9))	UINT8	RO	0x7000:1A, 8
1600:11	SubIndex 0x017	17. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x1B (Data Out 10))	UINT8	RO	0x7000:1B, 8
1600:12	SubIndex 0x018	18. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x1C (Data Out 11))	UINT8	RO	0x7000:1C, 8
1600:13	SubIndex 0x019	19. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x1D (Data Out 12))	UINT8	RO	0x7000:1D, 8
1600:14	SubIndex 0x020	20. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x1E (Data Out 13))	UINT8	RO	0x7000:1E, 8
1600:15	SubIndex 0x021	21. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x1F (Data Out 14))	UINT8	RO	0x7000:1F, 8
1600:16	SubIndex 0x022	22. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x20 (Data Out 15))	UINT8	RO	0x7000:20, 8
1600:17	SubIndex 0x023	23. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x21 (Data Out 16))	UINT8	RO	0x7000:21, 8
1600:18	SubIndex 0x024	24. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x22 (Data Out 17))	UINT8	RO	0x7000:22, 8
1600:19	SubIndex 0x025	25. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x23 (Data Out 18))	UINT8	RO	0x7000:23, 8
1600:1A	SubIndex 0x026	26. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x24 (Data Out 19))	UINT8	RO	0x7000:24, 8
1600:1B	SubIndex 0x027	27. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x25 (Data Out 20))	UINT8	RO	0x7000:25, 8
1600:1C	SubIndex 0x028	28. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x26 (Data Out 21))	UINT8	RO	0x7000:26, 8

Index 0x1601 COM RxPDO-Map Outputs Ch.2

Index (hex)	Name	Meaning	Data type	Flags	Default
1601:0	COM RxPDO-Map Outputs Ch.2	PDO Mapping RxPDO 2	UINT8	RO	0x1C (28_{dec})
1601:01	SubIndex 0x001	1. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x01 (Transmit request))	BOOLEAN	RO	0x7010:01, 1
1601:02	SubIndex 0x002	2. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x02 (Receive accepted))	BOOLEAN	RO	0x7010:02, 1
1601:03	SubIndex 0x003	3. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x03 (Init request))	BOOLEAN	RO	0x7010:03, 1
1601:04	SubIndex 0x004	4. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x04 (Send continuous))	BOOLEAN	RO	0x7010:04, 1
1601:05	SubIndex 0x005	5. PDO Mapping entry (4 bits align)	Align4	RO	0x0000:00, 4
1601:06	SubIndex 0x006	6. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x09 (Output length))	UINT8	RO	0x7010:09, 8
1601:07	SubIndex 0x007	7. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x11 (Data Out 0))	UINT8	RO	0x7010:11, 8
1601:08	SubIndex 0x008	8. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x12 (Data Out 1))	UINT8	RO	0x7010:12, 8
1601:09	SubIndex 0x009	9. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x13 (Data Out 2))	UINT8	RO	0x7010:13, 8
1601:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x14 (Data Out 3))	UINT8	RO	0x7010:14, 8
1601:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x15 (Data Out 4))	UINT8	RO	0x7010:15, 8
1601:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x16 (Data Out 5))	UINT8	RO	0x7010:16, 8
1601:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x17 (Data Out 6))	UINT8	RO	0x7010:17, 8
1601:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x18 (Data Out 7))	UINT8	RO	0x7010:18, 8
1601:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x19 (Data Out 8))	UINT8	RO	0x7010:19, 8
1601:10	SubIndex 0x016	16. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x1A (Data Out 9))	UINT8	RO	0x7010:1A, 8
1601:11	SubIndex 0x017	17. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x1B (Data Out 10))	UINT8	RO	0x7010:1B, 8
1601:12	SubIndex 0x018	18. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x1C (Data Out 11))	UINT8	RO	0x7010:1C, 8
1601:13	SubIndex 0x019	19. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x1D (Data Out 12))	UINT8	RO	0x7010:1D, 8
1601:14	SubIndex 0x020	20. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x1E (Data Out 13))	UINT8	RO	0x7010:1E, 8
1601:15	SubIndex 0x021	21. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x1F (Data Out 14))	UINT8	RO	0x7010:1F, 8
1601:16	SubIndex 0x022	22. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x20 (Data Out 15))	UINT8	RO	0x7010:20, 8
1601:17	SubIndex 0x023	23. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x21 (Data Out 16))	UINT8	RO	0x7010:21, 8
1601:18	SubIndex 0x024	24. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x22 (Data Out 17))	UINT8	RO	0x7010:22, 8
1601:19	SubIndex 0x025	25. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x23 (Data Out 18))	UINT8	RO	0x7010:23, 8
1601:1A	SubIndex 0x026	26. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x24 (Data Out 19))	UINT8	RO	0x7010:24, 8
1601:1B	SubIndex 0x027	27. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x25 (Data Out 20))	UINT8	RO	0x7010:25, 8
1601:1C	SubIndex 0x028	28. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x26 (Data Out 21))	UINT8	RO	0x7010:26, 8

Index 0x1604 COM RxPDO-Map Outputs Ch.1

Index (hex)	Name	Meaning	Data type	Flags	Default
1604:0	COM RxPDO-Map Outputs Ch.1	PDO Mapping RxPDO 1	UINT8	RO	0x17 (23_{dec})
1604:01	SubIndex 0x001	1. PDO Mapping entry (object 0x7001 (Ctrl Ch.1), entry 0x01 (Ctrl))	UINT16	RO	0x7001:01, 16
1604:02	SubIndex 0x002	2. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x11 (Data Out 0))	UINT8	RO	0x7000:11, 8
1604:03	SubIndex 0x003	3. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x12 (Data Out 1))	UINT8	RO	0x7000:12, 8
1604:04	SubIndex 0x004	4. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x13 (Data Out 2))	UINT8	RO	0x7000:13, 8
1604:05	SubIndex 0x005	5. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x14 (Data Out 3))	UINT8	RO	0x7000:14, 8
1604:06	SubIndex 0x006	6. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x15 (Data Out 4))	UINT8	RO	0x7000:15, 8
1604:07	SubIndex 0x007	7. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x16 (Data Out 5))	UINT8	RO	0x7000:16, 8
1604:08	SubIndex 0x008	8. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x17 (Data Out 6))	UINT8	RO	0x7000:17, 8
1604:09	SubIndex 0x009	9. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x18 (Data Out 7))	UINT8	RO	0x7000:18, 8
1604:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x19 (Data Out 8))	UINT8	RO	0x7000:19, 8
1604:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x1A (Data Out 9))	UINT8	RO	0x7000:1A, 8
1604:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x1B (Data Out 10))	UINT8	RO	0x7000:1B, 8
1604:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x1C (Data Out 11))	UINT8	RO	0x7000:1C, 8
1604:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x1D (Data Out 12))	UINT8	RO	0x7000:1D, 8
1604:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x1E (Data Out 13))	UINT8	RO	0x7000:1E, 8
1604:10	SubIndex 0x016	16. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x1F (Data Out 14))	UINT8	RO	0x7000:1F, 8
1604:11	SubIndex 0x017	17. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x20 (Data Out 15))	UINT8	RO	0x7000:20, 8
1604:12	SubIndex 0x018	18. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x21 (Data Out 16))	UINT8	RO	0x7000:21, 8
1604:13	SubIndex 0x019	19. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x22 (Data Out 17))	UINT8	RO	0x7000:22, 8
1604:14	SubIndex 0x020	20. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x23 (Data Out 18))	UINT8	RO	0x7000:23, 8
1604:15	SubIndex 0x021	21. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x24 (Data Out 19))	UINT8	RO	0x7000:24, 8
1604:16	SubIndex 0x022	22. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x25 (Data Out 20))	UINT8	RO	0x7000:25, 8
1604:17	SubIndex 0x023	23. PDO Mapping entry (object 0x7000 (COM Outputs Ch.1), entry 0x26 (Data Out 21))	UINT8	RO	0x7000:26, 8

Index 0x1605 COM RxPDO-Map Outputs Ch.2

Index (hex)	Name	Meaning	Data type	Flags	Default
1605:0	COM RxPDO-Map Outputs Ch.1	PDO Mapping RxPDO 1	UINT8	RO	0x17 (23_{dec})
1605:01	SubIndex 0x001	1. PDO Mapping entry (object 0x7011 (Ctrl Ch.2), entry 0x01 (Ctrl))	UINT16	RO	0x7011:01, 16
1605:02	SubIndex 0x002	2. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x11 (Data Out 0))	UINT8	RO	0x7010:11, 8
1605:03	SubIndex 0x003	3. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x12 (Data Out 1))	UINT8	RO	0x7010:12, 8
1605:04	SubIndex 0x004	4. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x13 (Data Out 2))	UINT8	RO	0x7010:13, 8
1605:05	SubIndex 0x005	5. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x14 (Data Out 3))	UINT8	RO	0x7010:14, 8
1605:06	SubIndex 0x006	6. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x15 (Data Out 4))	UINT8	RO	0x7010:15, 8
1605:07	SubIndex 0x007	7. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x16 (Data Out 5))	UINT8	RO	0x7010:16, 8
1605:08	SubIndex 0x008	8. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x17 (Data Out 6))	UINT8	RO	0x7010:17, 8
1605:09	SubIndex 0x009	9. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x18 (Data Out 7))	UINT8	RO	0x7010:18, 8
1605:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x19 (Data Out 8))	UINT8	RO	0x7010:19, 8
1605:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x1A (Data Out 9))	UINT8	RO	0x7010:1A, 8
1605:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x1B (Data Out 10))	UINT8	RO	0x7010:1B, 8
1605:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x1C (Data Out 11))	UINT8	RO	0x7010:1C, 8
1605:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x1D (Data Out 12))	UINT8	RO	0x7010:1D, 8
1605:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x1E (Data Out 13))	UINT8	RO	0x7010:1E, 8
1605:10	SubIndex 0x016	16. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x1F (Data Out 14))	UINT8	RO	0x7010:1F, 8
1605:11	SubIndex 0x017	17. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x20 (Data Out 15))	UINT8	RO	0x7010:20, 8
1605:12	SubIndex 0x018	18. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x21 (Data Out 16))	UINT8	RO	0x7010:21, 8
1605:13	SubIndex 0x019	19. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x22 (Data Out 17))	UINT8	RO	0x7010:22, 8
1605:14	SubIndex 0x020	20. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x23 (Data Out 18))	UINT8	RO	0x7010:23, 8
1605:15	SubIndex 0x021	21. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x24 (Data Out 19))	UINT8	RO	0x7010:24, 8
1605:16	SubIndex 0x022	22. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x25 (Data Out 20))	UINT8	RO	0x7010:25, 8
1605:17	SubIndex 0x023	23. PDO Mapping entry (object 0x7010 (COM Outputs Ch.2), entry 0x26 (Data Out 21))	UINT8	RO	0x7010:26, 8

