BECKHOFF New Automation Technology

Documentation | EN

EL6695

EtherCAT Bridge Terminal

2020-12-02 | Version: 1.4.5

Table of contents

1 Foreword

1.1 Notes on the documentation

Intended audience

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning these components.
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

1.2 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.

ℹ **Tip or pointer**

This symbol indicates information that contributes to better understanding.

1.3 Documentation Issue Status

Version	Comment
1.4.5	- Addenda within section “Symmetric PDO mapping” - Update structure
1.4.4	- Updated section “Symmetric PDO mapping” - Update structure
1.4.3	- Updated section “Distributed Clocks” - Update structure
1.4.2	- Updated section “Symmetric PDO mapping” - Update structure
1.4.1	- Update structure
1.4.0	- Addenda of example for FoE data throughput - Several additions/ corrections - Update structure - Update revision status
1.3.0	- Update section „Function and Operating modes“ (addenda of FoE operating mode)
1.2.0	- Updated section “Introduction” - Updated section “Function and operating modes” - Addenda chapter “Connection” - Updated section “Diagnostic LEDs”
1.1.0	- Updated section “Function and operating modes” - Updated section “Technical data” - Update structure - Update revision status
1.0.0	- First publication
0.9.8	- Section “Selective PDO mapping” supplemented with “Conversion to a global data type” (Provisional version – subject to change)
0.9.7	- Correction in section “AoE application for CoE access”
0.9.6	- Section “Function and operating modes” expanded
0.9.5	- Section “AoE application for CoE access” inserted in section “Transfer-capable mailbox protocols”
0.9	- Provisional documentation for the first series
0.1	- First created

1.4 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

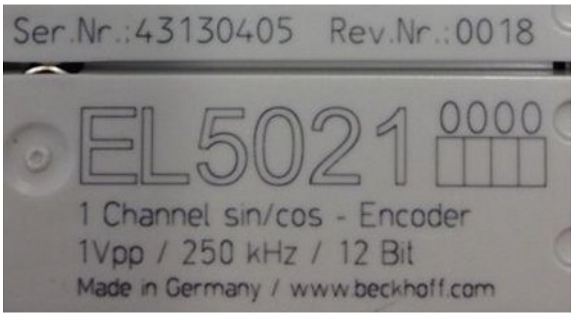

Ser.Nr.:43130405 Rev.Nr.:0018
EL5021 0000
1 Channel sin/cos - Encoder
1Vpp / 250 kHz / 12 Bit
Made in Germany / www.beckhoff.com

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

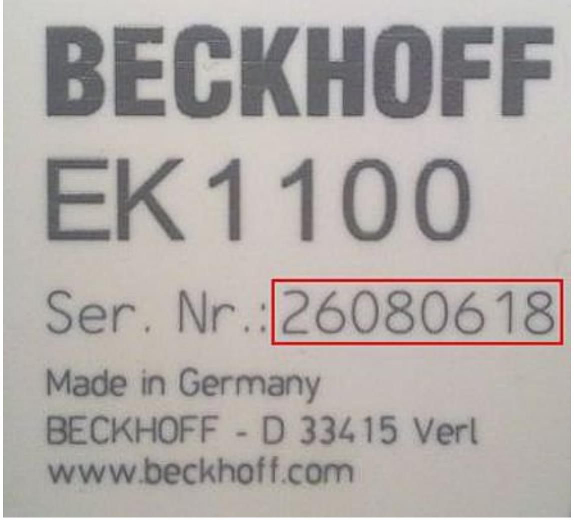


Fig. 2: EK1100 EtherCAT coupler, standard IP20 IO device with serial/ batch number

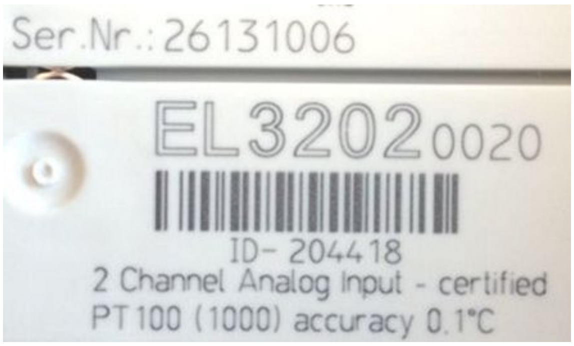

Ser.Nr.:26131006
EL3202 0020
ID- 204418
2 Channel Analog Input - certified
PT100 (1000) accuracy 0.1°C

Fig. 3: EL3202-0020 with serial/ batch number 26131006 and unique ID-number 204418

1.4.1 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.

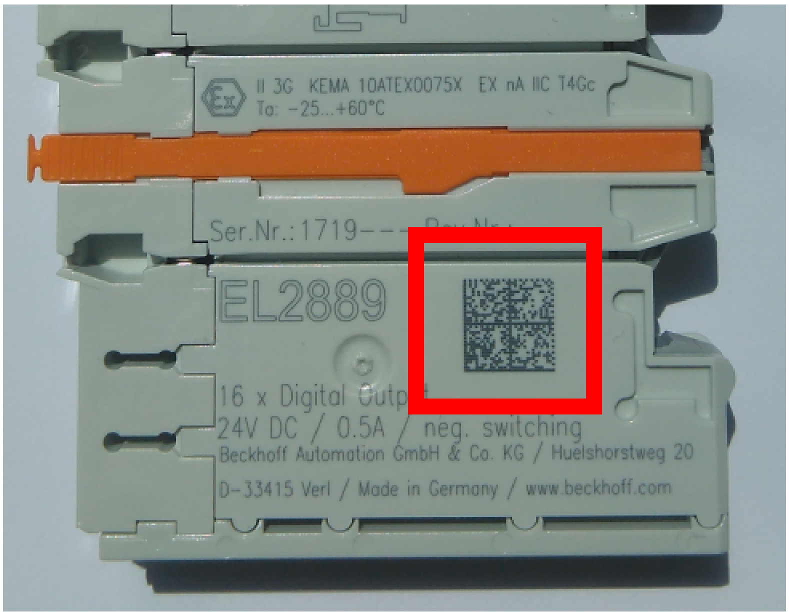

II 3G KEMA 10ATEX0075X EX nA IIC T4Gc
Ta: -25...+60°C
Ser.Nr.:1719---
EL2889
16 x Digital Output
24V DC / 0.5A / neg. switching
Beckhoff Automation GmbH & Co. KG / Huelshorstweg 20
D-33415 Verl / Made in Germany / www.beckhoff.com

Fig. 4: 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 or calibrated terminals	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

2.1 Introduction

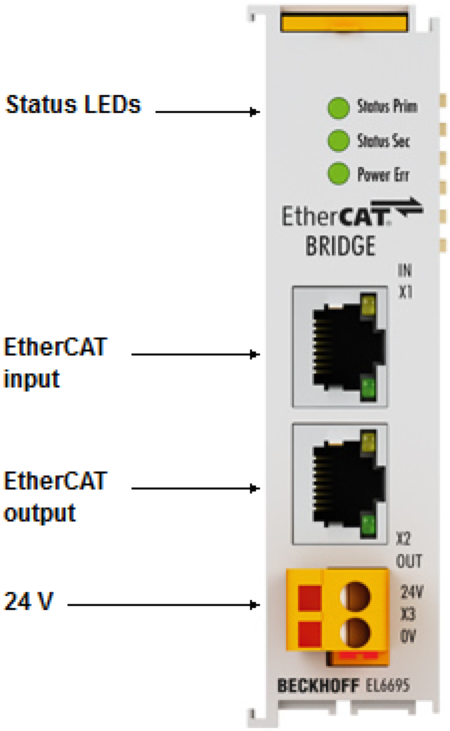


Fig. 5: EL6695

EtherCAT bridge terminal

The EL6695 EtherCAT bridge terminal enables real-time data exchange between EtherCAT strands with different masters. Asynchronous communication via various protocols is also supported. Synchronization of distributed clocks is possible in both directions. The EL6695 differs from the EL6692 (which will continue to be available) in terms of flexible CoE configuration, a device emulation option and a significant increase in data throughput. A convenient configuration interface is available in the TwinCAT System Manager “Extension”, like for the EL6692. The power supply for the secondary side (RJ 45) is via an external connection, the primary side is supplied via the E-bus. Thanks to the flexible CoE configuration, the bridge terminal can also be used to integrate a subordinate IPC systems as an EtherCAT slave. In this case, the user can define a parameter set in the CoE and present the subsystem externally as a user-defined EtherCAT slave.

2.2 Technical data

Technical data	EL6695
Ports	Primary side: E-bus (terminal strand), secondary side: 2 x 100 Mbit/s Ethernet RJ 45, in/out
Function	EtherCAT distributed clock synchronization, data exchange synchronous/asynchronous
Cable length	Secondary port: max. 100 m 100BASE-TX
Hardware diagnostics	Status LEDs
Power supply Current consumption	Primary: via the E-bus 400 mA typically Secondary: 24 V DC (-15%/+20%), 80 mA typically, pluggable Only one of the voltages is required for operation. If both voltages are present, the internal power supply unit preferentially uses the 24 V supply
Distributed Clocks	Yes, TwinCAT from TC3.1 b4018.4 is required
Electrical isolation	500 V (E-bus/secondary side)
Cyclic process data	max. 3 kbyte in each direction Notice: how many cyclical PDOs are supported mainly depends on the EcMaster used With TC2.11b2248 or TC3.1 b4018: 255 variables, maximum overall size MTU (~1400 bytes)
Supported asynchronous protocols	CoE, EoE, AoE, FoE, (VoE, SoE)
PDO transfer rate	Depends on the operating mode and the data quantity (typically 10 to 100 µs)
Minimum EtherCAT cycle time	50 µs
Special features	● Can be used as external reference clock in TwinCAT ● Synchronous data exchange ● Flexible CoE definition / CoE device emulation ● Can be used for direct two-sided DC synchronization ● PDO mapping (symmetric or selective) ● ADS routing (processing of ADS queries from two masters) ● Transfer rate up to 5 kbytes of user data per FoE
Operating/storage temperature	0 to +55 °C / -25 to +85 °C
Relative humidity	95 % no condensation
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
Protect. class / installation pos.	IP 20/variable
Dimensions	Width: 24 mm (side-by-side installation) Height: 100 mm Depth: 68 mm
Approvals	CE cULus [▶ 34]

2.2.1 Compare to Beckhoff EtherCAT Data-Exchange devices

	EL6695	EL6692	FC1100	FC1121	CXnnnn-B110
Design	24mm Terminal		PCIe-plug-in card		Integrated within embedded-PC
PDO Cyclic process-data	max. 3 kB every direction ([1] Execute TC2.11b2248 or TC3.1 b4018: 255 variables, max. entire scaling MTU (c.a.1400 Byte)	max. 480 Byte every direction	max. 1024 Byte every direction		max. 480 Byte every direction
PDO transmission speed (device-internal, without bus-cycle)	E.g.: 200 Byte in/200 Byte out: 15 µs typ E.g.: 1400 Byte in/1400 Byte out: 50 µs typ	typ. 1..4 ms E.g.: 200 Byte in/200 Byte out: 1 ms typ.	E.g.: 480 Byte in/480 Byte out: ca 300 µs ([2]		E.g.: 480 Byte in/480 Byte out: ca 250 µs auf CX5020 ([2]
Supported asynchronous protocols	CoE, EoE, AoE, FoE, (VoE, SoE)	CoE, AoE, EoE			
Mailbox	128-1498 Byte	128-1024 Byte	64-1024 Byte		
Mailbox-default settings	1024 Byte	256 Byte	1024 Byte		512 Byte
Minimal allowed EtherCAT cycletime ([3]	50 µs (SyncMan interrupt be used)	No limit (SyncMan interrupt not be used)			
DistributedClocks syncronization	yes				
In TwinCAT als externe Referenzuhr nutzbar	yes				
Specific properties	• Synchronous data-exchange • Flexible CoE-definition / CoE device-emulation • As for direct bidirectional DC-synchronization applicable • PDO mapping (symmetric or selective) • ADS Routing (proceeding of ADS requests by two masters) • Transmission of up to 5 kB user data by FoE • Independent power supply primary/ secondary	• For DC-synchronization indirectly over EC-Master applicable • ADS routing			

)[1] The number of supported cyclic PDO is dependent by the EtherCAT master

)[2] Transmission values are significant dependent by the IPC environment and control

)[3] This limit means an operational limit of the device. Indeed the lowest reasonable cycle time of both sides is amongst others dependent by the amount of data to be transferred (and also the PDO transmission time) – a sensible cycle time should be chosen as for the terminal is able to get/set data every cycle.

2.3 Operating conditions for installation

Please ensure that the EL6695 EtherCAT Terminal is only transported, stored and operated under the specified conditions (see technical data)!

⚠ **WARNING**

Invalid operating conditions must be avoided!

The EL6695 must not be used under the following operating conditions:

- under the influence of ionizing radiation
- in corrosive environments
- in an environment that leads to unacceptable soiling of the Bus Terminal

Safety instructions for installation

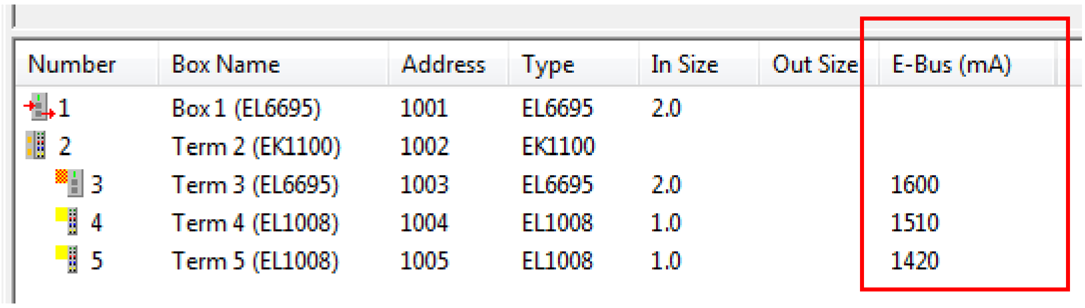
Before installing and commissioning the EtherCAT Terminals please read the safety instructions in the foreword of this documentation.

Transport / storage

When transporting or storing the EtherCAT Terminals, always use the original packaging in which they were delivered.

Electrical installation

Please note that the EL6695 preferentially uses the 24 V supply, if both E-bus and 24 V are available. In order to ensure that the EL6695 is supplied via the E-bus, it is therefore advisable commission it with the 24 V supply disconnected. Otherwise, overload may occur in the E-bus supply in the terminal strand in the event of a failure of the 24 V supply, if the E-bus supply was not pre-dimensioned correctly.



Number	Box Name	Address	Type	In Size	Out Size	E-Bus (mA)
1	Box 1 (EL6695)	1001	EL6695	2.0		
2	Term 2 (EK1100)	1002	EK1100			
3	Term 3 (EL6695)	1003	EL6695	2.0		1600
4	Term 4 (EL1008)	1004	EL1008	1.0		1510
5	Term 5 (EL1008)	1005	EL1008	1.0		1420

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

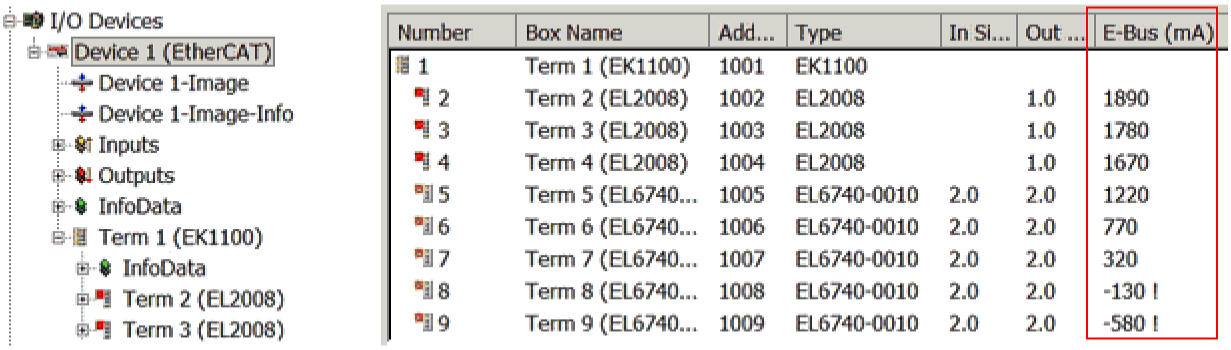
It is recommended to use the appropriate Beckhoff components e.g.
- cable sets ZK1090-9191-xxxx respectively
- RJ45 connector, field assembly ZS1090-0005
- EtherCAT cable, field assembly ZB9010, ZB9020

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. 6: 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.

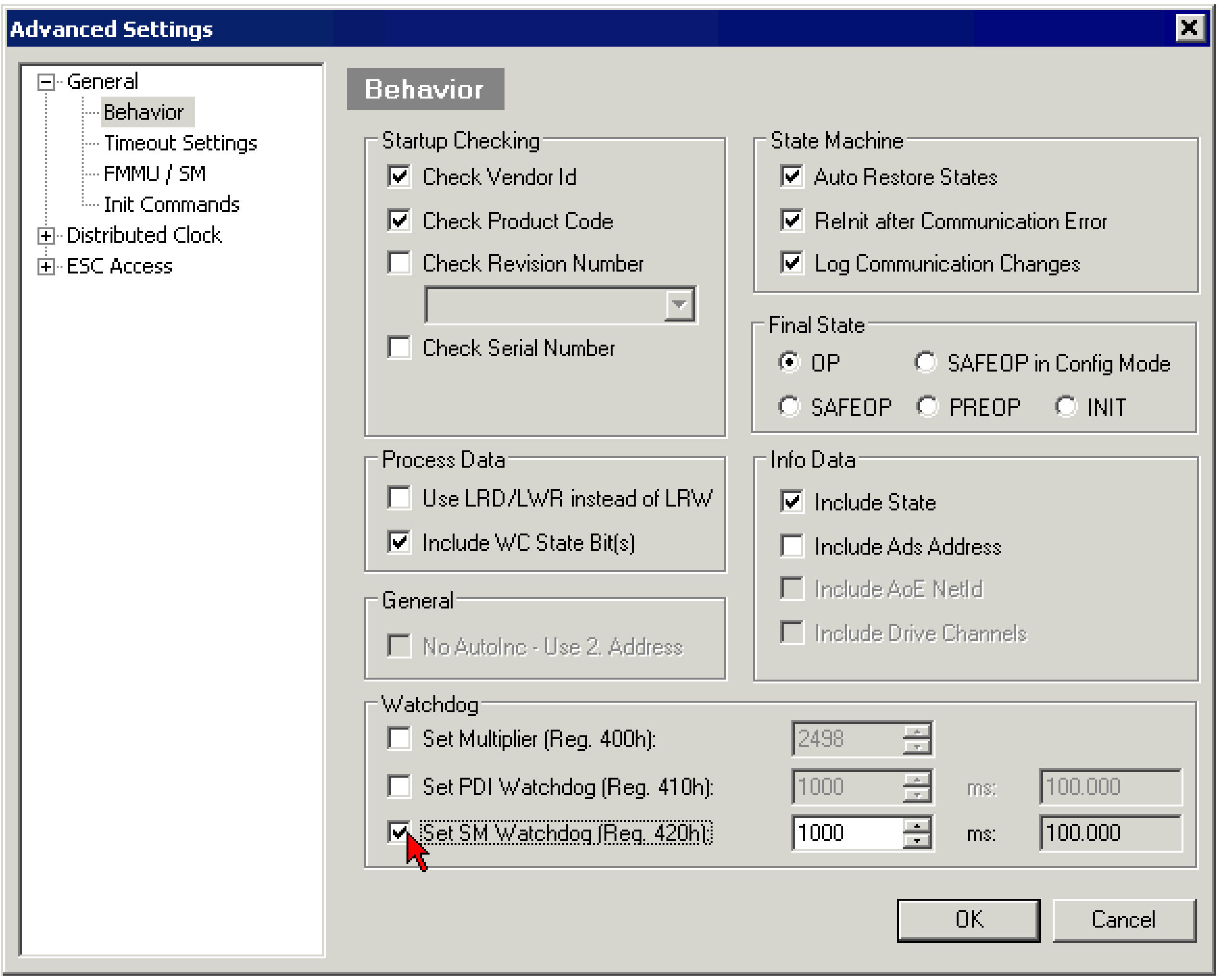
The SyncManager watchdog monitors correct and timely process data communication with the ESC from the EtherCAT side.

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.



Advanced Settings

- General
  - Behavior
  - Timeout Settings
  - FMMU / SM
  - Init Commands
- Distributed Clock
- ESC Access

Behavior

Startup Checking
☑ Check Vendor Id
☑ Check Product Code
□ Check Revision Number
□ Check Serial Number

State Machine
☑ Auto Restore States
☑ ReInit after Communication Error
☑ Log Communication Changes

Final State
● OP ○ SAFEOP in Config Mode
○ SAFEOP ○ PREOP ○ INIT

Process Data
□ Use LRD/LWR instead of LRW
☑ Include WC State Bit(s)

Info Data
☑ Include State
□ Include Ads Address
□ Include AoE NetId
□ Include Drive Channels

General
□ No AutoInc - Use 2. Address

Watchdog
□ Set Multiplier (Reg. 400h): 2498
□ Set PDI Watchdog (Reg. 410h): 1000 ms: 100.000
☑ Set SM Watchdog (Reg. 420h): 1000 ms: 100.000

Fig. 7: 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.

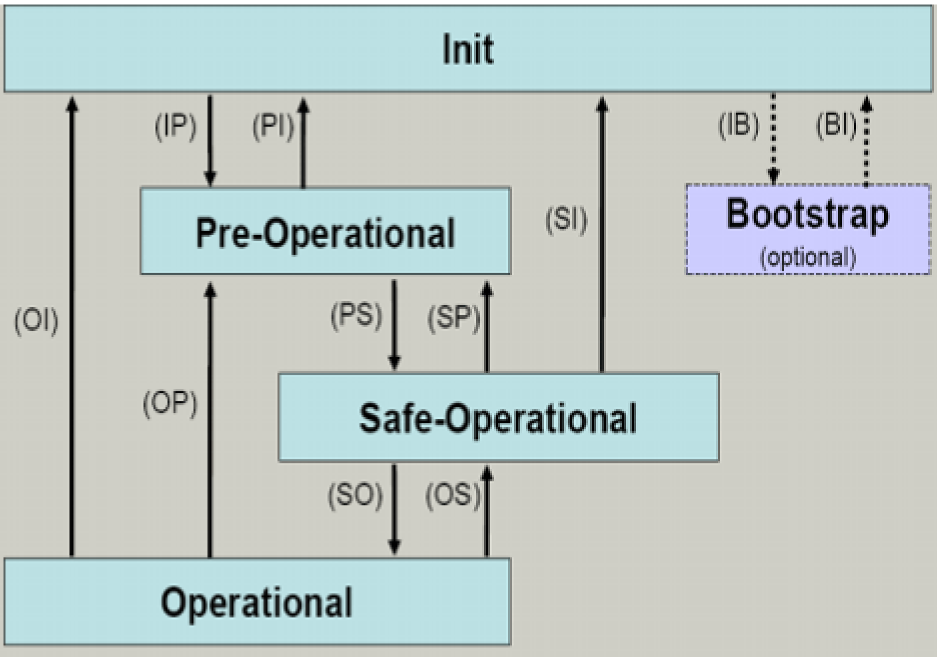


Fig. 8: States of the EtherCAT State Machine

Init

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

Pre-Operational (Pre-Op)

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

In *Pre-Op* state mailbox communication is possible, but not process data communication. The EtherCAT master initializes the sync manager channels for process data (from sync manager channel 2), the 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 [▶ 17] 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.

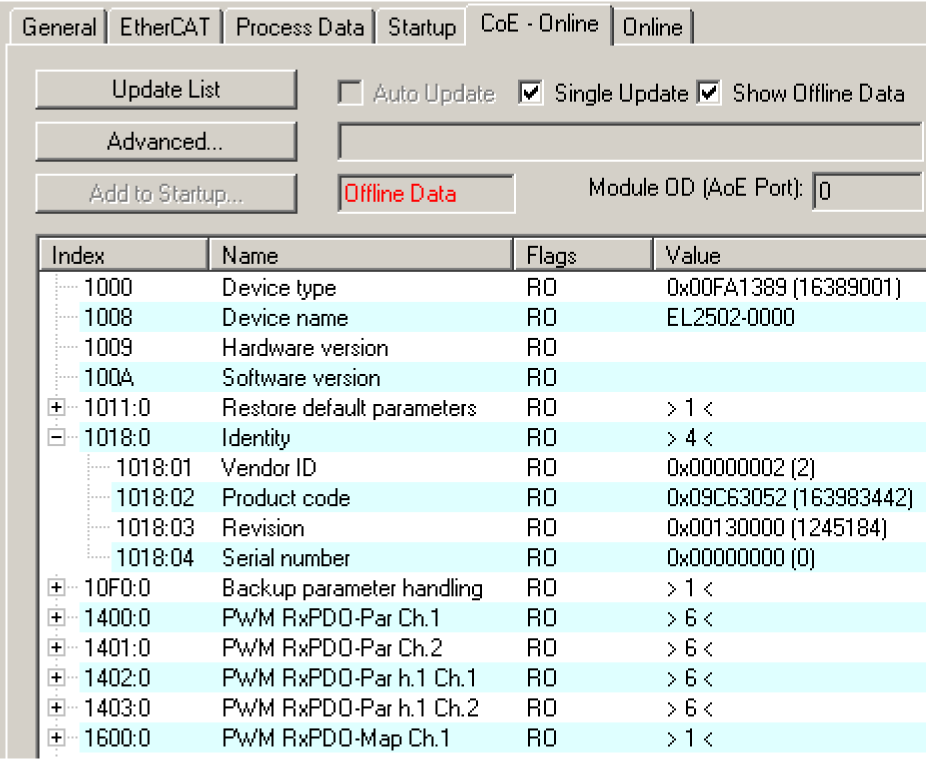
Other important ranges are:

- 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:



□ Auto Update ☑ Single Update ☑ Show Offline Data

Offline Data

Module OD (AoE Port): 0

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

Fig. 9: “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.

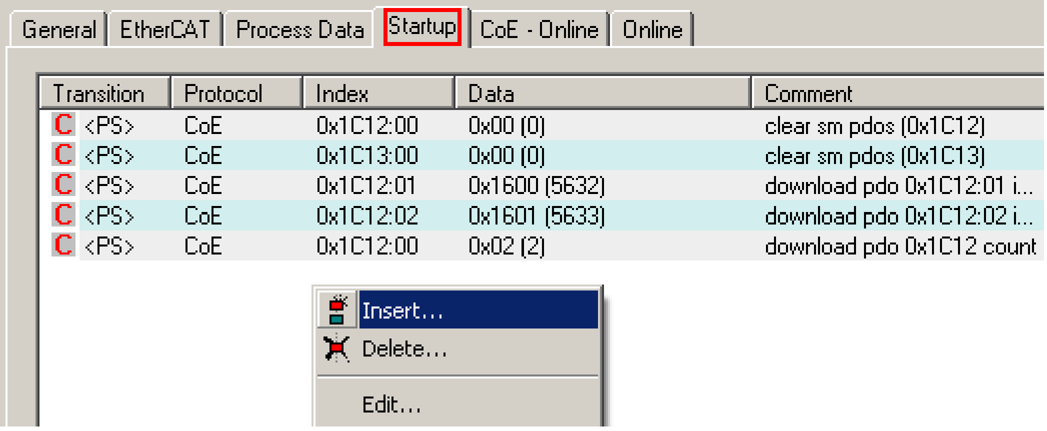
Startup list

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.



Transition	Protocol	Index	Data	Comment
C <PS>	CoE	0x1C12:00	0x00 (0)	clear sm pdos (0x1C12)
C <PS>	CoE	0x1C13:00	0x00 (0)	clear sm pdos (0x1C13)
C <PS>	CoE	0x1C12:01	0x1600 (5632)	download pdo 0x1C12:01 i...
C <PS>	CoE	0x1C12:02	0x1601 (5633)	download pdo 0x1C12:02 i...
C <PS>	CoE	0x1C12:00	0x02 (2)	download pdo 0x1C12 count

Fig. 10: Startup list in the TwinCAT System Manager

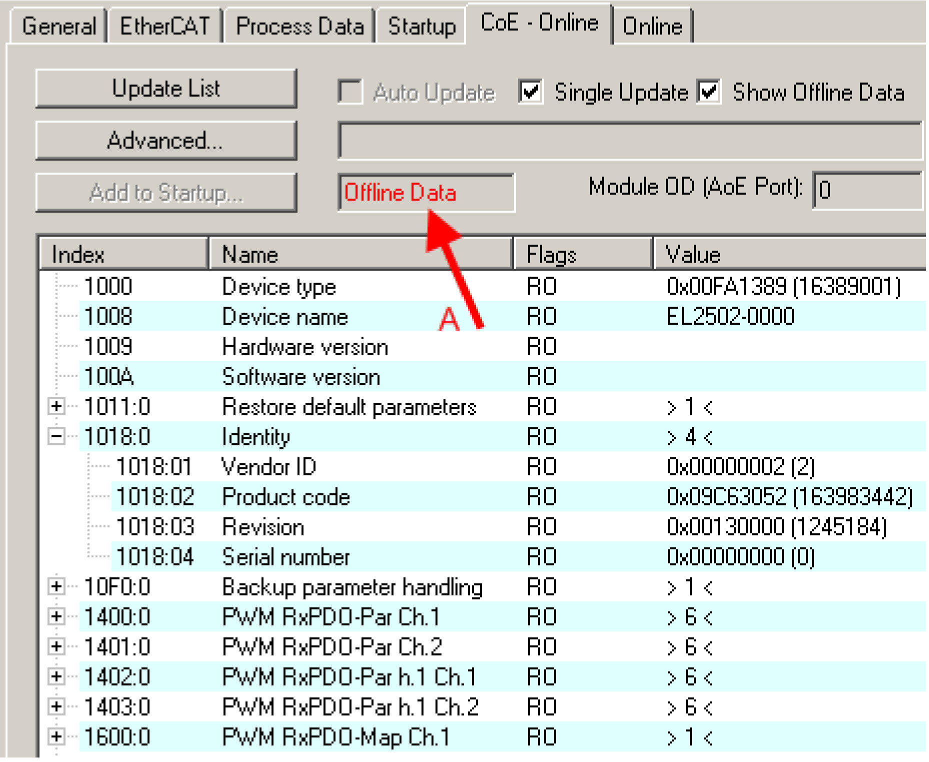
The Startup list may already contain values that were configured by the System Manager based on the ESI specifications. Additional application-specific entries can be created.

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.



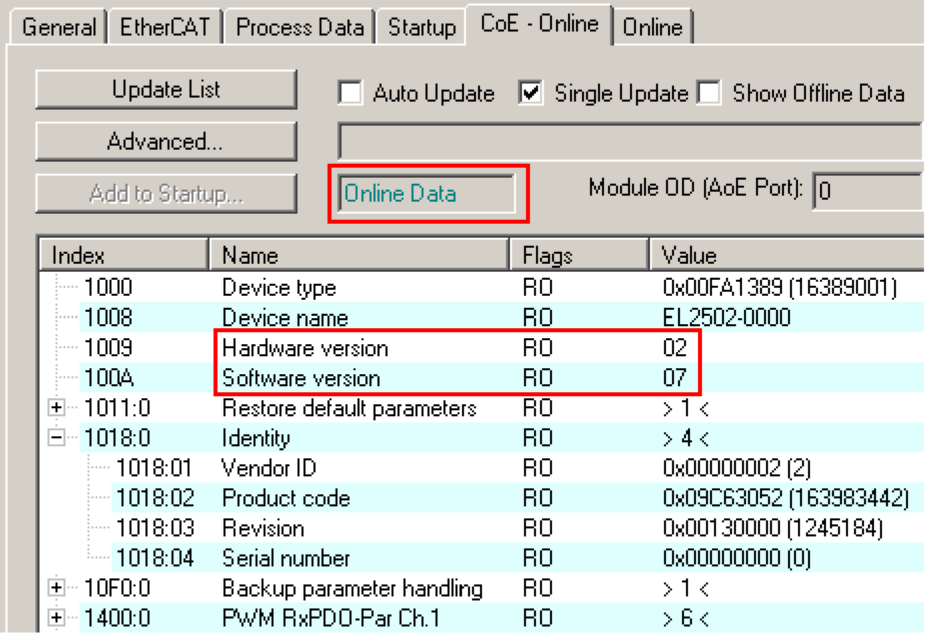
□ Auto Update ☑ Single Update ☑ Show Offline Data

Module OD (AoE Port): 0

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

Fig. 11: Offline list

- If the slave is online
  - The actual current slave list is read. This may take several seconds, depending on the size and cycle time.
  - The actual identity is displayed
  - The firmware and hardware version of the equipment according to the electronic information is displayed
  - **Online** is shown in green.



□ Auto Update ☑ Single Update □ Show Offline Data

Module OD (AoE Port): 0

Index	Name	Flags	Value
1000	Device type	RO	0x00FA1389 (16389001)
1008	Device name	RO	EL2502-0000
1009	Hardware version	RO	02
100A	Software version	RO	07
1011:0	Restore default parameters	RO	> 1 <
1018:0	Identity	RO	> 4 <
1018:01	Vendor ID	RO	0x00000002 (2)
1018:02	Product code	RO	0x09C63052 (163983442)
1018:03	Revision	RO	0x00130000 (1245184)
1018:04	Serial number	RO	0x00000000 (0)
10F0:0	Backup parameter handling	RO	> 1 <
1400:0	PWM RxPDO-Par Ch.1	RO	> 6 <

Fig. 12: Online list

Channel-based order

The CoE list is available in EtherCAT devices that usually feature several functionally equivalent channels. For example, a 4-channel analog 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.

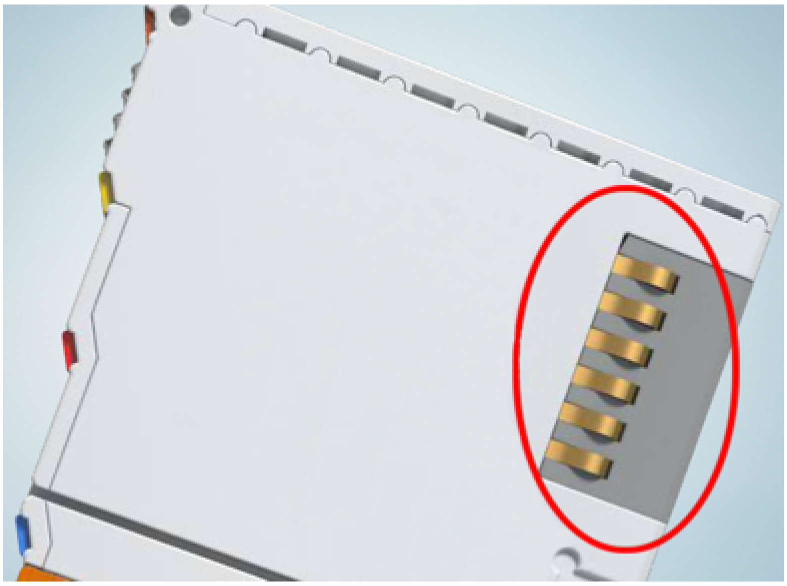


Fig. 13: Spring contacts of the Beckhoff I/O components

4.2 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).

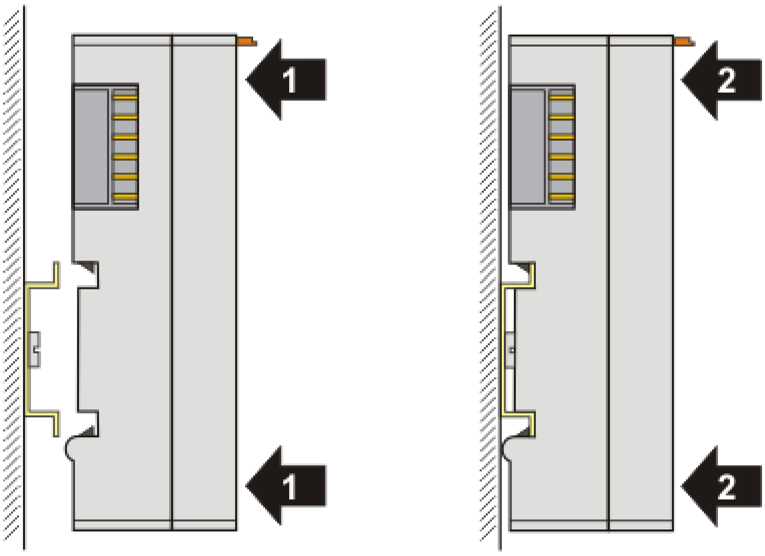
⚠ WARNING

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.

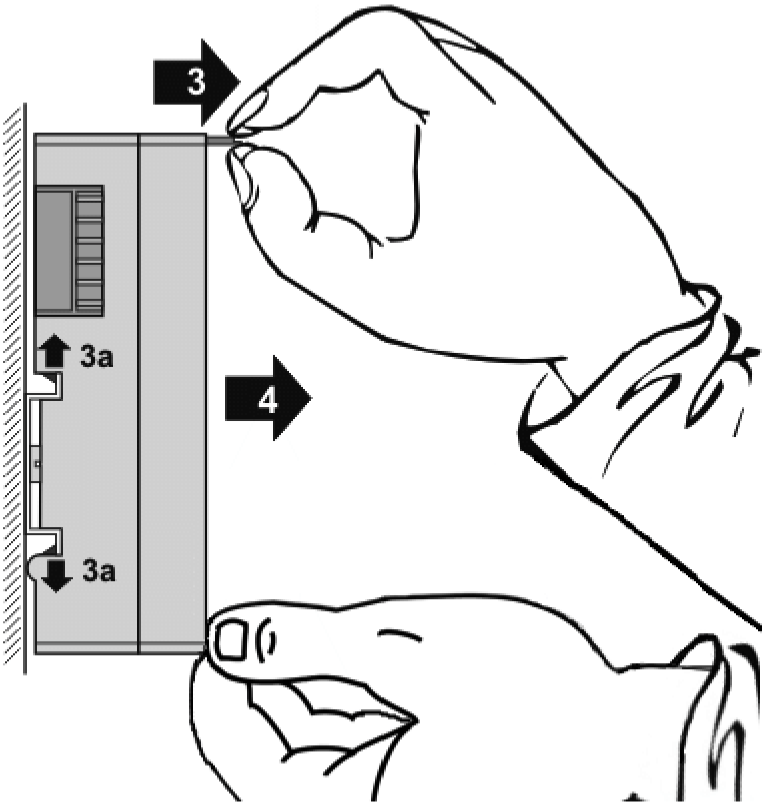


and press (1) the terminal module against the mounting rail until it latches in place on the mounting rail (2).

- 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 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)!

4.4 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.

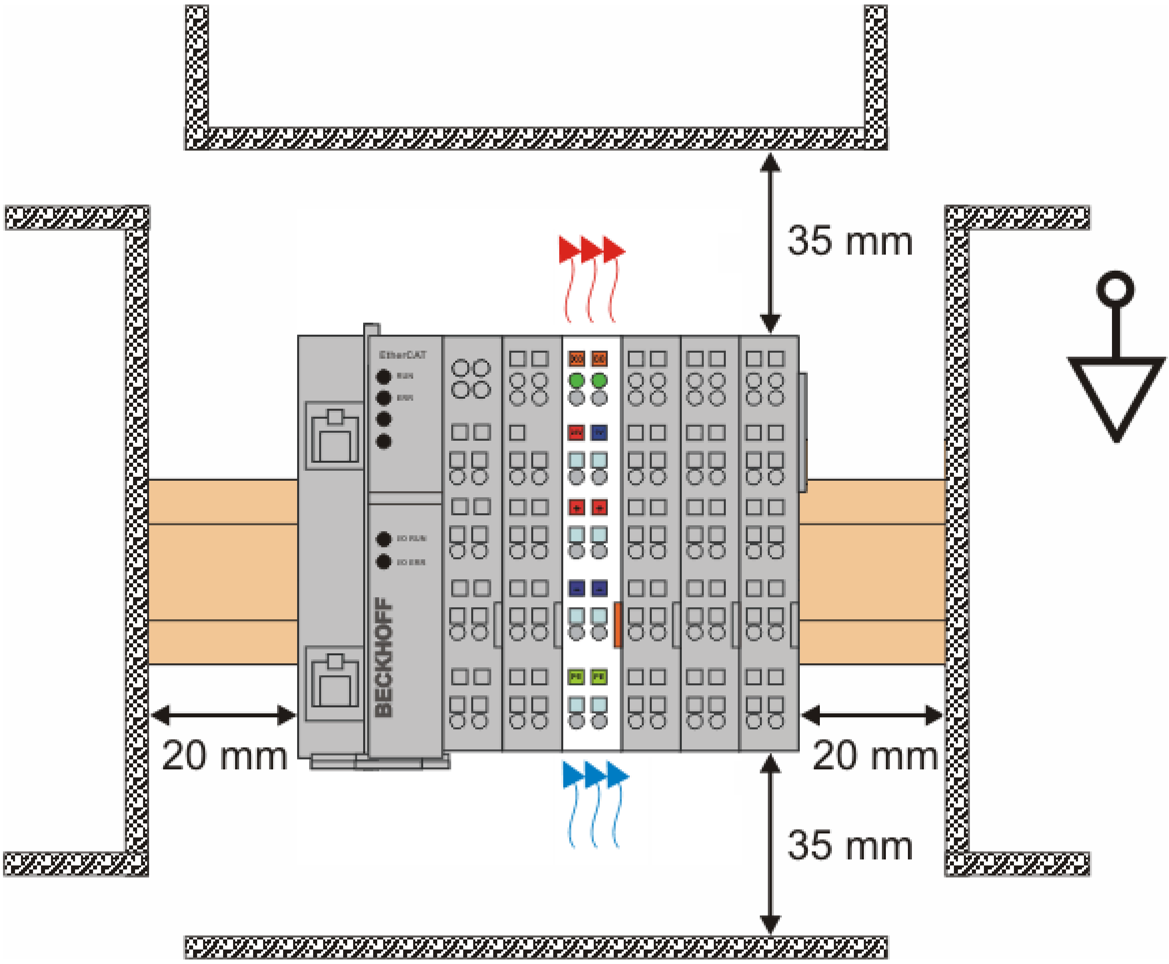


Fig. 14: Recommended distances for standard installation position

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.

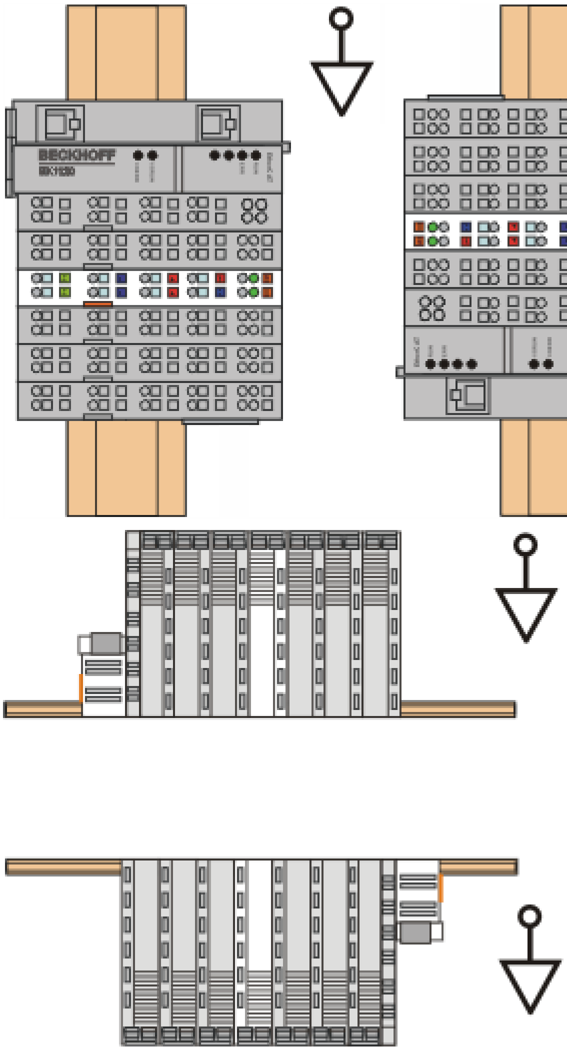


Fig. 15: Other installation positions

4.5 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)

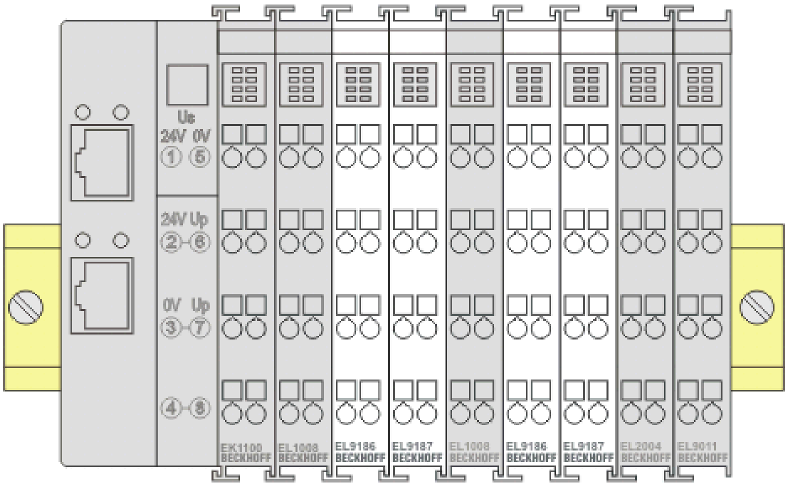


Fig. 16: Correct positioning

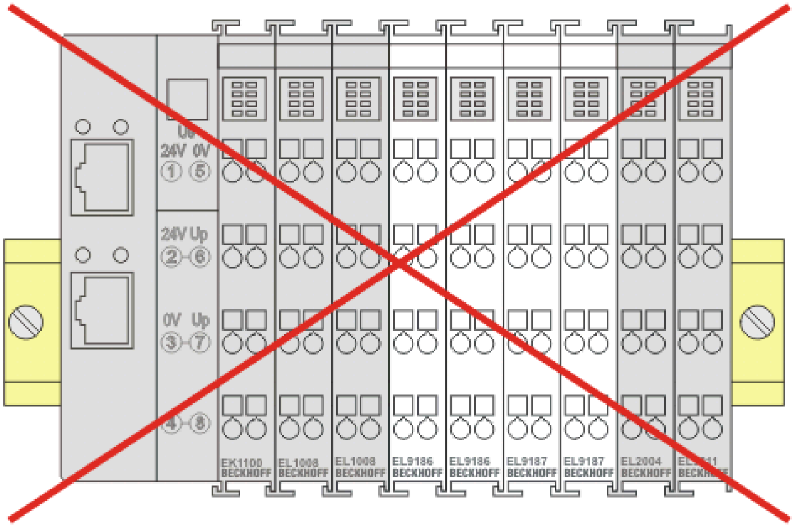


Fig. 17: Incorrect positioning

4.6 UL notice

c UL US	**Application** Beckhoff EtherCAT modules are intended for use with Beckhoff’s UL Listed EtherCAT System only.

c UL US	**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).

c UL US	**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:

4.7 Connection

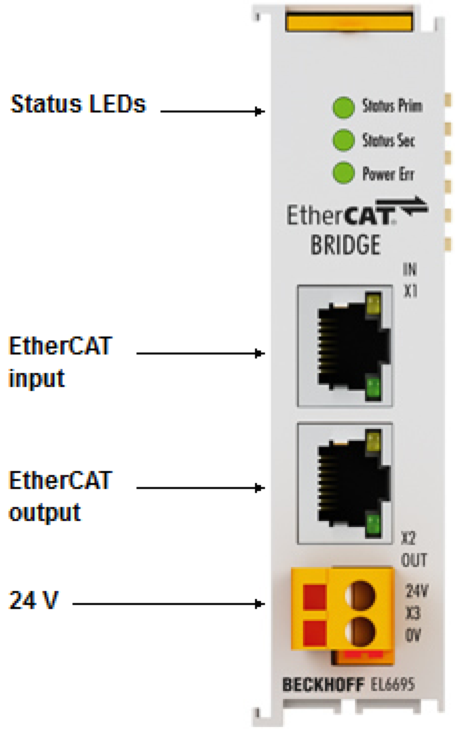


Fig. 18: Connection EL6695

Terminal point	Description
X1	EtherCAT Input (RJ45 with 10BASE-T/100BASE-TX Ethernet)
X2	EtherCAT output (RJ45 with 10BASE-T/100BASE-TX Ethernet)
X3	2-pole socket terminal connection (24 VDC), secondary side power supply

Diagnostic LEDs

For the LED description please refer to the chapter Diagnostic LEDs [▶ 175]

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:

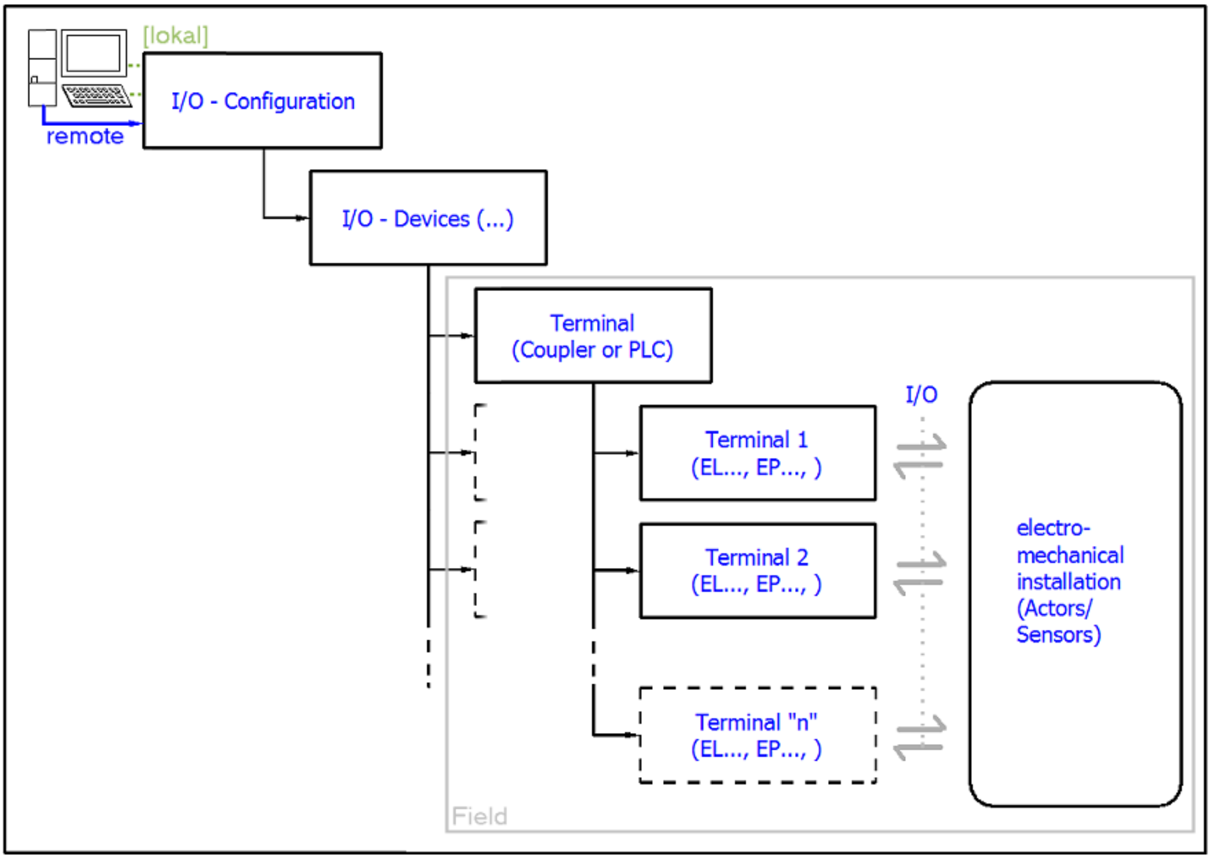


Fig. 19: Relationship between user side (commissioning) and installation

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)

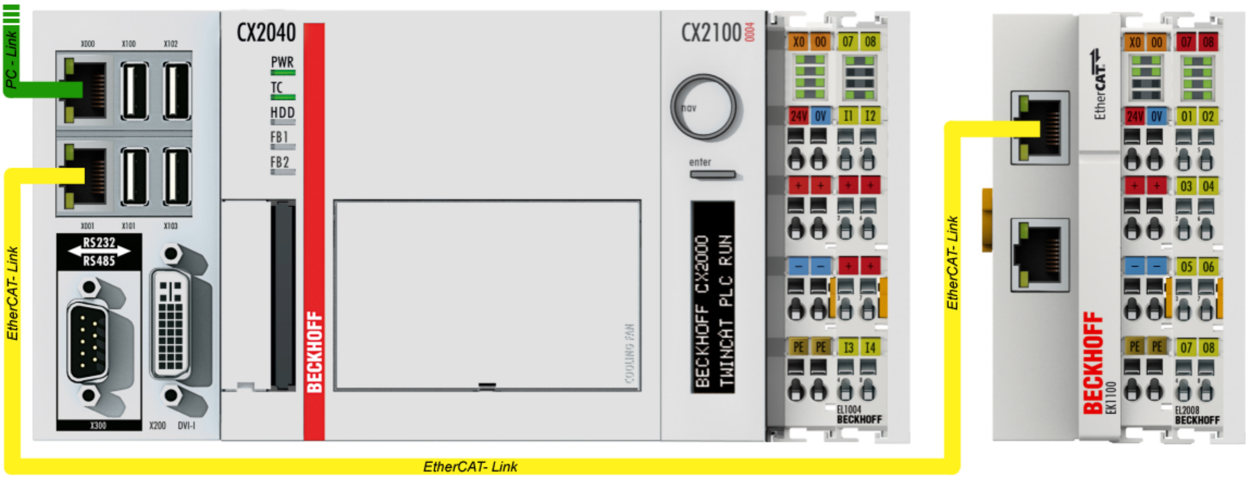


Fig. 20: Control configuration with Embedded PC, input (EL1004) and output (EL2008)

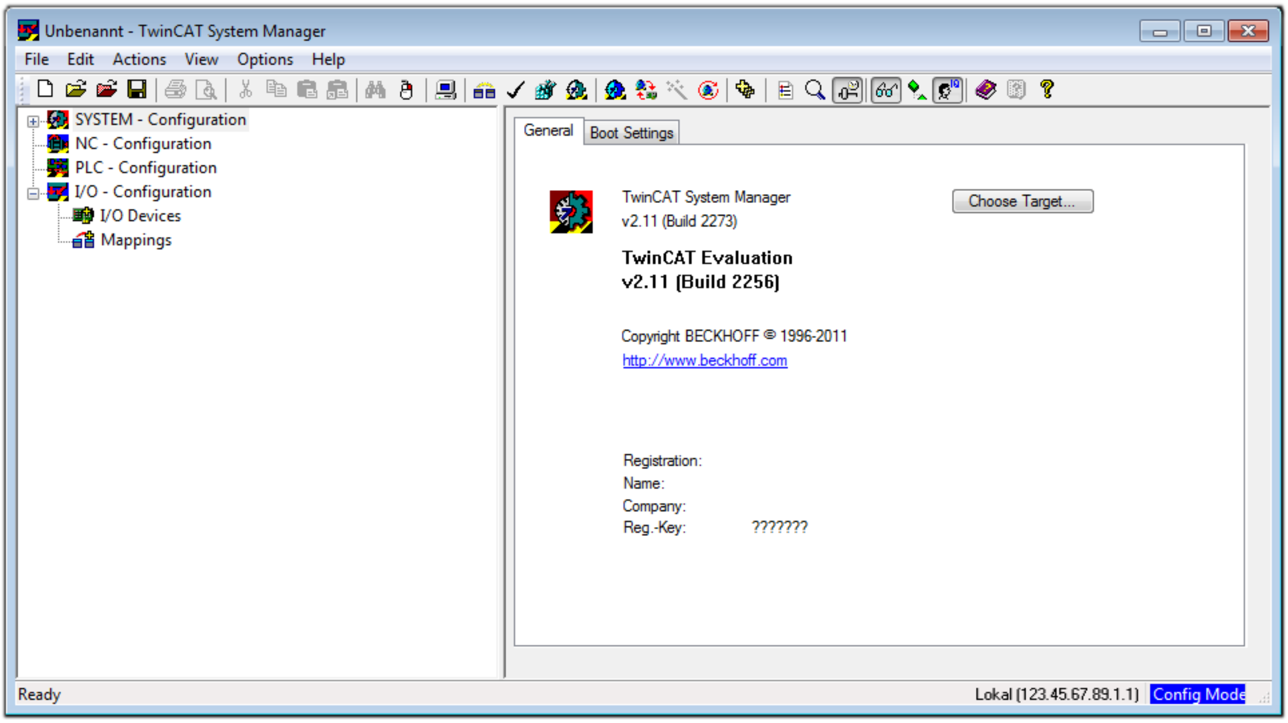
Note that all combinations of a configuration are possible; for example, the EL1004 terminal could also be connected after the coupler, or the EL2008 terminal could additionally be connected to the CX2040 on the right, in which case the EK1100 coupler wouldn’t be necessary.

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:



TwinCAT System Manager
v2.11 (Build 2273)

TwinCAT Evaluation
v2.11 (Build 2256)

Copyright BECKHOFF © 1996-2011
http://www.beckhoff.com

Registration:
Name:
Company:
Reg.-Key: ???????

Ready

Lokal (123.45.67.89.1.1) Config Mode

Fig. 21: 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 [▶ 41]”.

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:

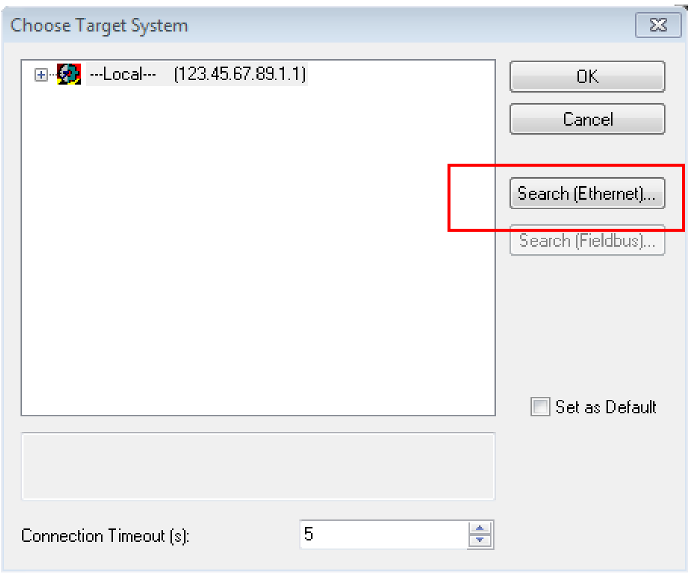
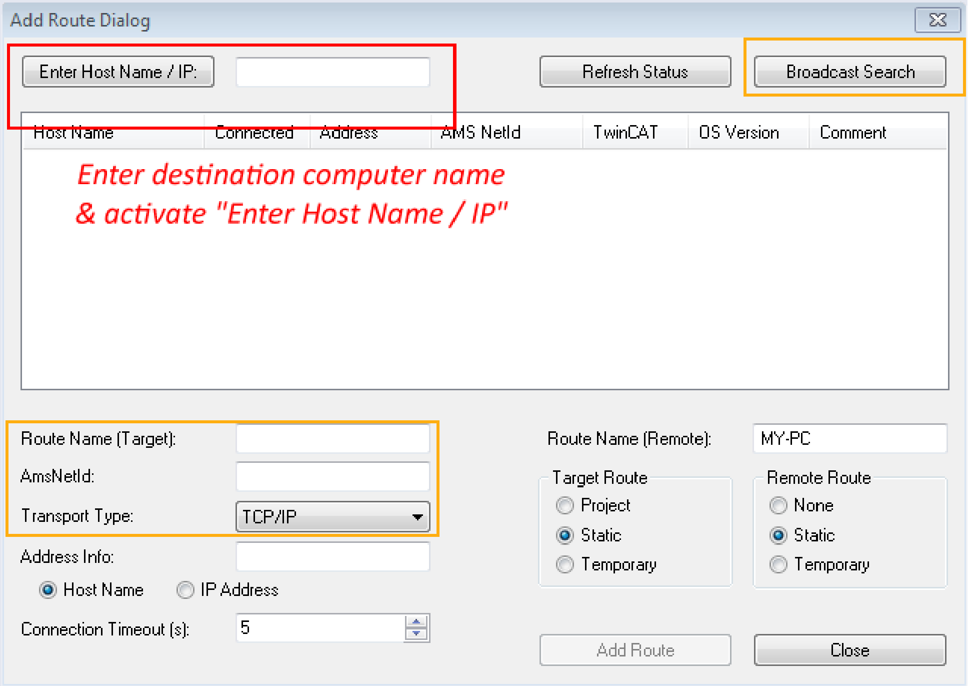


Fig. 22: Selection of the target system

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

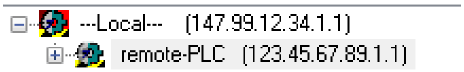
- enter the known computer name after “Enter Host Name / IP:” (as shown in red)
- perform a “Broadcast Search” (if the exact computer name is not known)
- enter the known computer IP or AmsNetID.



Enter destination computer name & activate "Enter Host Name / IP"

Fig. 23: 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).

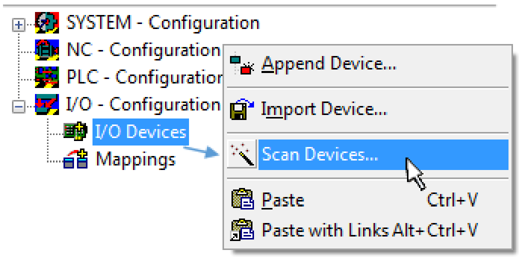
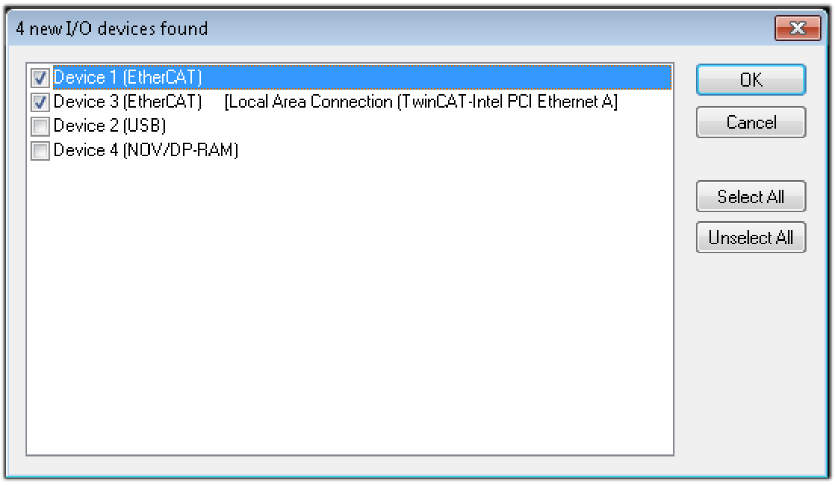


Fig. 24: Select “Scan Devices...”

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


4 new I/O devices found

☑ Device 1 (EtherCAT)
☑ Device 3 (EtherCAT) [Local Area Connection (TwinCAT-Intel PCI Ethernet A)
☐ Device 2 (USB)
☐ Device 4 (NOV/DP-RAM)

Fig. 25: 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 [▸ 37] described at the beginning of this section, the result is as follows:

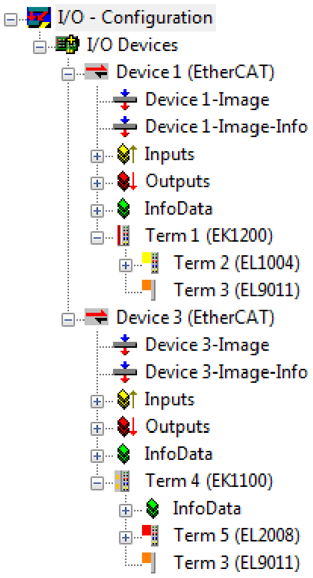


Fig. 26: Mapping of the configuration in the TwinCAT 2 System Manager

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:

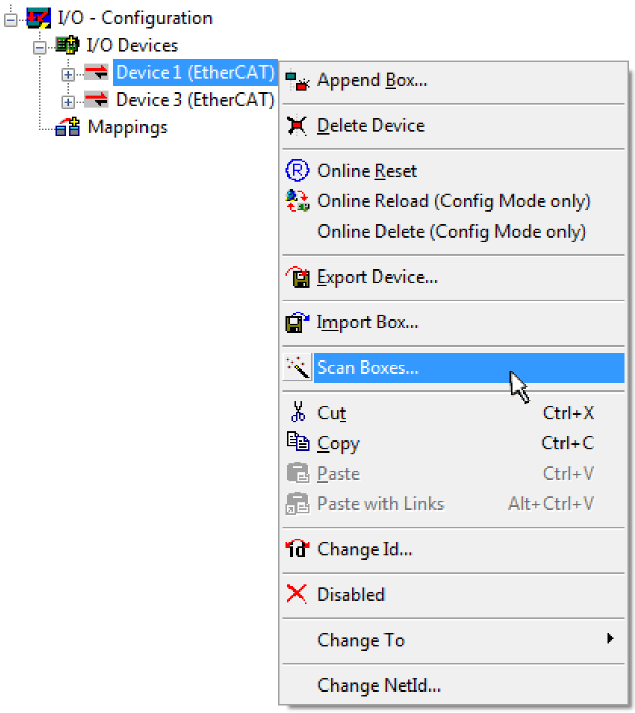


Fig. 27: Reading of individual terminals connected to a device

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

Programming and integrating the PLC

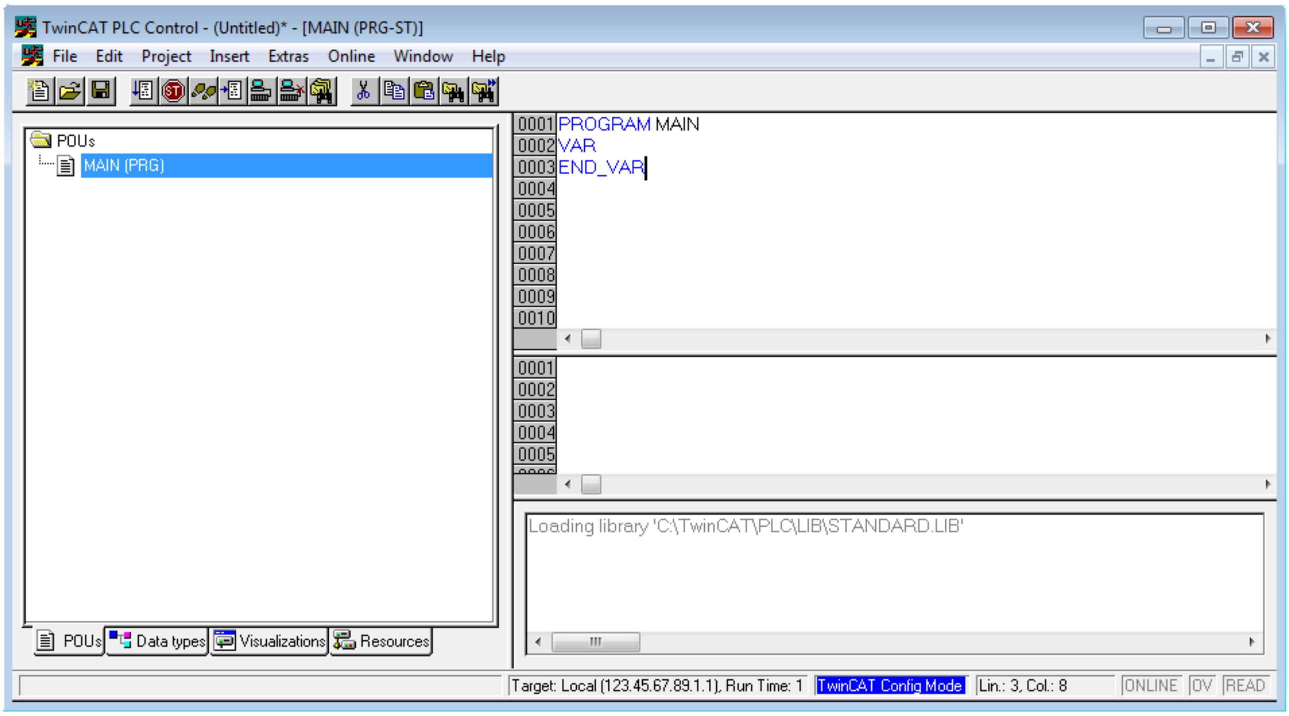
TwinCAT PLC Control is the development environment for the creation of the controller in different program environments: TwinCAT PLC Control supports all languages described in IEC 61131-3. There are two text-based languages and three graphical languages.

- **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:



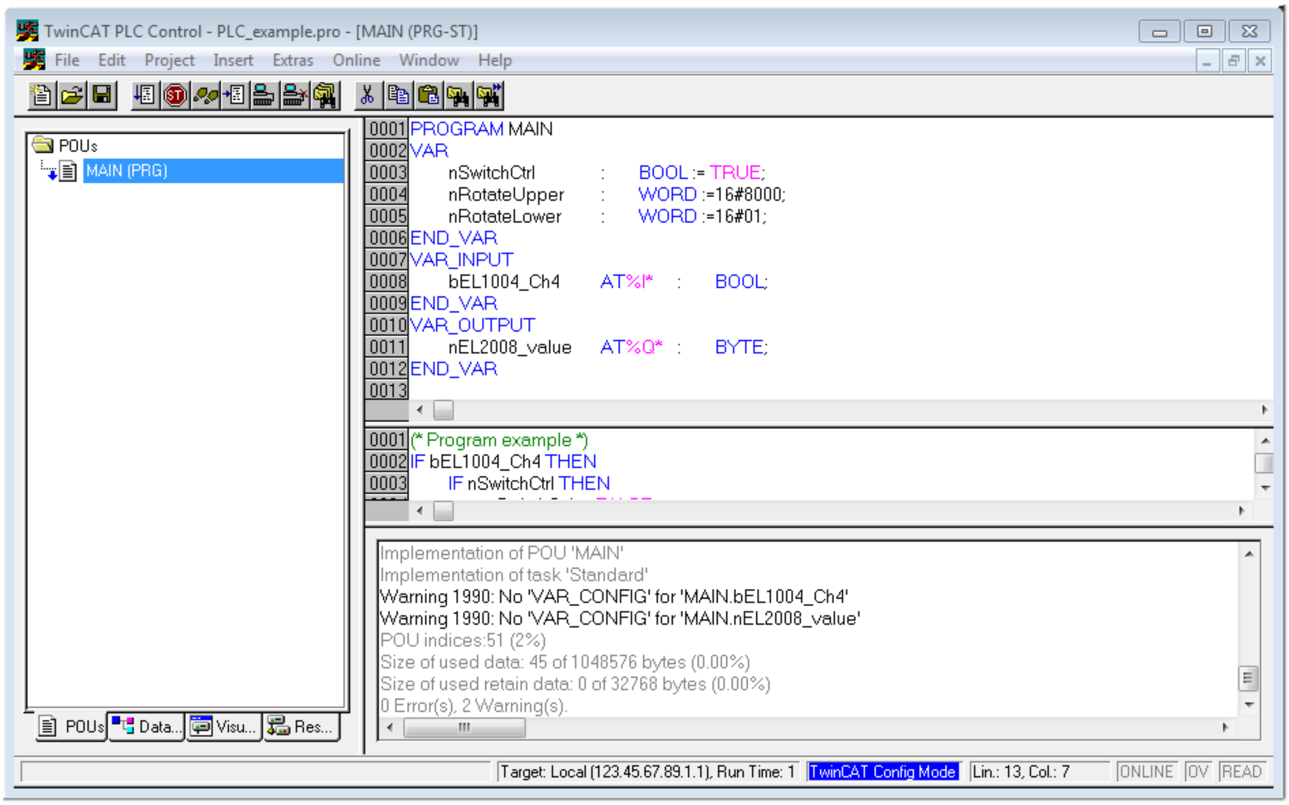
```
PROGRAM MAIN
VAR
END_VAR
```

Loading library 'C:\TwinCAT\PLC\LIB\STANDARD.LIB'

Target: Local (123.45.67.89.1.1), Run Time: 1 | TwinCAT Config Mode | Lin.: 3, Col.: 8 | ONLINE | OV | READ

Fig. 28: TwinCAT PLC Control after startup

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


TwinCAT PLC Control - PLC_example.pro - [MAIN (PRG-ST)]
File Edit Project Insert Extras Online Window Help
POUs
MAIN (PRG)
POUs
Data...
Visu...
Res...

```
PROGRAM MAIN
VAR
    nSwitchCtrl      :    BOOL := TRUE;
    nRotateUpper     :    WORD :=16#8000;
    nRotateLower     :    WORD :=16#01;
END_VAR
VAR_INPUT
    bEL1004_Ch4      AT%I*   :    BOOL;
END_VAR
VAR_OUTPUT
    nEL2008_value    AT%Q*   :    BYTE;
END_VAR
```

```
(* Program example *)
IF bEL1004_Ch4 THEN
    IF nSwitchCtrl THEN
```

```
Implementation of POU 'MAIN'
Implementation of task 'Standard'
Warning 1990: No 'VAR_CONFIG' for 'MAIN.bEL1004_Ch4'
Warning 1990: No 'VAR_CONFIG' for 'MAIN.nEL2008_value'
POU indices:51 (2%)
Size of used data: 45 of 1048576 bytes (0.00%)
Size of used retain data: 0 of 32768 bytes (0.00%)
0 Error(s), 2 Warning(s).
```

Fig. 29: 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…”:

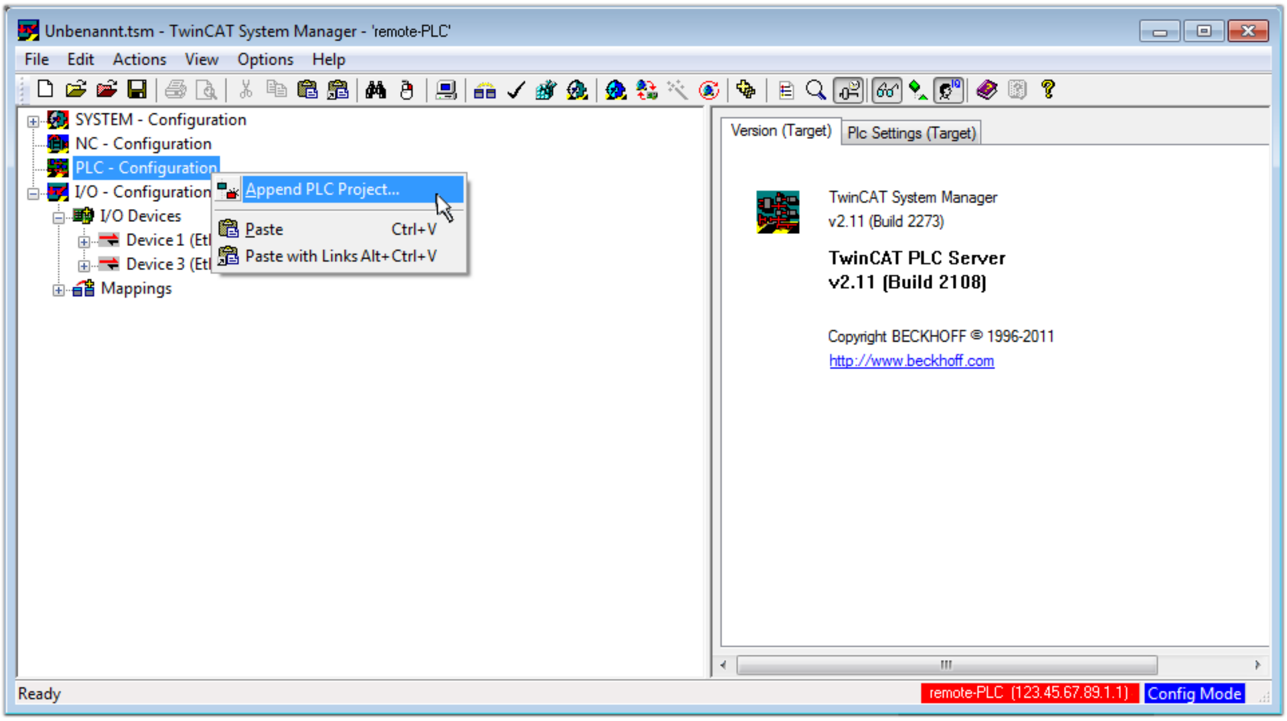


Fig. 30: Appending the TwinCAT PLC Control project

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:

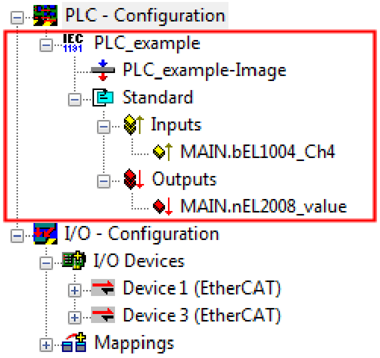
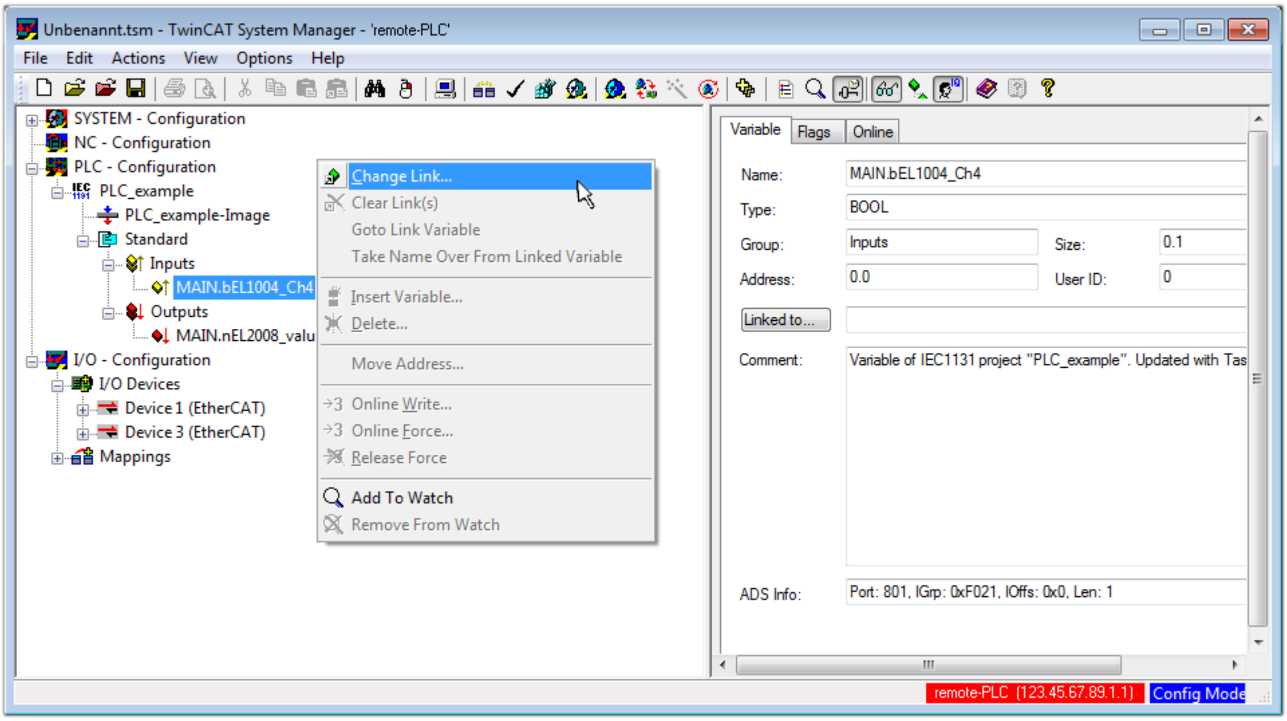


Fig. 31: PLC project integrated in the PLC configuration of the System Manager

The two variables “bEL1004_Ch4” and “nEL2008_value” can now be assigned to certain process objects of the I/O configuration.

Assigning variables

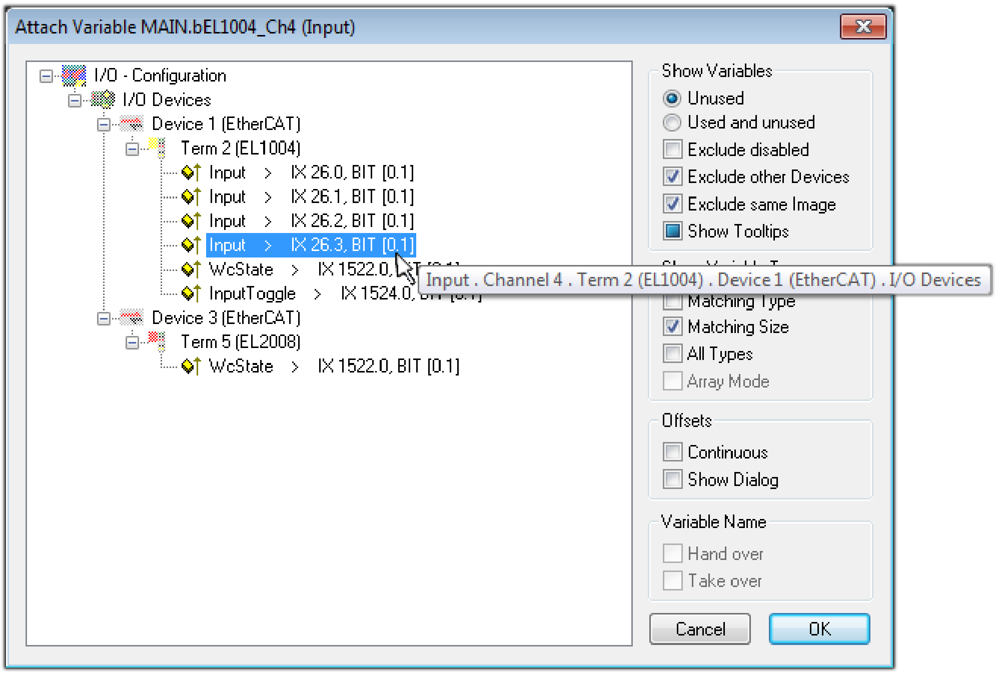
Open a window for selecting a suitable process object (PDO) via the context menu of a variable of the integrated project “PLC_example” and via “Modify Link...” “Standard”:



Name: MAIN.bEL1004_Ch4
Type: BOOL
Group: Inputs
Size: 0.1
Address: 0.0
User ID: 0
Comment: Variable of IEC1131 project "PLC_example". Updated with Tas
ADS Info: Port: 801, IGrp: 0xF021, IOffs: 0x0, Len: 1

Fig. 32: 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:



Attach Variable MAIN.bEL1004_Ch4 (Input)

- I/O - Configuration
  - I/O Devices
    - Device 1 (EtherCAT)
      - Term 2 (EL1004)
        - Input > IX 26.0, BIT [0.1]
        - Input > IX 26.1, BIT [0.1]
        - Input > IX 26.2, BIT [0.1]
        - Input > IX 26.3, BIT [0.1]
        - WcState > IX 1522.0, BIT [0.1]
        - InputToggle > IX 1524.0, BIT [0.1]
    - Device 3 (EtherCAT)
      - Term 5 (EL2008)
        - WcState > IX 1522.0, BIT [0.1]

Input . Channel 4 . Term 2 (EL1004) . Device 1 (EtherCAT) . I/O Devices

Show Variables
- ● Unused
- ○ Used and unused
- □ Exclude disabled
- ☑ Exclude other Devices
- ☑ Exclude same Image
- ■ Show Tooltips

Show Variable Types
- □ Matching Type
- ☑ Matching Size
- □ All Types
- □ Array Mode

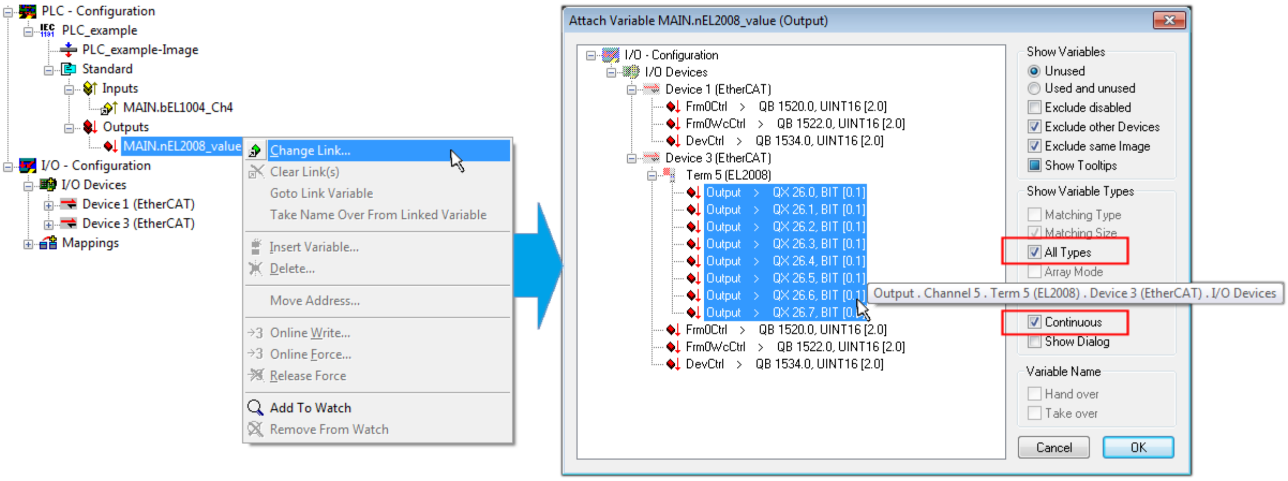
Offsets
- □ Continuous
- □ Show Dialog

Variable Name
- □ Hand over
- □ Take over

Cancel OK

Fig. 33: 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:



- PLC - Configuration
  - PLC_example
    - PLC_example-Image
    - Standard
      - Inputs
        - MAIN.bEL1004_Ch4
      - Outputs
        - MAIN.nEL2008_value
- I/O - Configuration
  - I/O Devices
    - Device 1 (EtherCAT)
    - Device 3 (EtherCAT)
  - Mappings

Attach Variable MAIN.nEL2008_value (Output)

- I/O - Configuration
  - I/O Devices
    - Device 1 (EtherCAT)
      - Frm0Ctrl > QB 1520.0, UINT16 [2.0]
      - Frm0WcCtrl > QB 1522.0, UINT16 [2.0]
      - DevCtrl > QB 1534.0, UINT16 [2.0]
    - Device 3 (EtherCAT)
      - Term 5 (EL2008)
        - Output > QX 26.0, BIT [0.1]
        - Output > QX 26.1, BIT [0.1]
        - Output > QX 26.2, BIT [0.1]
        - Output > QX 26.3, BIT [0.1]
        - Output > QX 26.4, BIT [0.1]
        - Output > QX 26.5, BIT [0.1]
        - Output > QX 26.6, BIT [0.1]
        - Output > QX 26.7, BIT [0.1]
      - Frm0Ctrl > QB 1520.0, UINT16 [2.0]
      - Frm0WcCtrl > QB 1522.0, UINT16 [2.0]
      - DevCtrl > QB 1534.0, UINT16 [2.0]

Output . Channel 5 . Term 5 (EL2008) . Device 3 (EtherCAT) . I/O Devices

Show Variables
- ● Unused
- ○ Used and unused
- □ Exclude disabled
- ☑ Exclude other Devices
- ☑ Exclude same Image
- ■ Show Tooltips

Show Variable Types
- □ Matching Type
- ☑ Matching Size
- ☑ All Types
- □ Array Mode
- ☑ Continuous
- □ Show Dialog

Variable Name
- □ Hand over
- □ Take over

Cancel OK

Fig. 34: 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:

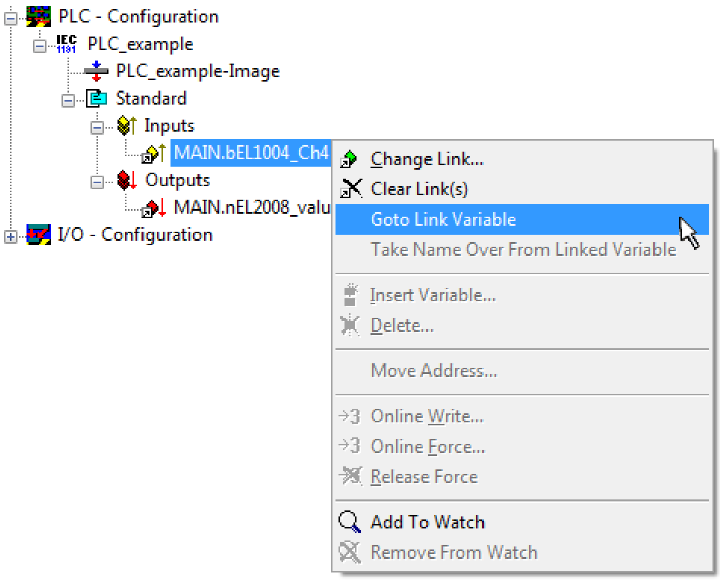
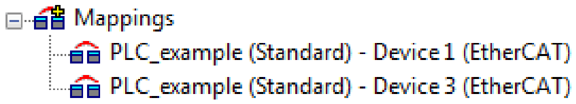


Fig. 35: Application of a “Goto Link” variable, using “MAIN.bEL1004_Ch4” as a sample

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:


Mappings
PLC_example (Standard) - Device 1 (EtherCAT)
PLC_example (Standard) - Device 3 (EtherCAT)

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 RTime 0% is displayed at the bottom right in the System Manager. The PLC system can then be started as described below.