Index 0x1A00 COM TxPDO-Map Inputs Ch.1

Index (hex)	Name	Meaning	Data type	Flags	Default
1A00:0	COM TxPDO-Map Inputs Ch.1	PDO Mapping TxPDO 1	UINT8	RO	0x1F (31_{dec})
1A00:01	SubIndex 0x001	1. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x01 (Transmit accepted))	BOOLEAN	RO	0x6000:01, 1
1A00:02	SubIndex 0x002	2. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x02 (Receive request))	BOOLEAN	RO	0x6000:02, 1
1A00:03	SubIndex 0x003	3. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x03 (Init accepted))	BOOLEAN	RO	0x6000:03, 1
1A00:04	SubIndex 0x004	4. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x04 (Buffer full))	BOOLEAN	RO	0x6000:04, 1
1A00:05	SubIndex 0x005	5. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x05 (Parity error))	BOOLEAN	RO	0x6000:05, 1
1A00:06	SubIndex 0x006	6. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x06 (Framing error))	BOOLEAN	RO	0x6000:06, 1
1A00:07	SubIndex 0x007	7. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x07 (Overrun error))	BOOLEAN	RO	0x6000:07, 1
1A00:08	SubIndex 0x008	8. PDO Mapping entry (1 bits align)	Align1	RO	0x0000:00, 1
1A00:09	SubIndex 0x009	9. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x09 (Input length))	UINT8	RO	0x6000:09, 8
1A00:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x11 (Data In 0))	UINT8	RO	0x6000:11, 8
1A00:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x12 (Data In 1))	UINT8	RO	0x6000:12, 8
1A00:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x13 (Data In 2))	UINT8	RO	0x6000:13, 8
1A00:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x14 (Data In 3))	UINT8	RO	0x6000:14, 8
1A00:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x15 (Data In 4))	UINT8	RO	0x6000:15, 8
1A00:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x16 (Data In 5))	UINT8	RO	0x6000:16, 8
1A00:10	SubIndex 0x016	16. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x17 (Data In 6))	UINT8	RO	0x6000:17, 8
1A00:11	SubIndex 0x017	17. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x18 (Data In 7))	UINT8	RO	0x6000:18, 8
1A00:12	SubIndex 0x018	18. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x19 (Data In 8))	UINT8	RO	0x6000:19, 8
1A00:13	SubIndex 0x019	19. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x1A (Data In 9))	UINT8	RO	0x6000:1A, 8
1A00:14	SubIndex 0x020	20. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x1B (Data In 10))	UINT8	RO	0x6000:1B, 8
1A00:15	SubIndex 0x021	21. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x1C (Data In 11))	UINT8	RO	0x6000:1C, 8
1A00:16	SubIndex 0x022	22. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x1D (Data In 12))	UINT8	RO	0x6000:1D, 8
1A00:17	SubIndex 0x023	23. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x1E (Data In 13))	UINT8	RO	0x6000:1E, 8
1A00:18	SubIndex 0x024	24. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x1F (Data In 14))	UINT8	RO	0x6000:1F, 8
1A00:19	SubIndex 0x025	25. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x20 (Data In 15))	UINT8	RO	0x6000:20, 8
1A00:1A	SubIndex 0x026	26. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x21 (Data In 16))	UINT8	RO	0x6000:21, 8
1A00:1B	SubIndex 0x027	27. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x22 (Data In 17))	UINT8	RO	0x6000:22, 8
1A00:1C	SubIndex 0x028	28. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x23 (Data In 18))	UINT8	RO	0x6000:23, 8
1A00:1D	SubIndex 0x029	29. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x24 (Data In 19))	UINT8	RO	0x6000:24, 8
1A00:1E	SubIndex 0x030	30. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x25 (Data In 20))	UINT8	RO	0x6000:25, 8
1A00:1F	SubIndex 0x031	31. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x26 (Data In 21))	UINT8	RO	0x6000:26, 8

Index 0x1A01 COM TxPDO-Map Inputs Ch.2

Index (hex)	Name	Meaning	Data type	Flags	Default
1A01:0	COM TxPDO-Map Inputs Ch.2	PDO Mapping TxPDO 2	UINT8	RO	0x1F (31_{dec})
1A01:01	SubIndex 0x001	1. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x01 (Transmit accepted))	BOOLEAN	RO	0x6010:01, 1
1A01:02	SubIndex 0x002	2. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x02 (Receive request))	BOOLEAN	RO	0x6010:02, 1
1A01:03	SubIndex 0x003	3. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x03 (Init accepted))	BOOLEAN	RO	0x6010:03, 1
1A01:04	SubIndex 0x004	4. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x04 (Buffer full))	BOOLEAN	RO	0x6010:04, 1
1A01:05	SubIndex 0x005	5. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x05 (Parity error))	BOOLEAN	RO	0x6010:05, 1
1A01:06	SubIndex 0x006	6. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x06 (Framing error))	BOOLEAN	RO	0x6010:06, 1
1A01:07	SubIndex 0x007	7. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x07 (Overrun error))	BOOLEAN	RO	0x6010:07, 1
1A01:08	SubIndex 0x008	8. PDO Mapping entry (1 bits align)	Align1	RO	0x0000:00, 1
1A01:09	SubIndex 0x009	9. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x09 (Input length))	UINT8	RO	0x6010:09, 8
1A01:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x11 (Data In 0))	UINT8	RO	0x6010:11, 8
1A01:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x12 (Data In 1))	UINT8	RO	0x6010:12, 8
1A01:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x13 (Data In 2))	UINT8	RO	0x6010:13, 8
1A01:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x14 (Data In 3))	UINT8	RO	0x6010:14, 8
1A01:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x15 (Data In 4))	UINT8	RO	0x6010:15, 8
1A01:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x16 (Data In 5))	UINT8	RO	0x6010:16, 8
1A01:10	SubIndex 0x016	16. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x17 (Data In 6))	UINT8	RO	0x6010:17, 8
1A01:11	SubIndex 0x017	17. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x18 (Data In 7))	UINT8	RO	0x6010:18, 8
1A01:12	SubIndex 0x018	18. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x19 (Data In 8))	UINT8	RO	0x6010:19, 8
1A01:13	SubIndex 0x019	19. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x1A (Data In 9))	UINT8	RO	0x6010:1A, 8
1A01:14	SubIndex 0x020	20. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x1B (Data In 10))	UINT8	RO	0x6010:1B, 8
1A01:15	SubIndex 0x021	21. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x1C (Data In 11))	UINT8	RO	0x6010:1C, 8
1A01:16	SubIndex 0x022	22. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x1D (Data In 12))	UINT8	RO	0x6010:1D, 8
1A01:17	SubIndex 0x023	23. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x1E (Data In 13))	UINT8	RO	0x6010:1E, 8
1A01:18	SubIndex 0x024	24. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x1F (Data In 14))	UINT8	RO	0x6010:1F, 8
1A01:19	SubIndex 0x025	25. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x20 (Data In 15))	UINT8	RO	0x6010:20, 8
1A01:1A	SubIndex 0x026	26. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x21 (Data In 16))	UINT8	RO	0x6010:21, 8
1A01:1B	SubIndex 0x027	27. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x22 (Data In 17))	UINT8	RO	0x6010:22, 8
1A01:1C	SubIndex 0x028	28. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x23 (Data In 18))	UINT8	RO	0x6010:23, 8
1A01:1D	SubIndex 0x029	29. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x24 (Data In 19))	UINT8	RO	0x6010:24, 8
1A01:1E	SubIndex 0x030	30. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x25 (Data In 20))	UINT8	RO	0x6010:25, 8
1A01:1F	SubIndex 0x031	31. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x26 (Data In 21))	UINT8	RO	0x6010:26, 8

Index 0x1A04 COM TxPDO-Map Inputs Ch.1

Index (hex)	Name	Meaning	Data type	Flags	Default
1A04:0	COM TxPDO-Map Inputs Ch.1	PDO Mapping TxPDO 1	UINT8	RO	0x17 (23_{dec})
1A04:01	SubIndex 0x001	1. PDO Mapping entry (object 0x6001 (Status Ch.1), entry 0x01 (Status))	UINT16	RO	0x6001:01, 16
1A04:02	SubIndex 0x002	2. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x11 (Data In 0))	UINT8	RO	0x6000:11, 8
1A04:03	SubIndex 0x003	3. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x12 (Data In 1))	UINT8	RO	0x6000:12, 8
1A04:04	SubIndex 0x004	4. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x13 (Data In 2))	UINT8	RO	0x6000:13, 8
1A04:05	SubIndex 0x005	5. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x14 (Data In 3))	UINT8	RO	0x6000:14, 8
1A04:06	SubIndex 0x006	6. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x15 (Data In 4))	UINT8	RO	0x6000:15, 8
1A04:07	SubIndex 0x007	7. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x16 (Data In 5))	UINT8	RO	0x6000:16, 8
1A04:08	SubIndex 0x008	8. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x17 (Data In 6))	UINT8	RO	0x6000:17, 8
1A04:09	SubIndex 0x009	9. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x18 (Data In 7))	UINT8	RO	0x6000:18, 8
1A04:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x19 (Data In 8))	UINT8	RO	0x6000:19, 8
1A04:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x1A (Data In 9))	UINT8	RO	0x6000:1A, 8
1A04:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x1B (Data In 10))	UINT8	RO	0x6000:1B, 8
1A04:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x1C (Data In 11))	UINT8	RO	0x6000:1C, 8
1A04:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x1D (Data In 12))	UINT8	RO	0x6000:1D, 8
1A04:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x1E (Data In 13))	UINT8	RO	0x6000:1E, 8
1A04:10	SubIndex 0x016	16. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x1F (Data In 14))	UINT8	RO	0x6000:1F, 8
1A04:11	SubIndex 0x017	17. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x20 (Data In 15))	UINT8	RO	0x6000:20, 8
1A04:12	SubIndex 0x018	17. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x21 (Data In 16))	UINT8	RO	0x6000:21, 8
1A04:13	SubIndex 0x019	19. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x22 (Data In 17))	UINT8	RO	0x6000:22, 8
1A04:14	SubIndex 0x020	20. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x23 (Data In 18))	UINT8	RO	0x6000:23, 8
1A04:15	SubIndex 0x021	21. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x24 (Data In 19))	UINT8	RO	0x6000:24, 8
1A04:16	SubIndex 0x022	22. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x25 (Data In 20))	UINT8	RO	0x6000:25, 8
1A04:17	SubIndex 0x023	23. PDO Mapping entry (object 0x6000 (COM Inputs Ch.1), entry 0x26 (Data In 21))	UINT8	RO	0x6000:26, 8