Starting the controller

Starting from a remote system, the PLC control has to be linked with the Embedded PC over Ethernet via “Online” → “Choose Run-Time System…”:

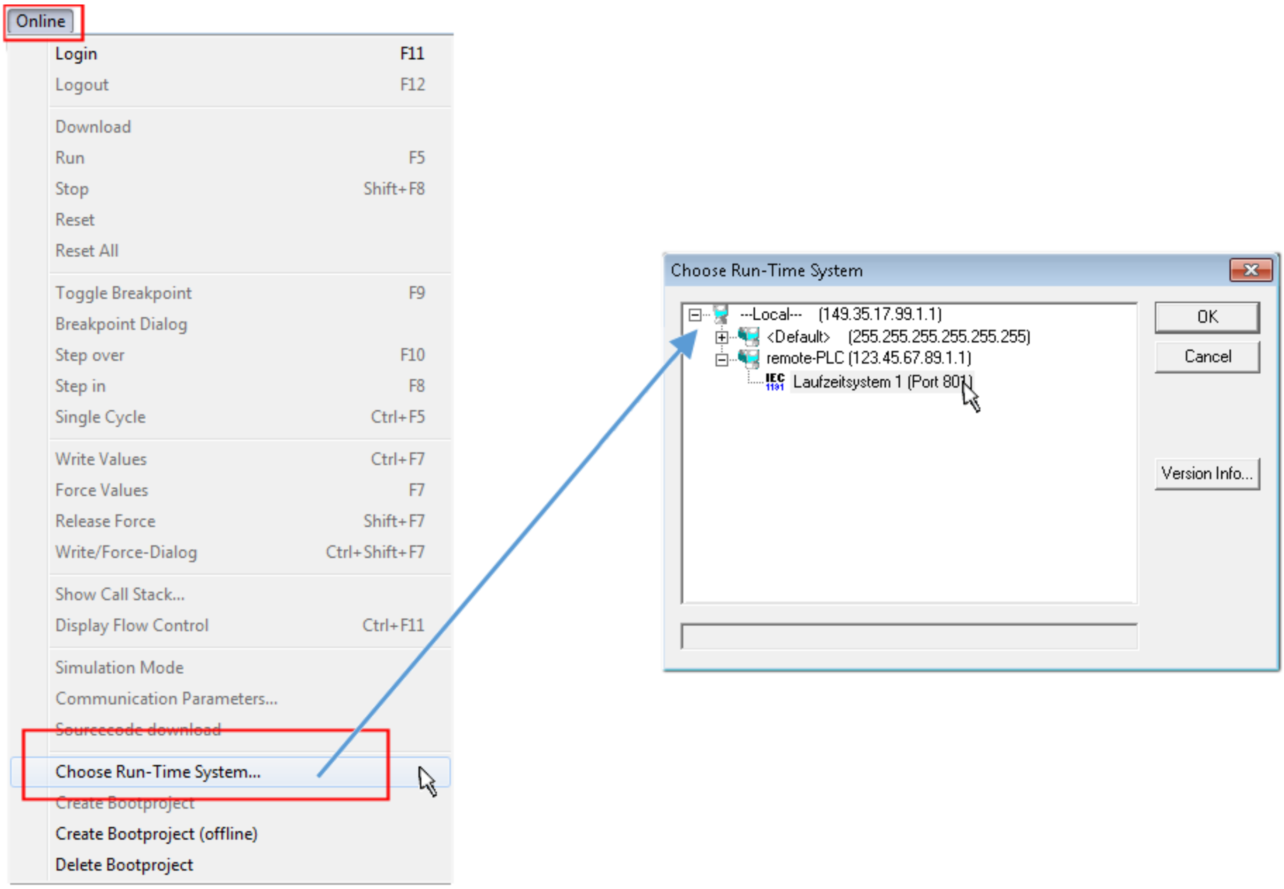
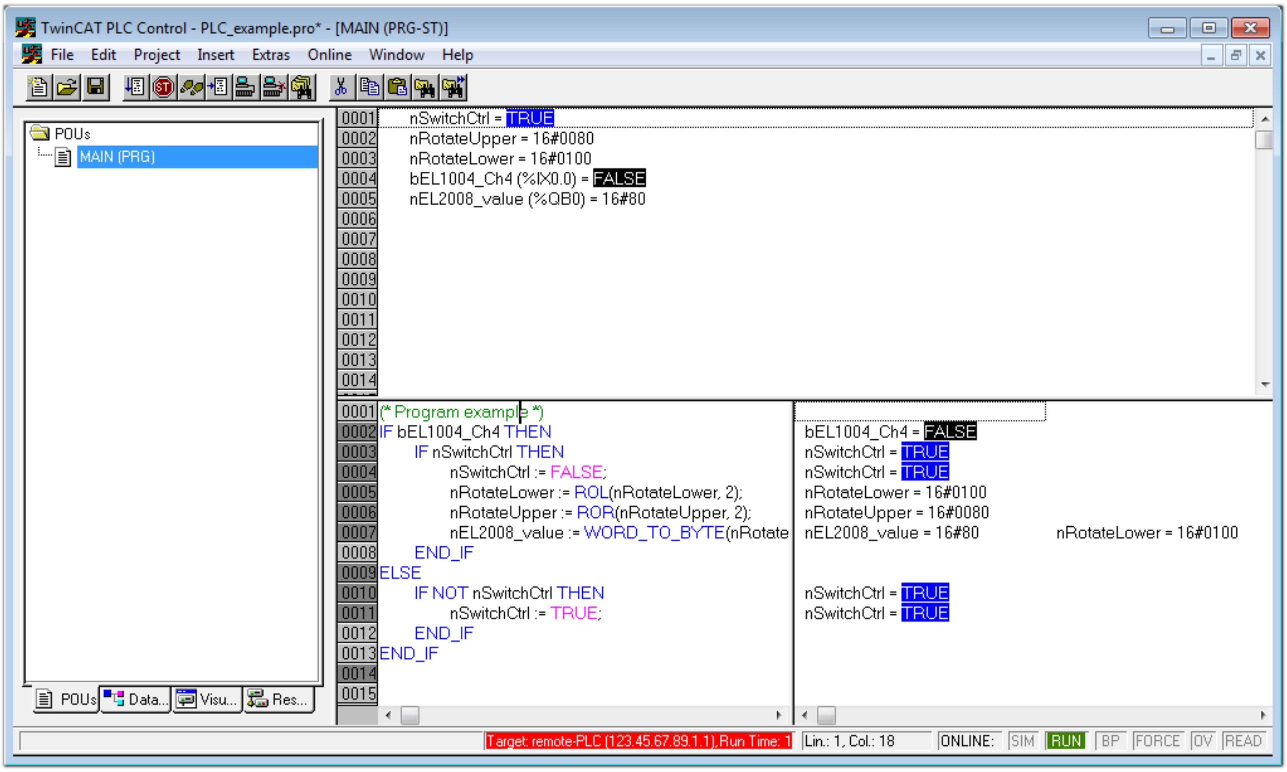


Fig. 36: Choose target system (remote)

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:


TwinCAT PLC Control - PLC_example.pro* - [MAIN (PRG-ST)]
File Edit Project Insert Extras Online Window Help
POUs
MAIN (PRG)
POUs Data... Visu... Res...
Lin.: 1, Col.: 18
ONLINE: SIM RUN BP FORCE OV READ

```
nSwitchCtrl = TRUE
nRotateUpper = 16#0080
nRotateLower = 16#0100
bEL1004_Ch4 (%IX0.0) = FALSE
nEL2008_value (%QB0) = 16#80
```

```
(* Program example *)
IF bEL1004_Ch4 THEN
    IF nSwitchCtrl THEN
        nSwitchCtrl := FALSE;
        nRotateLower := ROL(nRotateLower, 2);
        nRotateUpper := ROR(nRotateUpper, 2);
        nEL2008_value := WORD_TO_BYTE(nRotate
    END_IF
ELSE
    IF NOT nSwitchCtrl THEN
        nSwitchCtrl := TRUE;
    END_IF
END_IF
```

```
bEL1004_Ch4 = FALSE
nSwitchCtrl = TRUE
nSwitchCtrl = TRUE
nRotateLower = 16#0100
nRotateUpper = 16#0080
nEL2008_value = 16#80          nRotateLower = 16#0100

nSwitchCtrl = TRUE
nSwitchCtrl = TRUE
```

Target: remote-PLC (123.45.67.89.1.1),Run Time: 1

Fig. 37: PLC Control logged in, ready for program startup

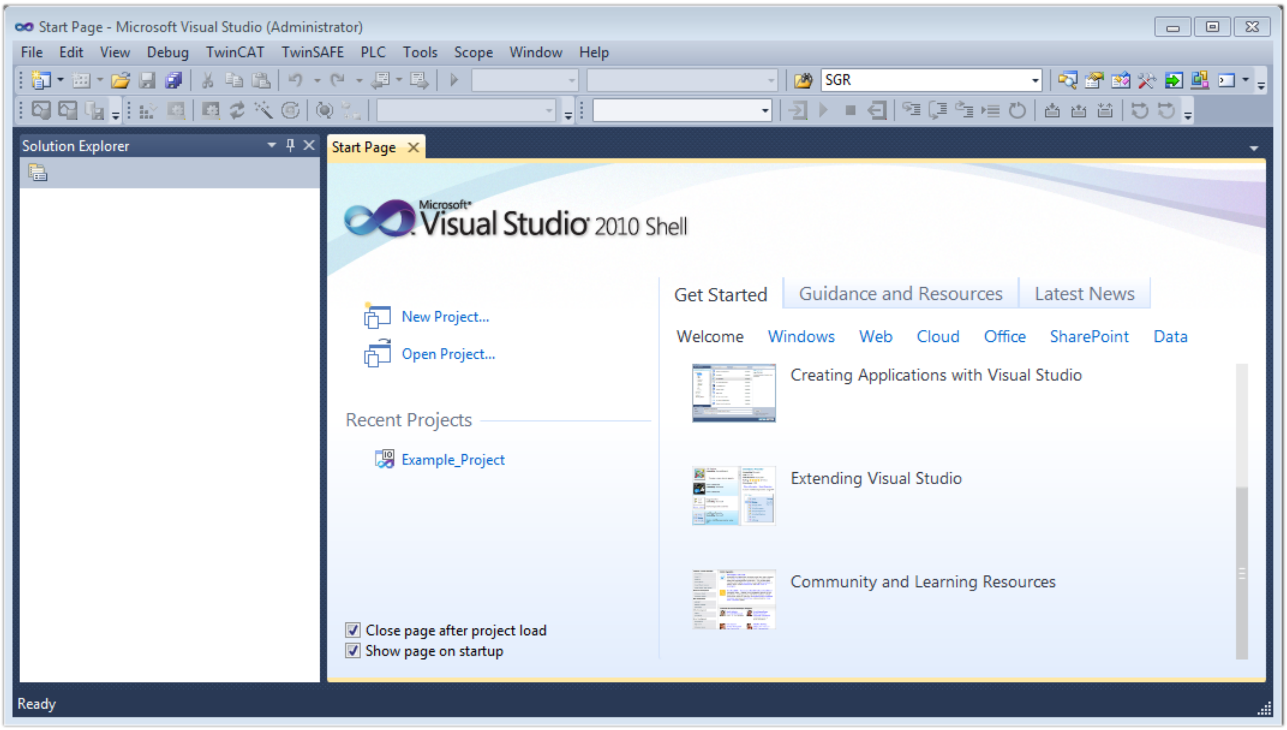
The PLC can now be started via “Online” → “Run”, F5 key or .

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.

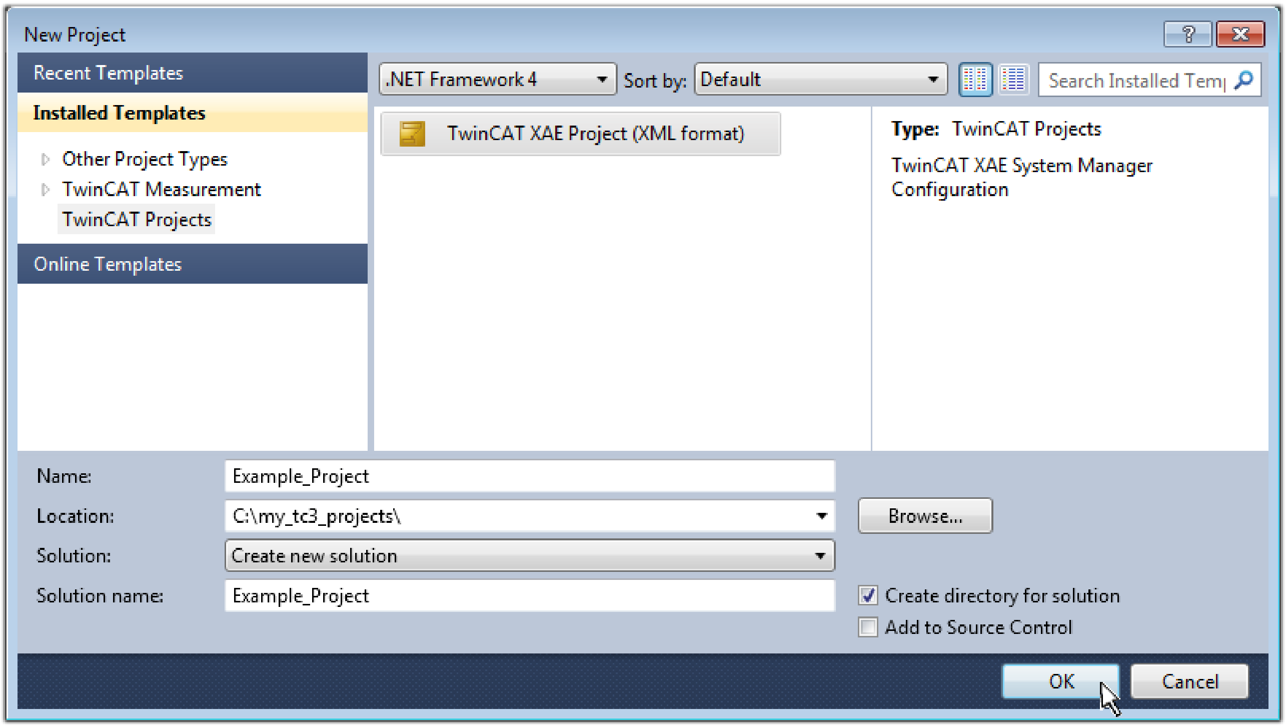
After successful installation of the TwinCAT system on the PC to be used for development, TwinCAT 3 (shell) displays the following user interface after startup:


Start Page - Microsoft Visual Studio (Administrator)
File Edit View Debug TwinCAT TwinSAFE PLC Tools Scope Window Help
SGR
Solution Explorer
Start Page
Microsoft Visual Studio 2010 Shell
New Project...
Open Project...
Recent Projects
Example_Project
Get Started
Guidance and Resources
Latest News
Welcome Windows Web Cloud Office SharePoint Data
Creating Applications with Visual Studio
Extending Visual Studio
Community and Learning Resources
Ready

☑ Close page after project load
☑ Show page on startup

Fig. 38: 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):



.NET Framework 4 Sort by: Default

TwinCAT XAE Project (XML format)

Type: TwinCAT Projects

TwinCAT XAE System Manager Configuration

Name: Example_Project

Location: C:\my_tc3_projects\

Solution: Create new solution

Solution name: Example_Project

☑ Create directory for solution

☐ Add to Source Control

Fig. 39: Create new TwinCAT project

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

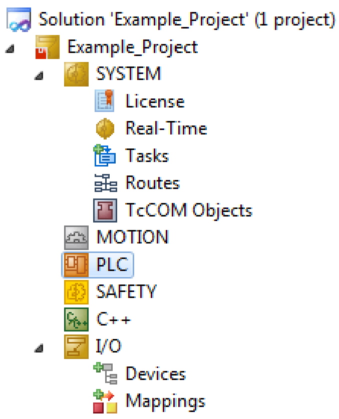
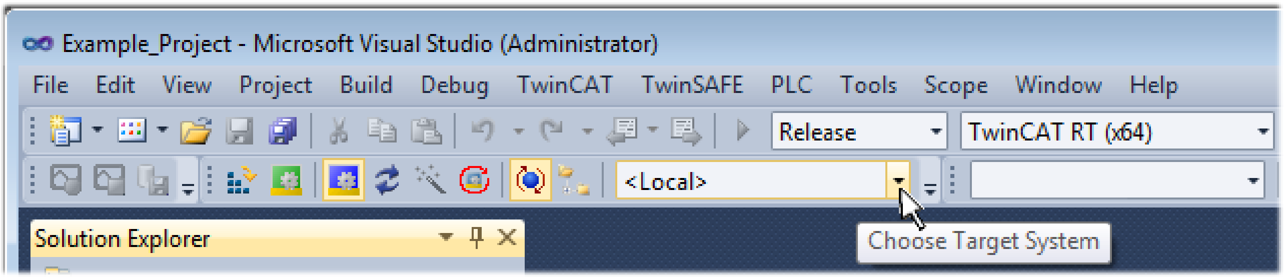


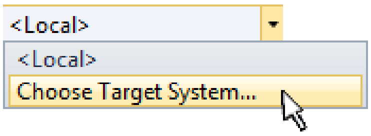
Fig. 40: New TwinCAT3 project in the project folder explorer

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 [▶ 52]".

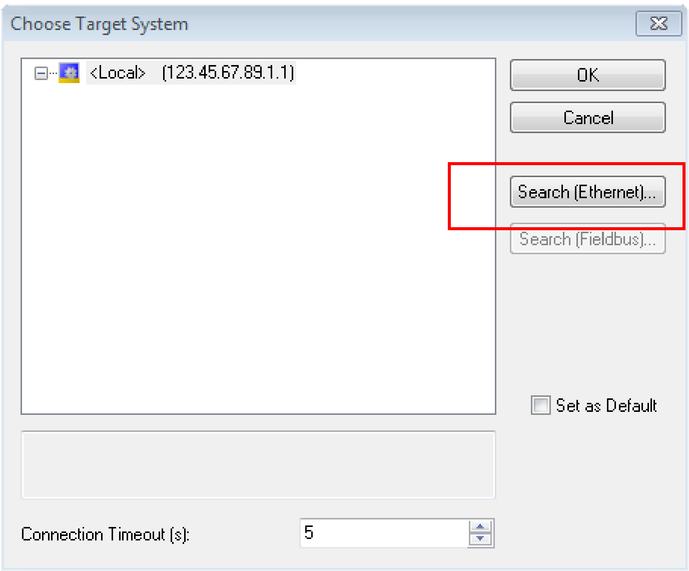
If the intention is to address the TwinCAT runtime environment installed on a PLC as development environment remotely from another system, the target system must be made known first. Via the symbol in the menu bar:


Example_Project - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug TwinCAT TwinSAFE PLC Tools Scope Window Help
Release
TwinCAT RT (x64)
<Local>
Solution Explorer
Choose Target System

expand the pull-down menu:



and open the following window:

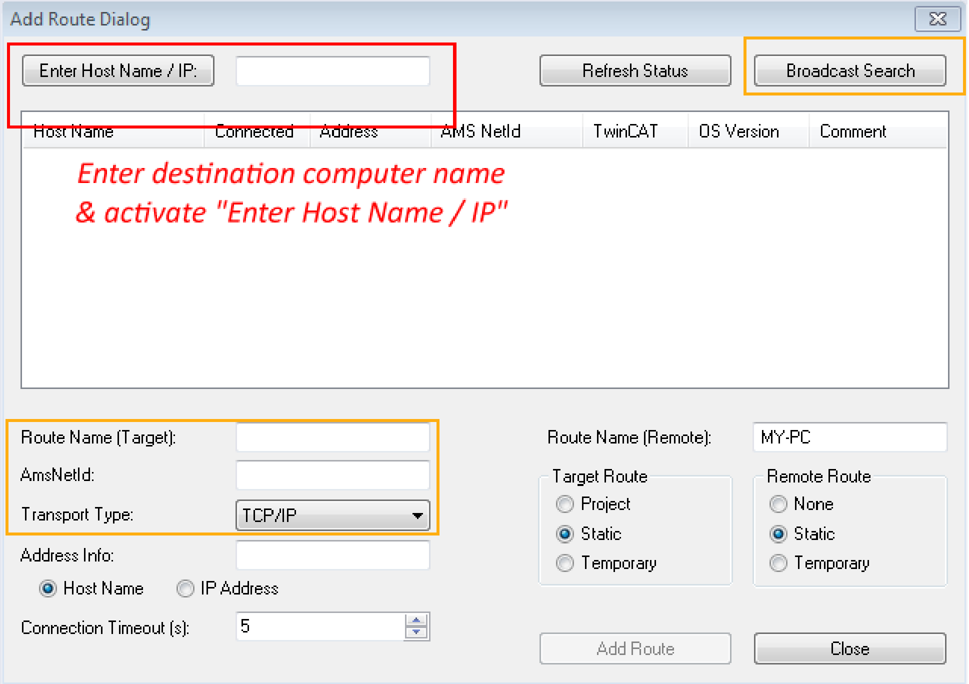


Connection Timeout (s): 5

Fig. 41: 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.



Enter destination computer name & activate "Enter Host Name / IP"

Route Name (Target):

AmsNetId:

Transport Type: TCP/IP

Address Info:

● Host Name ○ IP Address

Connection Timeout (s): 5

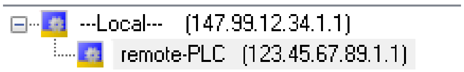
Route Name (Remote): MY-PC

Target Route: ○ Project ● Static ○ Temporary

Remote Route: ○ None ● Static ○ Temporary

Fig. 42: 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):



```
---Local---   (147.99.12.34.1.1)
    remote-PLC   (123.45.67.89.1.1)
```

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)”.

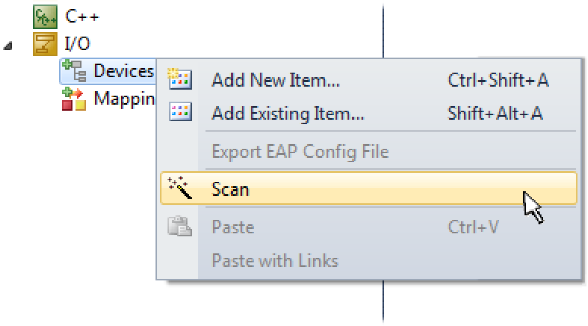
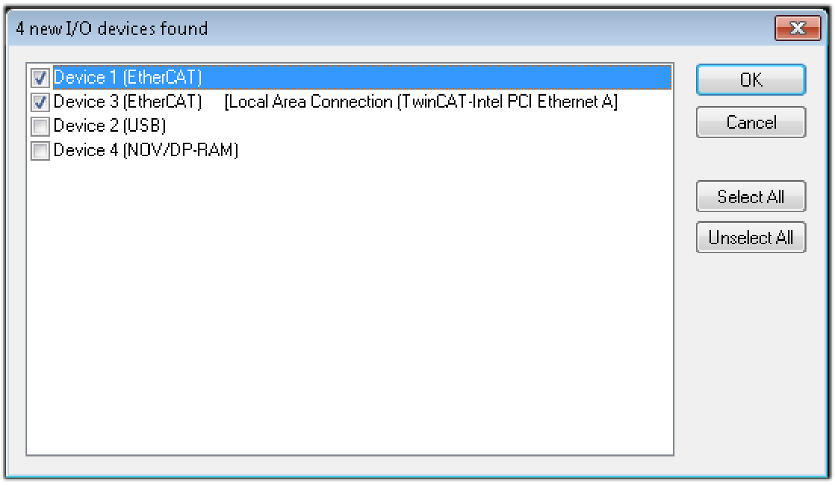


Fig. 43: Select “Scan”

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



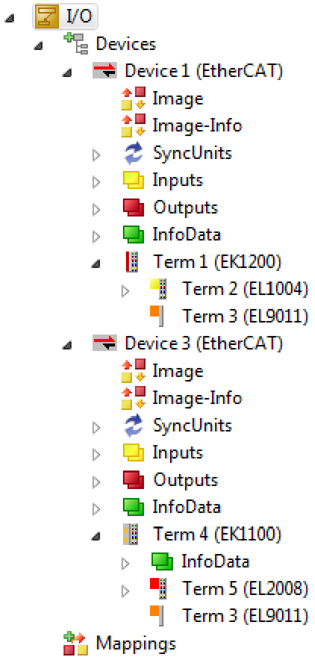
4 new I/O devices found

☑ Device 1 (EtherCAT)
☑ Device 3 (EtherCAT)   [Local Area Connection (TwinCAT-Intel PCI Ethernet A]
□ Device 2 (USB)
□ Device 4 (NOV/DP-RAM)

Fig. 44: 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 [▶ 37] described at the beginning of this section, the result is as follows:



```
I/O
  Devices
    Device 1 (EtherCAT)
        Image
        Image-Info
        SyncUnits
        Inputs
        Outputs
        InfoData
        Term 1 (EK1200)
          Term 2 (EL1004)
          Term 3 (EL9011)
    Device 3 (EtherCAT)
        Image
        Image-Info
        SyncUnits
        Inputs
        Outputs
        InfoData
        Term 4 (EK1100)
          InfoData
          Term 5 (EL2008)
          Term 3 (EL9011)
  Mappings
```

Fig. 45: 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:

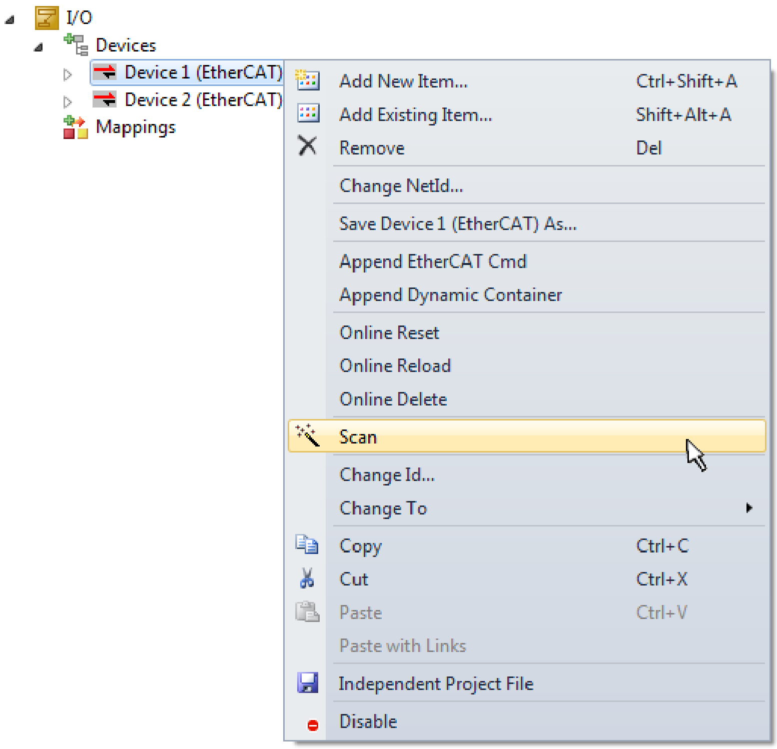


Fig. 46: Reading of individual terminals connected to a device

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….”:

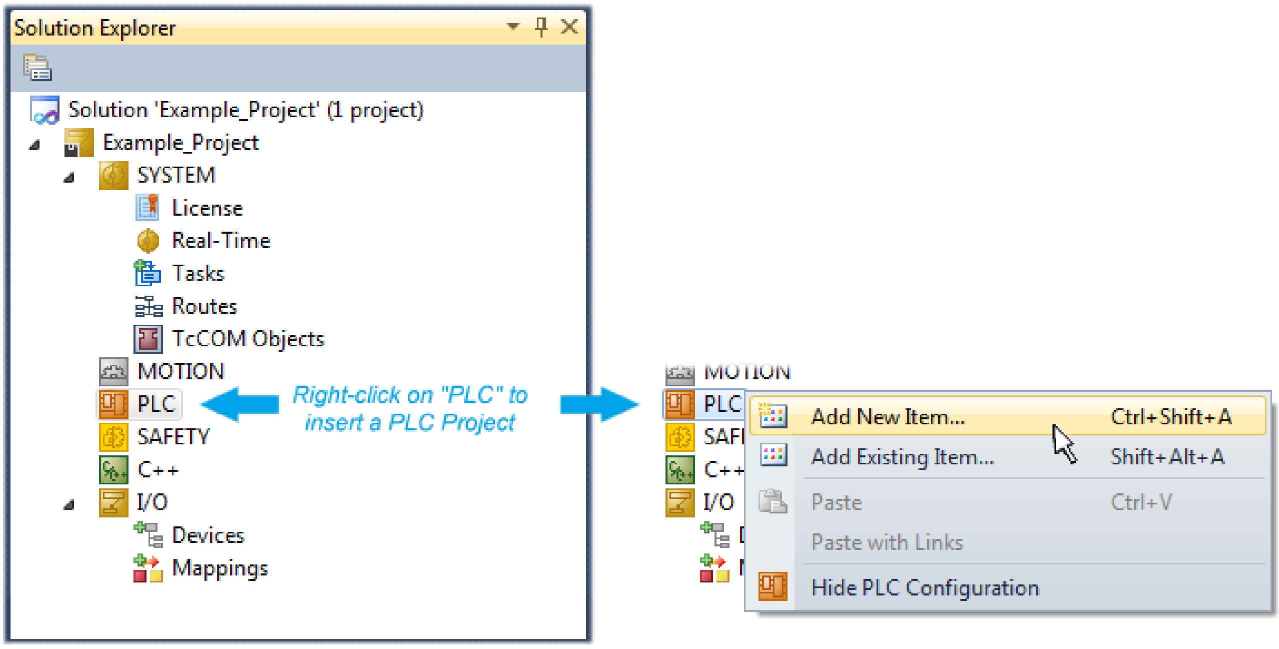
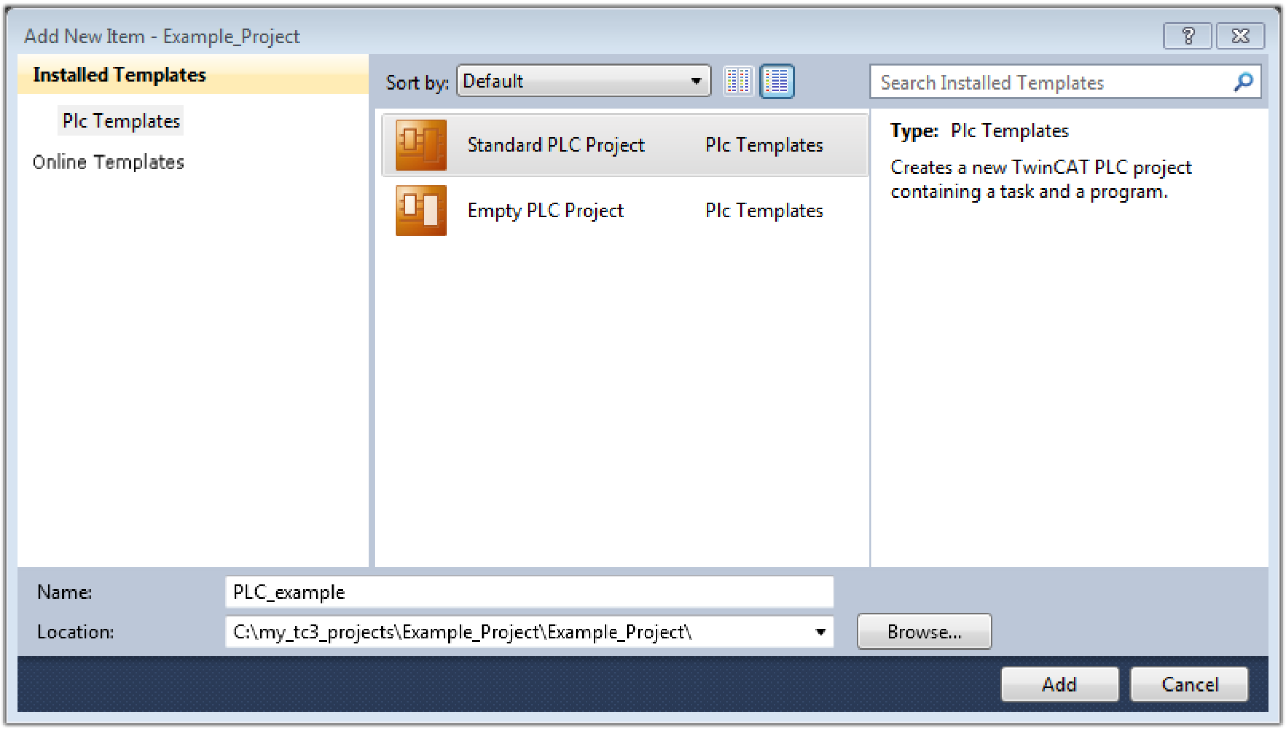


Fig. 47: Adding the programming environment in “PLC”

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


Add New Item - Example_Project
Installed Templates
Plc Templates
Online Templates
Sort by: Default
Search Installed Templates
Standard PLC Project Plc Templates
Empty PLC Project Plc Templates
Browse...
Add
Cancel

Type: Plc Templates

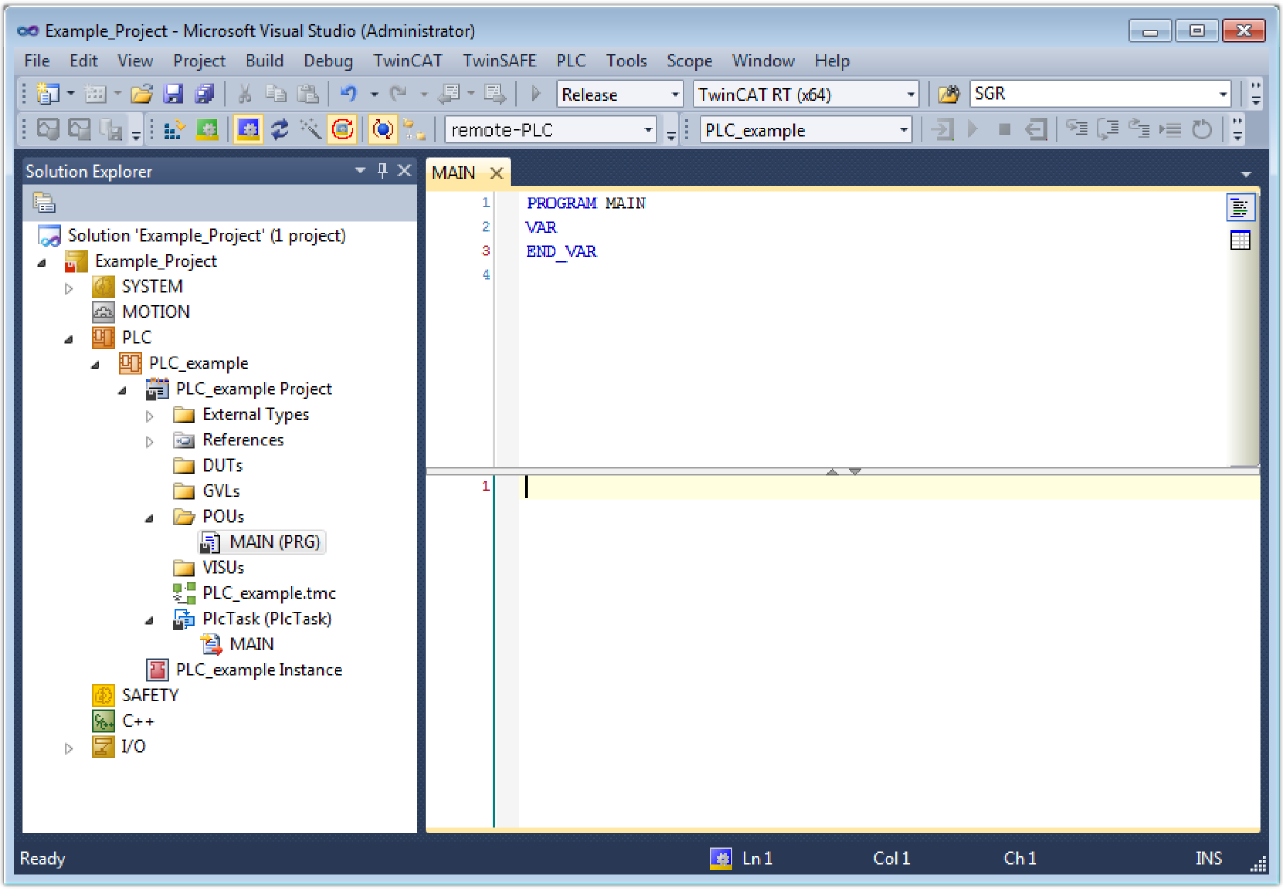
Creates a new TwinCAT PLC project containing a task and a program.

Name: PLC_example

Location: C:\my_tc3_projects\Example_Project\Example_Project\

Fig. 48: Specifying the name and directory for the PLC programming environment

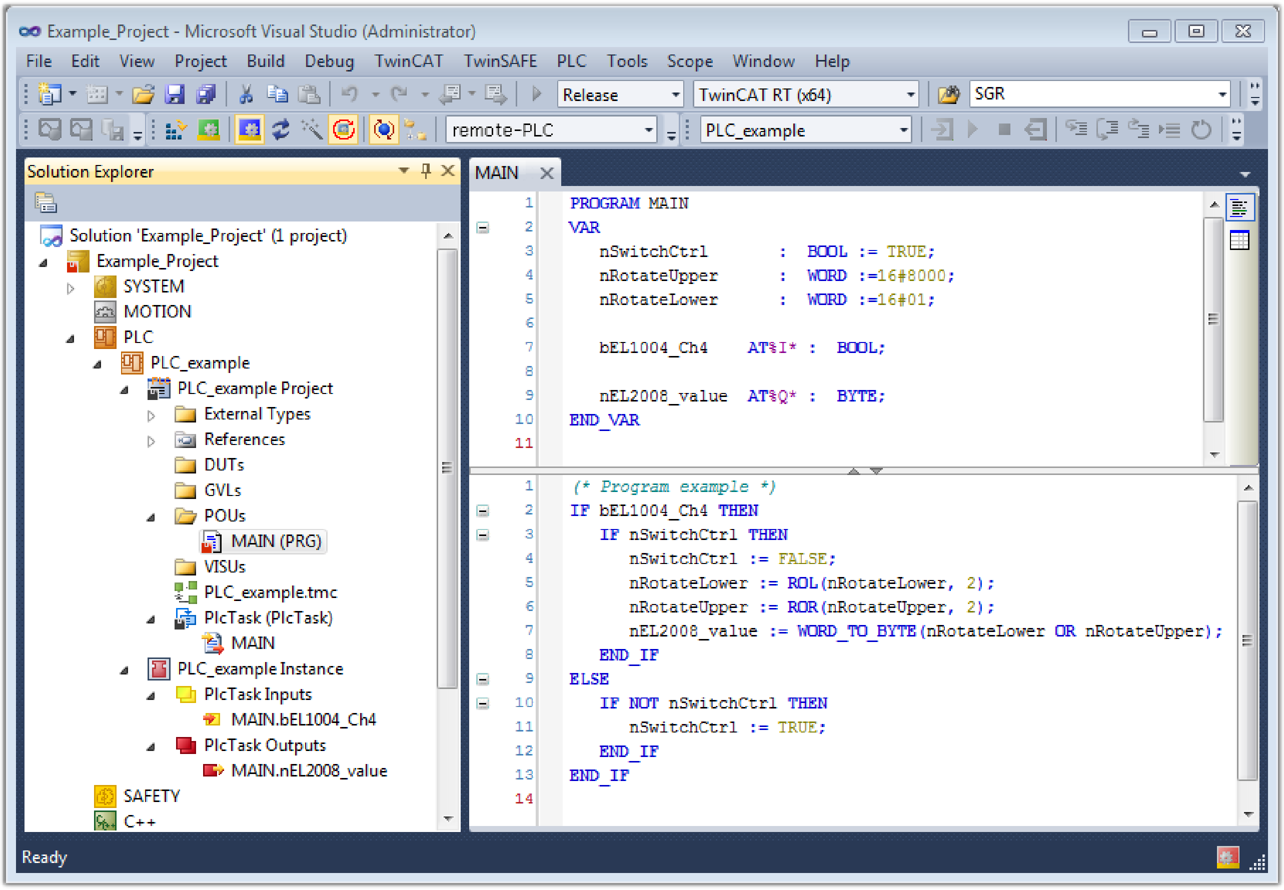
The “Main” program, which already exists by selecting “Standard PLC project”, can be opened by double-clicking on “PLC_example_project” in “POUs”. The following user interface is shown for an initial project:



```
PROGRAM MAIN
VAR
END_VAR
```

Fig. 49: Initial “Main” program of the standard PLC project

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



```
PROGRAM MAIN
VAR
    nSwitchCtrl       :  BOOL := TRUE;
    nRotateUpper      :  WORD :=16#8000;
    nRotateLower      :  WORD :=16#01;

    bEL1004_Ch4    AT%I* :  BOOL;

    nEL2008_value  AT%Q* :  BYTE;
END_VAR
```

```
(* Program example *)
IF bEL1004_Ch4 THEN
    IF nSwitchCtrl THEN
        nSwitchCtrl  := FALSE;
        nRotateLower := ROL(nRotateLower, 2);
        nRotateUpper := ROR(nRotateUpper, 2);
        nEL2008_value := WORD_TO_BYTE(nRotateLower OR nRotateUpper);
    END_IF
ELSE
    IF NOT nSwitchCtrl THEN
        nSwitchCtrl  := TRUE;
    END_IF
END_IF
```

Fig. 50: 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:

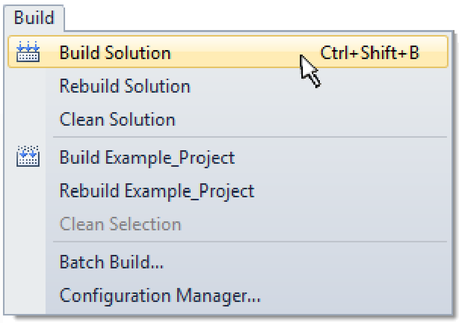
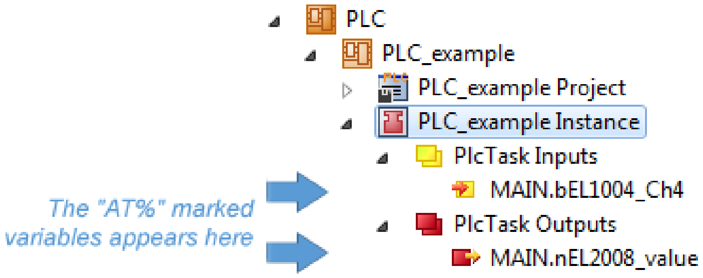


Fig. 51: Start program compilation

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:

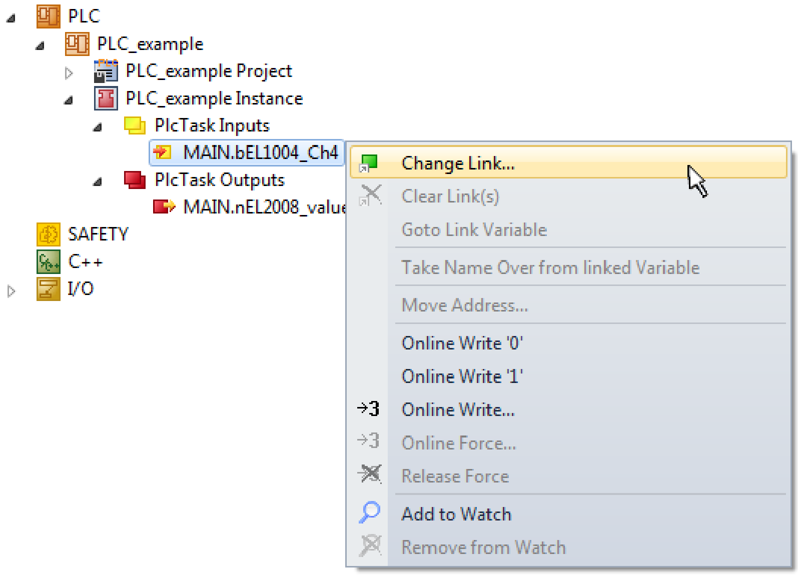


Fig. 52: Creating the links between PLC variables and process objects

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

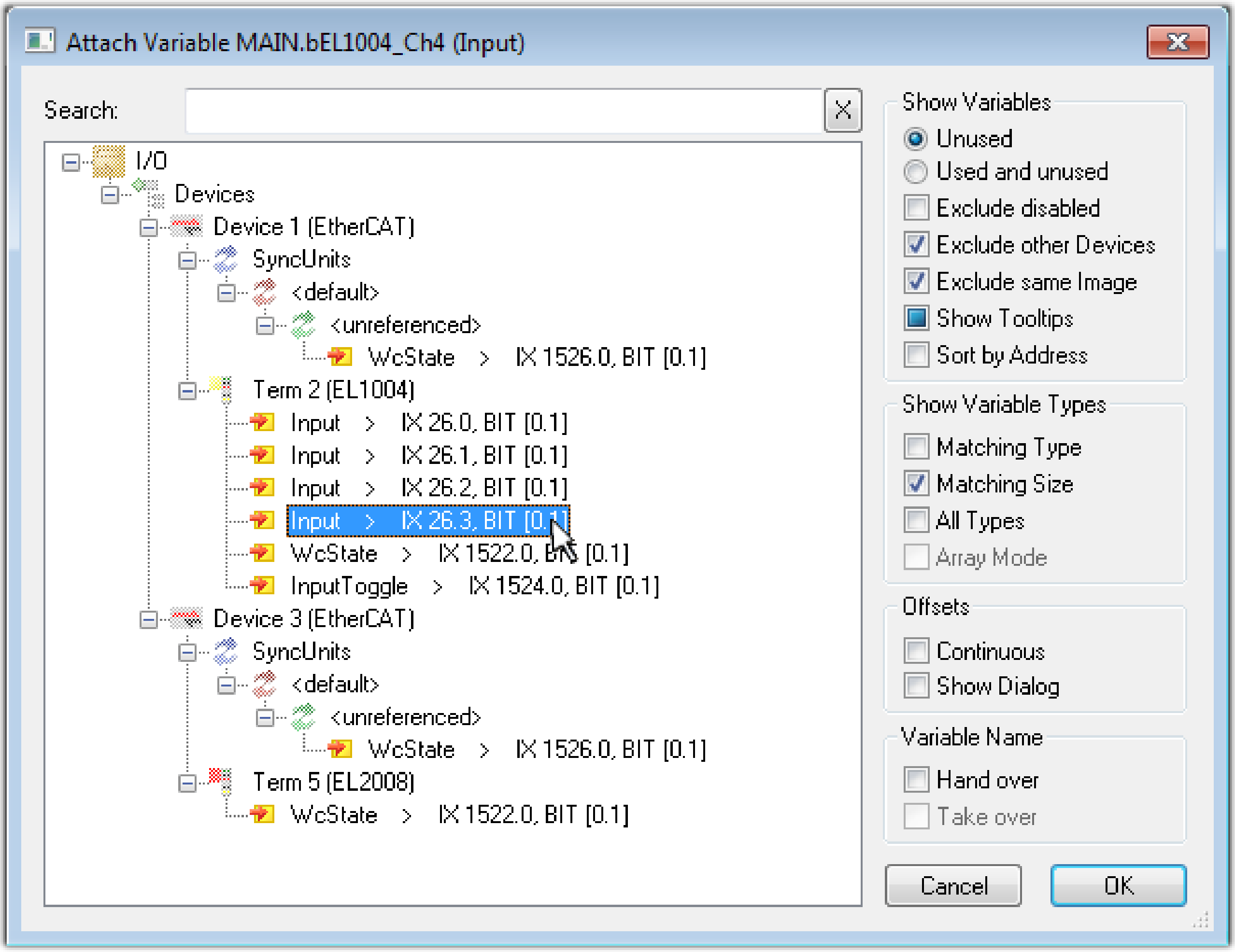
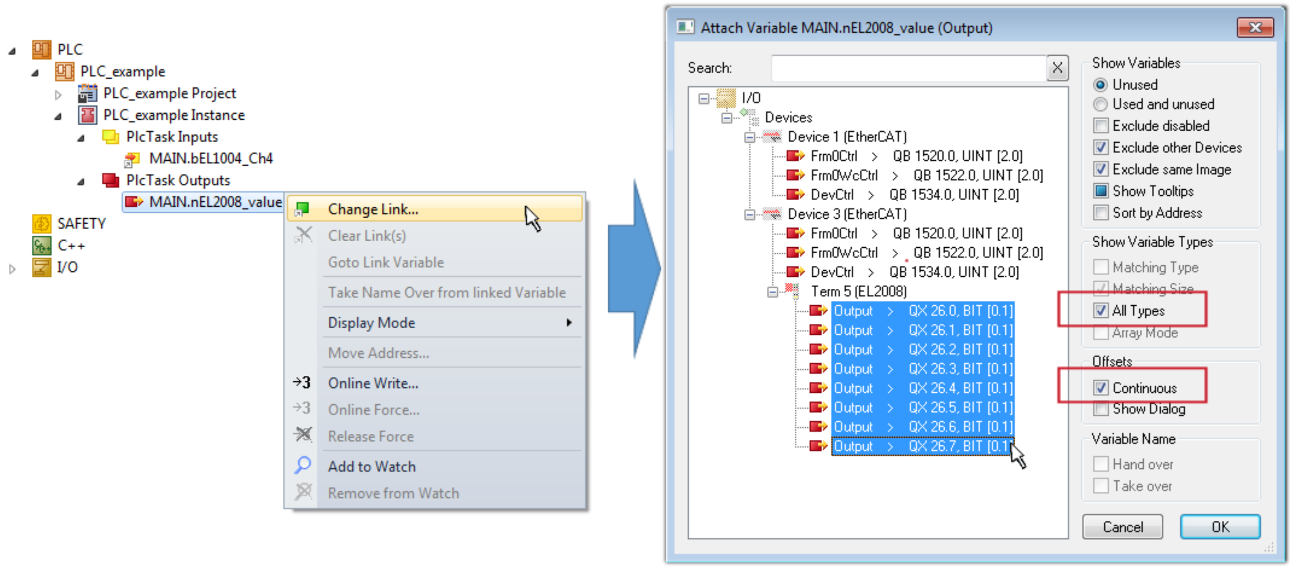


Fig. 53: Selecting PDO of type BOOL

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:


PLC
PLC_example
PLC_example Project
PLC_example Instance
PlcTask Inputs
MAIN.bEL1004_Ch4
PlcTask Outputs
MAIN.nEL2008_value
SAFETY
C++
I/O
Change Link...
Clear Link(s)
Goto Link Variable
Take Name Over from linked Variable
Display Mode
Move Address...
Online Write...
Online Force...
Release Force
Add to Watch
Remove from Watch
Attach Variable MAIN.nEL2008_value (Output)
Search:
I/O
Devices
Device 1 (EtherCAT)
Frm0Ctrl > QB 1520.0, UINT [2.0]
Frm0WcCtrl > QB 1522.0, UINT [2.0]
DevCtrl > QB 1534.0, UINT [2.0]
Device 3 (EtherCAT)
Frm0Ctrl > QB 1520.0, UINT [2.0]
Frm0WcCtrl > QB 1522.0, UINT [2.0]
DevCtrl > QB 1534.0, UINT [2.0]
Term 5 (EL2008)
Output > QX 26.0, BIT [0.1]
Output > QX 26.1, BIT [0.1]
Output > QX 26.2, BIT [0.1]
Output > QX 26.3, BIT [0.1]
Output > QX 26.4, BIT [0.1]
Output > QX 26.5, BIT [0.1]
Output > QX 26.6, BIT [0.1]
Output > QX 26.7, BIT [0.1]
Cancel
OK

Show Variables
● Unused
○ Used and unused
□ Exclude disabled
☑ Exclude other Devices
☑ Exclude same Image
■ Show Tooltips
□ Sort by Address

Show Variable Types
□ Matching Type
☑ Matching Size
☑ All Types
□ Array Mode

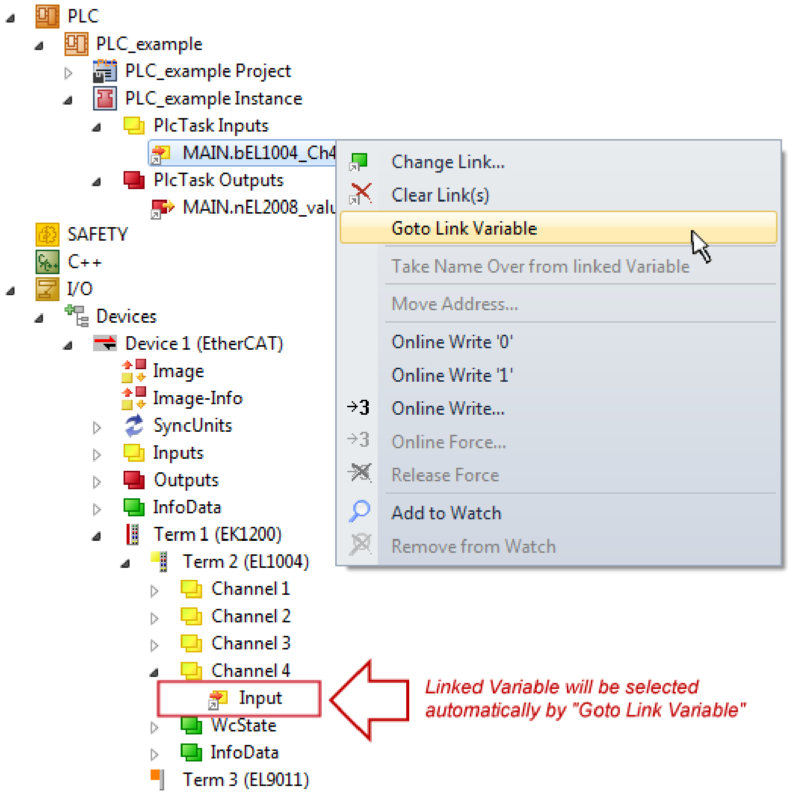
Offsets
☑ Continuous
□ Show Dialog

Variable Name
□ Hand over
□ Take over

Fig. 54: 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:



Linked Variable will be selected automatically by "Goto Link Variable"

Fig. 55: 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.

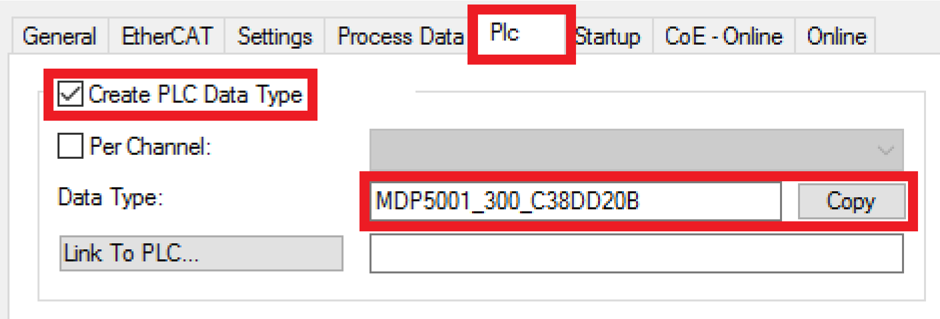
Note on the type of variable assignment

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.



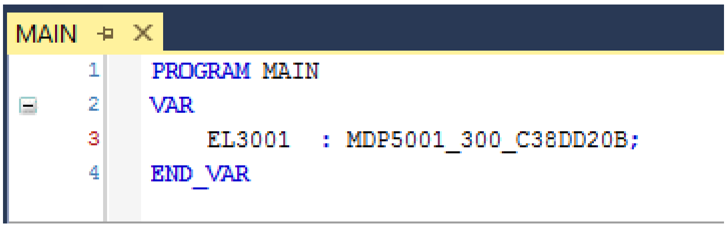
☑ Create PLC Data Type

□ Per Channel:

Data Type: MDP5001_300_C38DD20B

Fig. 56: Creating a PLC data type

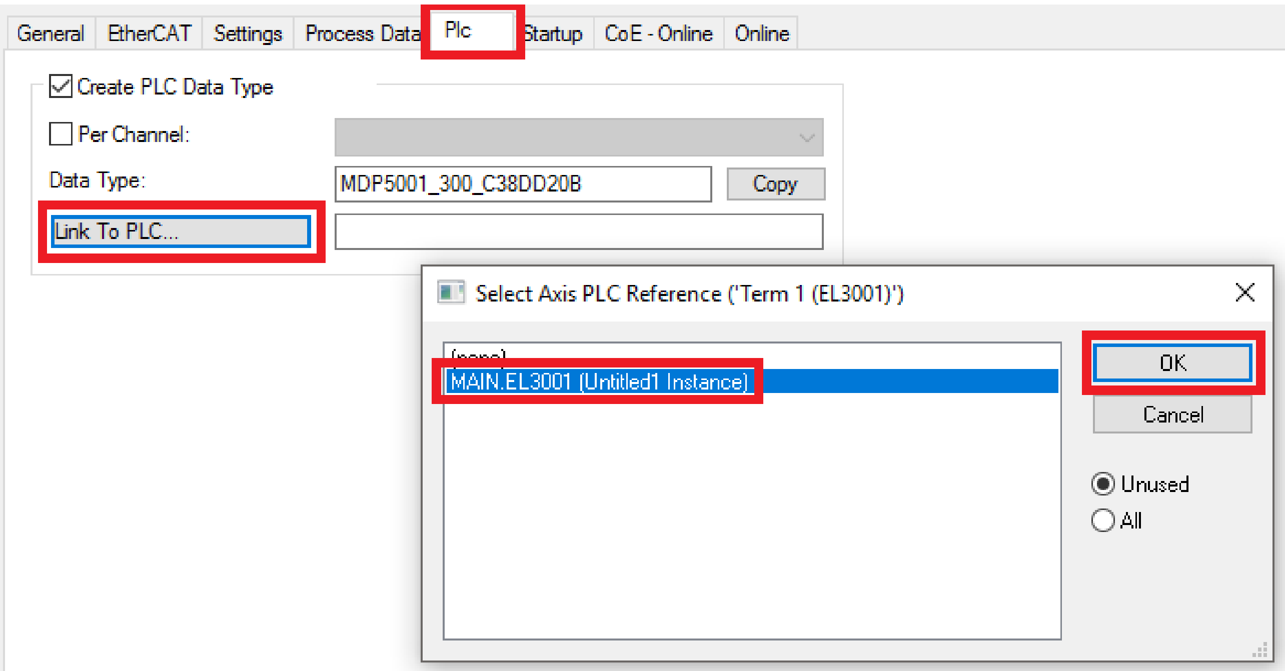
4. An instance of the data structure of the copied data type must then be created in the PLC.



```
PROGRAM MAIN
VAR
    EL3001   : MDP5001_300_C38DD20B;
END_VAR
```

Fig. 57: 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.



☑ Create PLC Data Type

□ Per Channel:

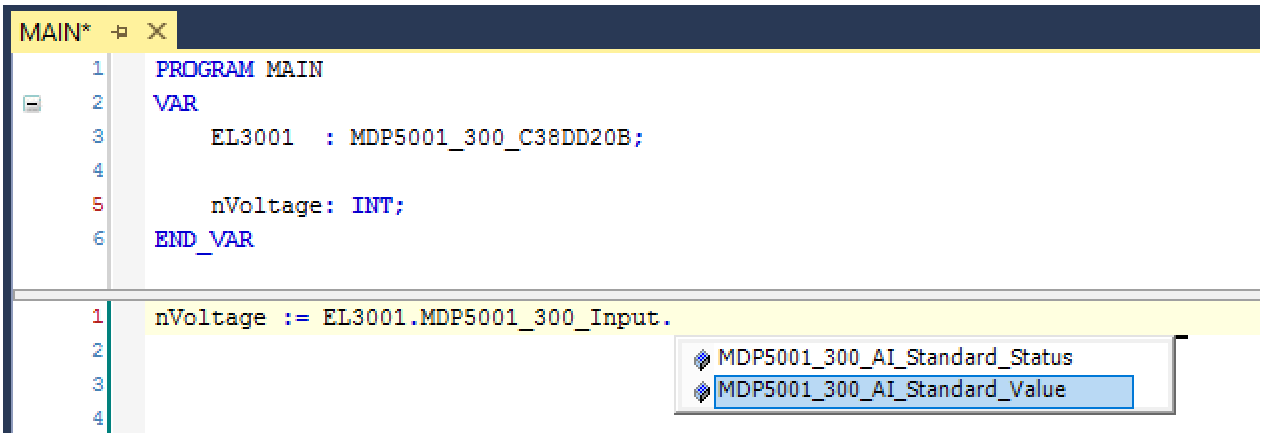
Data Type: MDP5001_300_C38DD20B

(none)
MAIN.EL3001 (Untitled1 Instance)

● Unused
○ All

Fig. 58: Linking the structure

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



```
PROGRAM MAIN
VAR
    EL3001  : MDP5001_300_C38DD20B;

    nVoltage: INT;
END_VAR
```

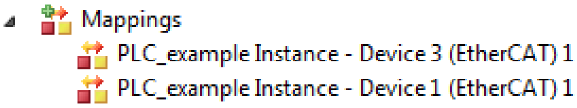
```
nVoltage := EL3001.MDP5001_300_Input.
```

MDP5001_300_AI_Standard_Status
MDP5001_300_AI_Standard_Value

Fig. 59: 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:

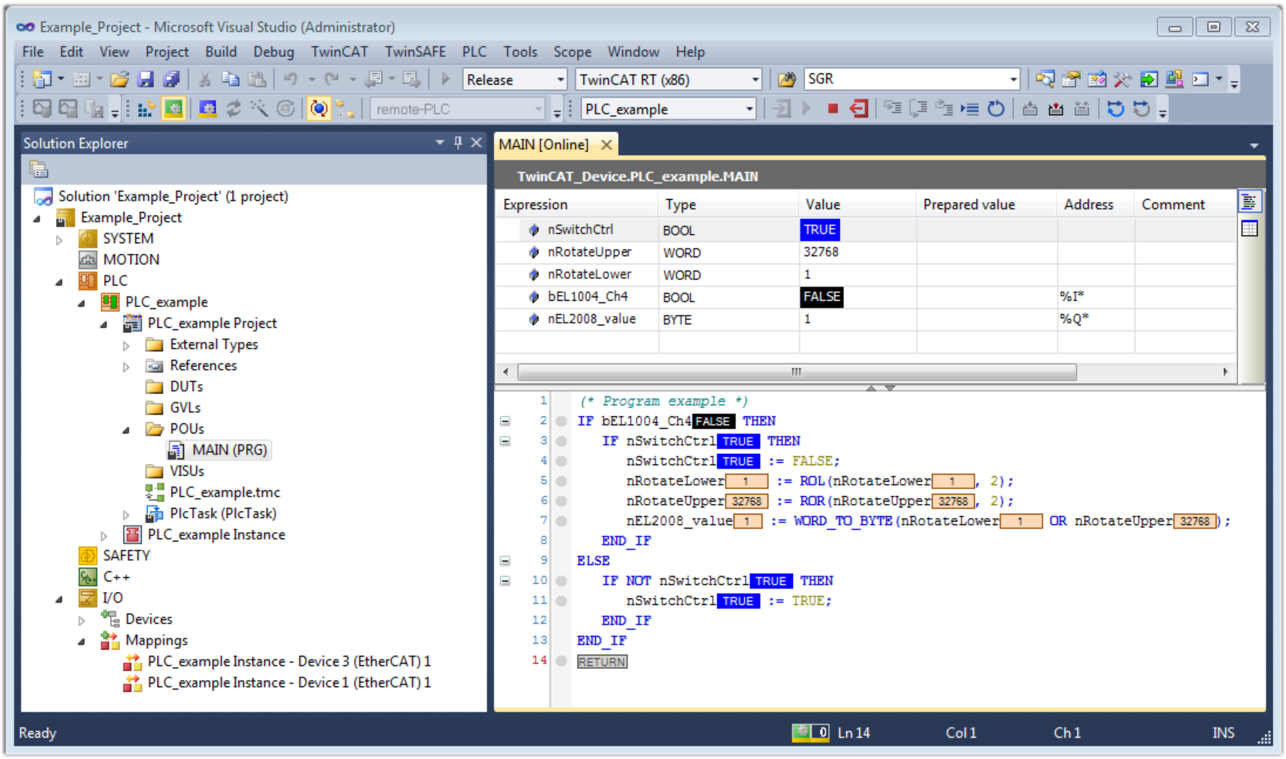


Mappings
PLC_example Instance - Device 3 (EtherCAT) 1
PLC_example Instance - Device 1 (EtherCAT) 1

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:



TwinCAT_Device.PLC_example.MAIN

Expression	Type	Value	Prepared value	Address	Comment
nSwitchCtrl	BOOL	TRUE			
nRotateUpper	WORD	32768			
nRotateLower	WORD	1			
bEL1004_Ch4	BOOL	FALSE		%I*	
nEL2008_value	BYTE	1		%Q*	

```
(* Program example *)
IF bEL1004_Ch4 FALSE THEN
   IF nSwitchCtrl TRUE THEN
      nSwitchCtrl TRUE := FALSE;
      nRotateLower 1 := ROL(nRotateLower 1 , 2);
      nRotateUpper 32768 := ROR(nRotateUpper 32768 , 2);
      nEL2008_value 1 := WORD_TO_BYTE(nRotateLower 1 OR nRotateUpper 32768 );
   END_IF
ELSE
   IF NOT nSwitchCtrl TRUE THEN
      nSwitchCtrl TRUE := TRUE;
   END_IF
END_IF
RETURN
```

Fig. 60: 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.

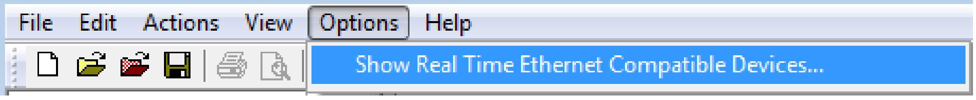


Fig. 61: System Manager “Options” (TwinCAT 2)

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

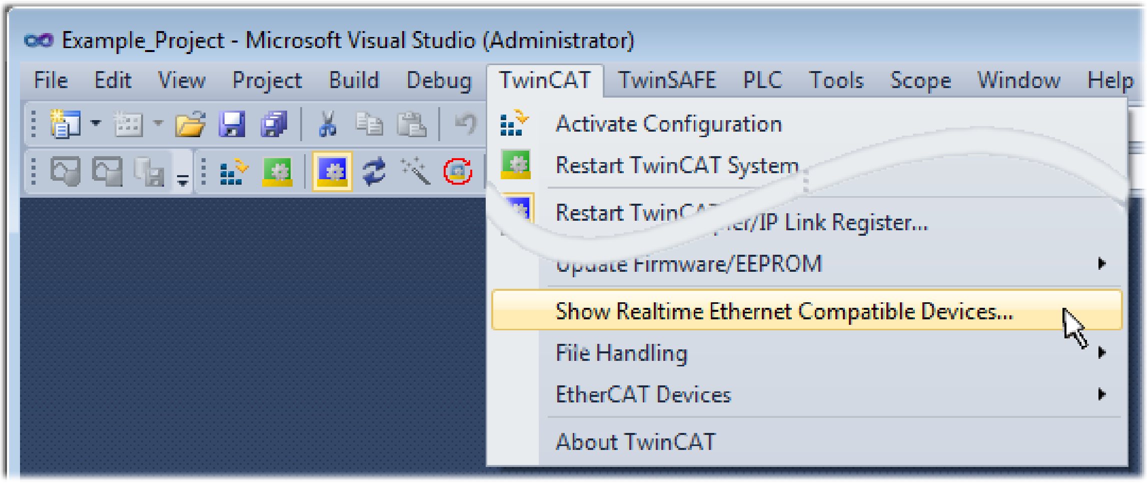
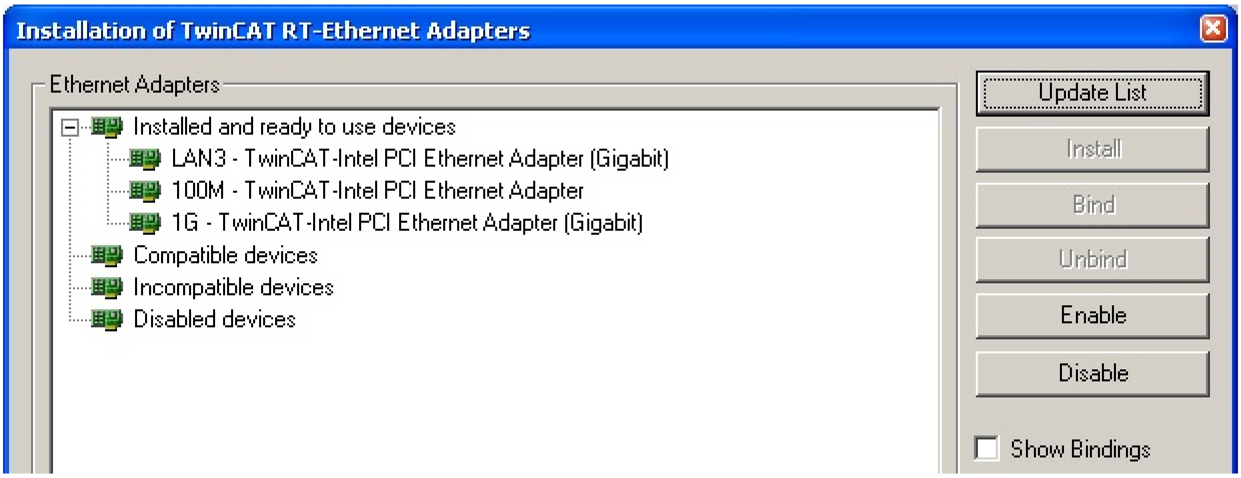


Fig. 62: Call up under VS Shell (TwinCAT 3)

The following dialog appears:



Installation of TwinCAT RT-Ethernet Adapters

Ethernet Adapters

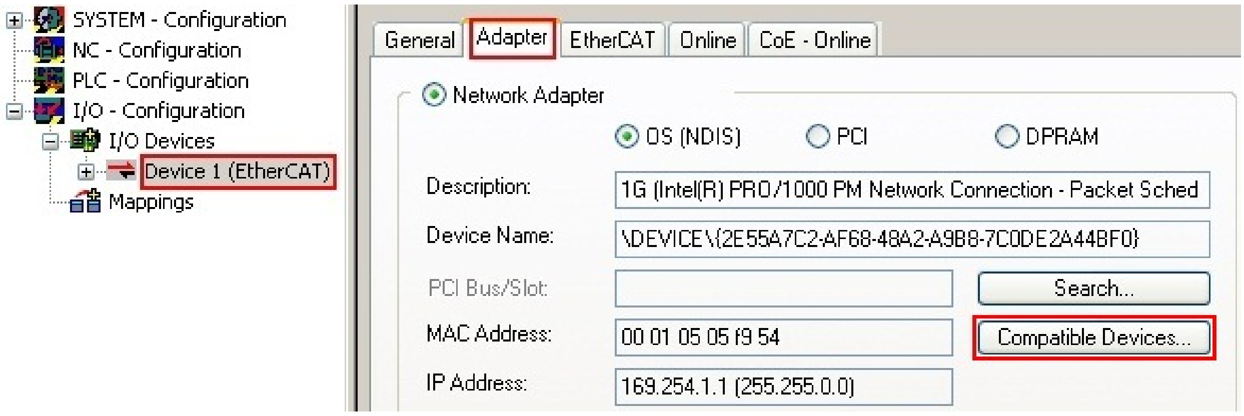
- Installed and ready to use devices
  - LAN3 - TwinCAT-Intel PCI Ethernet Adapter (Gigabit)
  - 100M - TwinCAT-Intel PCI Ethernet Adapter
  - 1G - TwinCAT-Intel PCI Ethernet Adapter (Gigabit)
- Compatible devices
- Incompatible devices
- Disabled devices

Fig. 63: 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” [▶ 72] in order to view the compatible ethernet ports via its EtherCAT properties (tab “Adapter”, button “Compatible Devices…”):



● Network Adapter

● OS (NDIS) ○ PCI ○ DPRAM

Description: 1G (Intel(R) PRO/1000 PM Network Connection - Packet Sched

Device Name: \DEVICE\{2E55A7C2-AF68-48A2-A9B8-7C0DE2A44BF0}

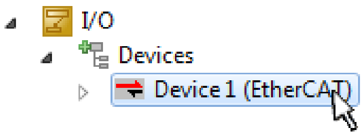
PCI Bus/Slot:

MAC Address: 00 01 05 05 f9 54

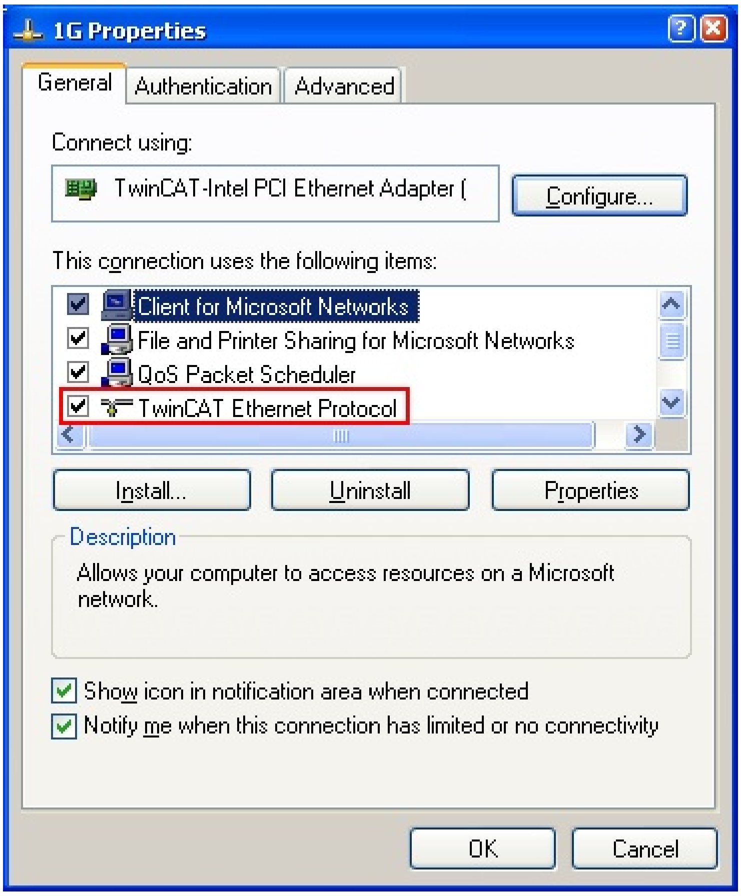
IP Address: 169.254.1.1 (255.255.0.0)

Fig. 64: 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)



1G Properties

General | Authentication | Advanced

Connect using:

TwinCAT-Intel PCI Ethernet Adapter (

Configure...

This connection uses the following items:

☑ Client for Microsoft Networks
☑ File and Printer Sharing for Microsoft Networks
☑ QoS Packet Scheduler
☑ TwinCAT Ethernet Protocol

Install... | Uninstall | Properties

Description

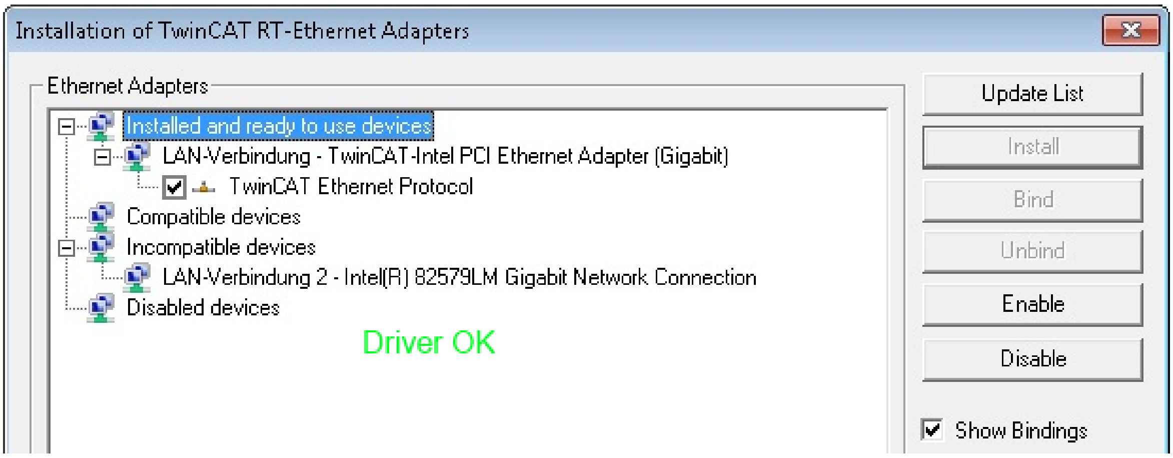
Allows your computer to access resources on a Microsoft network.

☑ Show icon in notification area when connected
☑ Notify me when this connection has limited or no connectivity

OK | Cancel

Fig. 65: Windows properties of the network interface

A correct setting of the driver could be:



Installation of TwinCAT RT-Ethernet Adapters

Ethernet Adapters

- Installed and ready to use devices
  - LAN-Verbindung - TwinCAT-Intel PCI Ethernet Adapter (Gigabit)
    - ☑ TwinCAT Ethernet Protocol
- Compatible devices
- Incompatible devices
  - LAN-Verbindung 2 - Intel(R) 82579LM Gigabit Network Connection
- Disabled devices

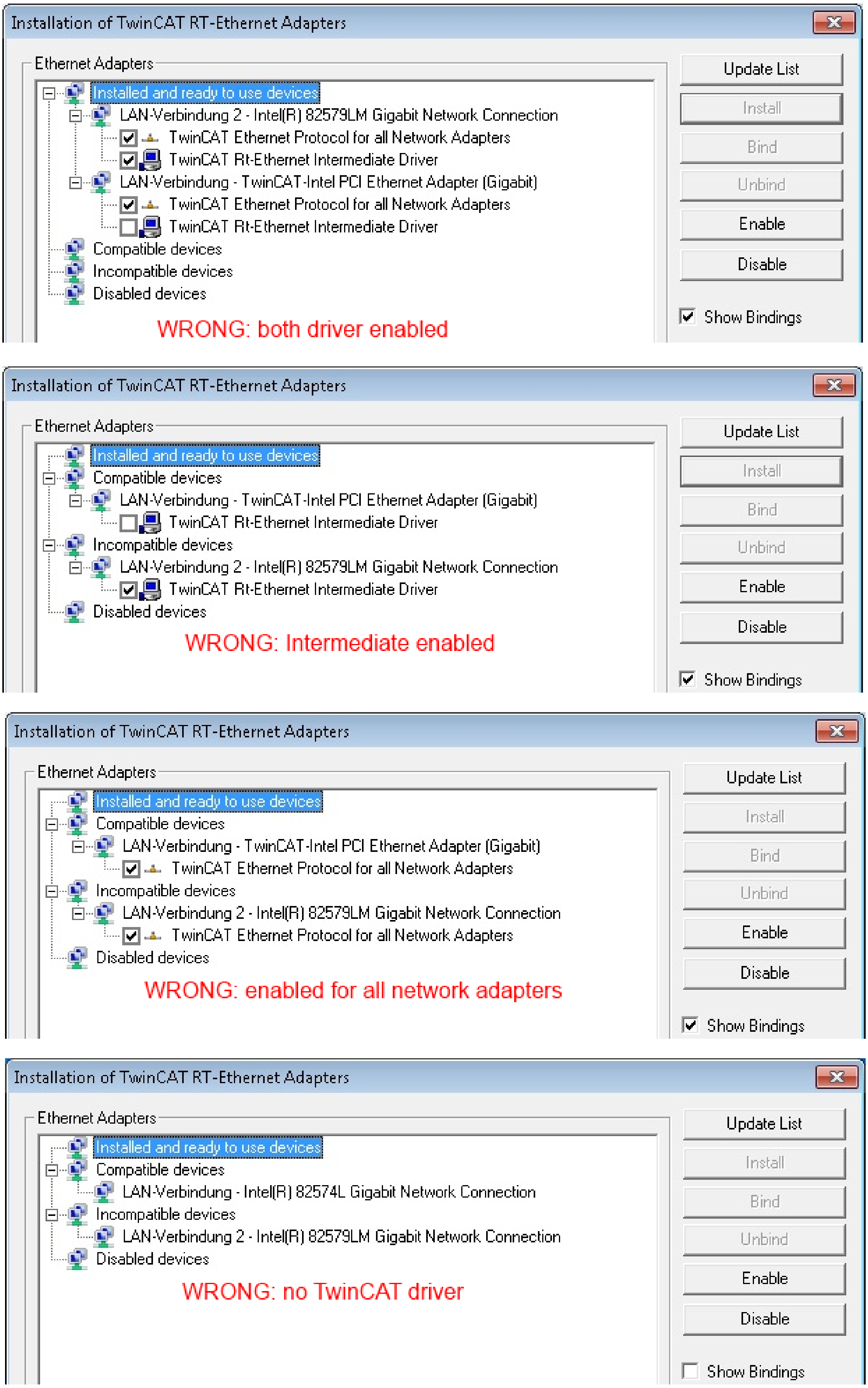
Driver OK

Update List | Install | Bind | Unbind | Enable | Disable

☑ Show Bindings

Fig. 66: Exemplary correct driver setting for the Ethernet port

Other possible settings have to be avoided:



Installation of TwinCAT RT-Ethernet Adapters

Ethernet Adapters

- Installed and ready to use devices
  - LAN-Verbindung 2 - Intel(R) 82579LM Gigabit Network Connection
    - ☑ TwinCAT Ethernet Protocol for all Network Adapters
    - ☑ TwinCAT Rt-Ethernet Intermediate Driver
  - LAN-Verbindung - TwinCAT-Intel PCI Ethernet Adapter (Gigabit)
    - ☑ TwinCAT Ethernet Protocol for all Network Adapters
    - □ TwinCAT Rt-Ethernet Intermediate Driver
- Compatible devices
- Incompatible devices
- Disabled devices

WRONG: both driver enabled

☑ Show Bindings

Installation of TwinCAT RT-Ethernet Adapters

Ethernet Adapters

- Installed and ready to use devices
- Compatible devices
  - LAN-Verbindung - TwinCAT-Intel PCI Ethernet Adapter (Gigabit)
    - □ TwinCAT Rt-Ethernet Intermediate Driver
- Incompatible devices
  - LAN-Verbindung 2 - Intel(R) 82579LM Gigabit Network Connection
    - ☑ TwinCAT Rt-Ethernet Intermediate Driver
- Disabled devices

WRONG: Intermediate enabled

☑ Show Bindings

Installation of TwinCAT RT-Ethernet Adapters

Ethernet Adapters

- Installed and ready to use devices
- Compatible devices
  - LAN-Verbindung - TwinCAT-Intel PCI Ethernet Adapter (Gigabit)
    - ☑ TwinCAT Ethernet Protocol for all Network Adapters
- Incompatible devices
  - LAN-Verbindung 2 - Intel(R) 82579LM Gigabit Network Connection
    - ☑ TwinCAT Ethernet Protocol for all Network Adapters
- Disabled devices

WRONG: enabled for all network adapters

☑ Show Bindings

Installation of TwinCAT RT-Ethernet Adapters

Ethernet Adapters

- Installed and ready to use devices
- Compatible devices
  - LAN-Verbindung - Intel(R) 82574L Gigabit Network Connection
- Incompatible devices
  - LAN-Verbindung 2 - Intel(R) 82579LM Gigabit Network Connection
- Disabled devices

WRONG: no TwinCAT driver

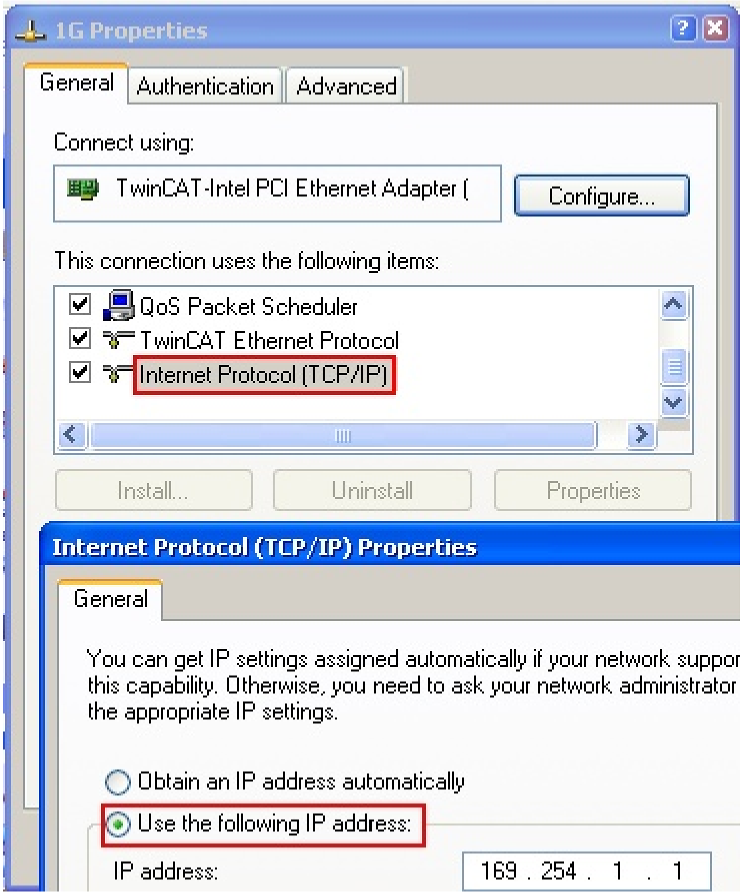
□ Show Bindings

Fig. 67: 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.



1G Properties

General | Authentication | Advanced

Connect using: TwinCAT-Intel PCI Ethernet Adapter (

This connection uses the following items:

- ☑ QoS Packet Scheduler
- ☑ TwinCAT Ethernet Protocol
- ☑ Internet Protocol (TCP/IP)

Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network suppor this capability. Otherwise, you need to ask your network administrator the appropriate IP settings.

○ Obtain an IP address automatically

● Use the following IP address:

IP address: 169 . 254 . 1 . 1

Fig. 68: 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 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”

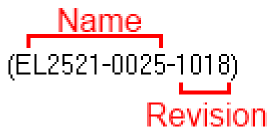
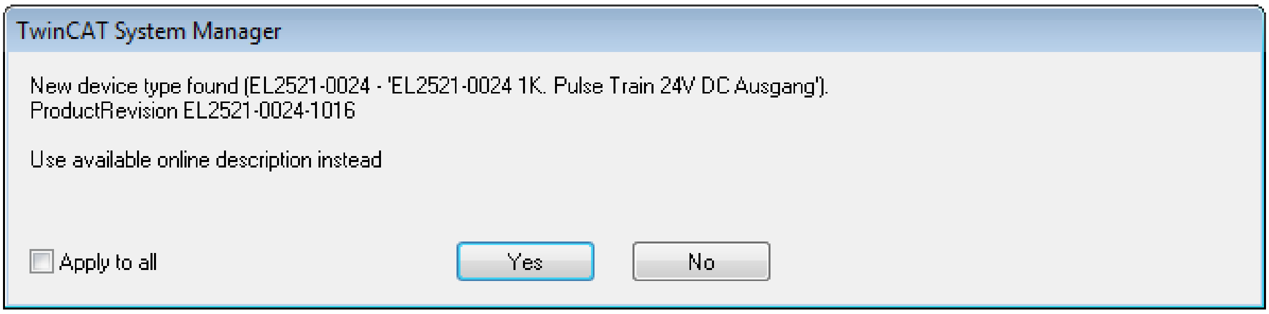


Fig. 69: Identifier structure

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.

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.

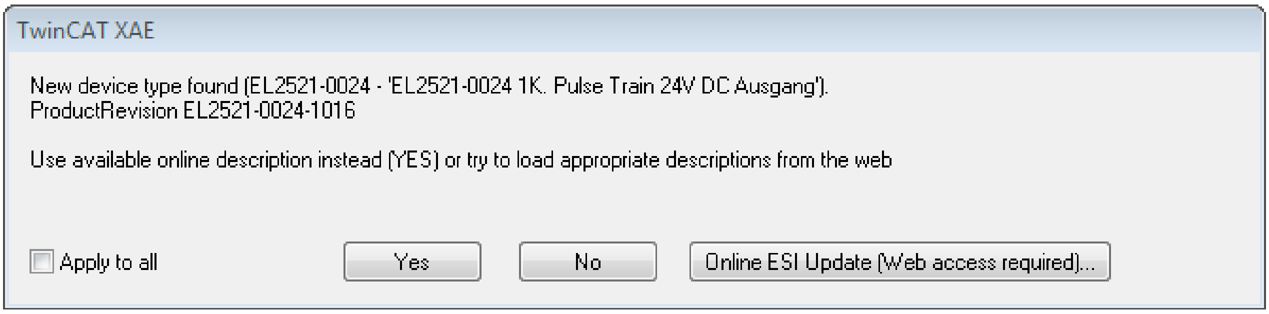

TwinCAT System Manager

New device type found [EL2521-0024 - 'EL2521-0024 1K. Pulse Train 24V DC Ausgang'].
ProductRevision EL2521-0024-1016

Use available online description instead

Fig. 70: OnlineDescription information window (TwinCAT 2)

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


TwinCAT XAE

New device type found [EL2521-0024 - 'EL2521-0024 1K. Pulse Train 24V DC Ausgang'].
ProductRevision EL2521-0024-1016

Use available online description instead (YES) or try to load appropriate descriptions from the web

Fig. 71: 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

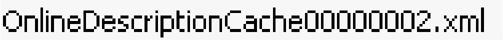
a) no ESI is present for the EL2521-0000 device at all, either for the revision 1019 or for an older revision. The ESI must then be requested from the manufacturer (in this case Beckhoff).

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 [▶ 72]".

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.