Index 0x1A05 COM TxPDO-Map Inputs Ch.2

Index (hex)	Name	Meaning	Data type	Flags	Default
1A05:0	COM TxPDO-Map Inputs Ch.2	PDO Mapping TxPDO 1	UINT8	RO	0x17 (23_{dec})
1A05:01	SubIndex 0x001	1. PDO Mapping entry (object 0x6011 (Status Ch.2), entry 0x01 (Status))	UINT16	RO	0x6011:01, 16
1A05:02	SubIndex 0x002	2. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x11 (Data In 0))	UINT8	RO	0x6010:11, 8
1A05:03	SubIndex 0x003	3. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x12 (Data In 1))	UINT8	RO	0x6010:12, 8
1A05:04	SubIndex 0x004	4. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x13 (Data In 2))	UINT8	RO	0x6010:13, 8
1A05:05	SubIndex 0x005	5. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x14 (Data In 3))	UINT8	RO	0x6010:14, 8
1A05:06	SubIndex 0x006	6. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x15 (Data In 4))	UINT8	RO	0x6010:15, 8
1A05:07	SubIndex 0x007	7. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x16 (Data In 5))	UINT8	RO	0x6010:16, 8
1A05:08	SubIndex 0x008	8. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x17 (Data In 6))	UINT8	RO	0x6010:17, 8
1A05:09	SubIndex 0x009	9. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x18 (Data In 7))	UINT8	RO	0x6010:18, 8
1A05:0A	SubIndex 0x010	10. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x19 (Data In 8))	UINT8	RO	0x6010:19, 8
1A05:0B	SubIndex 0x011	11. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x1A (Data In 9))	UINT8	RO	0x6010:1A, 8
1A05:0C	SubIndex 0x012	12. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x1B (Data In 10))	UINT8	RO	0x6010:1B, 8
1A05:0D	SubIndex 0x013	13. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x1C (Data In 11))	UINT8	RO	0x6010:1C, 8
1A05:0E	SubIndex 0x014	14. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x1D (Data In 12))	UINT8	RO	0x6010:1D, 8
1A05:0F	SubIndex 0x015	15. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x1E (Data In 13))	UINT8	RO	0x6010:1E, 8
1A05:10	SubIndex 0x016	16. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x1F (Data In 14))	UINT8	RO	0x6010:1F, 8
1A05:11	SubIndex 0x017	17. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x20 (Data In 15))	UINT8	RO	0x6010:20, 8
1A05:12	SubIndex 0x018	17. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x21 (Data In 16))	UINT8	RO	0x6010:21, 8
1A05:13	SubIndex 0x019	19. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x22 (Data In 17))	UINT8	RO	0x6010:22, 8
1A05:14	SubIndex 0x020	20. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x23 (Data In 18))	UINT8	RO	0x6010:23, 8
1A05:15	SubIndex 0x021	21. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x24 (Data In 19))	UINT8	RO	0x6010:24, 8
1A05:16	SubIndex 0x022	22. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x25 (Data In 20))	UINT8	RO	0x6010:25, 8
1A05:17	SubIndex 0x023	23. PDO Mapping entry (object 0x6010 (COM Inputs Ch.2), entry 0x26 (Data In 21))	UINT8	RO	0x6010:26, 8

Index 0x1C00 Sync manager type

Index (hex)	Name	Meaning	Data type	Flags	Default
1C00:0	Sync manager type	Using the sync managers	UINT8	RO	0x04 (4_{dec})
1C00:01	SubIndex 0x001	Sync-Manager Type Channel 1: Mailbox Write	UINT8	RO	0x01 (1_{dec})
1C00:02	SubIndex 0x002	Sync-Manager Type Channel 2: Mailbox Read	UINT8	RO	0x02 (2_{dec})
1C00:03	SubIndex 0x003	Sync-Manager Type Channel 3: Process Data Write (Outputs)	UINT8	RO	0x03 (3_{dec})
1C00:04	SubIndex 0x004	Sync-Manager Type Channel 4: Process Data Read (Inputs)	UINT8	RO	0x04 (4_{dec})

Index 0x1C12 RxPDO assign

Index (hex)	Name	Meaning	Data type	Flags	Default
1C12:0	RxPDO assign	PDO Assign Outputs	UINT8	RO	0x04 (4_{dec})
1C12:01	SubIndex 0x001	1st allocated RxPDO (contains the index 0x of the associated RxPDO mapping object)	UINT16	RO	0x1600 (5632_{dec})
1C12:02	SubIndex 0x002	2nd allocated RxPDO (contains the index 0x of the associated RxPDO mapping object)	UINT16	RO	0x1601 (5633_{dec})

Index 0x1C13 TxPDO assign

Index (hex)	Name	Meaning	Data type	Flags	Default
1C13:0	TxPDO assign	PDO Assign Inputs	UINT8	RO	0x04 (4_{dec})
1C13:01	SubIndex 0x001	1st allocated TxPDO (contains the index 0x of the associated TxPDO mapping object)	UINT16	RO	0x1A00 (6656_{dec})
1C13:02	SubIndex 0x002	2nd allocated TxPDO (contains the index 0x of the associated TxPDO mapping object)	UINT16	RO	0x1A01 (6657_{dec})

Index 0x1C32 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_{dec})
1C32:01	Sync mode	Current synchronization mode: • 0: Free Run • 1: Synchron with SM 2 Event • 2: DC-Mode - Synchron with SYNC0 Event • 3: DC-Mode - Synchron with SYNC1 Event	UINT16	RW	0x0000 (0_{dec})
1C32:02	Cycle time	Cycle time (in ns): • Free Run: Cycle time of the local timer • Synchronous with SM 2 event: Master cycle time • DC-Mode: SYNC0/SYNC1 Cycle Time	UINT32	RW	0x0003D090 (250000_{dec})
1C32:03	Shift time	Time between SYNC0 event and output of the outputs (in ns, DC mode only)	UINT32	RO	0x00000000 (0_{dec})
1C32:04	Sync modes supported	Supported synchronization modes: • Bit 0 = 1: free run is supported • Bit 1 = 1: Synchronous with SM 2 event is supported • Bit 2-3 = 01: DC mode is supported • Bit 4-5 = 10: Output shift with SYNC1 event (only DC mode) • Bit 14 = 1: dynamic times (measurement through writing of 1C32:08)	UINT16	RO	0xC007 (49159_{dec})
1C32:05	Minimum cycle time	Minimum cycle time (in ns)	UINT32	RO	0x00004E20 (20000_{dec})
1C32:06	Calc and copy time	Minimum time between SYNC0 and SYNC1 event (in ns, DC mode only)	UINT32	RO	0x00000000 (0_{dec})
1C32:08	Command	• 0: Measurement of the local cycle time is stopped • 1: Measurement of the local cycle time is started The entries 1C32:03, 1C32:05, 1C32:06, 1C32:09, 1C33:03, 1C33:06, and 1C33:09 are updated with the maximum measured values. For a subsequent measurement the measured values are reset	UINT16	RW	0x0000 (0_{dec})
1C32:09	Delay time	Time between SYNC1 event and output of the outputs (in ns, DC mode only)	UINT32	RO	0x00000000 (0_{dec})
1C32:0B	SM event missed counter	Number of missed SM events in OPERATIONAL (DC mode only)	UINT16	RO	0x0000 (0_{dec})
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_{dec})
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_{dec})
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_{dec})

Index 0x1C33 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_{dec})
1C33:01	Sync mode	Current synchronization mode: • 0: Free Run • 1: Synchronous with SM 3 event (no outputs available) • 2: DC - Synchronous with SYNC0 Event • 3: DC - Synchronous with SYNC1 Event • 34: Synchronous with SM 2 event (outputs available)	UINT16	RW	0x0000 (0_{dec})
1C33:02	Cycle time	as 1C32:02	UINT32	RW	0x0003D090 (250000_{dec})
1C33:03	Shift time	Time between SYNC0 event and reading of the inputs (in ns, only DC mode)	UINT32	RO	0x00000000 (0_{dec})
1C33:04	Sync modes supported	Supported synchronization modes: • Bit 0: free run is supported • Bit 1: synchronous with SM 2 event is supported (outputs available) • Bit 1: synchronous with SM 3 event is supported (no outputs available) • Bit 2-3 = 01: DC mode is supported • Bit 4-5 = 01: input shift through local event (outputs available) • Bit 4-5 = 10: input shift with SYNC1 event (no outputs available) • Bit 14 = 1: dynamic times (measurement through writing of 1C32:08 or 1C33:08)	UINT16	RO	0xC007 (49159_{dec})
1C33:05	Minimum cycle time	as 1C32:05	UINT32	RO	0x00004E20 (20000_{dec})
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_{dec})
1C33:08	Command	as 1C32:08	UINT16	RW	0x0000 (0_{dec})
1C33:09	Delay time	Time between SYNC1 event and reading of the inputs (in ns, only DC mode)	UINT32	RO	0x00000000 (0_{dec})
1C33:0B	SM event missed counter	as 1C32:11	UINT16	RO	0x0000 (0_{dec})
1C33:0C	Cycle exceeded counter	as 1C32:12	UINT16	RO	0x0000 (0_{dec})
1C33:0D	Shift too short counter	as 1C32:13	UINT16	RO	0x0000 (0_{dec})
1C33:20	Sync error	as 1C32:32	BOOLEAN	RO	0x00 (0_{dec})

7.1.3 Profile-specific objects (0x6000-0xFFFF) [from hardware version 03]

The profile-specific objects have the same meaning for all EtherCAT slaves that support the profile 5001.