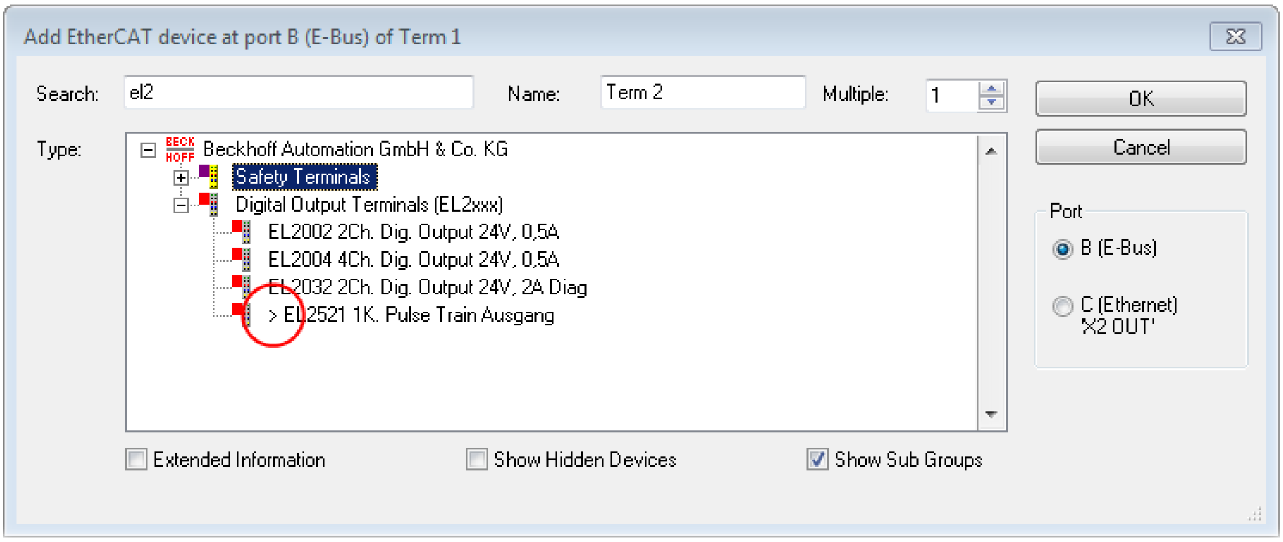
The System Manager creates for online recorded device descriptions a new file "OnlineDescription0000...xml" in its ESI directory, which contains all ESI descriptions that were read online.



OnlineDescriptionCache00000002.xml

Fig. 72: 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)*.



Add EtherCAT device at port B (E-Bus) of Term 1

Search: el2

Name: Term 2

Multiple: 1

Port: ● B (E-Bus) ○ C (Ethernet) 'X2 OUT'

□ Extended Information □ Show Hidden Devices ☑ Show Sub Groups

Type:
Beckhoff Automation GmbH & Co. KG
Safety Terminals
Digital Output Terminals (EL2xxx)
EL2002 2Ch. Dig. Output 24V, 0,5A
EL2004 4Ch. Dig. Output 24V, 0,5A
EL2032 2Ch. Dig. Output 24V, 2A Diag
> EL2521 1K. Pulse Train Ausgang
OK
Cancel

Fig. 73: 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

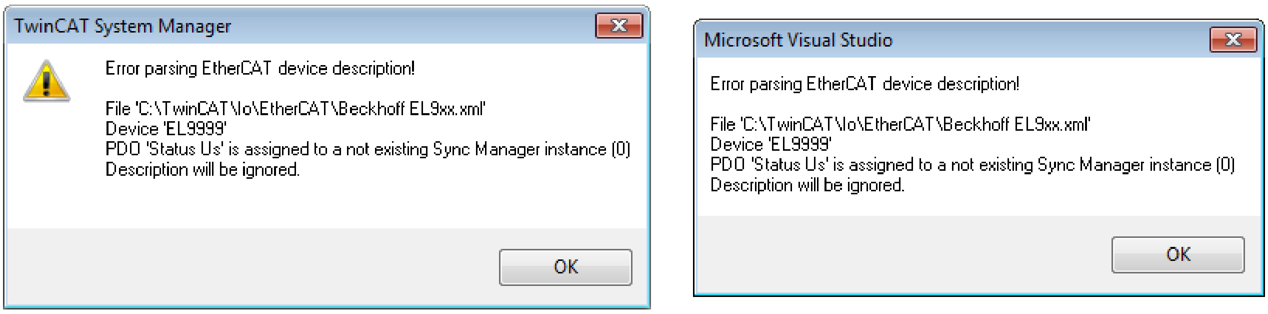
In addition to the file described above “OnlineDescription0000...xml”, a so called EtherCAT cache with new discovered devices is created by TwinCAT 3.x, e.g. under Windows 7:

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.



TwinCAT System Manager

Error parsing EtherCAT device description!

File 'C:\TwinCAT\Io\EtherCAT\Beckhoff EL9xx.xml'
Device 'EL9999'
PDO 'Status Us' is assigned to a not existing Sync Manager instance (0)
Description will be ignored.

Microsoft Visual Studio

Error parsing EtherCAT device description!

File 'C:\TwinCAT\Io\EtherCAT\Beckhoff EL9xx.xml'
Device 'EL9999'
PDO 'Status Us' is assigned to a not existing Sync Manager instance (0)
Description will be ignored.

Fig. 74: 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 OFFLINE configuration creation

Creating the EtherCAT device

Create an EtherCAT device in an empty System Manager window.

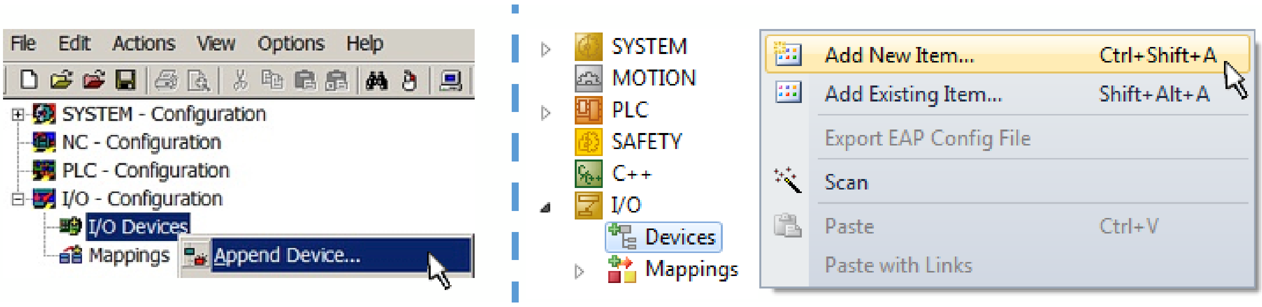


Fig. 75: Append EtherCAT device (left: TwinCAT 2; right: TwinCAT 3)

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”.

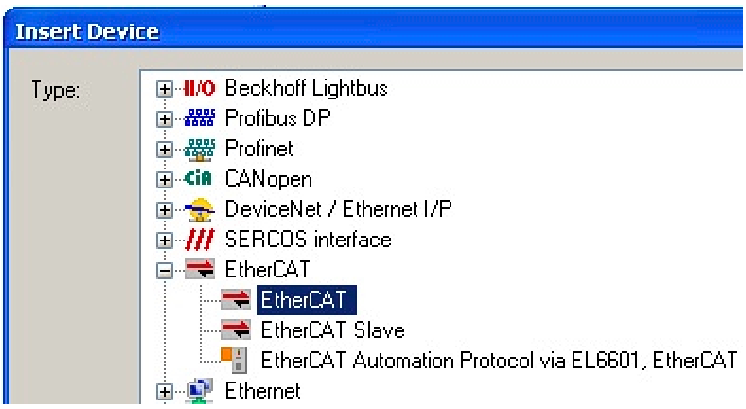


Fig. 76: Selecting the EtherCAT connection (TwinCAT 2.11, TwinCAT 3)

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

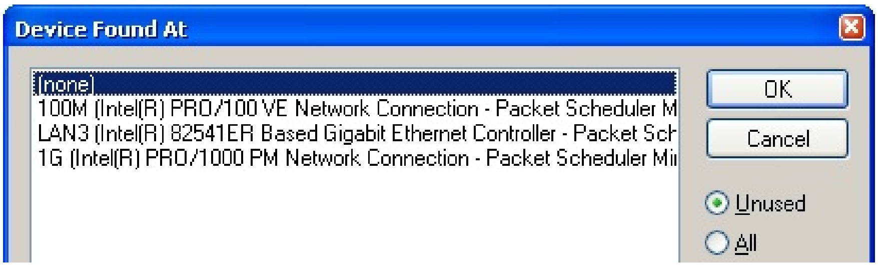
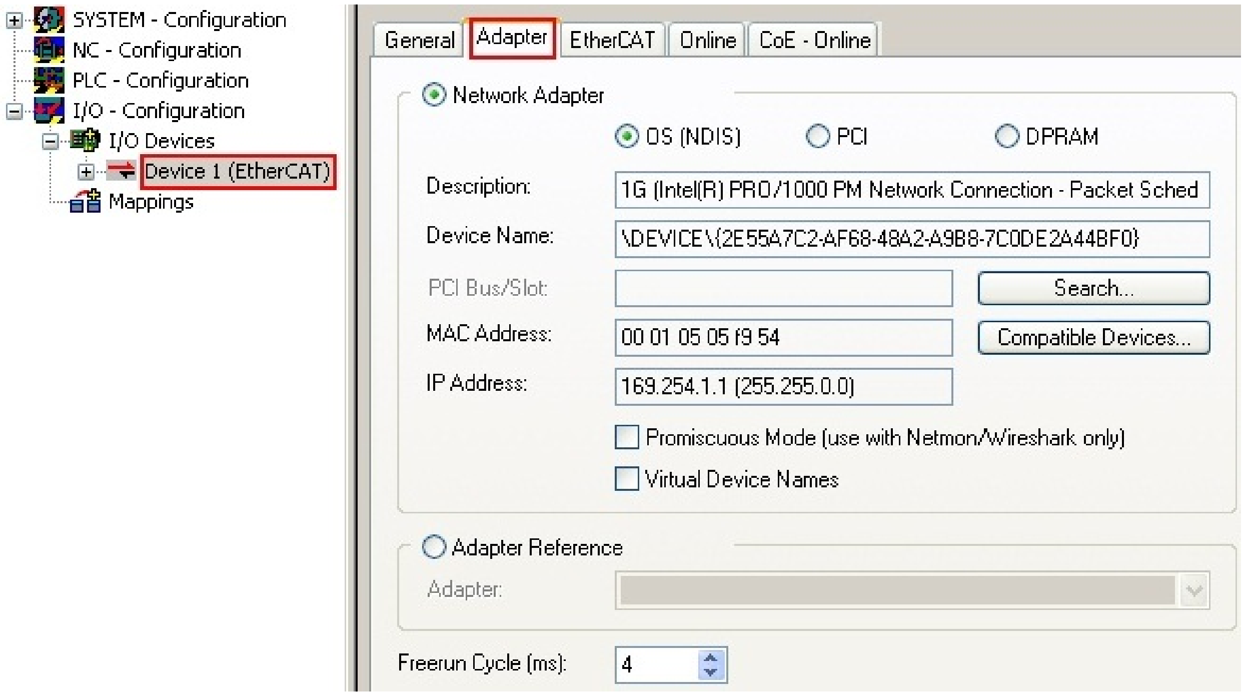

Device Found At
(none)
100M (Intel(R) PRO/100 VE Network Connection - Packet Scheduler M
LAN3 (Intel(R) 82541ER Based Gigabit Ethernet Controller - Packet Sch
1G (Intel(R) PRO/1000 PM Network Connection - Packet Scheduler Mi
OK
Cancel
● Unused
○ All

Fig. 77: 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)”.



● Network Adapter

● OS (NDIS) ○ PCI ○ DPRAM

Description: 1G (Intel(R) PRO/1000 PM Network Connection - Packet Sched

Device Name: \DEVICE\{2E55A7C2-AF68-48A2-A9B8-7C0DE2A44BF0}

PCI Bus/Slot:

MAC Address: 00 01 05 05 f9 54

IP Address: 169.254.1.1 (255.255.0.0)

□ Promiscuous Mode (use with Netmon/Wireshark only)

□ Virtual Device Names

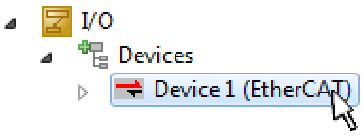
○ Adapter Reference

Adapter:

Freerun Cycle (ms): 4

Fig. 78: 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 [▶ 63].

Defining EtherCAT slaves

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

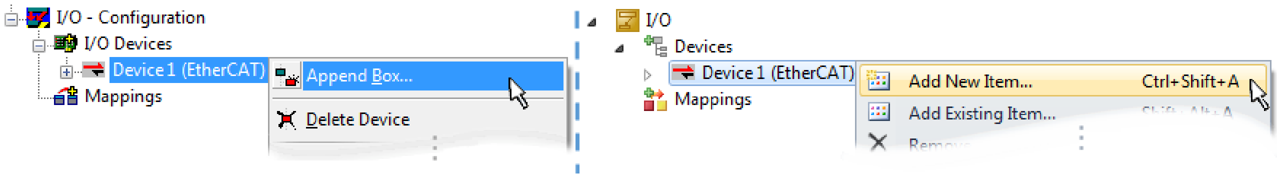


Fig. 79: Appending EtherCAT devices (left: TwinCAT 2; right: TwinCAT 3)

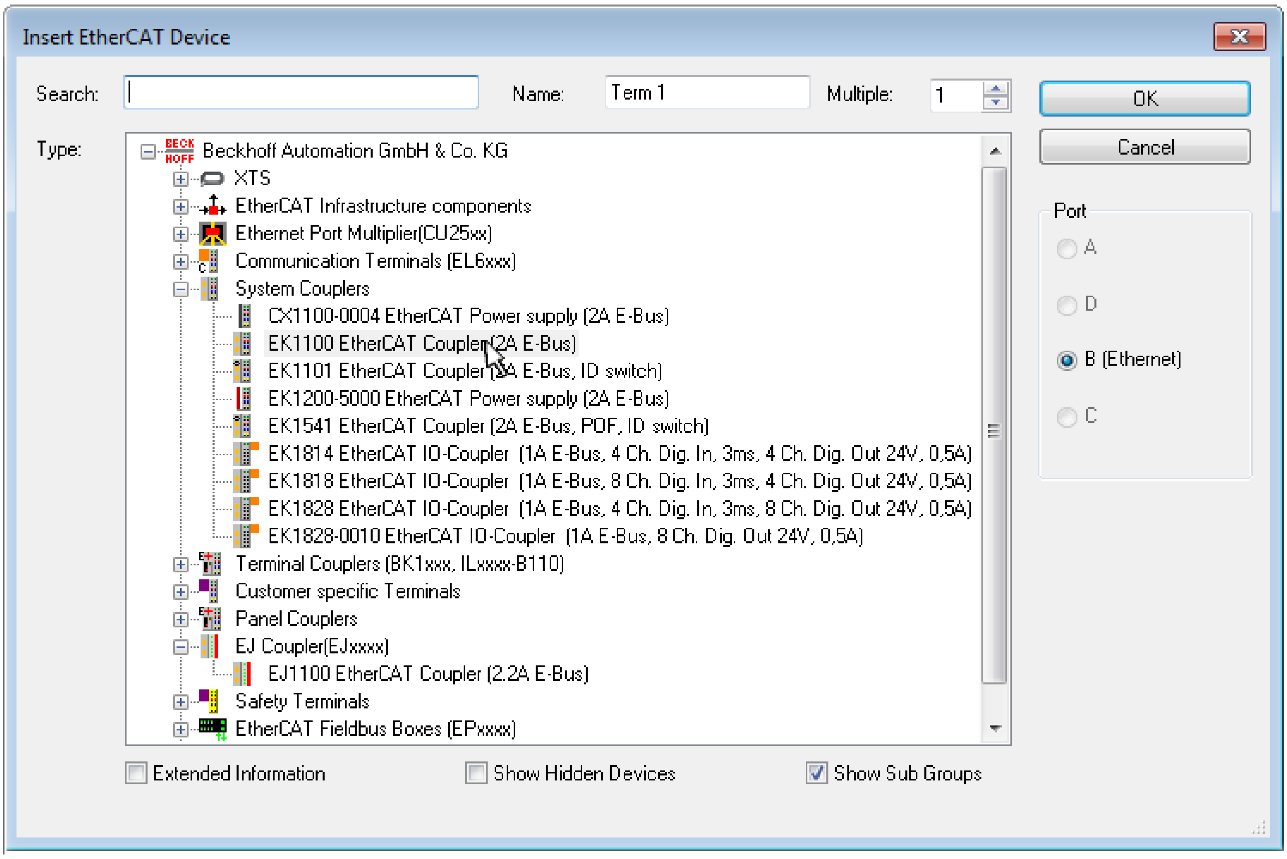
The dialog for selecting a new device opens. Only devices for which ESI files are available are displayed.

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).



Insert EtherCAT Device

Search:

Name: Term 1

Multiple: 1

Type:

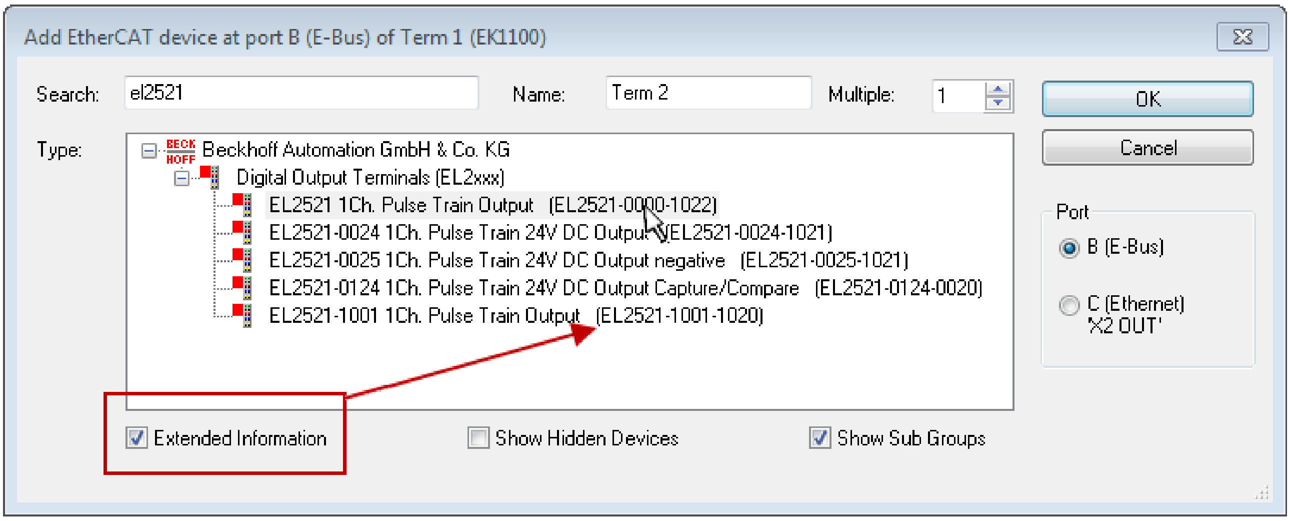
- Beckhoff Automation GmbH & Co. KG
  - XTS
  - EtherCAT Infrastructure components
  - Ethernet Port Multiplier(CU25xx)
  - Communication Terminals (EL6xxx)
  - System Couplers
    - CX1100-0004 EtherCAT Power supply (2A E-Bus)
    - EK1100 EtherCAT Coupler (2A E-Bus)
    - EK1101 EtherCAT Coupler (2A E-Bus, ID switch)
    - EK1200-5000 EtherCAT Power supply (2A E-Bus)
    - EK1541 EtherCAT Coupler (2A E-Bus, POF, ID switch)
    - EK1814 EtherCAT IO-Coupler (1A E-Bus, 4 Ch. Dig. In, 3ms, 4 Ch. Dig. Out 24V, 0,5A)
    - EK1818 EtherCAT IO-Coupler (1A E-Bus, 8 Ch. Dig. In, 3ms, 4 Ch. Dig. Out 24V, 0,5A)
    - EK1828 EtherCAT IO-Coupler (1A E-Bus, 4 Ch. Dig. In, 3ms, 8 Ch. Dig. Out 24V, 0,5A)
    - EK1828-0010 EtherCAT IO-Coupler (1A E-Bus, 8 Ch. Dig. Out 24V, 0,5A)
  - Terminal Couplers (BK1xxx, ILxxxx-B110)
  - Customer specific Terminals
  - Panel Couplers
  - EJ Coupler(EJxxxx)
    - EJ1100 EtherCAT Coupler (2.2A E-Bus)
  - Safety Terminals
  - EtherCAT Fieldbus Boxes (EPxxxx)

□ Extended Information □ Show Hidden Devices ☑ Show Sub Groups

Port: ○ A ○ D ● B (Ethernet) ○ C

Fig. 80: 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”.



Add EtherCAT device at port B (E-Bus) of Term 1 (EK1100)

Search: el2521

Name: Term 2

Multiple: 1

Type:

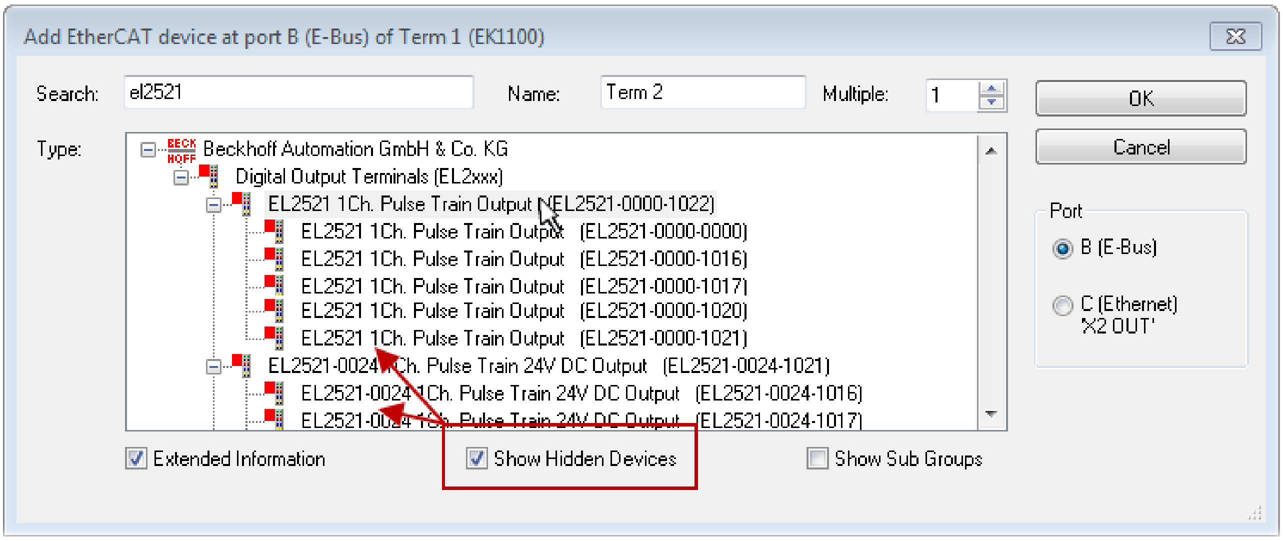
- Beckhoff Automation GmbH & Co. KG
  - Digital Output Terminals (EL2xxx)
    - EL2521 1Ch. Pulse Train Output (EL2521-0000-1022)
    - EL2521-0024 1Ch. Pulse Train 24V DC Output (EL2521-0024-1021)
    - EL2521-0025 1Ch. Pulse Train 24V DC Output negative (EL2521-0025-1021)
    - EL2521-0124 1Ch. Pulse Train 24V DC Output Capture/Compare (EL2521-0124-0020)
    - EL2521-1001 1Ch. Pulse Train Output (EL2521-1001-1020)

☑ Extended Information □ Show Hidden Devices ☑ Show Sub Groups

Port: ● B (E-Bus) ○ C (Ethernet) 'X2 OUT'

Fig. 81: 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”.



Add EtherCAT device at port B (E-Bus) of Term 1 (EK1100)

Search: el2521

Name: Term 2

Multiple: 1

Type:

- Beckhoff Automation GmbH & Co. KG
  - Digital Output Terminals (EL2xxx)
    - EL2521 1Ch. Pulse Train Output (EL2521-0000-1022)
      - EL2521 1Ch. Pulse Train Output (EL2521-0000-0000)
      - EL2521 1Ch. Pulse Train Output (EL2521-0000-1016)
      - EL2521 1Ch. Pulse Train Output (EL2521-0000-1017)
      - EL2521 1Ch. Pulse Train Output (EL2521-0000-1020)
      - EL2521 1Ch. Pulse Train Output (EL2521-0000-1021)
    - EL2521-0024 1Ch. Pulse Train 24V DC Output (EL2521-0024-1021)
      - EL2521-0024 1Ch. Pulse Train 24V DC Output (EL2521-0024-1016)
      - EL2521-0024 1Ch. Pulse Train 24V DC Output (EL2521-0024-1017)

☑ Extended Information ☑ Show Hidden Devices □ Show Sub Groups

Port

● B (E-Bus)

○ C (Ethernet) 'X2 OUT'

Fig. 82: 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.

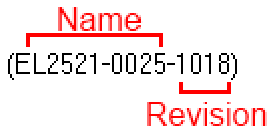
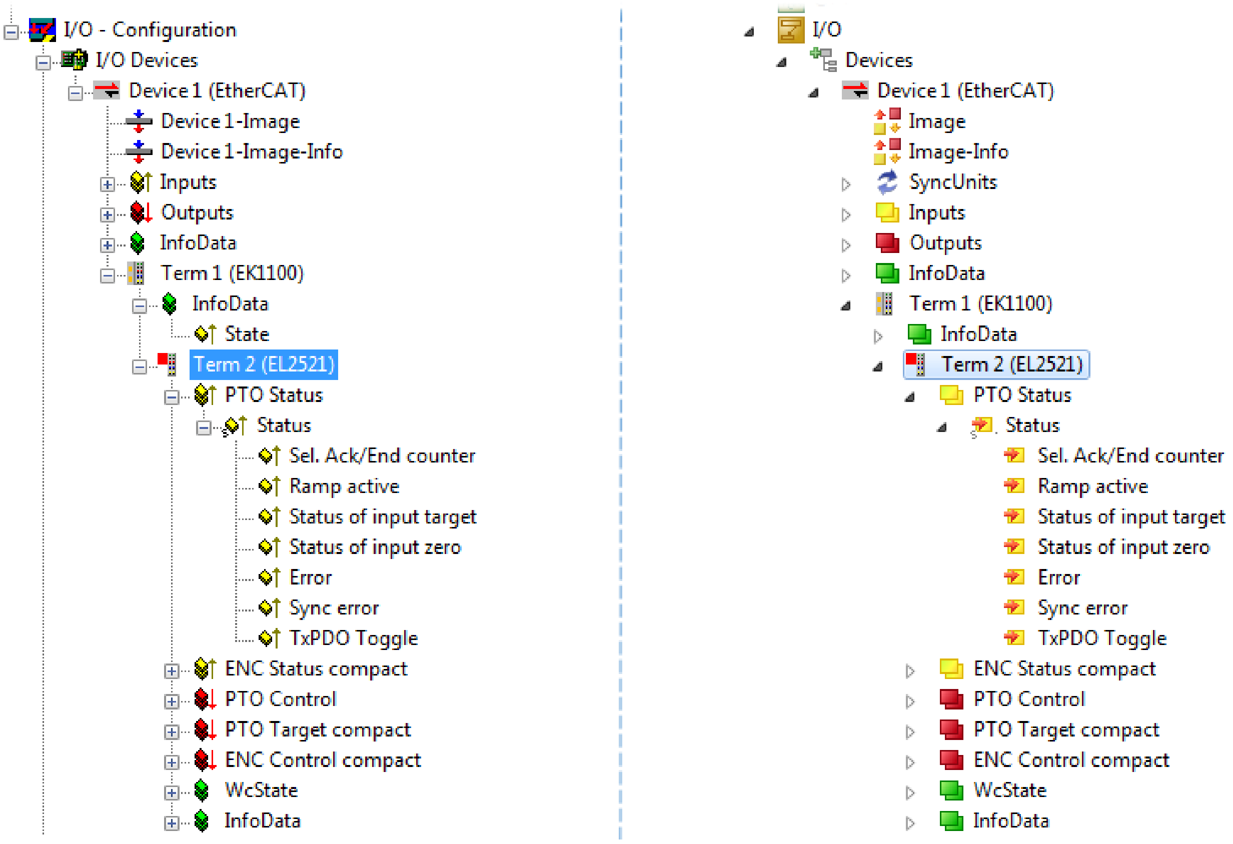


Fig. 83: Name/revision of the terminal

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, ...



- I/O - Configuration
  - I/O Devices
    - Device 1 (EtherCAT)
      - Device 1-Image
      - Device 1-Image-Info
      - Inputs
      - Outputs
      - InfoData
      - Term 1 (EK1100)
        - InfoData
          - State
        - Term 2 (EL2521)
          - PTO Status
            - Status
              - Sel. Ack/End counter
              - Ramp active
              - Status of input target
              - Status of input zero
              - Error
              - Sync error
              - TxPDO Toggle
          - ENC Status compact
          - PTO Control
          - PTO Target compact
          - ENC Control compact
          - WcState
          - InfoData

- I/O
  - Devices
    - Device 1 (EtherCAT)
      - Image
      - Image-Info
      - SyncUnits
      - Inputs
      - Outputs
      - InfoData
      - Term 1 (EK1100)
        - InfoData
        - Term 2 (EL2521)
          - PTO Status
            - Status
              - Sel. Ack/End counter
              - Ramp active
              - Status of input target
              - Status of input zero
              - Error
              - Sync error
              - TxPDO Toggle
          - ENC Status compact
          - PTO Control
          - PTO Target compact
          - ENC Control compact
          - WcState
          - InfoData

Fig. 84: EtherCAT terminal in the TwinCAT tree (left: TwinCAT 2; right: TwinCAT 3)

5.2.4 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.

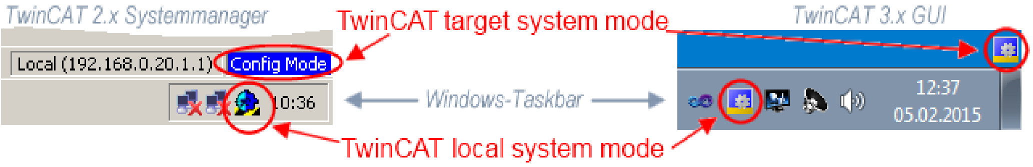


Fig. 85: Differentiation local/target system (left: TwinCAT 2; right: TwinCAT 3)

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

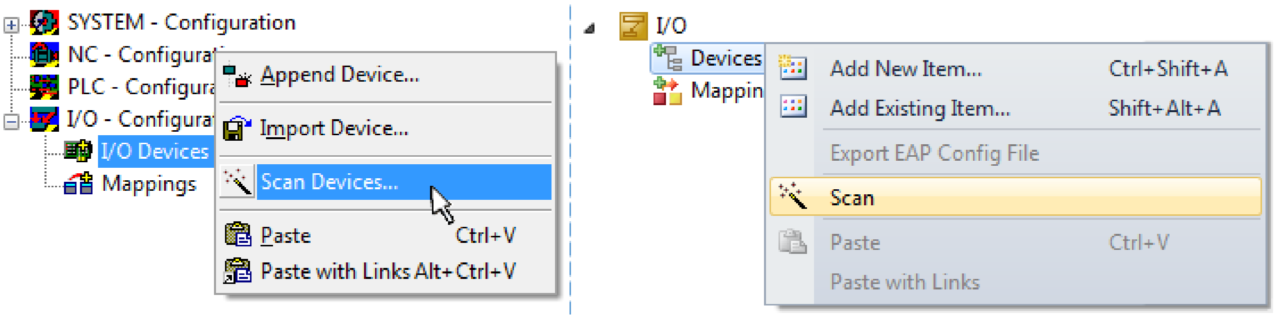
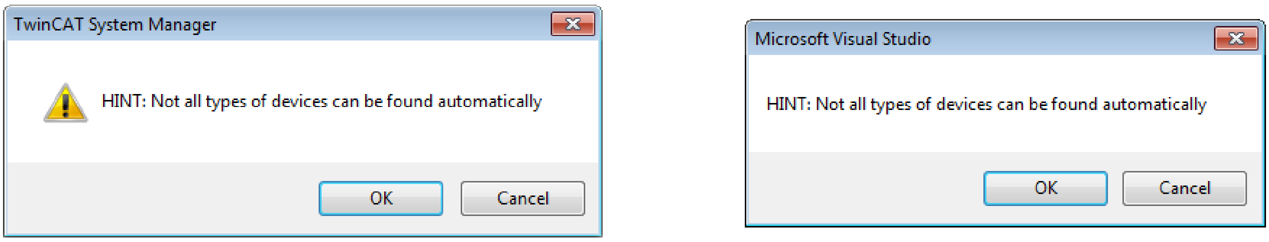


Fig. 86: Scan Devices (left: TwinCAT 2; right: TwinCAT 3)

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.

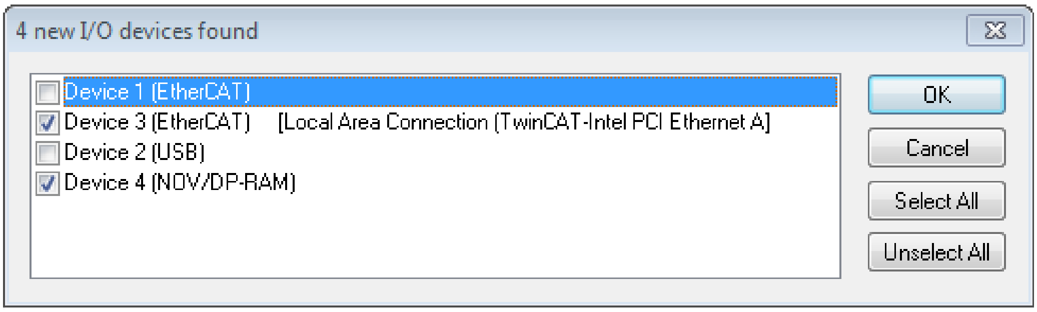


HINT: Not all types of devices can be found automatically

HINT: Not all types of devices can be found automatically

Fig. 87: 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” .


4 new I/O devices found

- □ Device 1 (EtherCAT)
- ☑ Device 3 (EtherCAT) [Local Area Connection (TwinCAT-Intel PCI Ethernet A]
- □ Device 2 (USB)
- ☑ Device 4 (NOV/DP-RAM)

Fig. 88: 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 [▶ 63].

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.

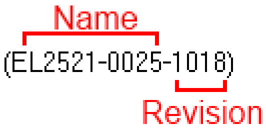


Fig. 89: Example default state

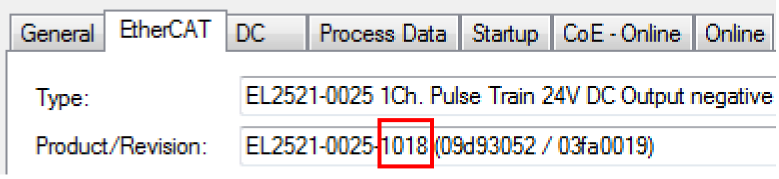
NOTE

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 [▶ 82] 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:



Type: EL2521-0025 1Ch. Pulse Train 24V DC Output negative

Product/Revision: EL2521-0025-1018 (09d93052 / 03fa0019)

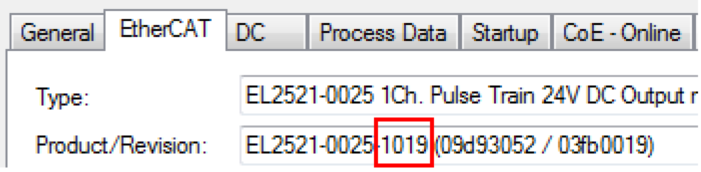
Fig. 90: Installing EthetCAT terminal with revision -1018

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 [▸ 82] 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:



Type: EL2521-0025 1Ch. Pulse Train 24V DC Output r

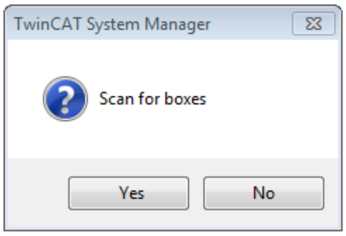
Product/Revision: EL2521-0025-1019 (09d93052 / 03fb0019)

Fig. 91: Detection of EtherCAT terminal with revision -1019

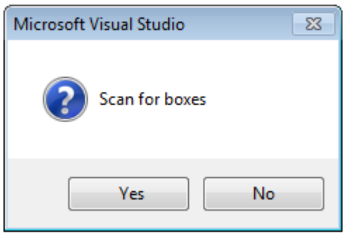
This is usually not noticed by the commissioning engineers. TwinCAT cannot signal anything either, since virtually a new configuration is created. According to the compatibility rule, however, this means that no EL2521-0025-**1018** should be built into this machine as a spare part (even if this nevertheless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



Scan for boxes



Scan for boxes

Fig. 92: Scan query after automatic creation of an EtherCAT device (left: TwinCAT 2; right: TwinCAT 3)

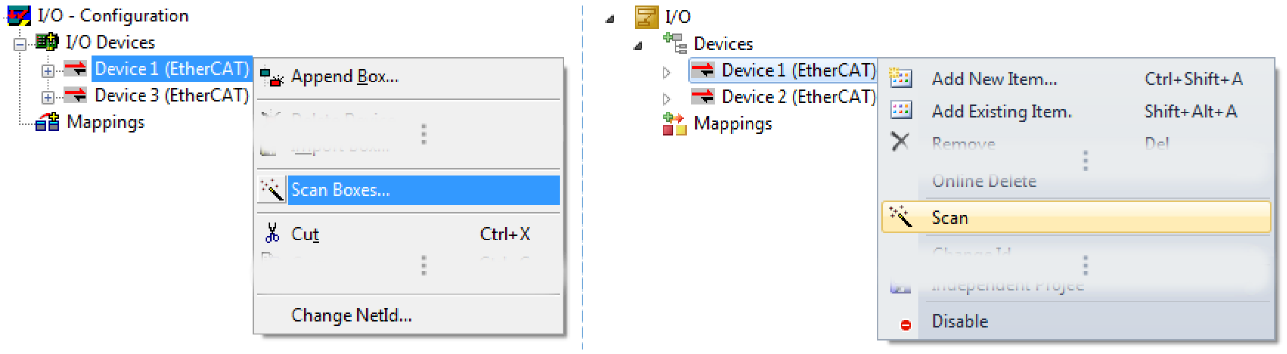


Fig. 93: Manual triggering of a device scan on a specified EtherCAT device (left: TwinCAT 2; right: TwinCAT 3)

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.

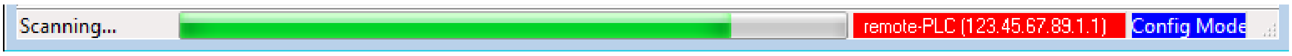


Fig. 94: *Scan progressexemplary by TwinCAT 2*

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

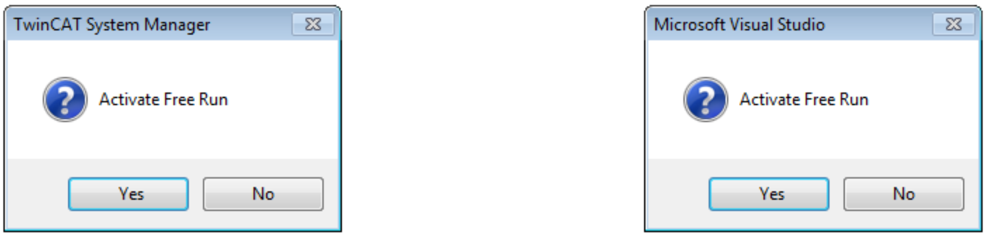


Fig. 95: Config/FreeRun query (left: TwinCAT 2; right: TwinCAT 3)

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).

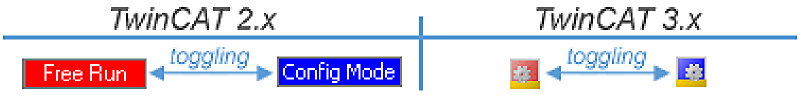


Fig. 96: Displaying of “Free Run” and “Config Mode” toggling right below in the status bar

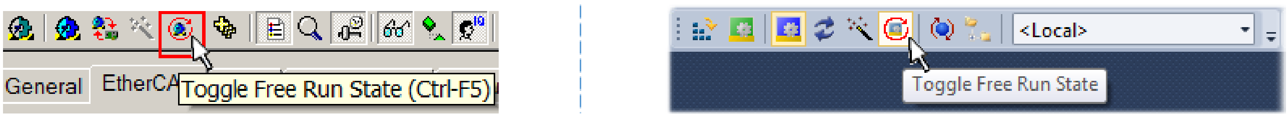
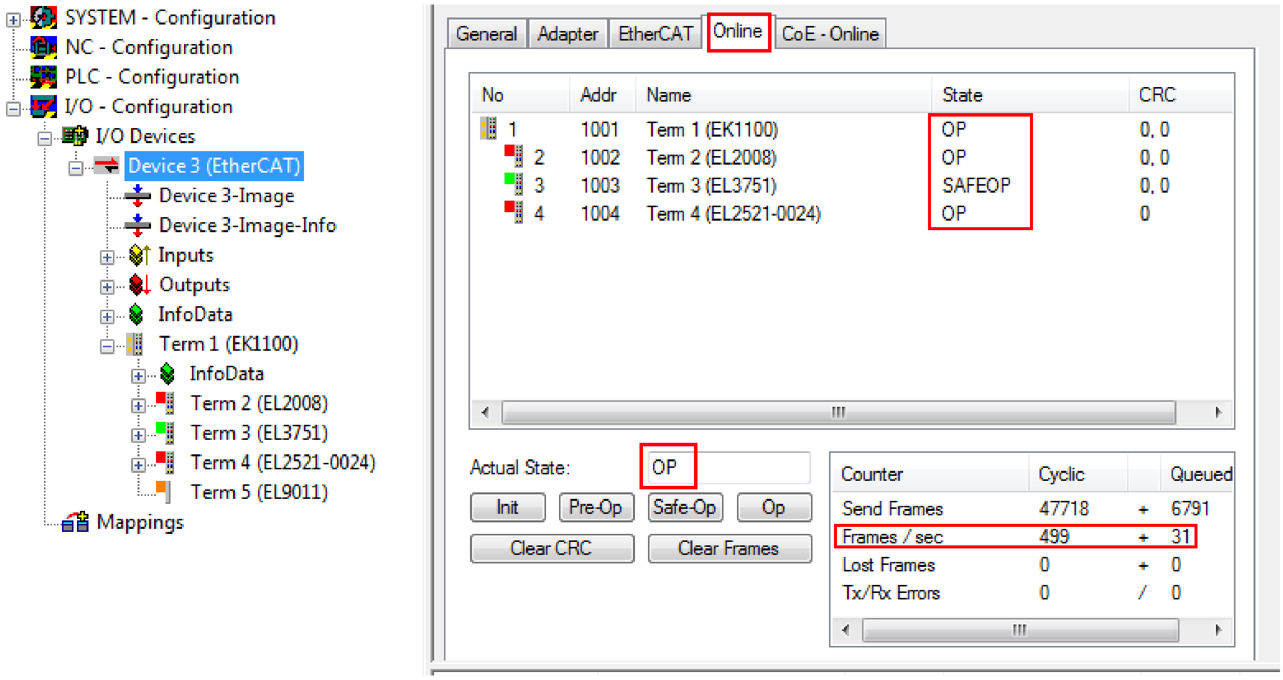


Fig. 97: TwinCAT can also be switched to this state by using a button (left: TwinCAT 2; right: TwinCAT 3)

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



SYSTEM - Configuration
NC - Configuration
PLC - Configuration
I/O - Configuration
I/O Devices
Device 3 (EtherCAT)
Device 3-Image
Device 3-Image-Info
Inputs
Outputs
InfoData
Term 1 (EK1100)
InfoData
Term 2 (EL2008)
Term 3 (EL3751)
Term 4 (EL2521-0024)
Term 5 (EL9011)
Mappings

No	Addr	Name	State	CRC
1	1001	Term 1 (EK1100)	OP	0, 0
2	1002	Term 2 (EL2008)	OP	0, 0
3	1003	Term 3 (EL3751)	SAFEOP	0, 0
4	1004	Term 4 (EL2521-0024)	OP	0

Actual State: OP

Counter	Cyclic		Queued
Send Frames	47718	+	6791
Frames / sec	499	+	31
Lost Frames	0	+	0
Tx/Rx Errors	0	/	0

Fig. 98: 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

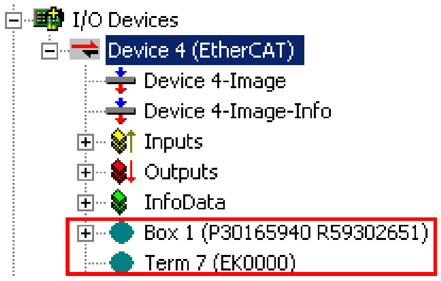
The configuration is now complete. It can be modified as described under manual procedure [▶ 72].

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.



I/O Devices
Device 4 (EtherCAT)
Device 4-Image
Device 4-Image-Info
Inputs
Outputs
InfoData
Box 1 (P30165940 R59302651)
Term 7 (EK0000)

Fig. 99: 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.

TwinCAT System Manager

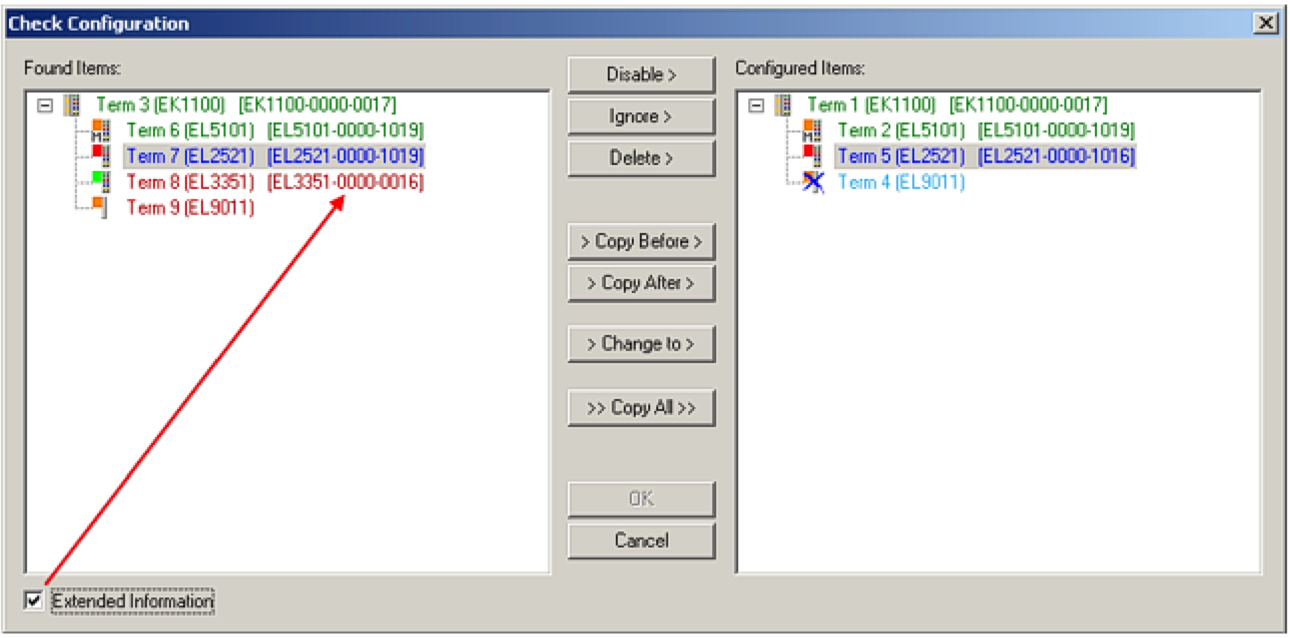
Configuration is identical

Microsoft Visual Studio

Configuration is identical

Fig. 100: 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.



Check Configuration

Found Items:

- Term 3 (EK1100) [EK1100-0000-0017]
  - Term 6 (EL5101) [EL5101-0000-1019]
  - Term 7 (EL2521) [EL2521-0000-1019]
  - Term 8 (EL3351) [EL3351-0000-0016]
  - Term 9 (EL9011)

Configured Items:

- Term 1 (EK1100) [EK1100-0000-0017]
  - Term 2 (EL5101) [EL5101-0000-1019]
  - Term 5 (EL2521) [EL2521-0000-1016]
  - Term 4 (EL9011)

☑ Extended Information

Fig. 101: 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.

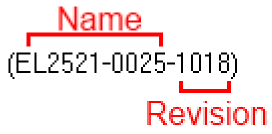


Fig. 102: Name/revision of the terminal

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, ...

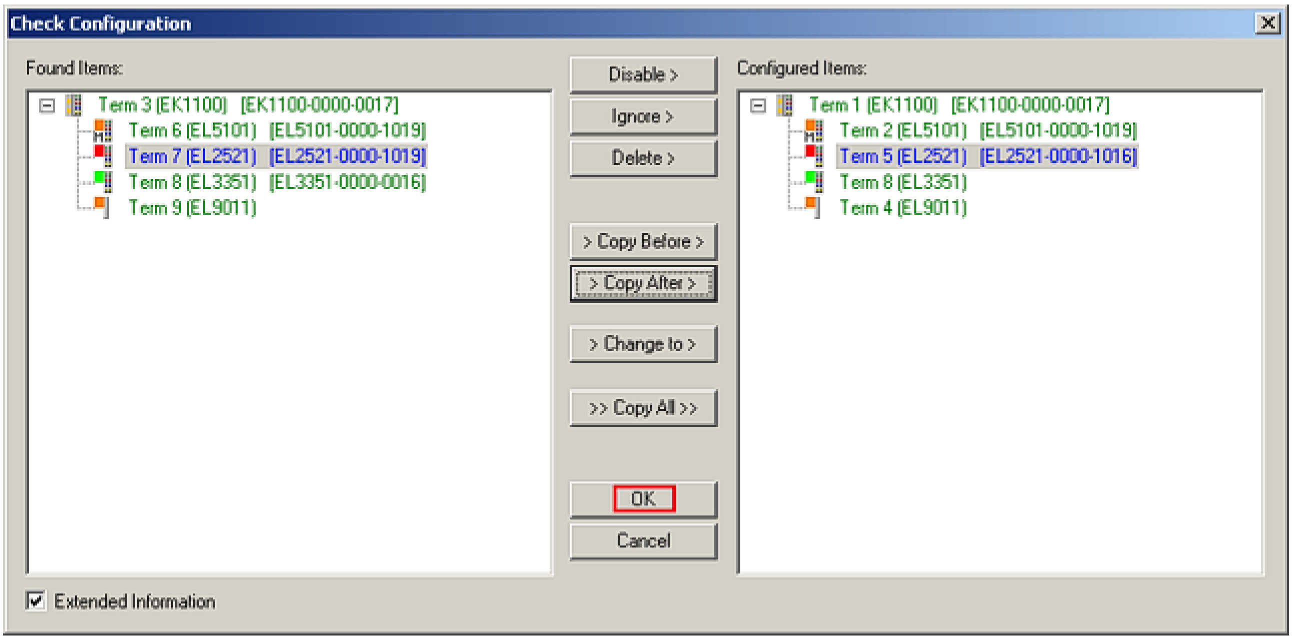

Check Configuration
Found Items:
Term 3 (EK1100) [EK1100-0000-0017]
Term 6 (EL5101) [EL5101-0000-1019]
Term 7 (EL2521) [EL2521-0000-1019]
Term 8 (EL3351) [EL3351-0000-0016]
Term 9 (EL9011)
Disable >
Ignore >
Delete >
> Copy Before >
> Copy After >
> Change to >
>> Copy All >>
OK
Cancel
Configured Items:
Term 1 (EK1100) [EK1100-0000-0017]
Term 2 (EL5101) [EL5101-0000-1019]
Term 5 (EL2521) [EL2521-0000-1016]
Term 8 (EL3351)
Term 4 (EL9011)
☑ Extended Information

Fig. 103: 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.

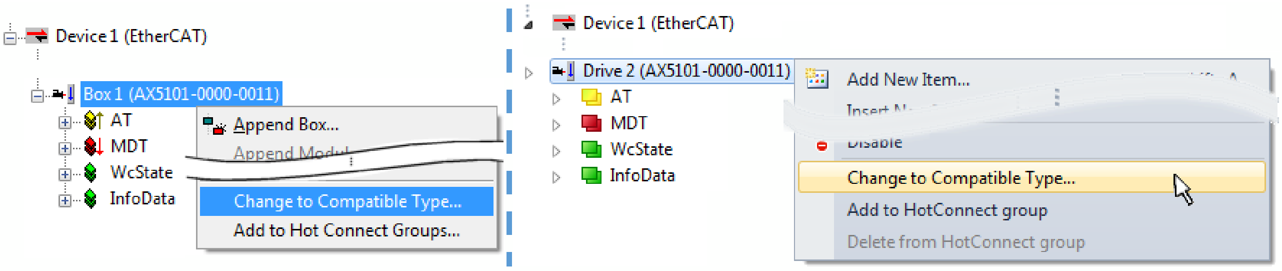


Fig. 104: Dialog “Change to Compatible Type…” (left: TwinCAT 2; right: TwinCAT 3)

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

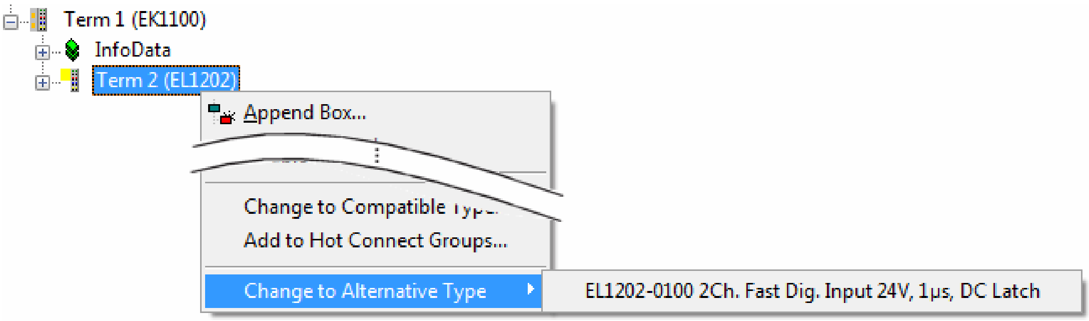

Term 1 (EK1100)
InfoData
Term 2 (EL1202)
Append Box...
Change to Compatible Type
Add to Hot Connect Groups...
Change to Alternative Type
EL1202-0100 2Ch. Fast Dig. Input 24V, 1µs, DC Latch

Fig. 105: 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).

5.2.5 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).

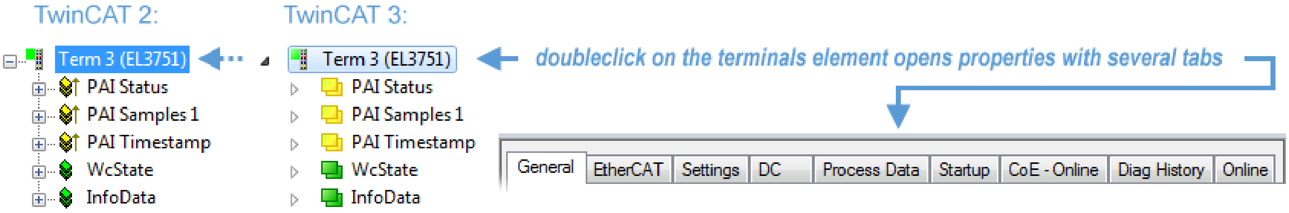
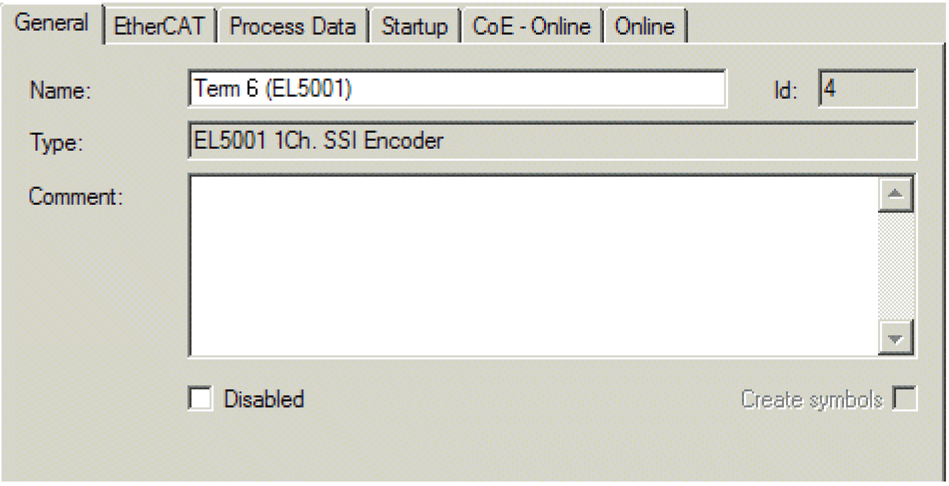


Fig. 106: Branch element as terminal EL3751

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

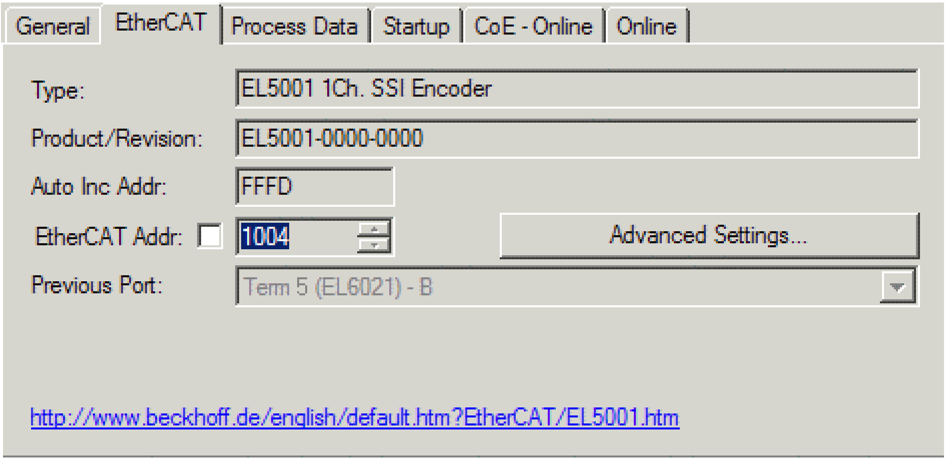


Name: Term 6 (EL5001)
Id: 4
Type: EL5001 1Ch. SSI Encoder
Comment:
☐ Disabled
Create symbols ☐

Fig. 107: “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



Type: EL5001 1Ch. SSI Encoder

Product/Revision: EL5001-0000-0000

Auto Inc Addr: FFFD

EtherCAT Addr: □ 1004

Previous Port: Term 5 (EL6021) - B

http://www.beckhoff.de/english/default.htm?EtherCAT/EL5001.htm

Fig. 108: "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.

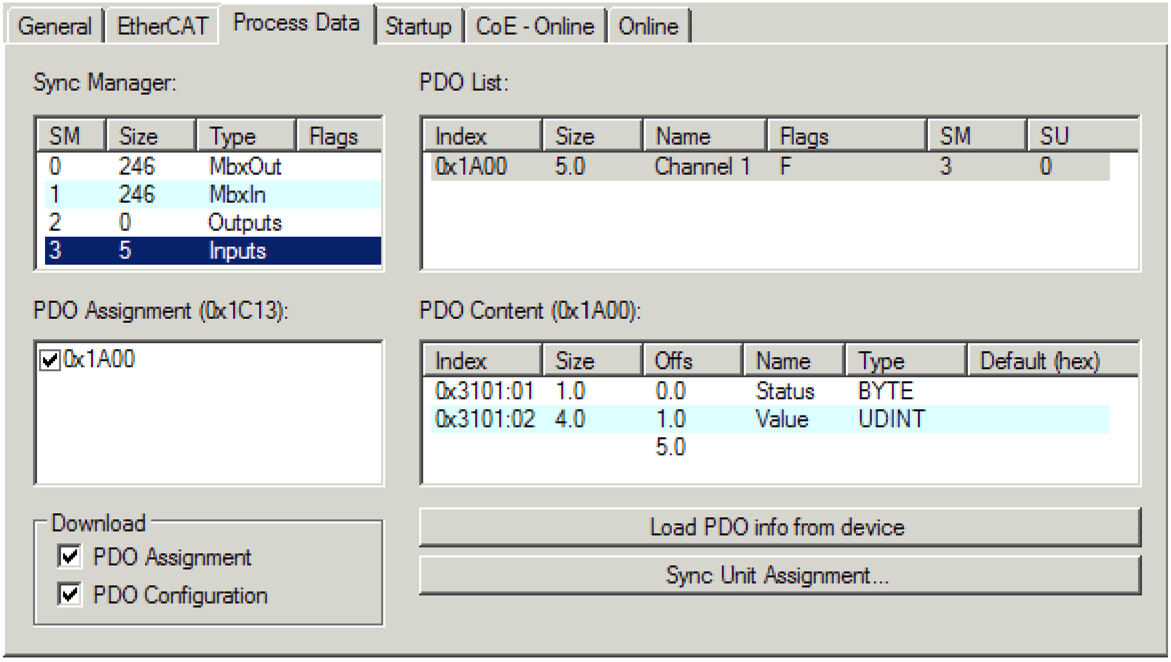
Previous Port Name and port of the EtherCAT device to which this device is connected. If it is possible to connect this device with another one without changing the order of the EtherCAT devices in the communication ring, then this combination field is activated and the EtherCAT device to which this 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.



Sync Manager:

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

PDO List:

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

PDO Assignment (0x1C13):

☑ 0x1A00

PDO Content (0x1A00):

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

Download

☑ PDO Assignment

☑ PDO Configuration

Fig. 109: “Process Data” tab

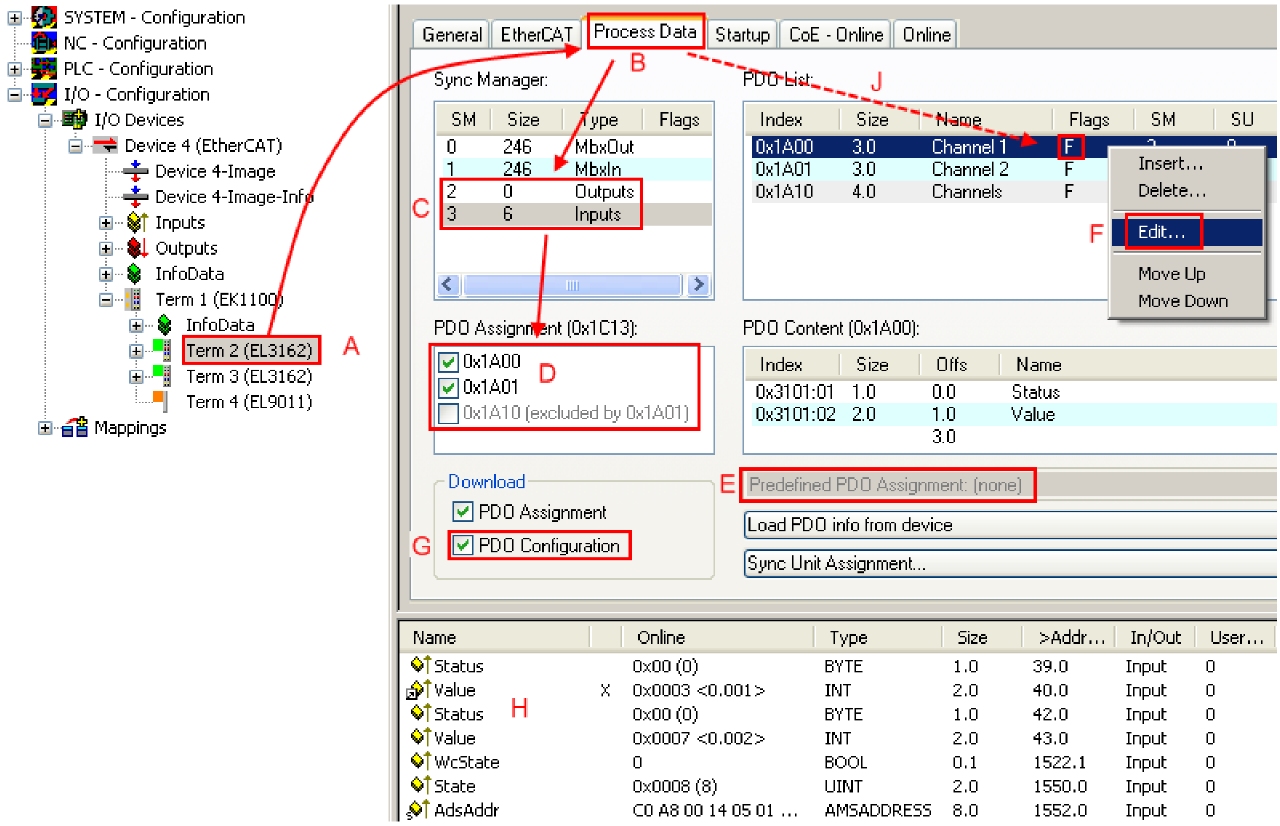
The process data (PDOs) transferred by an EtherCAT slave during each cycle are user data which the application expects to be updated cyclically or which are sent to the slave. To this end the EtherCAT master (Beckhoff TwinCAT) parameterizes each EtherCAT slave during the start-up phase to define which 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”).



Sync Manager:

SM	Size	Type	Flags
0	246	MbxOut	
1	246	MbxIn	
2	0	Outputs	
3	6	Inputs	

PDO List:

Index	Size	Name	Flags	SM	SU
0x1A00	3.0	Channel 1	F	2	0
0x1A01	3.0	Channel 2	F		
0x1A10	4.0	Channels	F		

PDO Assignment (0x1C13):

☑ 0x1A00
☑ 0x1A01
☐ 0x1A10 (excluded by 0x1A01)

PDO Content (0x1A00):

Index	Size	Offs	Name
0x3101:01	1.0	0.0	Status
0x3101:02	2.0	1.0	Value
		3.0	

Predefined PDO Assignment: (none)

☑ PDO Assignment
☑ PDO Configuration

Name		Online	Type	Size	>Addr...	In/Out	User...
Status		0x00 (0)	BYTE	1.0	39.0	Input	0
Value	X	0x0003 <0.001>	INT	2.0	40.0	Input	0
Status		0x00 (0)	BYTE	1.0	42.0	Input	0
Value		0x0007 <0.002>	INT	2.0	43.0	Input	0
WcState		0	BOOL	0.1	1522.1	Input	0
State		0x0008 (8)	UINT	2.0	1550.0	Input	0
AdsAddr		C0 A8 00 14 05 01 ...	AMSADDRESS	8.0	1552.0	Input	0

Fig. 110: Configuring the process data

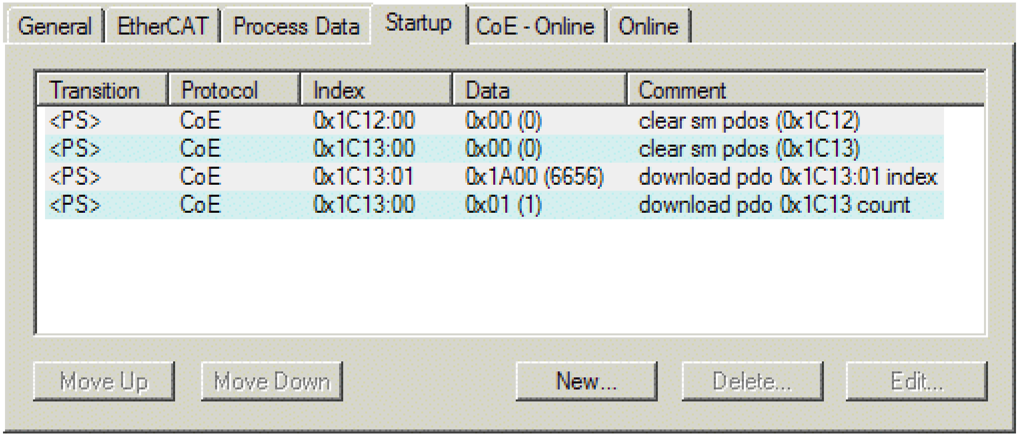
Manual modification of the process data

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 [▸ 93] 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.



Transition	Protocol	Index	Data	Comment
<PS>	CoE	0x1C12:00	0x00 (0)	clear sm pdos (0x1C12)
<PS>	CoE	0x1C13:00	0x00 (0)	clear sm pdos (0x1C13)
<PS>	CoE	0x1C13:01	0x1A00 (6656)	download pdo 0x1C13:01 index
<PS>	CoE	0x1C13:00	0x01 (1)	download pdo 0x1C13 count

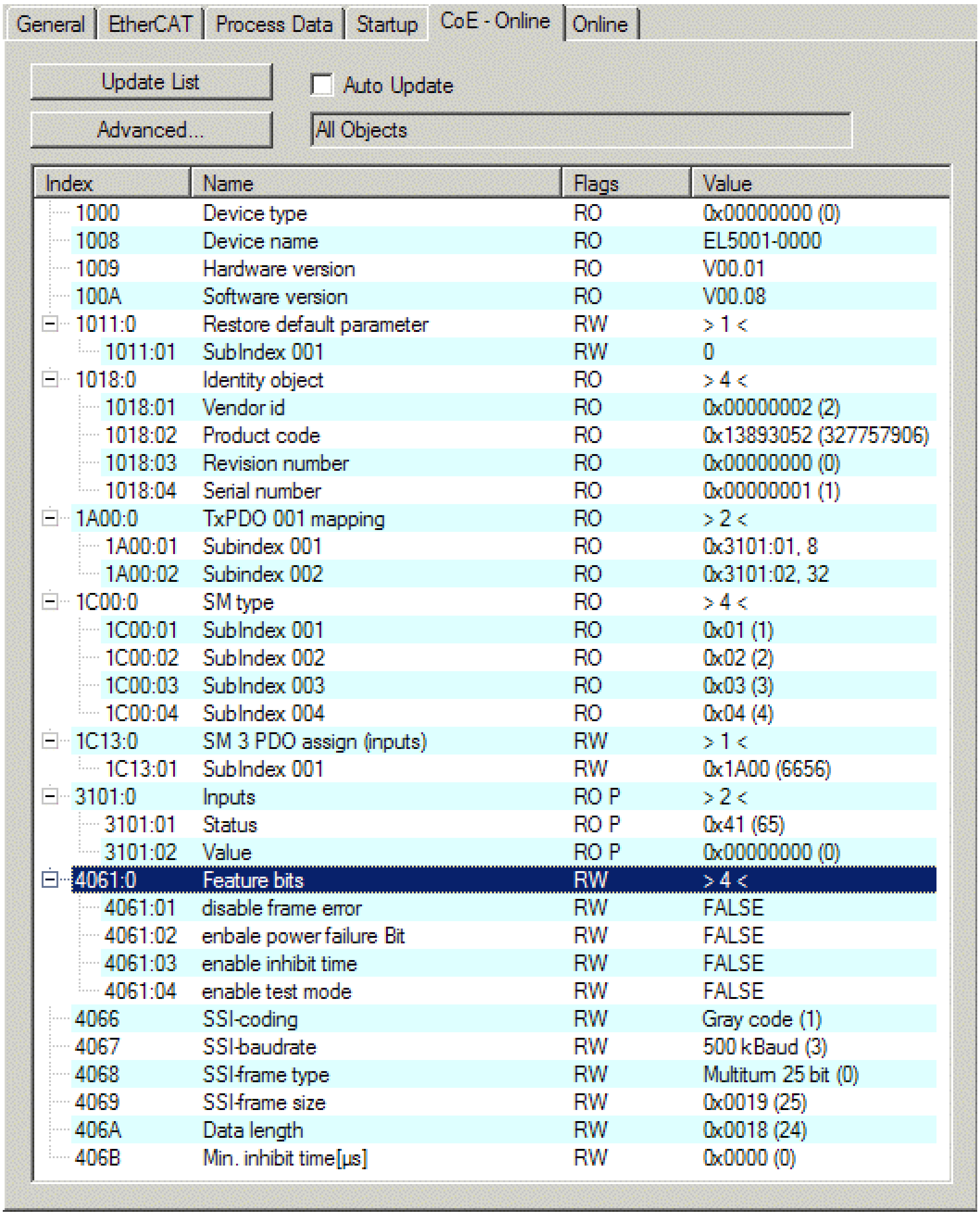
Fig. 111: “Startup” tab

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.



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	Multiturn 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. 112: “CoE - Online” tab

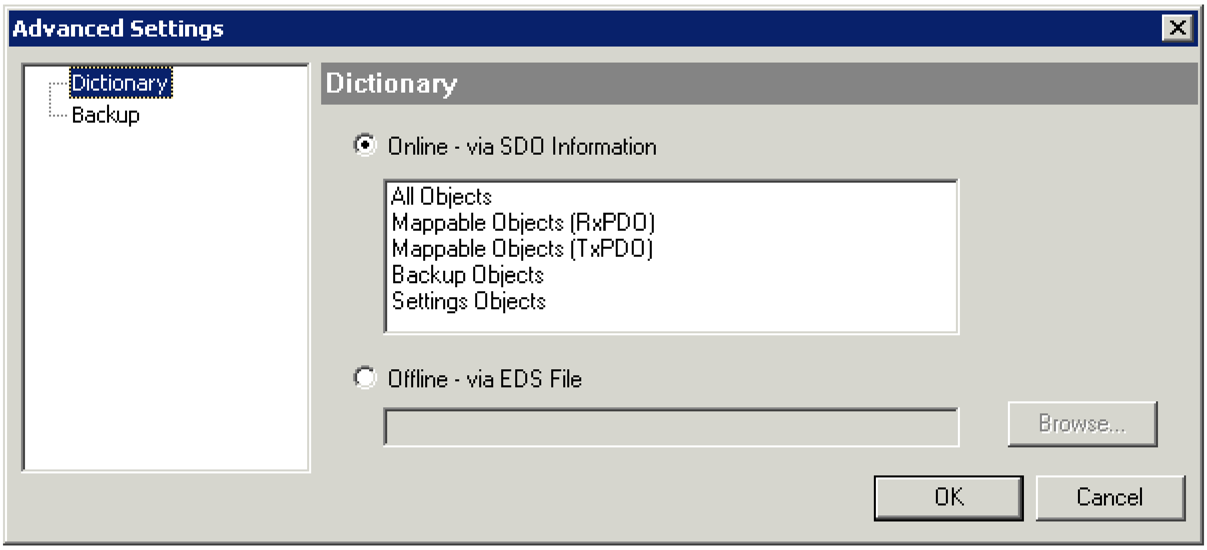
Object list display

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.



● Online - via SDO Information

All Objects
Mappable Objects (RxPDO)
Mappable Objects (TxPDO)
Backup Objects
Settings Objects

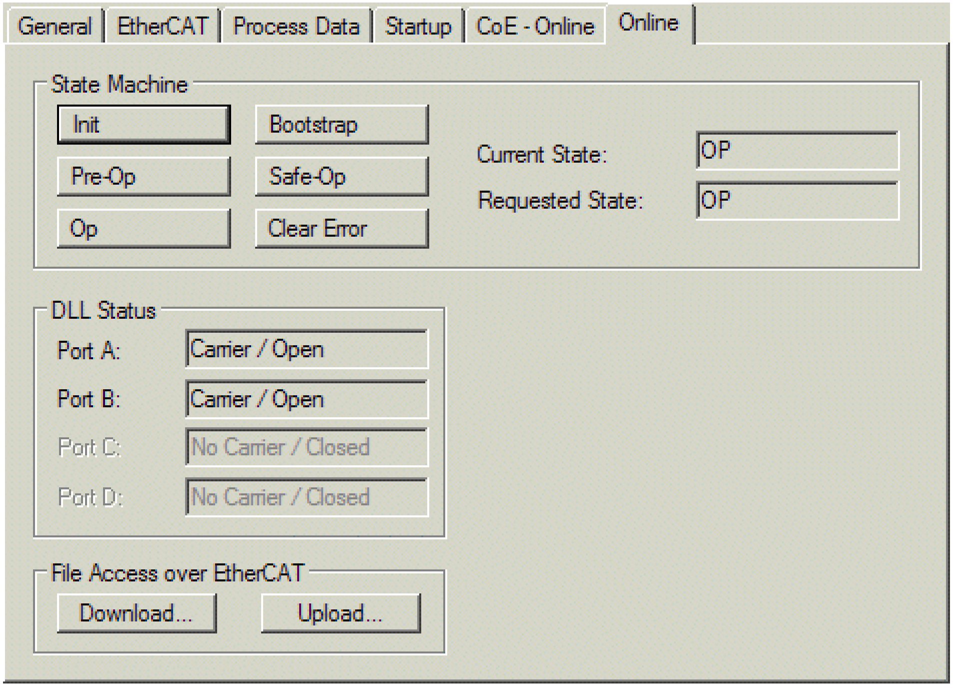
○ Offline - via EDS File

Fig. 113: 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



Current State: OP

Requested State: OP

Port A: Carrier / Open

Port B: Carrier / Open

Port C: No Carrier / Closed

Port D: No Carrier / Closed

Fig. 114: “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:

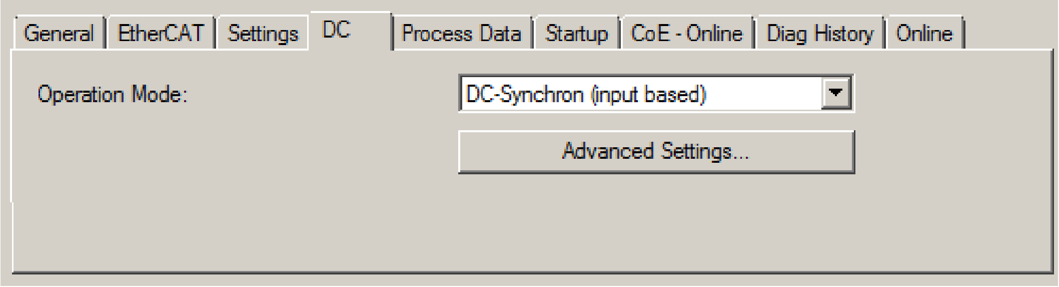
Status	Description
No Carrier / Open	No carrier signal is available at the port, but the port is open.
No Carrier / Closed	No carrier signal is available at the port, and the port is closed.
Carrier / Open	A carrier signal is available at the port, and the port is open.
Carrier / Closed	A carrier signal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



Operation Mode: DC-Synchron (input based)

Fig. 115: “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

5.2.5.1 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 [▶ 91]),

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 [▶ 88] 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.

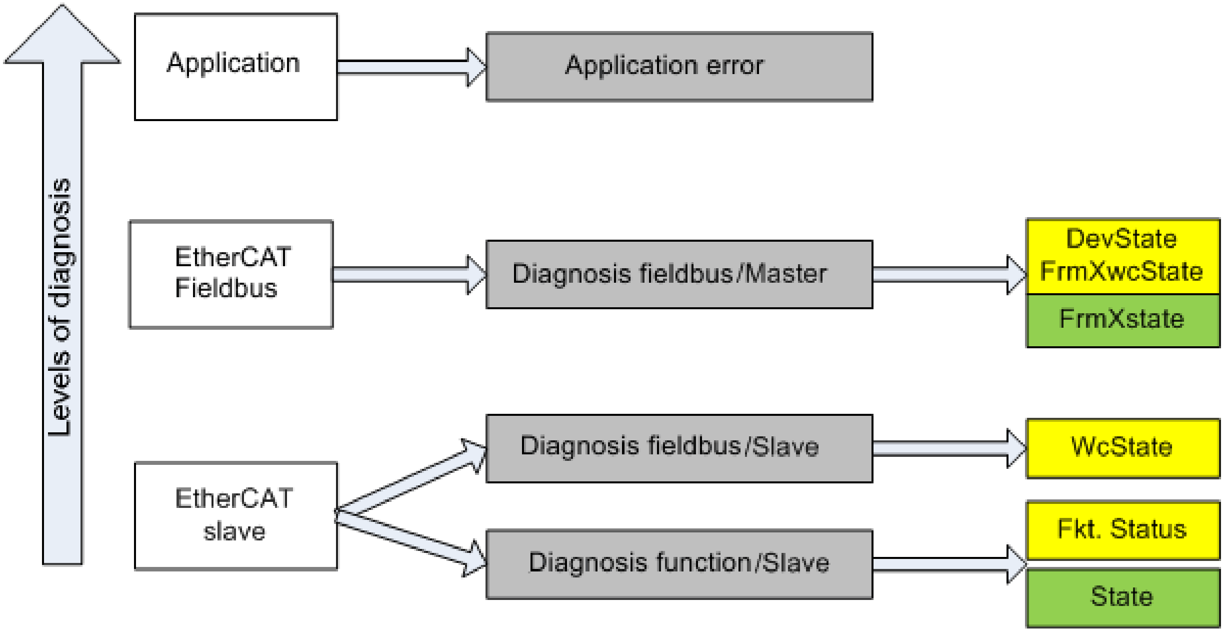


Fig. 116: Selection of the diagnostic information of an EtherCAT Slave

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.

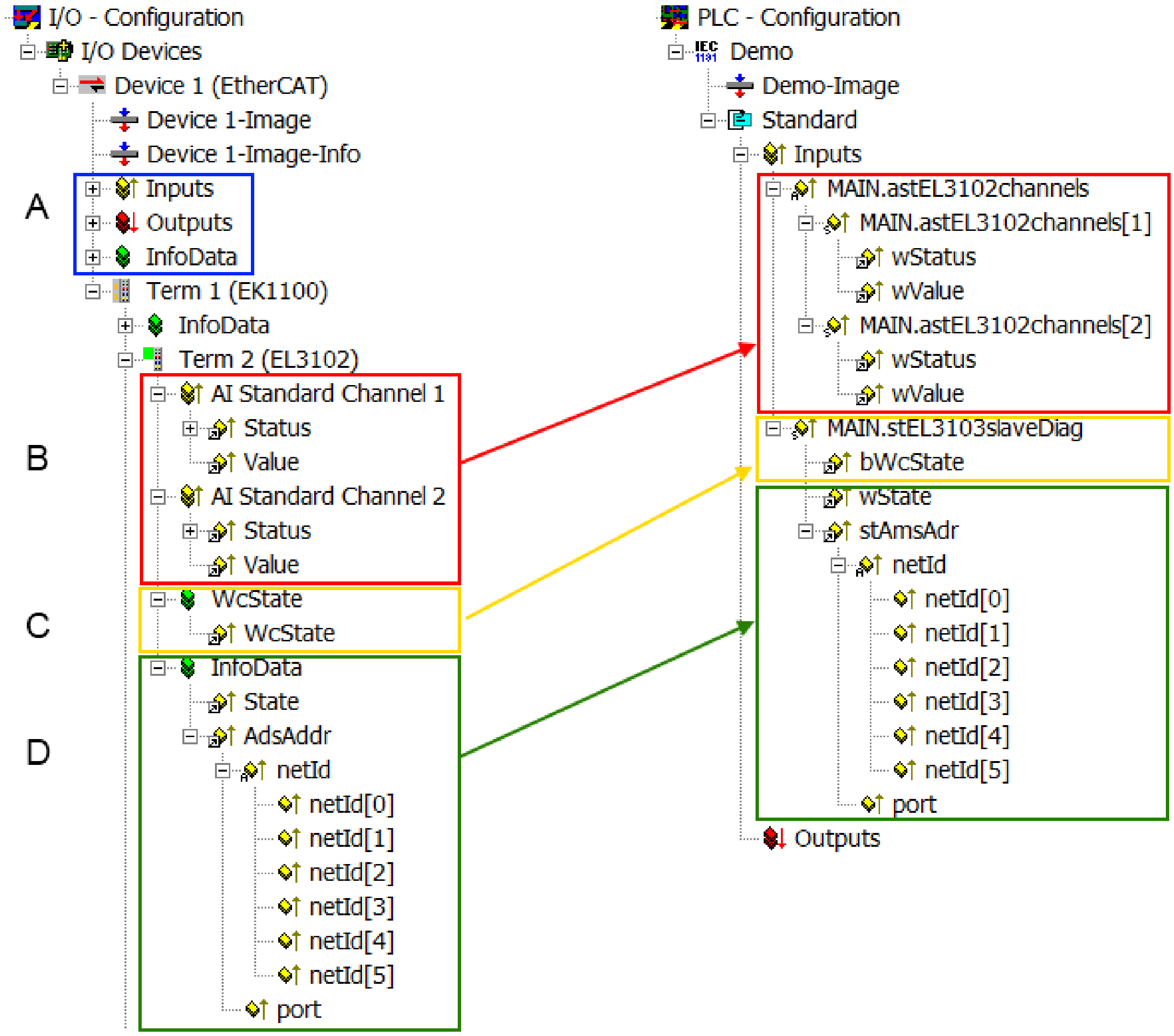


Fig. 117: Basic EtherCAT Slave Diagnosis in the PLC

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.

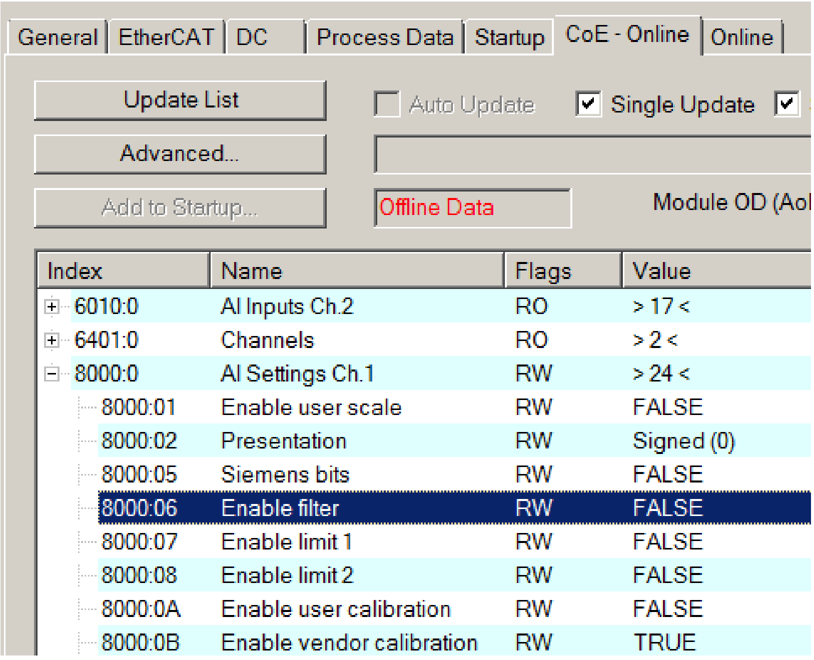
NOTE

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*:



□ Auto Update ☑ Single Update ☑

Index	Name	Flags	Value
⊞ 6010:0	AI Inputs Ch.2	RO	> 17 <
⊞ 6401:0	Channels	RO	> 2 <
⊟ 8000:0	AI Settings Ch.1	RW	> 24 <
8000:01	Enable user scale	RW	FALSE
8000:02	Presentation	RW	Signed (0)
8000:05	Siemens bits	RW	FALSE
8000:06	Enable filter	RW	FALSE
8000:07	Enable limit 1	RW	FALSE
8000:08	Enable limit 2	RW	FALSE
8000:0A	Enable user calibration	RW	FALSE
8000:0B	Enable vendor calibration	RW	TRUE

Fig. 118: EL3102, CoE directory

EtherCAT System Documentation

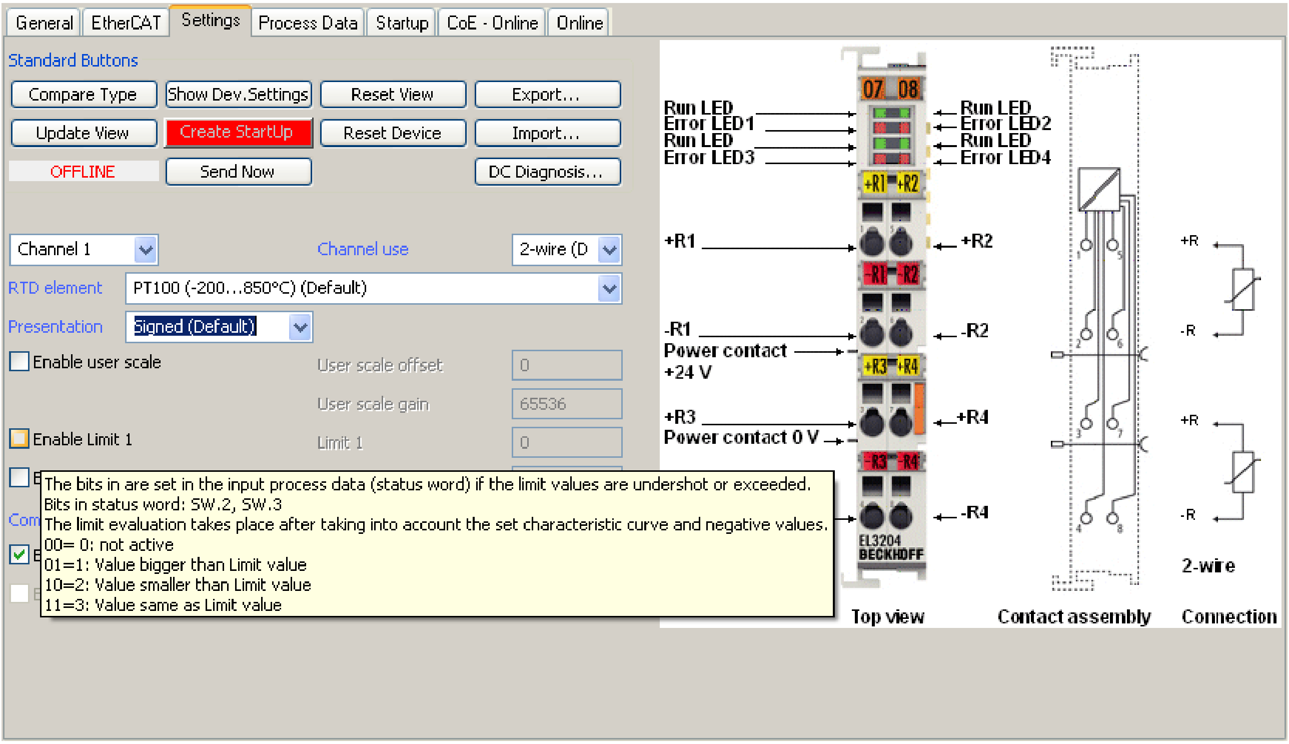
The comprehensive description in the EtherCAT System Documentation (EtherCAT Basics --> CoE Interface) must be observed!