Index 0x60n0 COM Inputs Ch. 1 (n = 0), Ch. 2 (n = 1)

Index (hex)	Name	Meaning	Data type	Flags	Default
60n0:0	COM Inputs Ch. 1 + Ch. 2	Max. SubIndex (hex)	UINT8	RO	0x26 (38_{dec})
60n0:01	Transmit accepted	The terminal acknowledges receipt of data by changing the state of this bit. Only now new data can be transferred from the controller to the terminal.	BOOLEAN	RO	0x00 (0_{dec})
60n0:02	Receive request	By changing the state of this bit, the terminal informs the controller that the DataIn bytes contain the number of bytes displayed in "Input length [▶ 186]". The controller must acknowledge receipt of the data by changing the state of the ReceiveAccepted [▶ 187] bit. Only then new data can be transferred from the terminal to the controller.	BOOLEAN	RO	0x00 (0_{dec})
60n0:03	Init accepted	0 The terminal is ready again for serial data exchange. 1 Initialization was completed by the terminal.	BOOLEAN	RO	0x00 (0_{dec})
60n0:04	Buffer full	The reception FIFO is full. All incoming data will be lost from this point on!	BOOLEAN	RO	0x00 (0_{dec})
60n0:05	Parity error	A parity error has occurred.	BOOLEAN	RO	0x00 (0_{dec})
60n0:06	Framing error	A framing error has occurred.	BOOLEAN	RO	0x00 (0_{dec})
60n0:07	Overrun error	An overrun error has occurred.	BOOLEAN	RO	0x00 (0_{dec})
60n0:09	Input length	Number of input bytes available for transfer from the terminal to the controller.	UINT8	RO	0x00 (0_{dec})
60n0:11	Data In 0	Input byte 0	UINT8	RO	0x00 (0_{dec})
...	...	....	...	...	...
60n0:26	Data In 21	Input byte 21	UINT8	RO	0x00 (0_{dec})

Index 0x60n1 Status Ch. 1 (n = 0), Ch. 2 (n = 1)

Index (hex)	Name	Meaning	Data type	Flags	Default
60n1:0	Status Ch. 1 + Ch. 2	Max. SubIndex (hex)	UINT8	RO	0x01 (01_{dec})
60n1:01	Status	Status word	UINT16	RO	0x00 (0_{dec})

Index 0x70n0 COM Outputs Ch. 1 (n = 0), Ch. 2 (n = 1)

Index (hex)	Name	Meaning	Data type	Flags	Default
70n0:0	COM Outputs Ch. 1 + Ch. 2	Max. SubIndex (hex)	UINT8	RO	0x26 (38_{dec})
70n0:01	Transmit request	By changing the state of this bit, the controller informs the terminal that the DataOut bytes contain the number of bytes displayed in "Output length [▶ 187]". The terminal acknowledges receipt of the data by changing the state of the "TransmitAccepted [▶ 186]" bit. Only now new data can be transferred from the controller to the terminal.	BOOLEAN	RO	0x00 (0_{dec})
70n0:02	Receive accepted	The controller acknowledges receipt of data by changing the state of this bit. Only then new data can be transferred from the terminal to the controller.	BOOLEAN	RO	0x00 (0_{dec})
70n0:03	Init request	0 The controller once again requests the terminal to prepare for serial data exchange. 1 The controller requests terminal for initialization. The transmit and receive functions will be blocked, the FIFO pointer will be reset and the interface will be initialized with the values of the responsible Settings object. The execution of the initialization will be acknowledged by the terminal with the 'Init accepted [▶ 186]' bit.	BOOLEAN	RO	0x00 (0_{dec})
70n0:04	Send continuous	Continuous sending of data from the FIFO. The send buffer is filled (up to 128 bytes) by the controller. The filled buffer contents will be sent on the rising edge of the bit. If the data has been transmitted, the terminal informs the controller by setting the "Init accepted [▶ 186]" bit. "Init accepted [▶ 186]" is cleared with "SendContinuous [▶ 187]".	BOOLEAN	RO	0x00 (0_{dec})
70n0:09	Output length	Number of output bytes available for transfer from the controller to the terminal.	UINT8	RO	0x00 (0_{dec})
70n0:11	Data Out 0	Output byte 0	UINT8	RO	0x00 (0_{dec})
...	...	....	...	...	...
70n0:26	Data Out 21	Output byte 21	UINT8	RO	0x00 (0_{dec})

Index 0x70n1 Ctrl Ch. 1 (n = 0), Ch. 2 (n = 1)

Index (hex)	Name	Meaning	Data type	Flags	Default
70n1:0	Crtl Ch. 1 + Ch. 2	Max. SubIndex (hex)	UINT8	RO	0x01 (01_{dec})
70n1:01	Status	Control word	UINT16	RO	0x00 (0_{dec})

Index 0xA0n0 COM Diag data Ch. 1 (n = 0), Ch. 2 (n = 1)

Index (hex)	Name	Meaning	Data type	Flags	Default
A0n0:0	COM Diag data Ch. 1 + Ch. 2	Max. SubIndex (hex)	UINT8	RO	0x12 (18_{dec})
A0n0:01	Buffer overflow	A buffer overflow has occurred.	BOOLEAN	RO	0x00 (0_{dec})
A0n0:02	Parity error	A parity error has occurred.	BOOLEAN	RO	0x00 (0_{dec})
A0n0:03	Framing error	A framing error has occurred	BOOLEAN	RO	0x00 (0_{dec})
A0n0:04	Overrun error	An overrun error has occurred.	BOOLEAN	RO	0x00 (0_{dec})
A0n0:05	Buffer full	The reception FIFO is full. All incoming data will be lost from this point on!	BOOLEAN	RO	0x00 (0_{dec})
A0n0:11	Data bytes in send buffer	Number of data bytes in the send FIFO	UINT16	RO	0x0000 (0_{dec})
A0n0:21	Data bytes in receive buffer	Number of data bytes in the receive FIFO	UINT16	RO	0x0000 (0_{dec})

Index 0xF000 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_{dec})
F000:01	Module Index 0xdistance	Index (hex) interval of the objects of the individual channels	UINT16	RO	0x0010 (16_{dec})
F000:02	Maximum number of modules	Number of channels	UINT16	RO	0x0004 (4_{dec})

Index 0xF008 Code word

Index (hex)	Name	Meaning	Data type	Flags	Default
F008:0	Code word	reserved	UINT32	RW	0x00000000 (0_{dec})

Index 0xF010 Module list

Index (hex)	Name	Meaning	Data type	Flags	Default
F010:0	Module list	Max. SubIndex (hex)	UINT8	RW	0x04 (4_{dec})
F010:01	SubIndex 0x001	-	UINT32	RW	0x00000258 (600_{dec})
F010:02	SubIndex 0x002	-	UINT32	RW	0x00000258 (600_{dec})

7.2 Control and status data

The control and status data are located in the first 16 bits of the input and output process image. Communication between the terminal and the controller is controlled via this data.

Status data

Bit position	Name	Meaning	Data type
0	Transmit accepted	The terminal acknowledges receipt of data by changing the state of this bit. Only now new data can be transferred from the controller to the terminal.	BOOLEAN
1	Receive request	By changing the state of this bit, the terminal informs the controller that the DataIn bytes contain the number of bytes displayed in “Input length”. The controller must acknowledge receipt of the data by changing the state of the ReceiveAccepted bit. Only then new data can be transferred from the terminal to the controller.	BOOLEAN
2	Init accepted	0 The terminal is ready again for serial data exchange. 1 Initialization was completed by the terminal.	BOOLEAN
3	Buffer full	The reception FIFO is full. All incoming data will be lost from this point on!	BOOLEAN
4	Parity error	A parity error has occurred.	BOOLEAN
5	Framing error	A framing error has occurred.	BOOLEAN
6	Overrun error	An overrun error has occurred.	BOOLEAN
7	-		
8...15	Input length	Number of input bytes available for transfer from the terminal to the controller.	UINT8

Control data

Bit position	Name	Meaning		Data type
0	Transmit request	By changing the state of this bit, the controller informs the terminal that the DataOut bytes contain the number of bytes displayed in “Output length”. The terminal acknowledges receipt of the data by changing the state of the “TransmitAccepted” bit. Only now new data can be transferred from the controller to the terminal.		BOOLEAN
1	Receive accepted	The controller acknowledges receipt of data by changing the state of this bit. Only then new data can be transferred from the terminal to the controller.		BOOLEAN
2	Init request	0	The controller once again requests the terminal to prepare for serial data exchange.	BOOLEAN
		1	The controller requests terminal for initialization. The transmit and receive functions will be blocked, the FIFO pointer will be reset and the interface will be initialized with the values of the responsible Settings object. The execution of the initialization will be acknowledged by the terminal with the ‘Init accepted’ bit.	
3	Send continuous	Continuous sending of data from the FIFO. The send buffer is filled (up to 128 bytes) by the controller. The filled buffer contents will be sent on the rising edge of the bit. If the data has been transmitted, the terminal informs the controller by setting the “Init accepted” bit. “Init accepted” is cleared with “SendContinuous”.		BOOLEAN
4...7	-			
8...15	Output length	Number of output bytes available for transfer from the controller to the terminal.		UINT8

PDO Assignment

The terminal makes two input/output process images available for each channel. These differ only in the representation of the control/status data.

In the case of bitwise representation, the data is made available as shown in the tables above. In the case of word-wise representation, the first 16 bits of the process data are combined in one word.

The process data objects begin as standard with a control/status word. This PDO assignment is required in order to use the ‘TwinCAT PLC Serial Communication’ library.

Status Inputs	Control Outputs	Representation
0x1A00	0x1600	bitwise Ch. 1
0x1A01	0x1601	bitwise Ch. 2
0x1A02	0x1602	bitwise Ch. 3
0x1A03	0x1603	bitwise Ch. 4
0x1A04	0x1604	word-wise Ch 1
0x1A05	0x1605	word-wise Ch 2
0x1A06	0x1606	word-wise Ch 3
0x1A07	0x1607	word-wise Ch 4

8 Appendix

8.1 EtherCAT AL Status Codes

For detailed information please refer to the EtherCAT system description.

8.2 Firmware compatibility

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.

NOTE

Risk of damage to the device

Pay attention to the instructions for firmware updates on the separate page [▶ 191].

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!

EL6001			
Hardware (HW)	**Firmware (FW)**	**Revision no.**	**Date of release**
00 - 02	01		05/2006
	02		12/2006
	03		04/2008
	04		04/2008
04 - 13*	05	EL6001-0000-0016	12/2009
	06		04/2010
	07	EL6001-0000-0017	03/2011
		EL6001-0000-0018	10/2012
	08	EL6001-0000-0019	05/2014
	09	EL6001-0000-0020	08/2014
	10		05/2015
	11*		06/2017

EL6002			
Hardware (HW)	**Firmware (FW)**	**Revision no.**	**Date of release**
00 – 16*	01	EL6002-0000-0016	12/2009
	02		06/2010
	03	EL6002-0000-0017	11/2012
		EL6002-0000-0018	08/2013
	04*	EL6002-0000-0019	05/2014

EL6021			
Hardware (HW)	**Firmware (FW)**	**Revision no.**	**Date of release**
00 - 02	01		05/2005
	02		12/2006
	03		04/2008
03 - 13*	04	EL6021-0000-0016	11/2009
	05		04/2010
		EL6021-0000-0017	10/2012
	06	EL6021-0000-0018	08/2013
	07	EL6021-0000-0019	06/2014
	08	EL6021-0000-0020	10/2014
	09*		01/2020

EL6022			
Hardware (HW)	**Firmware (FW)**	**Revision no.**	**Date of release**
00 - 17*	01	EL6022-0000-0016	01/2010
	02		06/2010
	03*		09/2011
		EL6022-0000-0017	08/2012
		EL6022-0000-0018	08/2013
		EL6022-0000-0019	03/2015

*) 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.

8.3 Firmware Update EL/ES/EM/ELM/EPxxxx

This section describes the device update for Beckhoff EtherCAT slaves from the EL/ES, ELM, EM, EK and EP series. A firmware update should only be carried out after consultation with Beckhoff support.

Storage locations

An EtherCAT slave stores operating data in up to three locations:

- 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.
- In addition, each EtherCAT slave has a memory chip, a so-called **ESI-EEPROM**, for storing its own device description (ESI: EtherCAT Slave Information). On power-up this description is loaded and the EtherCAT communication is set up accordingly. The device description is available from the download area of the Beckhoff website at (https://www.beckhoff.de). All ESI files are accessible there as zip files.

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. ELxxxx-xxxx_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

NOTE

Risk of damage to the device!

✓ Note the following when downloading new device files

a) Firmware downloads to an EtherCAT device must not be interrupted

b) Flawless EtherCAT communication must be ensured. CRC errors or LostFrames must be avoided.

c) 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.

8.3.1 Device description ESI file/XML

NOTE

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:

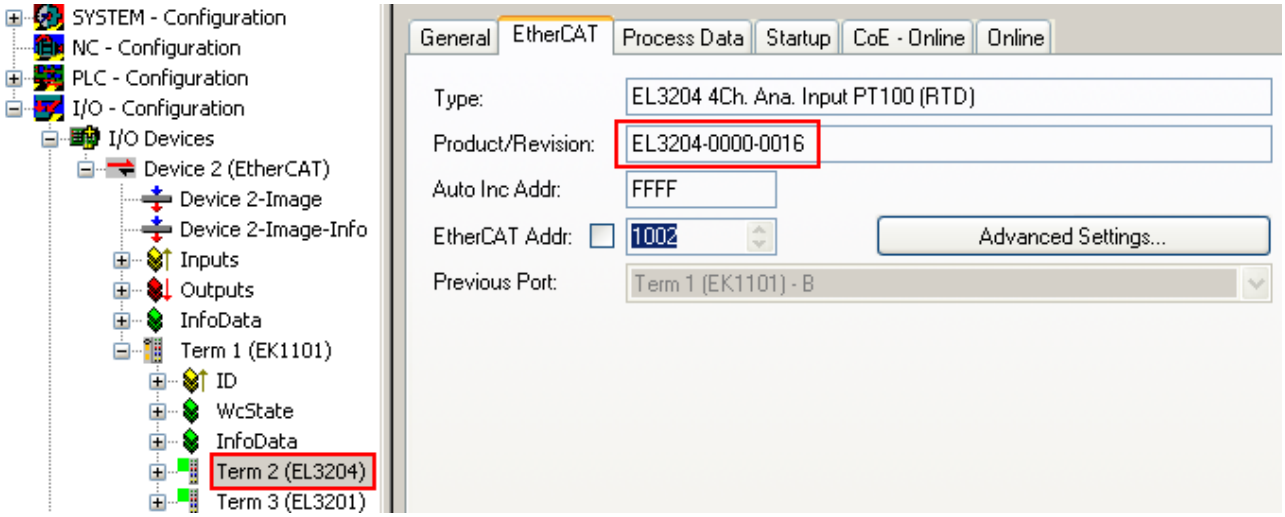


Fig. 169: Device identifier consisting of name EL3204-0000 and revision -0016

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.

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:

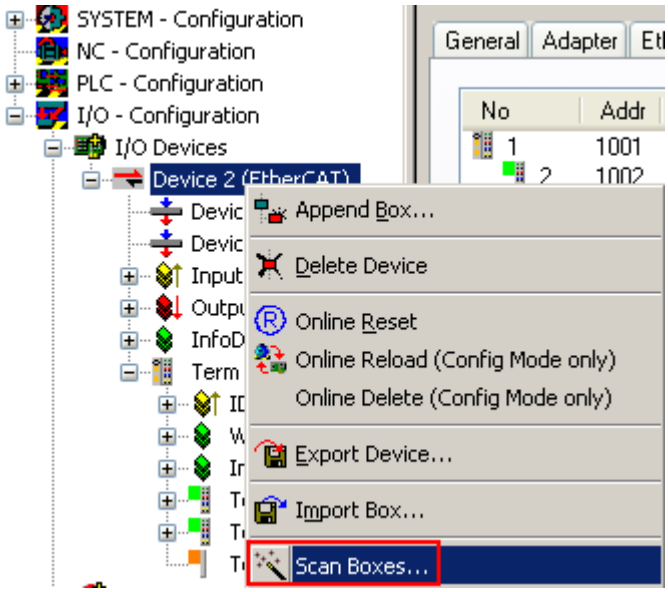


Fig. 170: Scan the subordinate field by right-clicking on the EtherCAT device

If the found field matches the configured field, the display shows

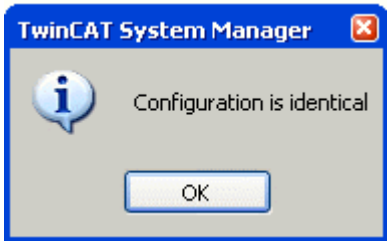


Fig. 171: Configuration is identical

otherwise a change dialog appears for entering the actual data in the configuration.

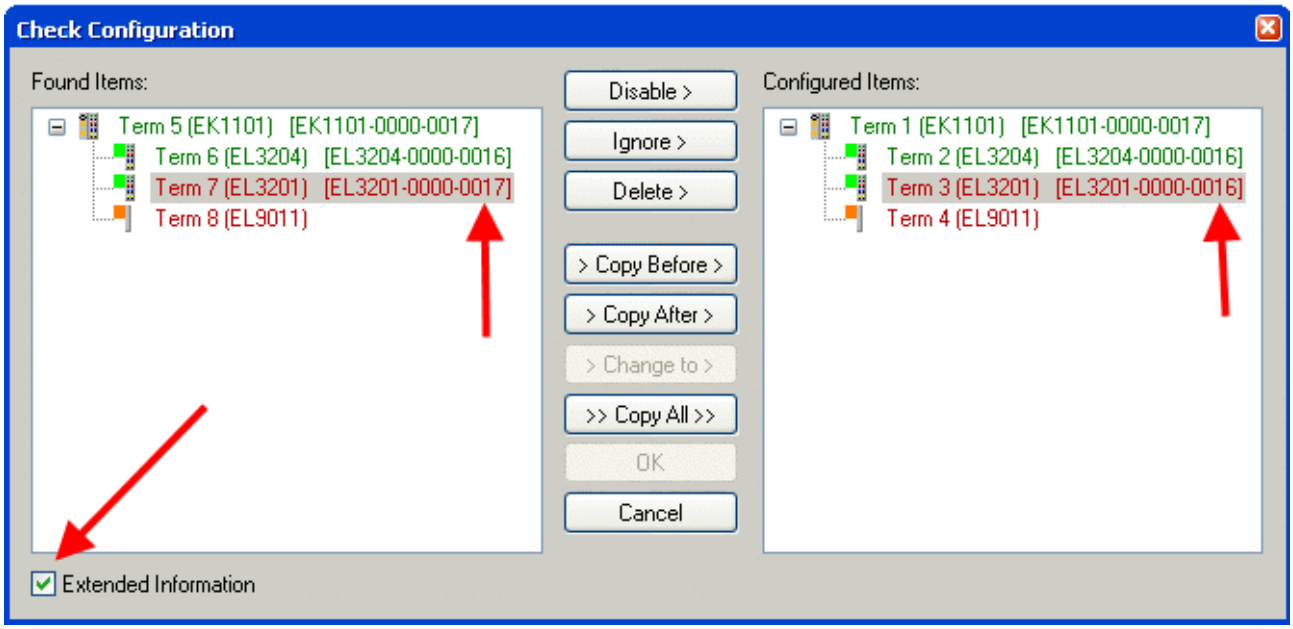


Fig. 172: Change dialog

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*

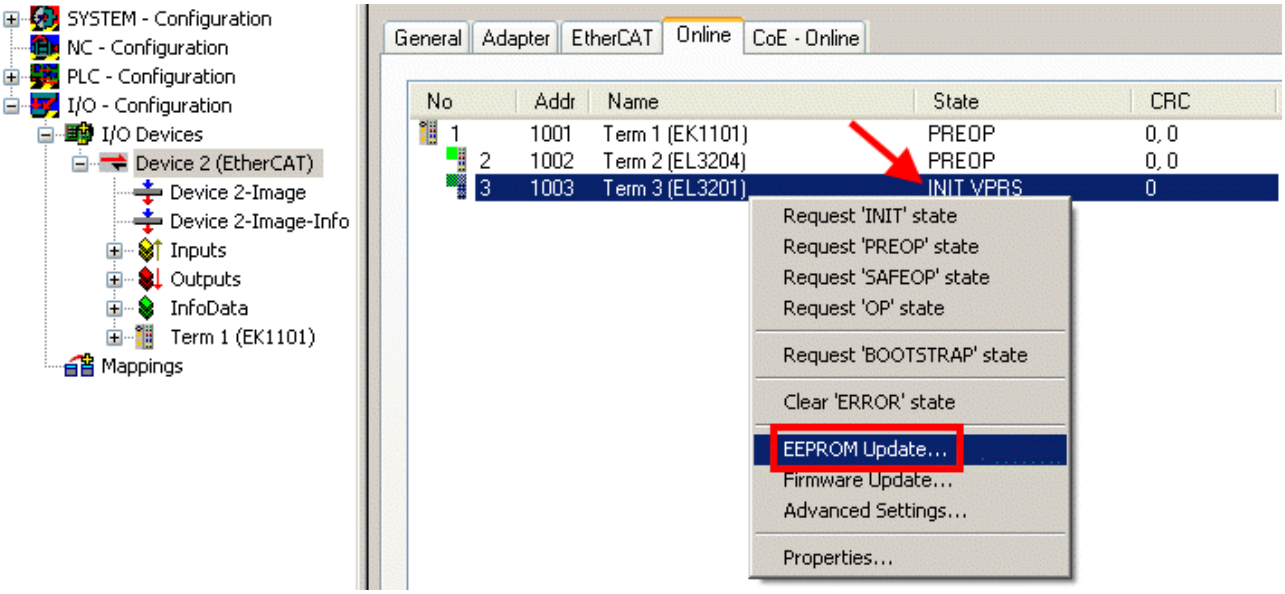


Fig. 173: EEPROM Update

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.