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.



Channel 1

Channel use: 2-wire (D

RTD element: PT100 (-200...850°C) (Default)

Presentation: Signed (Default)

□ Enable user scale — User scale offset: 0

User scale gain: 65536

□ Enable Limit 1 — Limit 1: 0

The bits in are set in the input process data (status word) if the limit values are undershot or exceeded.
Bits in status word: SW.2, SW.3
The limit evaluation takes place after taking into account the set characteristic curve and negative values.
00= 0: not active
01=1: Value bigger than Limit value
10=2: Value smaller than Limit value
11=3: Value same as Limit value

Fig. 119: 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 [▶ 19]" in this connection. Depending how much configuration has to be done, and on the overall communication, booting can take up to a few seconds.

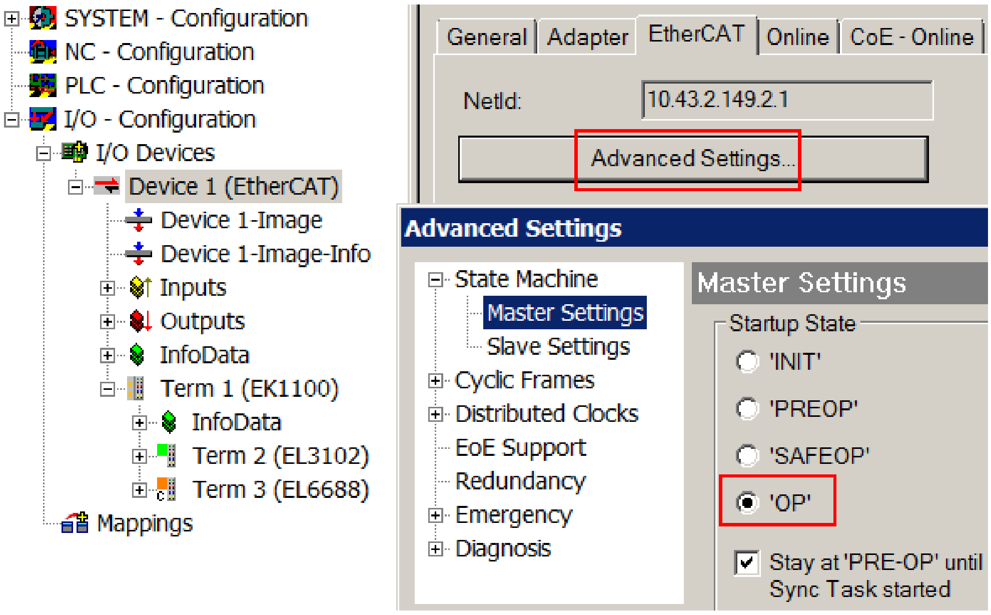
The EtherCAT Master itself must go through these routines when starting, until it has reached at least the OP target state.

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.



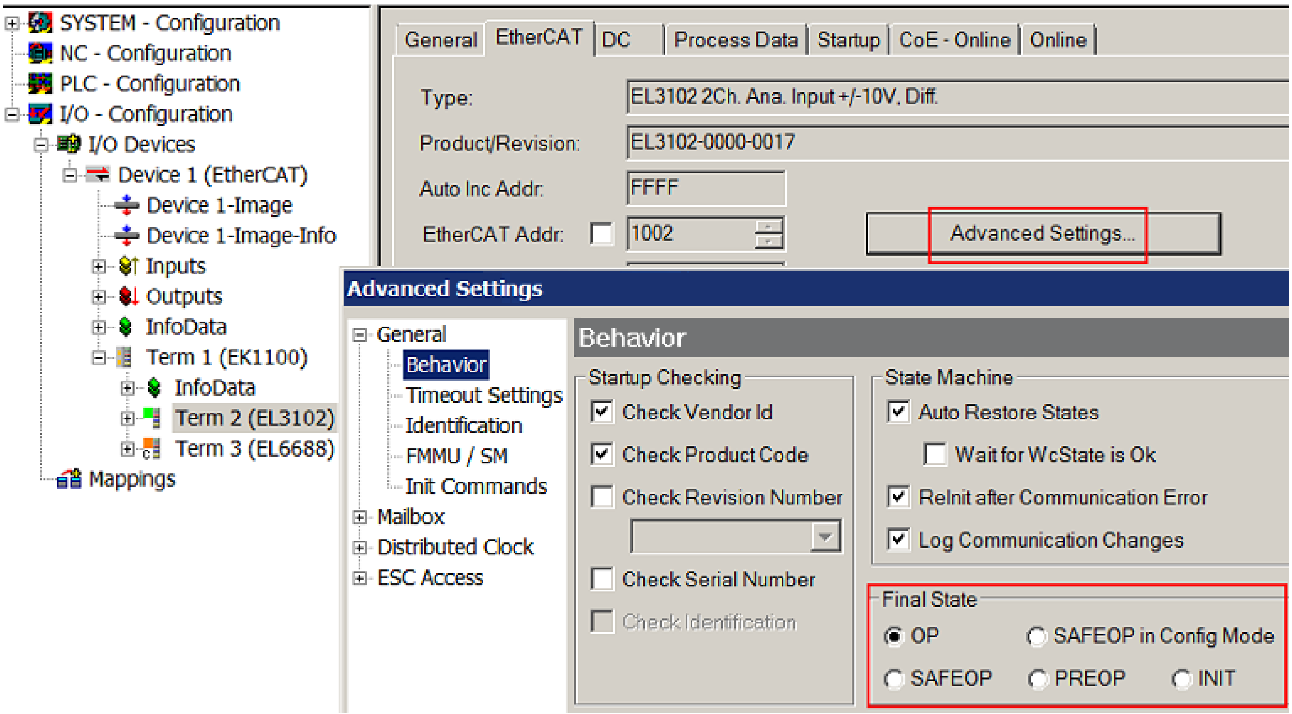
NetId: 10.43.2.149.2.1

Startup State
○ 'INIT'
○ 'PREOP'
○ 'SAFEOP'
● 'OP'

☑ Stay at 'PRE-OP' until Sync Task started

Fig. 120: 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.



Type: EL3102 2Ch. Ana. Input +/-10V, Diff.

Product/Revision: EL3102-0000-0017

Auto Inc Addr: FFFF

EtherCAT Addr: □ 1002

Startup Checking
☑ Check Vendor Id
☑ Check Product Code
□ Check Revision Number
□ Check Serial Number
□ Check Identification

State Machine
☑ Auto Restore States
□ Wait for WcState is Ok
☑ ReInit after Communication Error
☑ Log Communication Changes

Final State
● OP ○ SAFEOP in Config Mode
○ SAFEOP ○ PREOP ○ INIT

Fig. 121: Default target state in the Slave

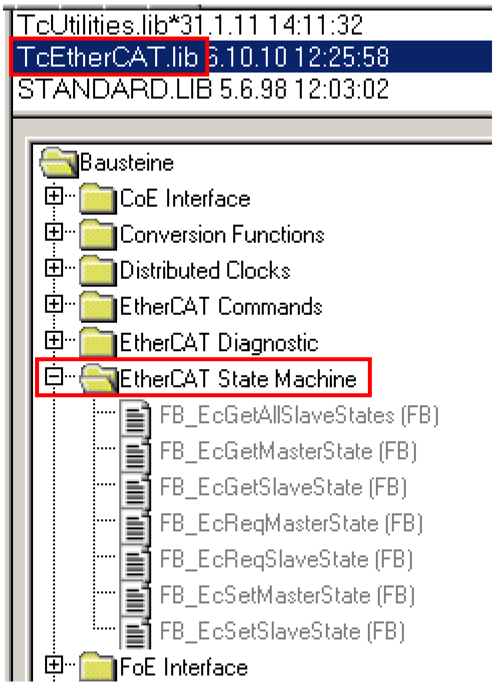
Manual Control

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.



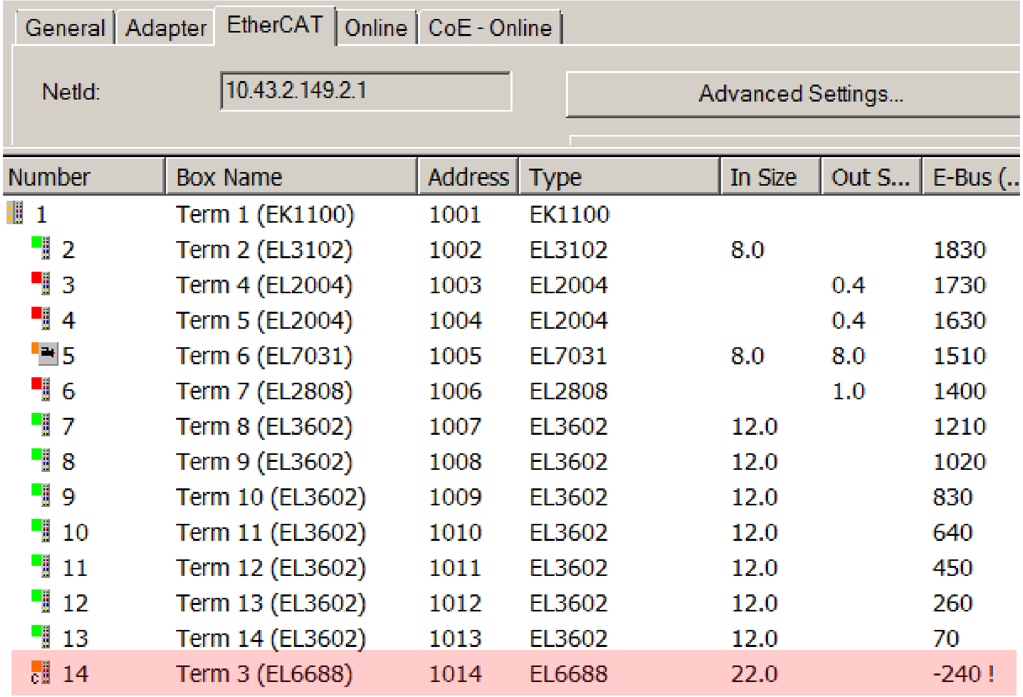
TcUtilities.lib*31.1.11 14:11:32
TcEtherCAT.lib 6.10.10 12:25:58
STANDARD.LIB 5.6.98 12:03:02

Fig. 122: 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.

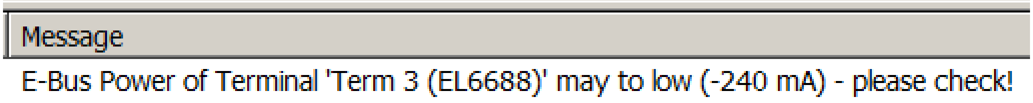


NetId: 10.43.2.149.2.1

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. 123: 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. 124: 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 CoE object overview

5.4.1 Standard CoE objects

Standard CoE objects

The objects with index (hex) 1000 to 1018:04 are referred to as standard CoE objects. They contain general information about the respective device and the respective terminal, such as device name, software and hardware version and manufacturer-specific IDs (serial number, revision, vendor ID and product code).

Index (hex)	Name	Meaning	Data type	Flags	Default value
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	0x00001389 (5001$_{dec}$)

Index (hex)	Name	Meaning	Data type	Flags	Default value
1008:0	Device name	Device name of the EtherCAT slave	STRING	RO	EL6695-0000

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

Index (hex)	Name	Meaning	Data type	Flags	Default value
100A:0	Software version	Firmware version of the EtherCAT slave	STRING	RO	02

Index (hex)	Name	Meaning	Data type	Flags	Default value
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	0x1A243052 (438579282$_{dec}$)
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	0x00000064 (100$_{dec}$)
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$_{dec}$)

5.4.2 Terminal-specific CoE objects

Terminal-specific CoE objects

The terminal-specific CoE objects of the EL6695 bridge terminal start from index (hex) 0x10F4. The objects 0x10F4 and 0x10F5 contain the status and the settings for the external synchronization.

Index (hex)	Name	Meaning	Data type	Flags	Default value
10F4:0	External synchronization status	Information about the synchronization status	UINT8	RO	0x13 (19$_{dec}$)
10F4:01	Sync Mode	Synchronization mode 0 = no synchronization 1 = secondary side is Sync Master 2 = primary side is Sync Master	BIT2	RO P	0x00 (0$_{dec}$)
10F4:0E	Control value update toggle	Bit toggles when a new control value is available	BOOLEAN	RO P	0x00 (0$_{dec}$)
10F4:0F	Time stamp update toggle	Bit toggles when new DC data were supplied	BOOLEAN	RO P	0x00 (0$_{dec}$)
10F4:10	External device not connected	0 = other side is connected to its EtherCAT fieldbus 1 = other side is not connected to its EtherCAT fieldbus	BOOLEAN	RO P	0x00 (0$_{dec}$)
10F4:11	Internal time stamp	Distributed clocks time on the current side	UINT64	RO P	-
10F4:12	External time stamp	Distributed clocks time on the other side (remote side)	UINT64	RO P	-
10F4:13	Control Value for DC Master Clock	Offset for correction of the lower priority reference clock	INT32	RO P	0x00000000 (0$_{dec}$)

Index (hex)	Name	Meaning	Data type	Flags	Default value
10F5:0	External synchronization settings	Setting for synchronizing the EtherCAT bridge	UINT8	RO	0x12 (18$_{dec}$)
10F5:01	Sync master	0: Sync Master is on the primary side 1: Sync Master is on the secondary side	BOOLEAN	RW (PREOP)	0x00 (0$_{dec}$)
10F5:02	32 Bit time stamps	0: 64-bit Timestamps 1: 32-bit Timestamps	BOOLEAN	RW	0x00 (0$_{dec}$)
10F5:11	Control Interval (ms)	Interval in ms for calculating the control value	UINT16	RW	0x0000 (0$_{dec}$)
10F5:12	Additional System Time	Additional DC time for calculating the control value	UINT64	RW	0x0000000000000000 (0$_{dec}$)

Index (hex)	Name	Meaning	Data type	Flags	Default value
1608:0	RxPDO-Map	PDO mapping RxPDO 1 (PDO mapping of the declared output process data)	UINT8	RW	0x00 (0_{dec})
..1608:FF	-		-	-	-
… 161F:00…		-			
161F:FF					

Index (hex)	Name	Meaning	Data type	Flags	Default value
1801:0	TxPDO-Par External Sync Compact	PDO parameter TxPDO 2	UINT8	RO	0x09 (9_{dec})
1801:06	Exclude TxPDOs	Specifies the TxPDOs (index (hex) of TxPDO mapping objects) that must not be transferred together with TxPDO 2	OCTET-STRING[4]	RO	02 1A 03 1A
1801:07	TxPDO State	The TxPDO state is set if it was not possible to correctly read in the associated input data	BOOLEAN	RO P	0x00 (0_{dec})
1801:09	TxPDO Toggle	The TxPDO toggle is toggled with each update the corresponding input data	BOOLEAN	RO P	0x00 (0_{dec})

Index (hex)	Name	Meaning	Data type	Flags	Default value
1802:0	TxPDO-Par External Sync	PDO parameter TxPDO 3	UINT8	RO	0x09 (9_{dec})
1802:06	Exclude TxPDOs	Specifies the TxPDOs (index (hex) of TxPDO mapping objects) that must not be transferred together with TxPDO 3	OCTET-STRING[4]	RO	01 1A 03 1A
1802:07	TxPDO State	The TxPDO state is set if it was not possible to correctly read in the associated input data	BOOLEAN	RO P	0x00 (0_{dec})
1802:09	TxPDO Toggle	The TxPDO toggle is toggled with each update the corresponding input data	BOOLEAN	RO P	0x00 (0dec)

Index (hex)	Name	Meaning	Data type	Flags	Default value
1803:0	TxPDO-Par External Sync (32 Bit)	PDO parameter TxPDO 4	UINT8	RO	0x09 (9_{dec})
1803:06	Exclude TxPDOs	Specifies the TxPDOs (index (hex) of TxPDO mapping objects) that must not be transferred together with TxPDO 4	OCTET-STRING[4]	RO	01 1A 02 1A
1803:07	TxPDO State	The TxPDO state is set if it was not possible to correctly read in the associated input data	BOOLEAN	RO P	0x00 (0_{dec})
1803:09	TxPDO Toggle	The TxPDO toggle is toggled with each update the corresponding input data	BOOLEAN	RO P	0x00 (0_{dec})

Index (hex)	Name	Meaning	Data type	Flags	Default value
1A01:0	TxPDO-Map External Sync Compact	PDO Mapping TxPDO 1	UINT8	RW	0x05 (5_{dec})
1A01:01	SubIndex (hex) 001	12 bit align	UINT32	RW	0x0000:00, 12
1A01:02	SubIndex (hex) 002	PDO Mapping entry (object 0x1801, entry 0x09)	UINT32	RW	0x1801:09, 1
1A01:03	SubIndex (hex) 003	PDO Mapping entry (object 0x1801, entry 0x07)	UINT32	RW	0x1801:07, 1
1A01:04	SubIndex (hex) 004	1 bit align	UINT32	RW	0x0000:00, 1
1A01:05	SubIndex (hex) 005	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x10)	UINT32	RW	0x10F4:10, 1

Index (hex)	Name	Meaning	Data type	Flags	Default value
1A02:0	TxPDO-Map External Sync	PDO Mapping TxPDO 2	UINT8	RW	0x09 (9_{dec})
1A02:01	SubIndex (hex) 001	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x02)	UINT32	RW	0x10F4:01, 2
1A02:02	SubIndex (hex) 002	10 bit align	UINT32	RW	0x0000:00, 10
1A02:03	SubIndex (hex) 003	PDO Mapping entry (object 0x1802, entry 0x09)	UINT32	RW	0x1802:09, 1
1A02:04	SubIndex (hex) 004	PDO Mapping entry (object 0x1802, entry 0x07)	UINT32	RW	0x1802:07, 1
1A02:05	SubIndex (hex) 005	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x0F (Time stamp update toggle))	UINT32	RW	0x10F4:0F, 1
1A02:06	SubIndex (hex) 006	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x10 (External device not connected))	UINT32	RW	0x10F4:10, 1
1A02:07	SubIndex (hex) 007	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x11 (Internal time stamp))	UINT32	RW	0x10F4:11, 64
1A02:08	SubIndex (hex) 008	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x12 (External time stamp))	UINT32	RW	0x10F4:12, 64
1A02:09	SubIndex (hex) 009	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x13 (Reserved))	UINT32	RW	0x10F4:13, 32

Index (hex)	Name	Meaning	Data type	Flags	Default value
1A03:0	TxPDO-Map External Sync	PDO Mapping TxPDO 3	UINT8	RW	0x09 (9_{dec})
1A03:01	SubIndex (hex) 001	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x02)	UINT32	RW	0x10F4:01, 2
1A03:02	SubIndex (hex) 002	10 bit align	UINT32	RW	0x0000:00, 10
1A03:03	SubIndex (hex) 003	PDO Mapping entry (object 0x1803, entry 0x09)	UINT32	RW	0x1803:09, 1
1A03:04	SubIndex (hex) 004	PDO Mapping entry (object 0x1803, entry 0x07)	UINT32	RW	0x1803:07, 1
1A03:05	SubIndex (hex) 005	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x0F (Time stamp update toggle))	UINT32	RW	0x10F4:0F, 1
1A03:06	SubIndex (hex) 006	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x10 (External device not connected))	UINT32	RW	0x10F4:10, 1
1A03:07	SubIndex (hex) 007	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x11 (Internal time stamp))	UINT32	RW	0x10F4:11, 32
1A03:08	SubIndex (hex) 008	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x12 (External time stamp))	UINT32	RW	0x10F4:12, 32
1A03:09	SubIndex (hex) 009	PDO Mapping entry (object 0x10F4 (External synchronization status), entry 0x13 (Reserved))	UINT32	RW	0x10F4:13, 32

Index (hex)	Name	Meaning	Data type	Flags	Default value
1A04:0	Active TxPDOs-Map	PDO Mapping TxPDO 4	UINT32	RW	0x05 (5_{dec})
1A04:01	SubIndex (hex) 01	PDO Mapping entry (object 0xF640:01(Remote Write Cycles u16Count))	UINT32	RW	10 01 40 F6
1A04:02	SubIndex (hex) 02	PDO Mapping entry (object 0xF630:01 Active TxPdo Info PDO 1-8)	UINT32	RW	10 01 30 F6
1A04:03	SubIndex (hex) 03	PDO Mapping entry (object 0xF630:02 Active TxPdo Info PDO 9-16)	UINT32	RW	10 02 30 F6
1A04:04	SubIndex (hex) 04	PDO Mapping entry (object 0xF630:03 Active TxPdo Info PDO 17-24)	UINT32	RW	10 03 30 F6
1A04:05	SubIndex (hex) 05	PDO Mapping entry (object 0xF630:04 Active TxPdo Info PDO 25-32)	UINT32	RW	10 04 30 F6

Index (hex)	Name	Meaning	Data type	Flags	Default value
1A05:0	FoE Info-Map	PDO Mapping TxPDO 5	UINT8	RW	0x01 (1$_{dec}$)
1A05:01		PDO Mapping entry (object 0xF650:01 Active TxPdo Info PDO 25-32)	UINT32	RW	10 01 50 F6

Index (hex)	Name	Meaning	Data type	Flags	Default value
1A08:0	TxPDO-Map	PDO mapping TxPDO 1 (PDO mapping of the declared input process data)	UINT8	RW	0x00 (0$_{dec}$)
..1A08:FF	-		-	-	-
… 1A1F:00…		-			
1A1F:FF					

Index (hex)	Name	Meaning	Data type	Flags	Default value
1C00:0	Sync-manager type	Sync-Manager Type Channels (Mailbox/ Process Data, Read/ Write)	UINT8	RO	0x04 (4$_{dec}$)
1C00:01	SubIndex (hex) 001	Sync-Manager Type Channel 1: Mailbox Write	UINT8	RO	0x01 (1$_{dec}$)
1C00:02	SubIndex (hex) 002	Sync-Manager Type Channel 2: Mailbox Read	UINT8	RO	0x02 (2$_{dec}$)
1C00:03	SubIndex (hex) 003	Sync-Manager Type Channel 3: Process Data Write (Outputs)	UINT8	RO	0x03 (3$_{dec}$)
1C00:04	SubIndex (hex) 004	Sync-Manager Type Channel 4: Process Data Read (Inputs)	UINT8	RO	0x04 (4$_{dec}$)

Index (hex)	Name	Meaning	Data type	Flags	Default value
1C12:0	RxPDO assign	PDO Assign Outputs	UINT8	RW	0x00 (0$_{dec}$)
1C32:01	SubIndex (hex) 001	1st allocated RxPDO (contains the index (hex) of the associated RxPDO mapping object)	UINT16	RW	0x0000 (0$_{dec}$)

Index (hex)	Name	Meaning	Data type	Flags	Default value
1C13:0	TxPDO assign	PDO Assign Inputs	UINT8	RW	0x01 (1$_{dec}$)
1C13:01	SubIndex (hex) 001	1st allocated TxPDO (contains the index (hex) of the associated TxPDO mapping object)	UINT16	RW	0x1A01 (6657$_{dec}$)
1C13:02	SubIndex (hex) 002	2nd allocated TxPDO (contains the index (hex) of the associated TxPDO mapping object)	UINT16	RW	0x0000 (0$_{dec}$)

Index (hex)	Name	Meaning	Data type	Flags	Default value
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: Synchronous with SM 2 event 2: DC-Mode - Synchronous with SYNC0 Event 3: DC-Mode - Synchronous 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	0x00000000 (0$_{dec}$)
1C32:03	Shift time	Time between SYNC0 event and output of the outputs (in ns, DC mode only)	UINT32	RW	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	0x00000000 (0$_{dec}$)
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	Maximum 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$_{dec}$)

Index (hex)	Name	Meaning	Data type	Flags	Default value
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 (hex)	Name	Meaning	Data type	Flags	Default value
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	0x00000000 (0$_{dec}$)
1C33:03	Shift time	Time between SYNC0 event and reading of the inputs (in ns, only DC mode)	UINT32	RW	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	0x00000000 (0$_{dec}$)
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	Maximum 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$_{dec}$)

Index (hex)	Name	Meaning	Data type	Flags	Default value
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})

5.4.3 Profile-specific CoE objects

Reading of CoE objects 0x6000 and 0x7000

The data that are output during reading of CoE objects 0x6000 and 0x7000 are not the real process data. The actual data can only be read indirectly via PDO allocation.

Index (hex)	Name	Meaning	Data type	Flags	Default value
6000:0	Input Data	Declared input process data (dynamically created)	UINT8	RO	0x00 (0_{dec})
6000:01					
…					
6000:FF					

Index (hex)	Name	Meaning	Data type	Flags	Default value
7000:0	Output Data	Declared output process data (dynamically created)	UINT8	RO	0x00 (0_{dec})
7000:01					
…					
7000:FF					

Index (hex)	Name	Meaning	Data type	Flags	Default value
F000:0	Modular device profile	General information for the modular device profile	UINT8	RO	0x02 (2_{dec})
F000:01	Module Index (hex) distance	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	0x0001 (1_{dec})

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

Index (hex)	Name	Meaning	Data type	Flags	Default value
F010:0	Module list	Max. subIndex (hex)	UINT8	RW	0x01 (1_{dec})
F010:01	SubIndex (hex) 001	-	UINT32	RW	0x00000000 (0_{dec})

Currently 64 PDOs, 2 bit per Tx-PDO

bit 0 → 1=mapping present (p)

bit 1 → 1=mapping active (a)

Index (hex)	Name															
F630	Active TxPdo															
Mapping:	a	p	a	p	a	p	a	p	a	p	a	p	a	p	a	p
0	0															
1	1A07		1A06		1A05		1A04		1A03		1A02		1A01		1A00	
2	1A0F		1A0E		1A0D		1A0C		1A0B		1A0A		1A09		1A08	
3	1A17		1A16		1A15		1A14		1A13		1A12		1A11		1A10	
4	1A1F		1A1E		1A1D		1A1C		1A1B		1A1A		1A19		1A18	
…	..		..		..		..		..		..		..		..	
8	1A3F		1A3E		1A3D		1A3C		1A3B		1A3A		1A39		1A38	

Index (hex)	Bit	Name	Meaning	Comment
F800:0	-	Device Config		
F800:01	0x8000	AoE	1 = protocol disabled	
	0x4000	EoE		
	0x2000	FoE		
	0x1000	SoE		
	0x0800	VoE		
	0x0400	other		
	0x0200	-		If 0, the system checks whether the MBX protocol ECAT-EEPROM is present
	…			
	0x0002	-		Suppress error message
	0x0001	-		1: If not routed: error message suppressed
	…			
0xF800:02	0x0100	FoE Buffer	Enable	
	…			
	0x0002	OP State	Deactivate dependence on other side	Deactivate dependence of the operating state on the secondary side (only device emulation)
	0x0001	OP State		Deactivate dependence of the operating state on the primary side (only device emulation)
0xF800:03	0x2000		SW Restart	Reboot during state change from Any to Init (A → I)
	…			

Index (hex)	Name	Meaning	Data type	Flags	Default value
F820:0	ADS Server Settings	Destination Net ID / Port	Object	RW	0x02 (2_{dec})
F820:01	Net ID	Destination Net ID	Array [0..5]	RW	0x00, ..
F820:02	Port	Destination Port	UINT16	RW	0x0000

Index (hex)	Name	Meaning	Data type	Flags	Default value
F821:0	EL6695 ADS Settings	Source Net ID / Port	Object	RO	0x02 (2_{dec})
F821:01	Net ID	Source Net ID	Array [0..5]	RO	0x00, ..
F821:02	Port	Source Port	UINT16	RO	0x0000

Index (hex)	Name	Meaning	Data type	Flags	Default value
FA20:0	Device Diag	Contains diagnostic information on status, CPU and heap load, information on sent data packets and internal copy time	UINT16	RD	0x1D (29$_{dec}$)
FA20:01	Status		UINT16	RD	
FA20:02	CPU usage [%]		UINT16	RD	
FA20:03	Heap Usage [%]		UINT16	RD	
FA20:04	AOE Packets		UINT16	RD	
FA20:05	EOE Packets		UINT16	RD	
FA20:06	FOE Packets		UINT16	RD	
FA20:07	SOE Packets		UINT16	RD	
FA20:08	VOE Packets		UINT16	RD	
FA20:09	Other Packets		UINT16	RD	
FA20:0A	Mbx Info		UINT16	RD	
FA20:0B	PD Copy time (my)	Time in µs	UINT16	RD	
FA20:0C	PD Copy time (remote)		UINT16	RD	
FA20:0D..	Info 2		UINT16	RD	
FA20:1D	Info 18		UINT16	RD	

6 Function and operating modes

Operating modes for EL terminals define a basic operating principle depending on the process data settings.

6.1 Basic function principles

The EL6695 EtherCAT bridge terminal enables synchronous real-time data exchange between EtherCAT strands and two, possibly different masters. Asynchronous communication via various acyclic protocols such as AoE, FoE, EoE, VoE etc. is also supported.

The EL6695 offers three basic functions:

- Synchronous PDO data exchange
- Asynchronous data exchange
- Distributed clock synchronization

During synchronous data exchange of synchronous process data, the EtherCAT master copies predefined process data from one bridge side to the other. Status variables on both sides provide information about missing data and the status of the other side.

Synchronization of distributed clocks is possible in both directions. It is important to define which side contains the higher-level reference clock, the so-called grandmaster clock. The EL6695 then transfers time values to the subordinate EtherCAT master on the other side, so that it can adjust its EtherCAT system time, if it supports this feature.

The EL6695 differs from the EL6692 (which will continue to be available) in terms of extended functions such as flexible CoE configuration, a device emulation option and in particular a significant increase in data throughput.

A user-friendly configuration interface is available in the TwinCAT System Manager for TwinCAT 3 (in preparation for TwinCAT 2). This extension is not absolutely necessary for general operation, although it is required for EmulationMode.

The internal energy supply for the EL6695 is provided redundantly from the primary side (E-bus) or the secondary side (RJ45). The EL6695 is fully functional on both EtherCAT sides, as soon as power is supplied on at least one of the two sides. The power supply for the secondary side (RJ45) is via an external connection, the primary side is supplied via the E-bus. If both supplies are active, the 24 V supply is used preferentially.

The EL6695 effectively consists of two complete EtherCAT slaves in one housing: the EL6695-0000 on the primary side (E-bus), and the EL6695-0002 on the secondary side with network cable connection (RJ45).

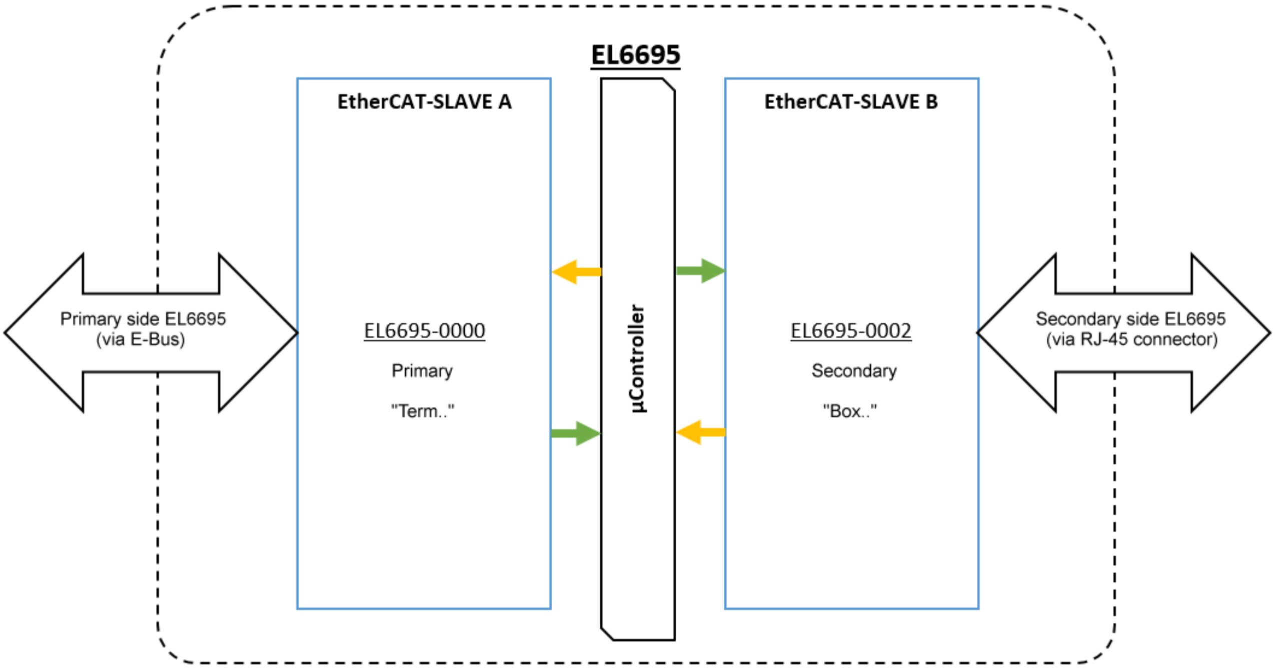
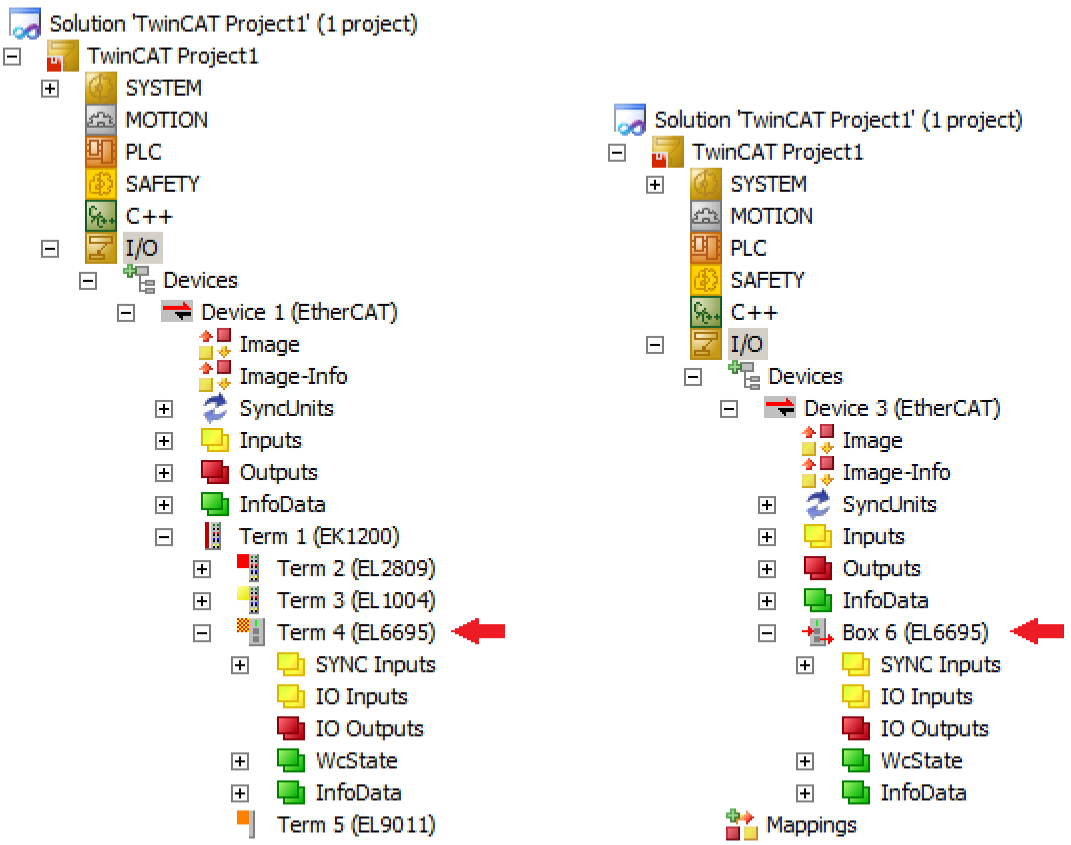


Fig. 125: Basic functionality of the bridge

In terms of functionality the two bridge sides are equivalent, i.e. all functions can be operated or used from both sides. Nevertheless, the terms “primary” for the E-bus side and “secondary” for the RJ45 side continue to be used for localization reasons, even though for the EL6695 primary/secondary do not relate to any weighting or ranking.

6.1.1 Accessing the EL6695 via TwinCAT

The context menu (right-click, then select **Add New Item...**) can be used to create a configuration or integrate the EL6695 bridge terminal in a TwinCAT project. Alternatively, right-click on the respective EtherCAT Device and select the Scan function from the context menu of device n (EtherCAT), which automatically reads all connected devices and boxes and integrates them into the structure (see diagram):



- Solution 'TwinCAT Project1' (1 project)
  - TwinCAT Project1
    - SYSTEM
    - MOTION
    - PLC
    - SAFETY
    - C++
    - I/O
      - Devices
        - Device 1 (EtherCAT)
          - Image
          - Image-Info
          - SyncUnits
          - Inputs
          - Outputs
          - InfoData
          - Term 1 (EK1200)
            - Term 2 (EL2809)
            - Term 3 (EL1004)
            - Term 4 (EL6695)
              - SYNC Inputs
              - IO Inputs
              - IO Outputs
              - WcState
              - InfoData
            - Term 5 (EL9011)

- Solution 'TwinCAT Project1' (1 project)
  - TwinCAT Project1
    - SYSTEM
    - MOTION
    - PLC
    - SAFETY
    - C++
    - I/O
      - Devices
        - Device 3 (EtherCAT)
          - Image
          - Image-Info
          - SyncUnits
          - Inputs
          - Outputs
          - InfoData
          - Box 6 (EL6695)
            - SYNC Inputs
            - IO Inputs
            - IO Outputs
            - WcState
            - InfoData
      - Mappings

Fig. 126: After “Scan”: EL6695 in the terminal segment (left) and via EtherCAT (via RJ-45 X1) as box (right)

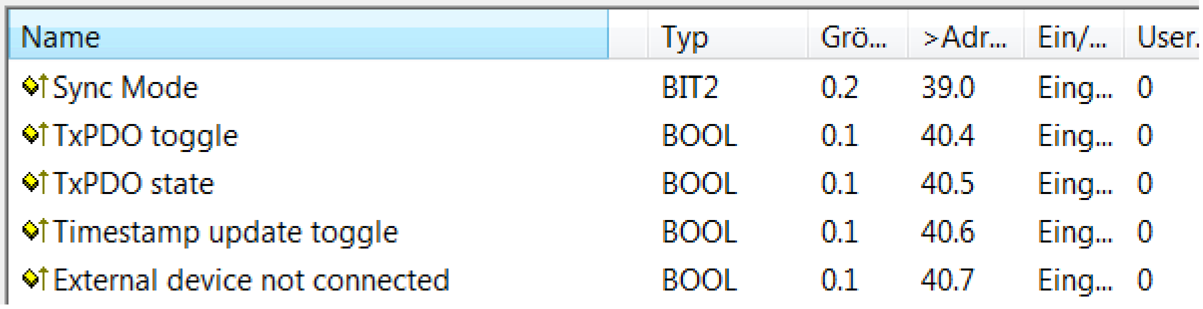
On the E-bus side, the EL6695 can be seen as a “Terminal” in the terminal segment. Via the RJ45 connection it is represented as a “Box” in the configuration (cf. EtherCAT slave). The initial state of the EL6695 bridge terminal after reading/creation is:

- Free Run => DC support disabled
- No process data variables preconfigured
- No distributed clocks information

6.2 Compatibility with EL6692

The EL6695 should essentially be seen as a new development, although it supports an operating mode that is compatible with the EL6692, in which case it uses the same default process data (status/diagnostics). However, please note that, due to its different time characteristics, the EL6695 is not fully exchange-compatible with the EL6692.

By selecting the diagnostic data in 0x1A02 or 0x1A03, the diagnostic data familiar from the EL6692 such as SyncMode, TxPdoToggle, TxPDO state, “Timestamp update toggle” and “External device not connected” can be displayed.



Name	Typ	Grö...	>Adr...	Ein/...	User.
Sync Mode	BIT2	0.2	39.0	Eing...	0
TxPDO toggle	BOOL	0.1	40.4	Eing...	0
TxPDO state	BOOL	0.1	40.5	Eing...	0
Timestamp update toggle	BOOL	0.1	40.6	Eing...	0
External device not connected	BOOL	0.1	40.7	Eing...	0

Fig. 127: EL6695 compatibility mode (default PDO)

6.3 State machine EL6695

Both sides (primary and secondary) of the EL6695 bridge terminal support the following EtherCAT states: INIT, PreOP, SafeOP, OP, BOOTSTRAP.

Status changes on one side do not affect the status of the other side. In EmulationMode this is not the case, see there.

INIT state

The EL6695 bridge terminal processes a one-sided, “normal” INIT state request on the respective side without effect on the other side. After a fundamental terminal configuration change (e.g. through FW update, change in device emulation, change in partial object directory) the terminal restarts on **both** sides.

6.4 Cyclic process data PDO

Two terms are introduced for the EL6695, for the purpose of differentiation

- “Symmetric PDO mapping”: the same process data, in terms of size and sequence, are created on both sides
- “Selective PDO mapping”: one side loads a maximum volume of process data into the EL6695, from which the master on the other side selects a subset and requests it as cyclic process data.

Key data (as of 2015-06, FW04, TwinCAT 3.1 b4018.4)

● Max. 255 variables per direction (max. 255 entries in 0x1A08:nn and 0x1608:nn managed through TwinCAT, if PDOs are created manually)

● Maximum total PDO size: MTU (~1500 bytes), i.e. 1 Ethernet frame for direction, corresponding to approx. 1408 bytes user data. Larger quantity on request, requires adaptation in TwinCAT

● Minimum PDO size is 1 byte, bit-PDOs are not permitted

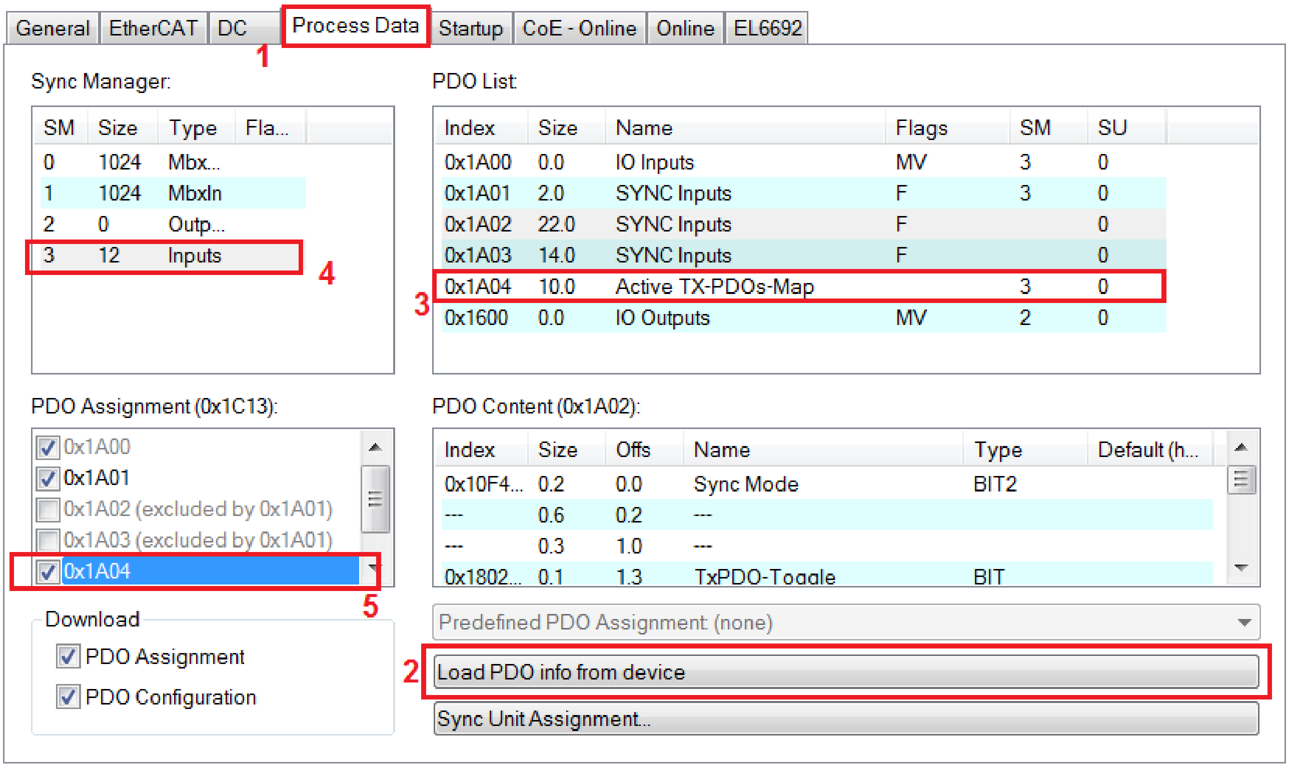
Changes in the configuration do not take effect until the respective EL6695 side is restarted.