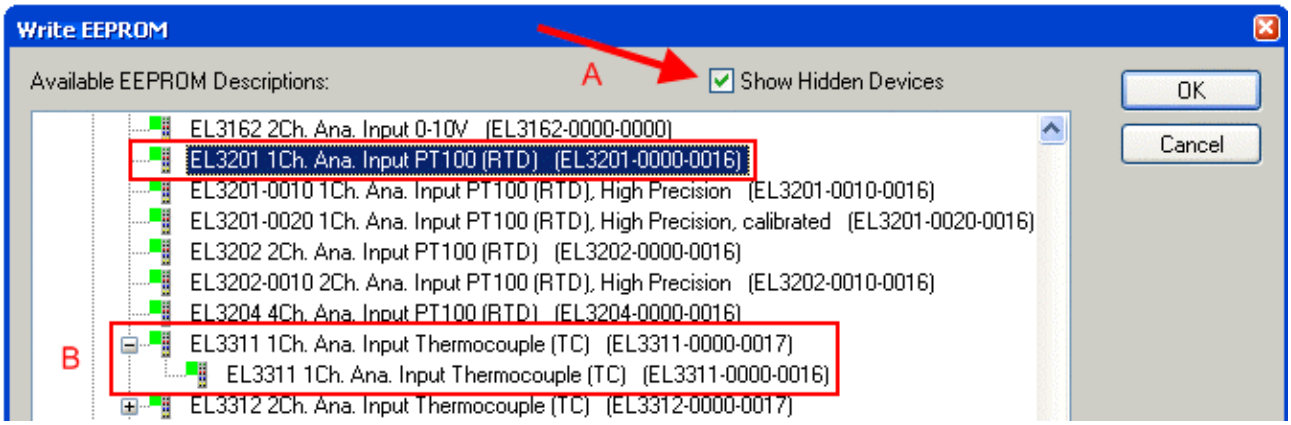


Fig. 174: Selecting the new ESI

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.

8.3.2 Firmware explanation

Determining the firmware version

Determining the version on laser inscription

Beckhoff EtherCAT slaves feature serial numbers applied by laser. The serial number has the following structure: **KK YY FF HH**

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

Example with ser. no.: 12 10 03 02:

12 - week of production 12
10 - year of production 2010
03 - firmware version 03
02 - hardware version 02

Determining the version via the 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.

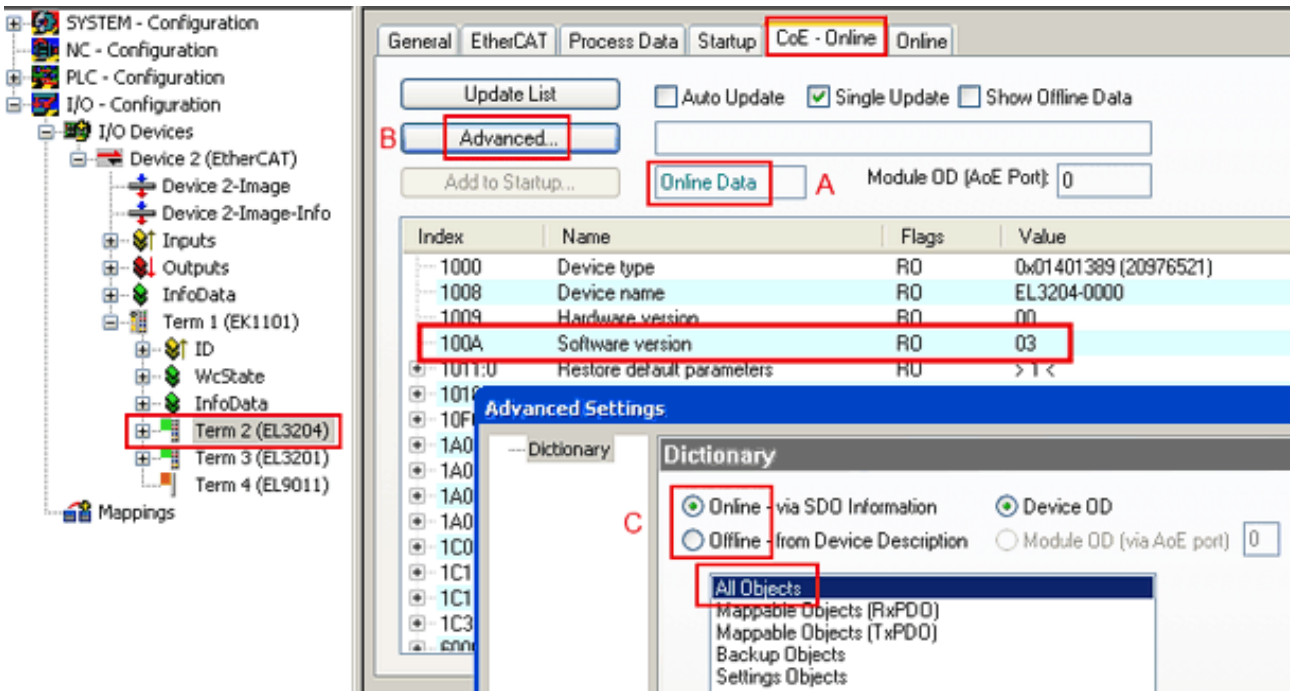


Fig. 175: Display of EL3204 firmware version

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 *AllObjects*.

8.3.3 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.*

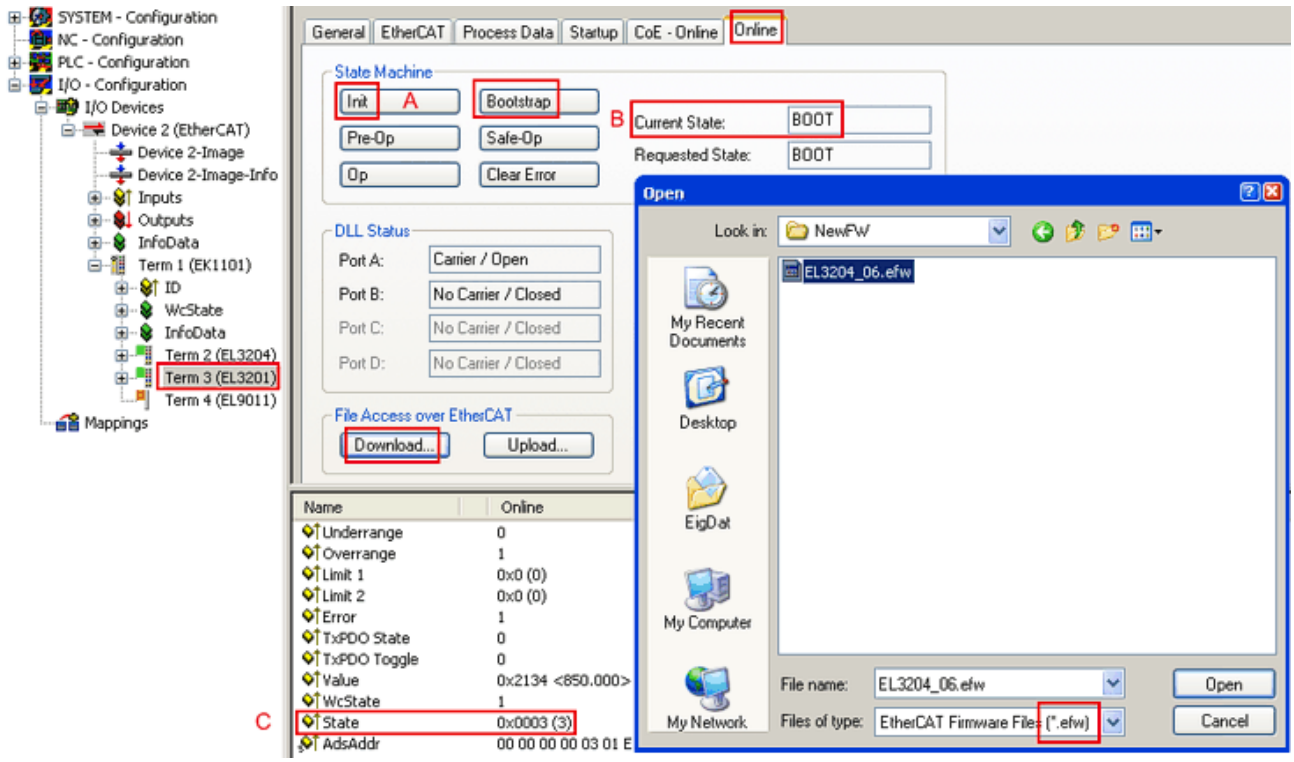
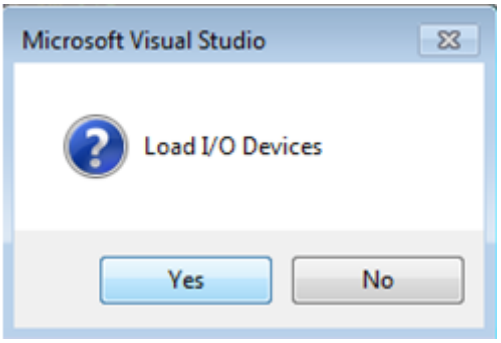
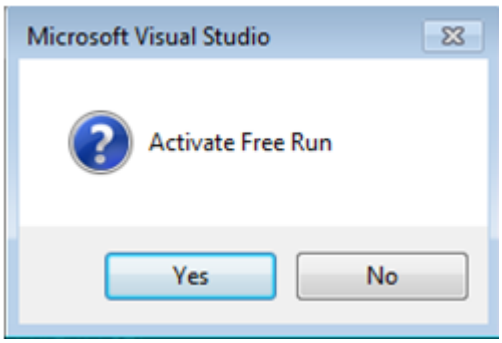


Fig. 176: Firmware Update

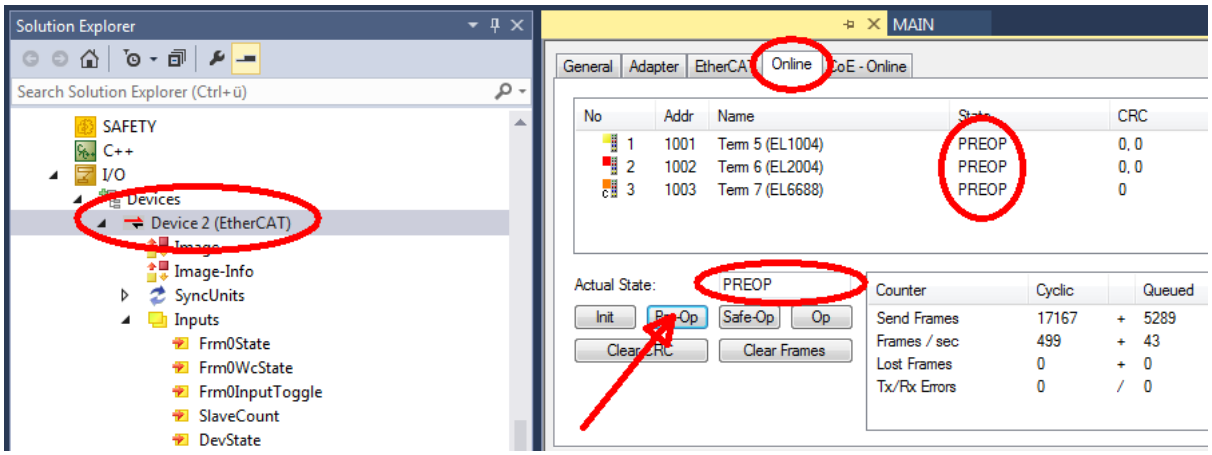
Proceed as follows, unless instructed otherwise by Beckhoff support. Valid for TwinCAT 2 and 3 as EtherCAT master.

- Switch TwinCAT system to ConfigMode/FreeRun with cycle time >= 1 ms (default in ConfigMode is 4 ms). A FW-Update during real time operation is not recommended.



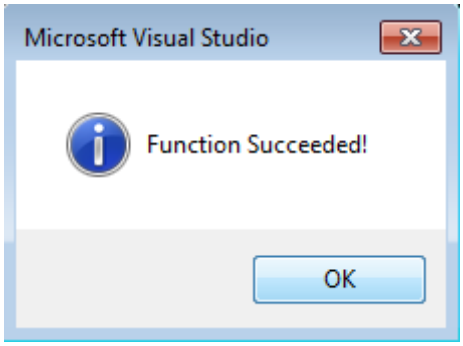


- Switch EtherCAT Master to PreOP



- 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 pass word will not be neccessary 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.

8.3.4 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 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.

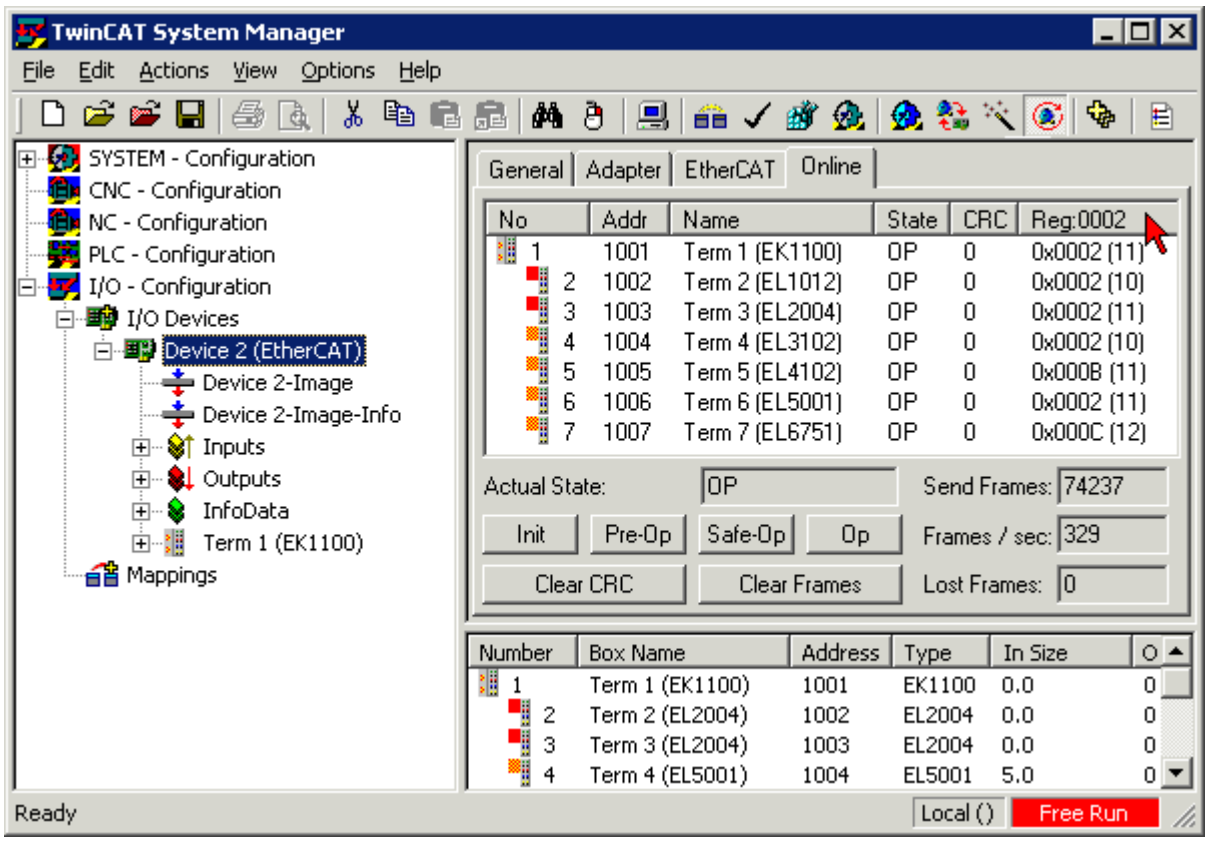


Fig. 177: FPGA firmware version definition

If the column *Reg:0002* is not displayed, right-click the table header and select *Properties* in the context menu.

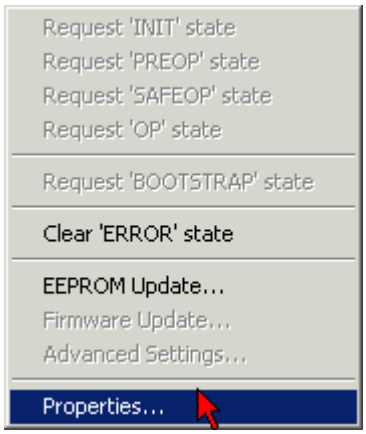


Fig. 178: Context menu *Properties*

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.

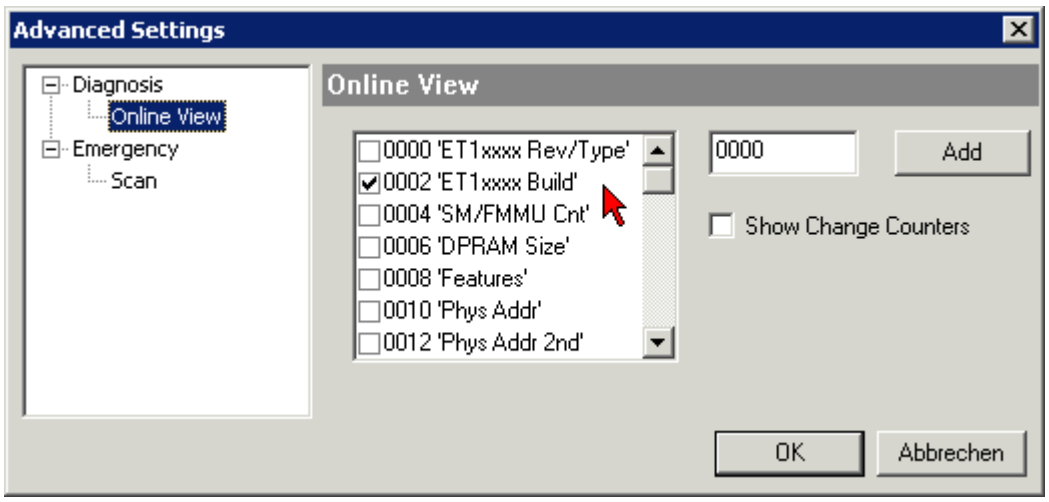


Fig. 179: Dialog *Advanced Settings*

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.

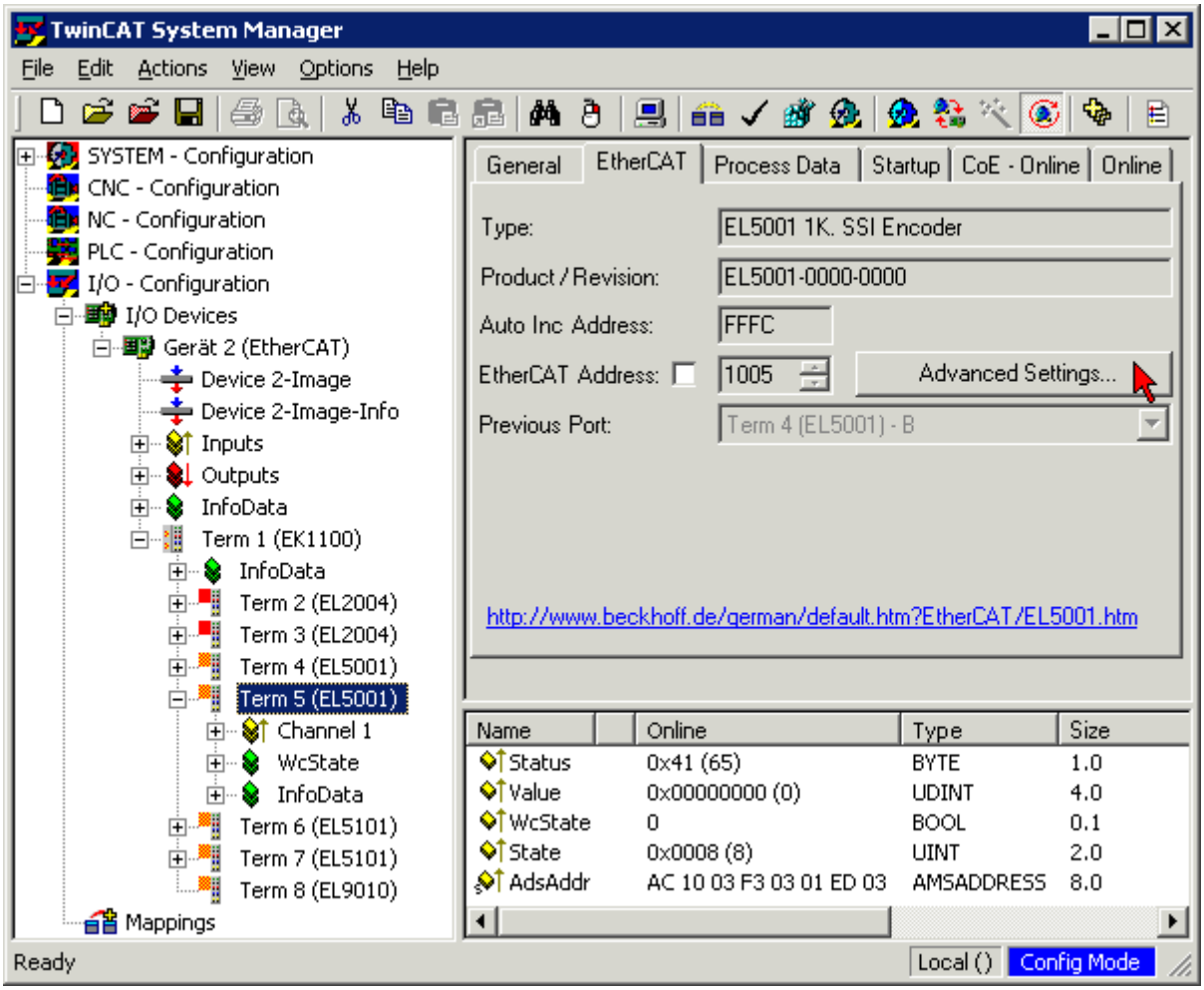
Older firmware versions can only be updated by the manufacturer!

Updating an EtherCAT device

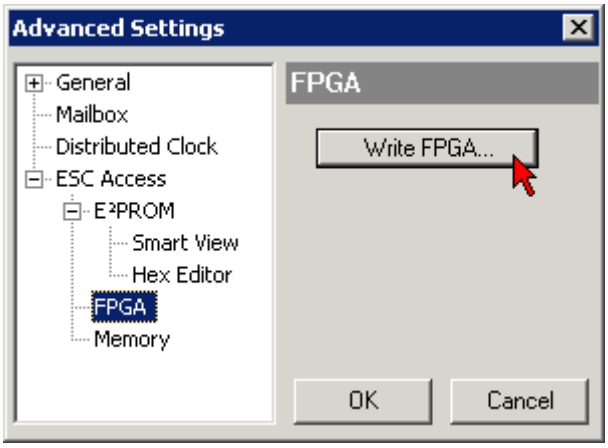
The following sequence order have to be met if no other specifications are given (e.g. by the Beckhoff support):

- Switch TwinCAT system to ConfigMode/FreeRun with cycle time >= 1 ms (default in ConfigMode is 4 ms). A FW-Update during real time operation is not recommended.

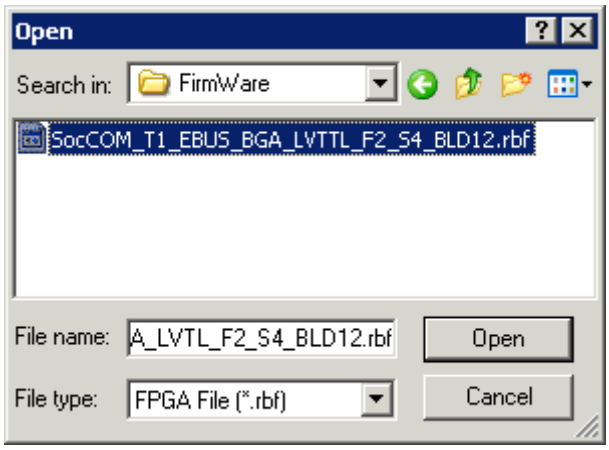
- In the TwinCAT System Manager select the terminal for which the FPGA firmware is to be updated (in the example: Terminal 5: EL5001) and
  click the *Advanced Settings* button in the *EtherCAT* tab:



- The *Advanced Settings* dialog appears. Under *ESC Access/E²PROM*/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

NOTE

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!

8.3.5 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.

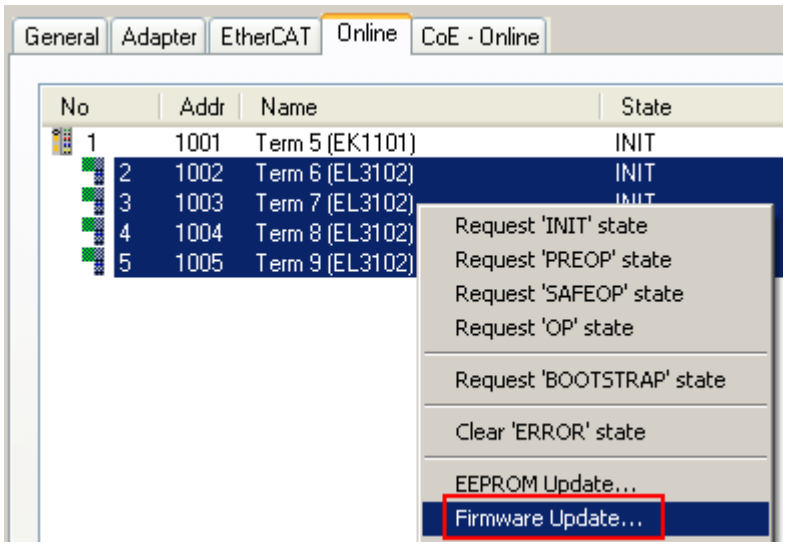


Fig. 180: Multiple selection and firmware update

Select the required slaves and carry out the firmware update in BOOTSTRAP mode as described above.

8.4 Restoring the delivery state

To restore the delivery state for backup objects in ELxxxx terminals, the CoE object Restore default parameters, *SubIndex 001* can be selected in the TwinCAT System Manager (Config mode) (see Fig. *Selecting the Restore default parameters PDO*)

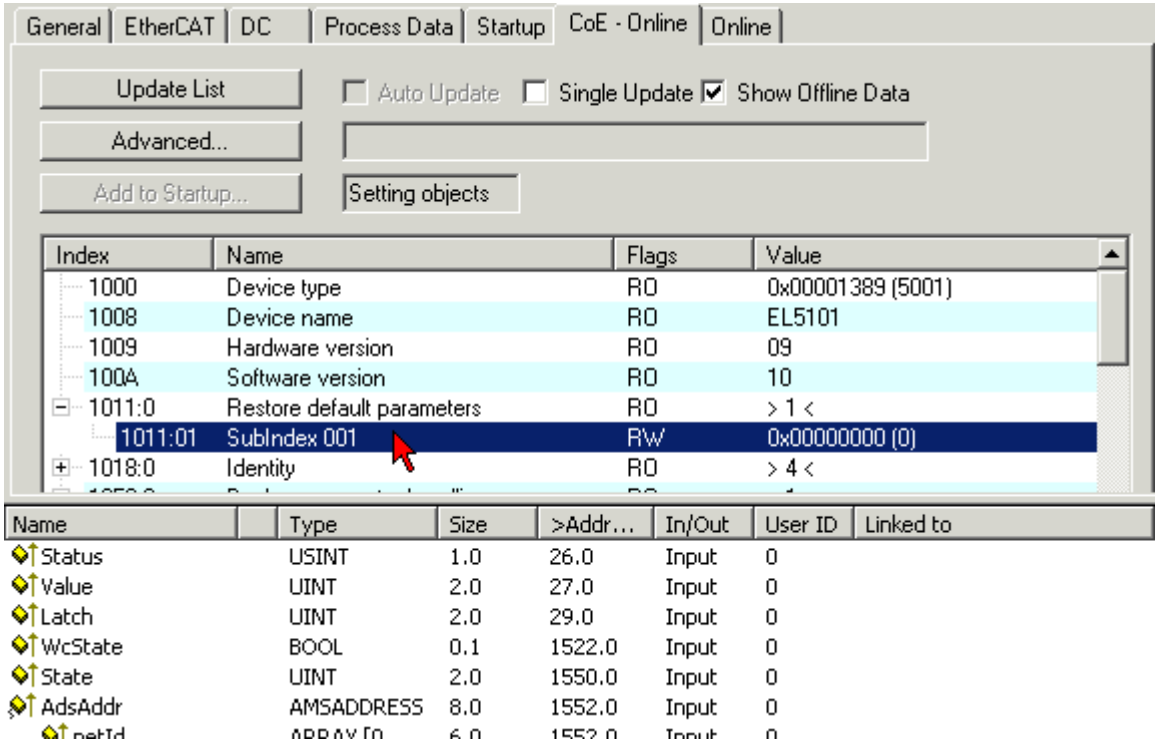


Fig. 181: Selecting the *Restore default parameters* PDO

Double-click on SubIndex 001 to enter the Set Value dialog. Enter the value **1684107116** in field *Dec* or the value **0x64616F6C** in field *Hex* and confirm with *OK* (Fig. *Entering a restore value in the Set Value dialog*). All backup objects are reset to the delivery state.

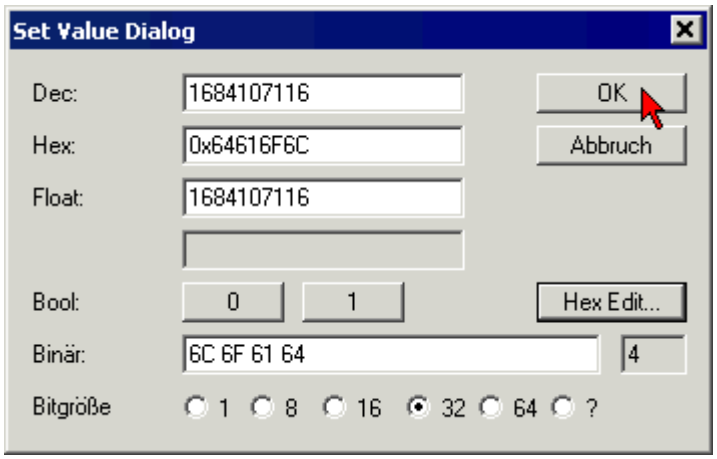


Fig. 182: Entering a restore value in the Set Value dialog

Alternative restore value

In some older terminals the backup objects can be switched with an alternative restore value: Decimal value: 1819238756, Hexadecimal value: 0x6C6F6164An incorrect entry for the restore value has no effect.

8.5 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:

http://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Castel Maggiore (BO) 40013
Via Lirone 60/i

e-mail: info@eurotecsrl.com
Tel: +39 051/6321595
Website: www.eurotecsrl.com

C.F.: 04103780377
P.I.: 01596581205