For each 6000-type object, the system searches for a corresponding 7000-type object on the other side – e.g. 6001:05 → 7001:05. The size of the object on the other side (7000-type) must be equal to or greater than the 6000-type object. In so far it is also possible to copy a subset 7000 → 6000, cf. “selective PDO mapping”.

6.4.1 Flow Control

The flow control has to accept the application, i.e. the EL6695 does not employ a handshake mechanism. However, in both PDO modes a counter can be displayed, which is incremented with each write access from the other side: CoE object 0xF640. To ensure that this counter can be mapped as cyclic PDO, it should be mapped as follows via a startup entry:

In order to be able to map the CoE object 0xF640, first all online CoE objects should be displayed, as described in section [▸ 23]. The PDO data of the device can then be reloaded via the Process Data tab, so that the PDO list shows the entry Active TX PDOs Map. If the entry Inputs is now selected under Sync Manager, the object 0x1A04 can be ticked under PDO assignment, and the counter of CoE object 0xF640 is mapped into the process image.



Sync Manager:

SM	Size	Type	Fla...
0	1024	Mbx...	
1	1024	MbxIn	
2	0	Outp...	
3	12	Inputs	

PDO List:

Index	Size	Name	Flags	SM	SU
0x1A00	0.0	IO Inputs	MV	3	0
0x1A01	2.0	SYNC Inputs	F	3	0
0x1A02	22.0	SYNC Inputs	F		0
0x1A03	14.0	SYNC Inputs	F		0
0x1A04	10.0	Active TX-PDOs-Map		3	0
0x1600	0.0	IO Outputs	MV	2	0

PDO Assignment (0x1C13):

- ☑ 0x1A00
- ☑ 0x1A01
- □ 0x1A02 (excluded by 0x1A01)
- □ 0x1A03 (excluded by 0x1A01)
- ☑ 0x1A04

PDO Content (0x1A02):

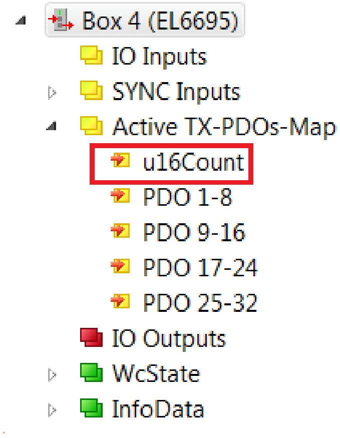
Index	Size	Offs	Name	Type	Default (h...
0x10F4...	0.2	0.0	Sync Mode	BIT2	
---	0.6	0.2	---		
---	0.3	1.0	---		
0x1802...	0.1	1.3	TxPDO-Toggle	BIT	

Download

- ☑ PDO Assignment
- ☑ PDO Configuration

Predefined PDO Assignment: (none)

Fig. 128: Loading of PDO data from the device creates the entry Active TX PDOs Map



- Box 4 (EL6695)
  - IO Inputs
  - SYNC Inputs
  - Active TX-PDOs-Map
    - u16Count
    - PDO 1-8
    - PDO 9-16
    - PDO 17-24
    - PDO 25-32
  - IO Outputs
  - WcState
  - InfoData

Fig. 129: PDO structure

Data throughput (example)

The following diagram illustrates the data throughput of the EL6695 in standard configuration and with two optimization methods, i.e. optimization through synchronization and optimization through a separate IO update, including synchronization:

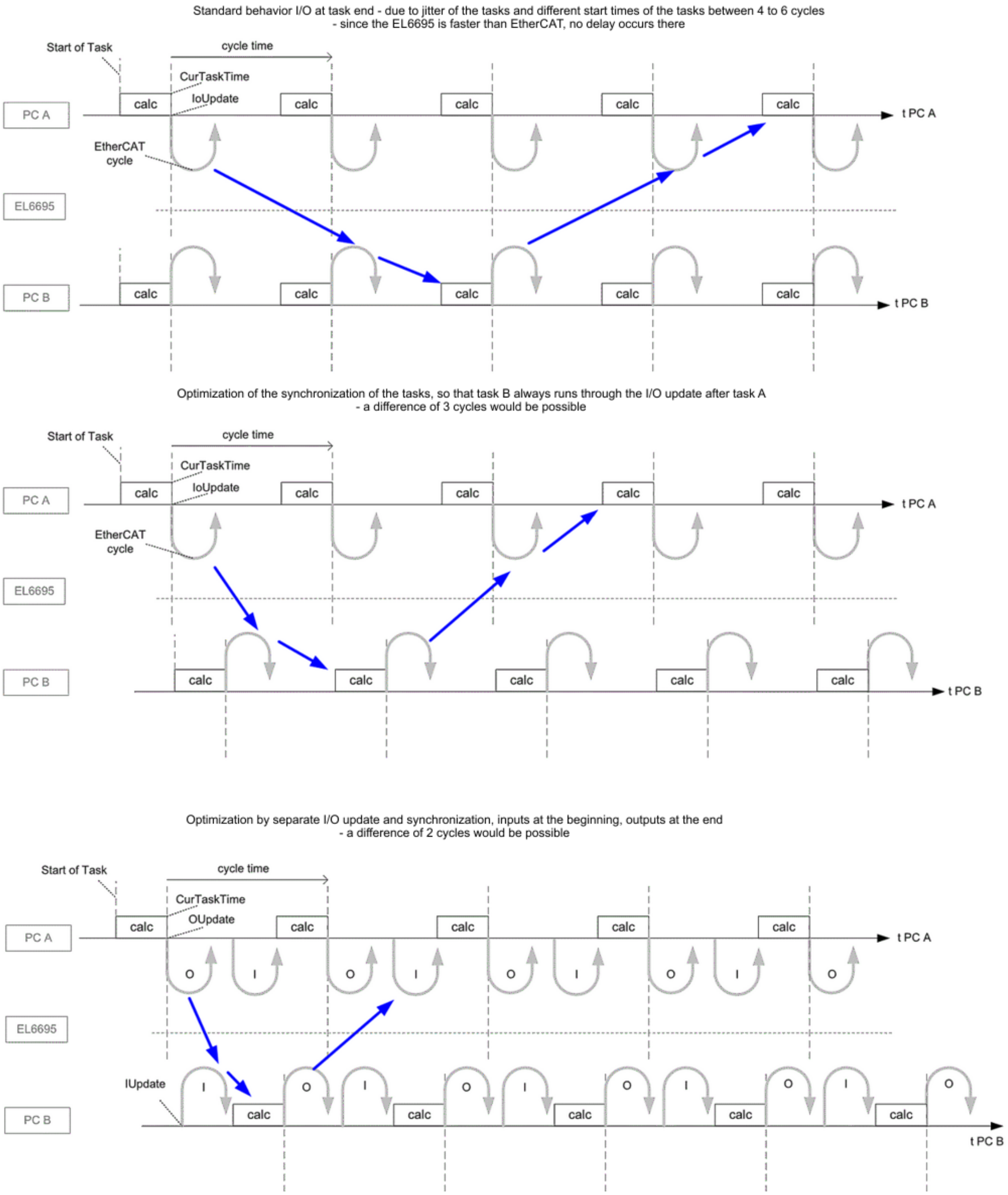

Standard behavior I/O at task end - due to jitter of the tasks and different start times of the tasks between 4 to 6 cycles
- since the EL6695 is faster than EtherCAT, no delay occurs there
Start of Task
cycle time
CurTaskTime
IoUpdate
calc
PC A
t PC A
EtherCAT cycle
EL6695
PC B
t PC B
Optimization of the synchronization of the tasks, so that task B always runs through the I/O update after task A
- a difference of 3 cycles would be possible
Optimization by separate I/O update and synchronization, inputs at the beginning, outputs at the end
- a difference of 2 cycles would be possible
OUpdate
IUpdate
O
I

Fig. 130: Data throughput: standard and with optimizations

The time it takes to transport the configured process data from one EtherCAT side to the other depends on the data quantity.

The internal data transport is triggered by reading on one side. The figure below illustrates the sequence:
- SyncManagers 2+3 operate in 3-buffer mode
- If an EtherCAT master fetches data from SyncManager 3, buffer 3, via an EtherCAT frame (A), immediately after this read process is complete (B) the EL66995 starts copying new data from the other side into the next free buffer, in this case buffer 1. The “Copy Time” this process takes can be read online.
My/remote refers to the “I” side, depending on whether the current online CoE is that of the primary side or the secondary side.

- The data remain there until they are retrieved by EtherCAT A.
- If the cycle time is long, relatively old data may remain there for nearly a whole cycle. The transport timing can be shifted through a CoE setting. Through this delay the system can be optimized such that data stored by system B in SM2 are transferred internally shortly afterwards to side A, SM3, in which case the data which the passing EtherCAT frame A then fetches are relatively fresh.
It is important to ensure minimum system jitter on sides A and B.

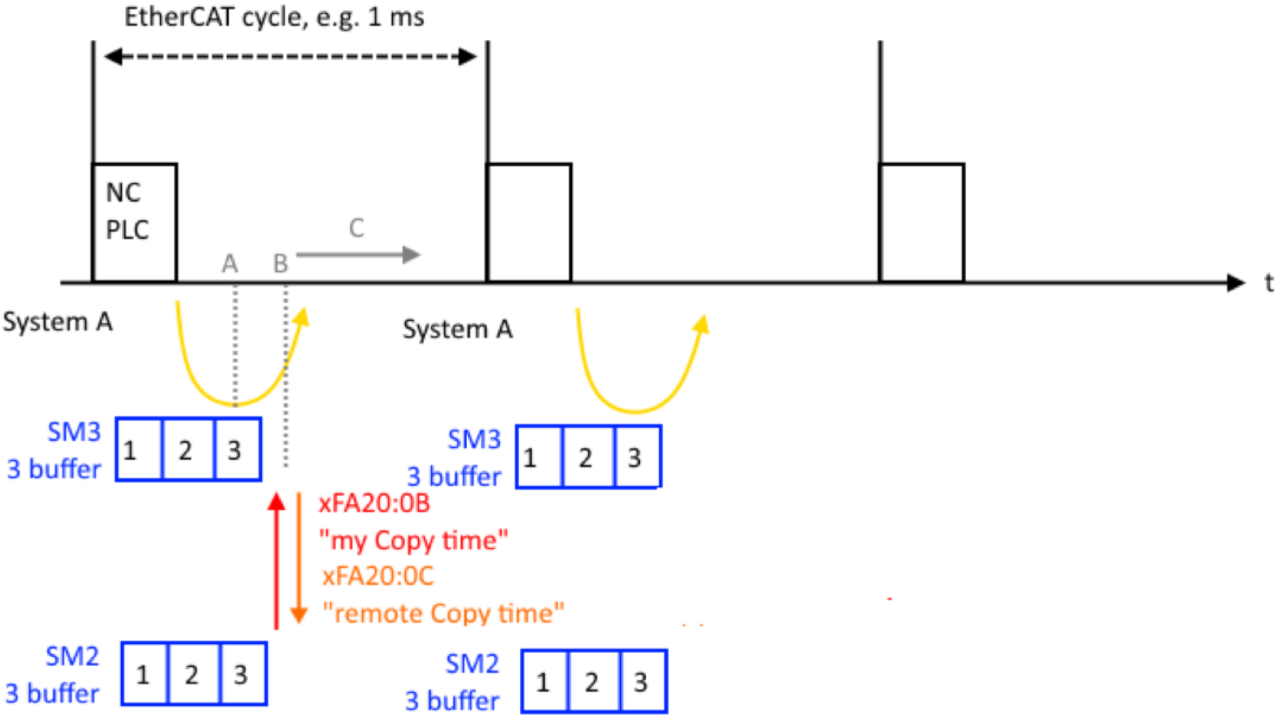


Fig. 131: EL6695 data transport sequence

The example below illustrates a measurement with the following configuration:
- PLC on the primary side sends a PDO set
- EL6695 transports the set to the other side
- PLC on the primary side fetches the data and may resend it in modified form
- EL6695 transports the data to the other side
- PLC of the primary side fetches the data and counts the PLC cycles up to now

In this example no optimizations were carried out (PDO delay, DC synchronization).

Number of PDO bytes	Task cycle time	Number of task cycles for both directions (average values)	Resulting transfer time (one direction)	Copying times for the input and output data within the EL6695; from CoE object 0xFA20 Device Diag. (average values)
200	50 µs	4.4	141.1 µs	14.3 µs
	100 µs	3.03	151.5 µs	14.3 µs
1400	150 µs	8.9	667.5 µs	42.9 µs
	200 µs	6.0	600 µs	42.3 µs

To realize hard-coupled cyclic data transfer, distributed clocks coupling of the two EtherCAT sides is advisable, in order to avoid beat effects during the data transport.

Notice: In practice, the EL6695 always involves two fieldbuses with their cycle times. Optimum timing is required (DC synchronization, shift times adjusted) in order to realize a PC A ->PC B transport time that is as short as that of a direct RealtimeEthernet link with publisher/subscriber. Even then, optimization is only possible for one direction, and in addition there is a delay due to the internal transport time.

6.4.2 Symmetric PDO mapping

Creating general process data / simple data exchange (symmetric PDO mapping)

Procedure:

Right-click on **IO Inputs** or **IO Outputs** within the EL6695 directory tree to show the context menu. Select **Insert Variable…** (TwinCAT 2) or **Add New Item** (TwinCAT 3) to create new variables / process data (see following figure).

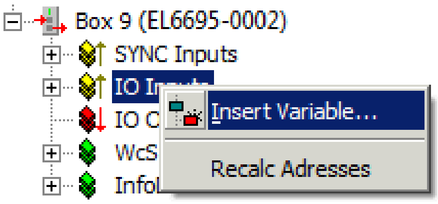
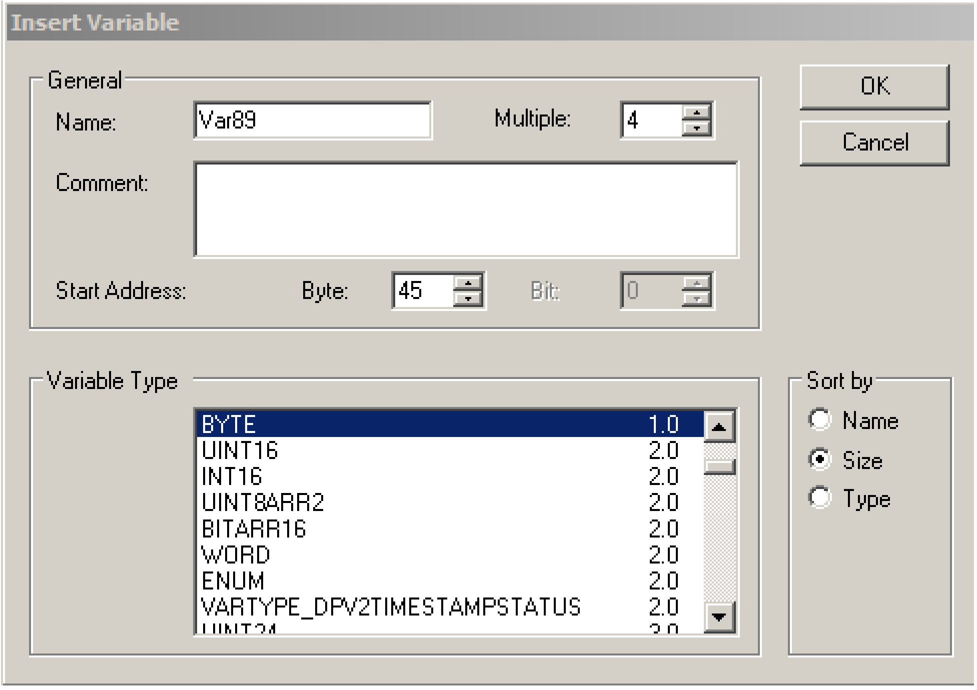


Fig. 132: Context menu for IO inputs: Adding new variables



Name: Var89
Multiple: 4
Start Address: Byte: 45 Bit: 0

Sort by: ○ Name ● Size ○ Type

BYTE	1.0
UINT16	2.0
INT16	2.0
UINT8ARR2	2.0
BITARR16	2.0
WORD	2.0
ENUM	2.0
VARTYPE_DPV2TIMESTAMPSTATUS	2.0
UINT24	3.0

Fig. 133: Dialog for adding new variables (here 4 x BYTE), beginning with the name Var89

In this dialog a name can now be assigned for the variable, and the data type can be selected from a wide range of possible types. Several variables of the same type can be declared simultaneously via the **Multiple** selection box, and a defined start address can be set. Once the above dialog is confirmed, the process image shown below appears.

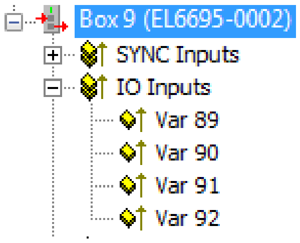


Fig. 134: Newly added variables on the secondary side of the EL6695

The variable names are incremented automatically, as shown here. Output variables are created accordingly.

On the primary side, i.e. the “remote side” of the EL6695, four suitable output variables of the same type should now be created for the data exchange. These variable do not have to have the same name, i.e. they can be called Var85 to Var88.

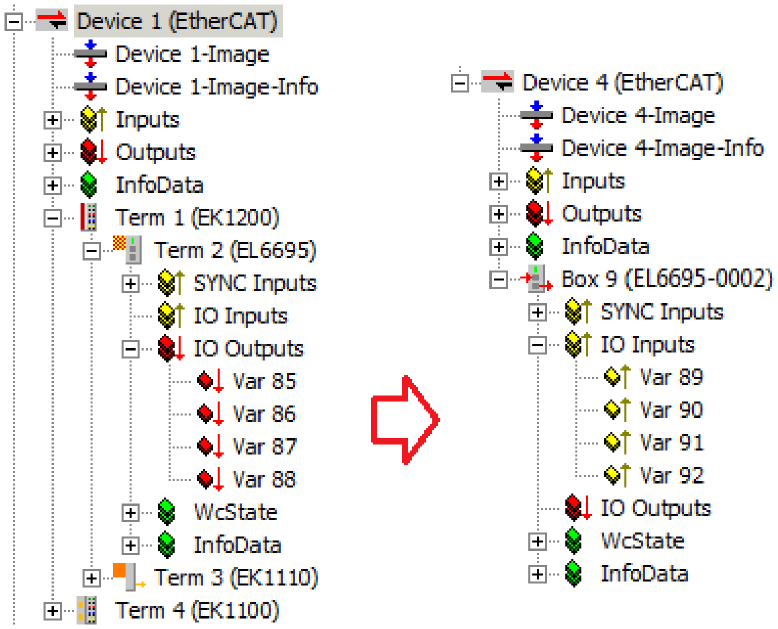


Fig. 135: Creating process data

Once here too 4 x BYTE variables have been added to the outputs on the primary side of the EL6695, all values are output continuously via the four input variables, once the variables have been written, e.g. via a PLC program. If, for example, a value is written into variable Var86, the same value is written to the sequence corresponding output variable Var90 in the mapping of the remote side of the EL6695 (as long as the terminal is in OP state).

Transfer direction of two controls PLC1 to PLC2

Generally the terminal will be used to transfer data between two controls. Given that the bridge maps an output as an input and an input as an output the following manner of transmission is the result:

- Output (PLC1) → Output EL6695 ⇛ Input EL6695-0002 remote side → Input (PLC2)
- Input (PLC1) → Input EL6695 ⇛ Output EL6695-0002 remote side → Output (PLC2)

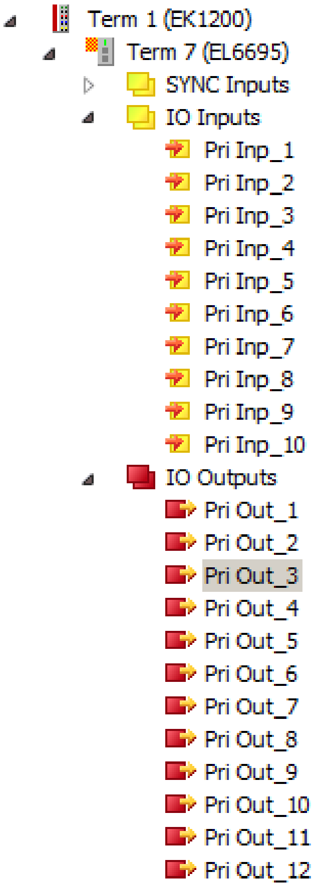
In each case, the EL6695 bridge terminal “maps” output value to input value in byte order and vice versa (without other configurations or settings). Thereby the data type of the used variables have to be the same on every side.

Automatic configuration via the configuration interface/extension:

The **EL6695** extension with the **Create Configuration** button makes the created variables readable online on the other side, so that they do not have to be created separately there. Obviously, this function is recommended to use for a wide process image and is available by the plug-In [EL6695], only. Additionally, the EL6695 have to be ready to operate and accessible "online“ from both sides. The procedure for TwinCAT 3 will be described below by created variables on the primary side automatic mirrored on the secondary side:

- Precondition: TwinCAT is in „Free-Run“/ „Config Mode" (see symbol down-right

  ,), an EL6695 is present primary sided below „Device“ (Term)
- **A)** Variables would be created on the primary side (e.g. 10 input and 12 output variables):



- Term 1 (EK1200)
  - Term 7 (EL6695)
    - SYNC Inputs
    - IO Inputs
      - Pri Inp_1
      - Pri Inp_2
      - Pri Inp_3
      - Pri Inp_4
      - Pri Inp_5
      - Pri Inp_6
      - Pri Inp_7
      - Pri Inp_8
      - Pri Inp_9
      - Pri Inp_10
    - IO Outputs
      - Pri Out_1
      - Pri Out_2
      - Pri Out_3
      - Pri Out_4
      - Pri Out_5
      - Pri Out_6
      - Pri Out_7
      - Pri Out_8
      - Pri Out_9
      - Pri Out_10
      - Pri Out_11
      - Pri Out_12

Fig. 136: Exemplary applied variables on the primary side of the EL6695

- **B)** The Terminal is selected on primary side; by pressing the button [Create configuration] under „Process Data“ within the [EL6695] tab a "mirrored" set of variables will be created for the other side, in this case the secondary side *internally* created by the terminal; thus they aren't visible yet

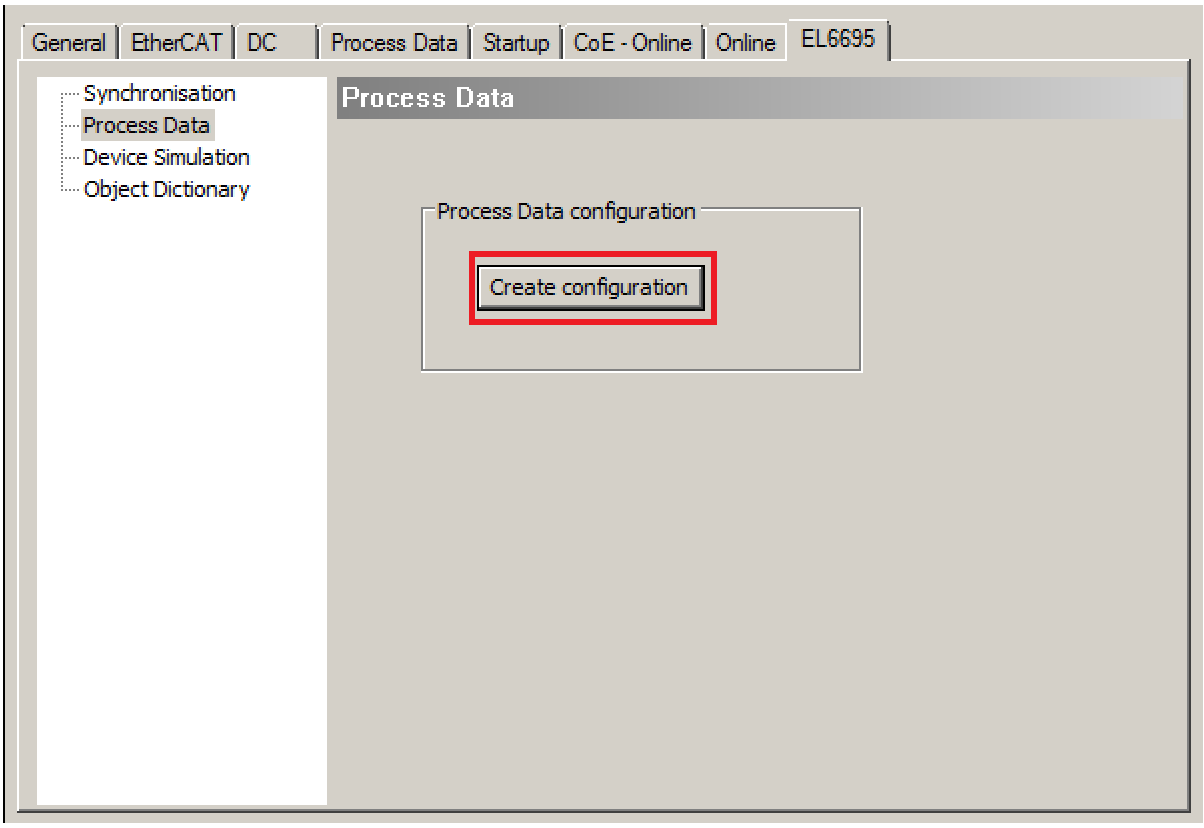
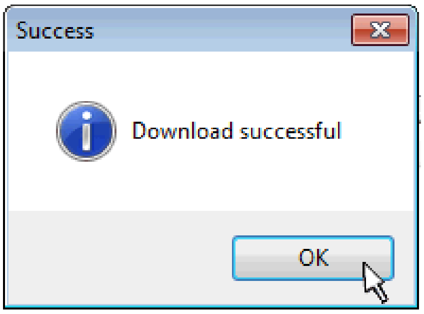


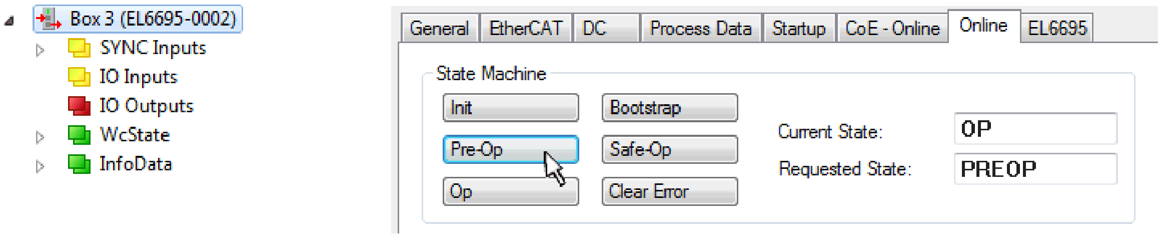
Fig. 137: [Create configuration] under „Process Data“ in the EL6695 tab

- **C)** The successful execution of this action should be receipted respectively:



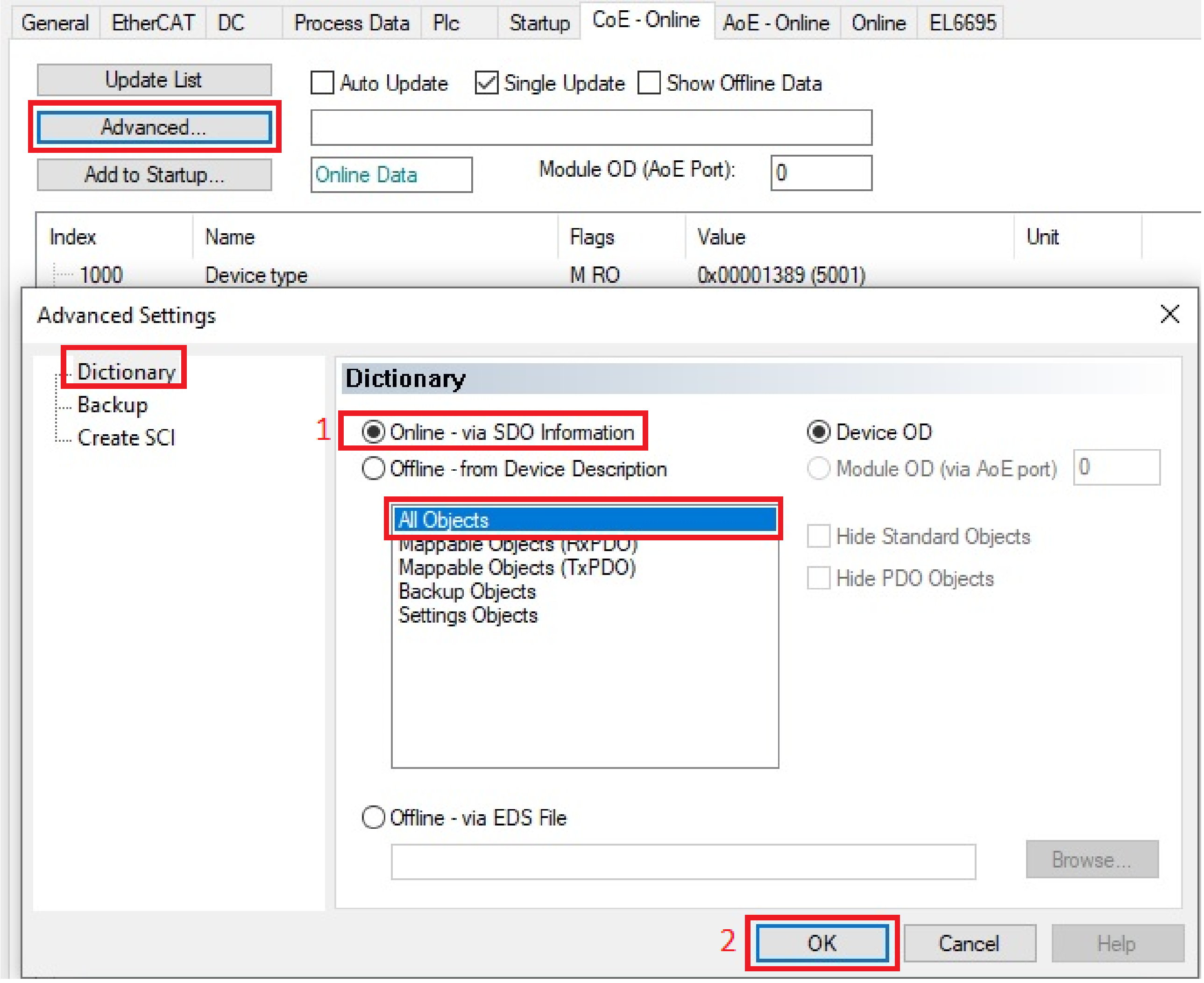
Download successful

- **D)** The primary side must now be set via INIT to PreOp and OP.
- **E)** The further steps are done on the secondary side. In order for the terminal to take over the data of the variables through an internal initialization, the terminal must be read in again via a scan process on the device on the secondary side. It may be in Error PreOp status because the PDOs of the primary and secondary side do not match. It must be manually set to PreOp status - not OP status - so that the StartUp list is not sent. Therefore FreeRun must not be activated.



Current State: OP
Requested State: PREOP

- **F)** First, for further proceeding, the object dictionary structure have to be transferred from the terminal into TwinCAT. This is done from the [CoE – Online] tab by selecting “Advanced...” and performing the following steps (1 and 2) as shown:



☐ Auto Update ☑ Single Update ☐ Show Offline Data

Module OD (AoE Port): 0

Index	Name	Flags	Value	Unit
1000	Device type	M RO	0x00001389 (5001)	

Dictionary

● Online - via SDO Information

○ Offline - from Device Description

All Objects
Mappable Objects (RxPDO)
Mappable Objects (TxPDO)
Backup Objects
Settings Objects

○ Offline - via EDS File

● Device OD

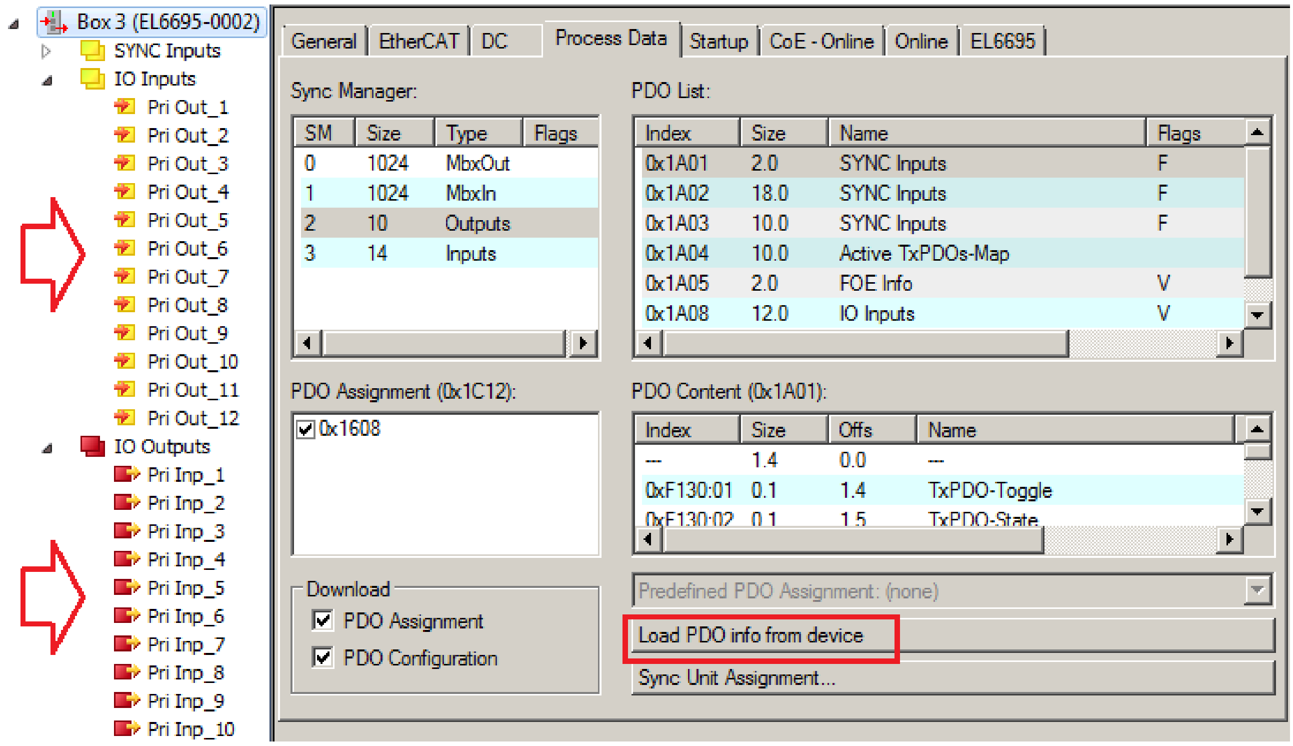
○ Module OD (via AoE port) 0

☐ Hide Standard Objects

☐ Hide PDO Objects

Fig. 138: Loading the object dictionary structure from the device

- **G)** Under the [Process Data] tab, the "mirrored" variable configuration is created from the other side using [Load PDO info from device] and then visible in the "Solution Explorer" (left):



- Box 3 (EL6695-0002)
  - SYNC Inputs
  - IO Inputs
    - Pri Out_1
    - Pri Out_2
    - Pri Out_3
    - Pri Out_4
    - Pri Out_5
    - Pri Out_6
    - Pri Out_7
    - Pri Out_8
    - Pri Out_9
    - Pri Out_10
    - Pri Out_11
    - Pri Out_12
  - IO Outputs
    - Pri Inp_1
    - Pri Inp_2
    - Pri Inp_3
    - Pri Inp_4
    - Pri Inp_5
    - Pri Inp_6
    - Pri Inp_7
    - Pri Inp_8
    - Pri Inp_9
    - Pri Inp_10

Sync Manager:

SM	Size	Type	Flags
0	1024	MbxOut	
1	1024	MbxIn	
2	10	Outputs	
3	14	Inputs	

PDO List:

Index	Size	Name	Flags
0x1A01	2.0	SYNC Inputs	F
0x1A02	18.0	SYNC Inputs	F
0x1A03	10.0	SYNC Inputs	F
0x1A04	10.0	Active TxPDOs-Map	
0x1A05	2.0	FOE Info	V
0x1A08	12.0	IO Inputs	V

PDO Assignment (0x1C12):

☑ 0x1608

PDO Content (0x1A01):

Index	Size	Offs	Name
---	1.4	0.0	---
0xF130:01	0.1	1.4	TxPDO-Toggle
0xF130:02	0.1	1.5	TxPDO-State

Download

☑ PDO Assignment

☑ PDO Configuration

Fig. 139: Variables of the primary side mirror-fit transmitted to the secondary side by "Load PDO info from device"

- **H)** If necessary, reload the configuration using "Reload Devices" ( )n the TwinCAT menu and activate FreeRun.

According to the procedure described here the variables could also be created for the primary side, if a set of I/O variables, i.e. a PDO-Configuration is present on the secondary side.

Restriction Variables

The following PDO/variable types are to be used for this procedure:
Byte, UINT, UDINT

Use without TwinCAT

If no TwinCAT EtherCAT Master with [Load PDO info from device] is available on the other side, the PDO upload mechanism must be reproduced there:

- Read PDO assignment objects 0x1C12, 0x1C13
- then read out the PDO Mapping Objects 0x160x (outputs), 0x1A0x (inputs) and
- read out the PDOs 0x60xx, 0x70xx accordingly

If a scan procedure is not possible/ not desired

To let variables be created on the secondary side, after steps **B)** and **C)** of [Create configuration], the terminal on the primary side must be set manually into the INIT state and then again into the OP state ("Online" tab: click on Init, then: Safe-Op, Pre-Op, Op). On the secondary side the respective variant of the EL6695 (-0002) can then be inserted instead. Besides the procedure is like described above since **E)**.

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.

→ Example configuration: https://infosys.beckhoff.com/content/1033/el6695/Resources/zip/3521535883.zip

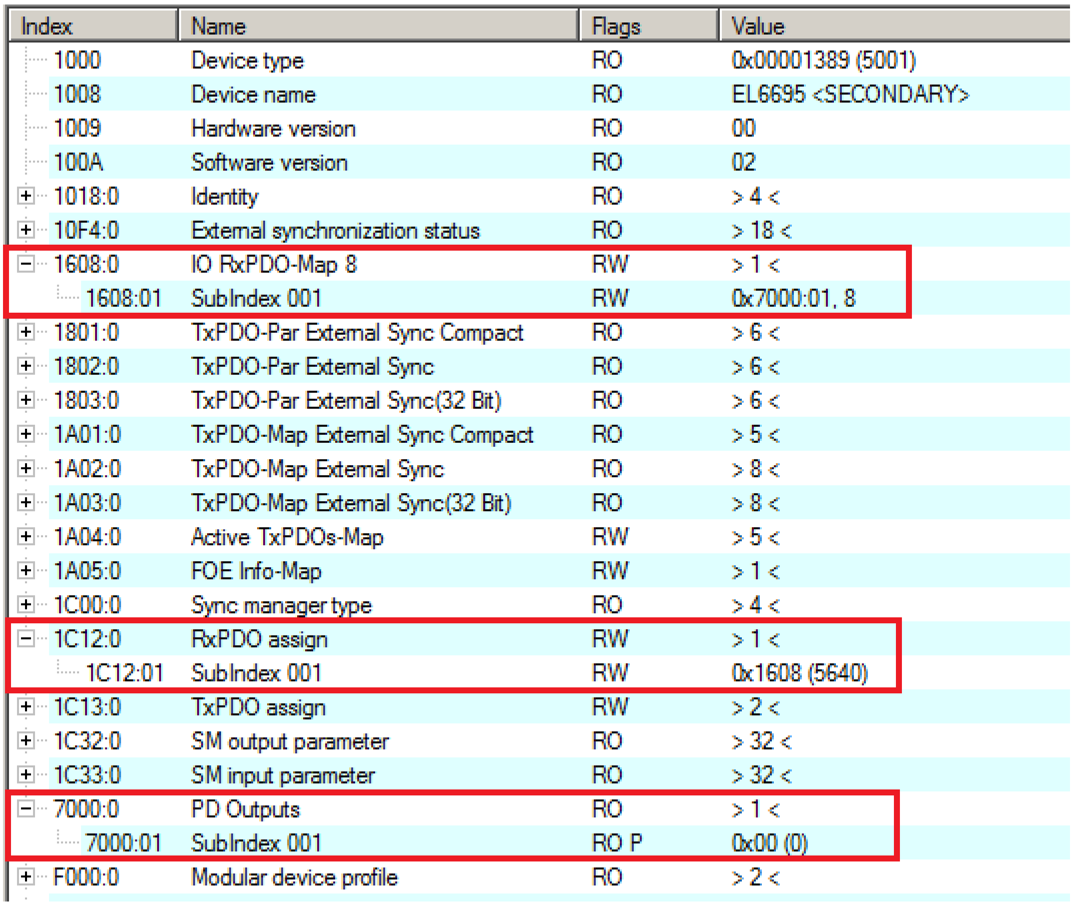
6.4.3 Selective PDO mapping

Selective PDO mapping

Users can configure customized cyclic data exchange through selective PDO mapping. To this end a “full” set of process data is created on one side of the terminal. On the other side a subset of this set can now be defined for cyclic reading. This process is referred to as selective PDO mapping.

The default PDO assignment and mapping is removed, and the user has to create a custom PDO assignment and mapping. For this purpose, it is important to know the structure of a PDO assignment or mapping.

Basically (standard) there are two CoE objects 0x1C12 “Rx PDO Assign” and 0x1C13 “Tx PDO Assign”, in which the assignments for the data input (RxPDO) and the data output (TxPDO) for the secondary side and the primary side are defined. They contain a reference to CoE objects 0x1608 “IO RxPDO Map” and 0x1A08 “IO TxPDO Map”, in which the mapping the inputs and outputs is defined. All four of these CoE objects, 0x1C12/0x1C13 and 0x1608/0x1A08, have an RW flag (RW = read/write). The two mapping objects 0x1608 and 0x1A08 initially refer to the CoE objects 0x7000 “PD Outputs” and 0x6000 “PD Inputs” (process data inputs/ outputs), in which process data (structures) are stored. “Initial” refers to automatic configuration by TwinCAT, when the user adds new input or output data on the primary or secondary side. This can also be viewed in the [Startup] tab. This is where all the links for object references are created in the transition from PREOP to SAFEOP (P → S). The two “final” CoE objects 0x6000 and 0x7000 are “RO” (read only), since within the device the copy processes for input and output data are carried out in OP state.



Index	Name	Flags	Value
1000	Device type	RO	0x00001389 (5001)
1008	Device name	RO	EL6695 <SECONDARY>
1009	Hardware version	RO	00
100A	Software version	RO	02
1018:0	Identity	RO	> 4 <
10F4:0	External synchronization status	RO	> 18 <
1608:0	IO RxPDO-Map 8	RW	> 1 <
1608:01	SubIndex 001	RW	0x7000:01, 8
1801:0	TxPDO-Par External Sync Compact	RO	> 6 <
1802:0	TxPDO-Par External Sync	RO	> 6 <
1803:0	TxPDO-Par External Sync(32 Bit)	RO	> 6 <
1A01:0	TxPDO-Map External Sync Compact	RO	> 5 <
1A02:0	TxPDO-Map External Sync	RO	> 8 <
1A03:0	TxPDO-Map External Sync(32 Bit)	RO	> 8 <
1A04:0	Active TxPDOs-Map	RW	> 5 <
1A05:0	FOE Info-Map	RW	> 1 <
1C00:0	Sync manager type	RO	> 4 <
1C12:0	RxPDO assign	RW	> 1 <
1C12:01	SubIndex 001	RW	0x1608 (5640)
1C13:0	TxPDO assign	RW	> 2 <
1C32:0	SM output parameter	RO	> 32 <
1C33:0	SM input parameter	RO	> 32 <
7000:0	PD Outputs	RO	> 1 <
7000:01	SubIndex 001	RO P	0x00 (0)
F000:0	Modular device profile	RO	> 2 <

Fig. 140: Basic mapping of a 1 byte variable via 0x1C12 to 0x7000 “PD Outputs”

Overview of the (initial) linked objects:

- **RxPDO Assign**: 0x1C12 → RxPDO Map 0x1608 → 0x7000 PD Outputs
- **TxPDO Assign**: 0x1C13 → TxPDO Map 0x1A08 → 0x6000 PD Inputs

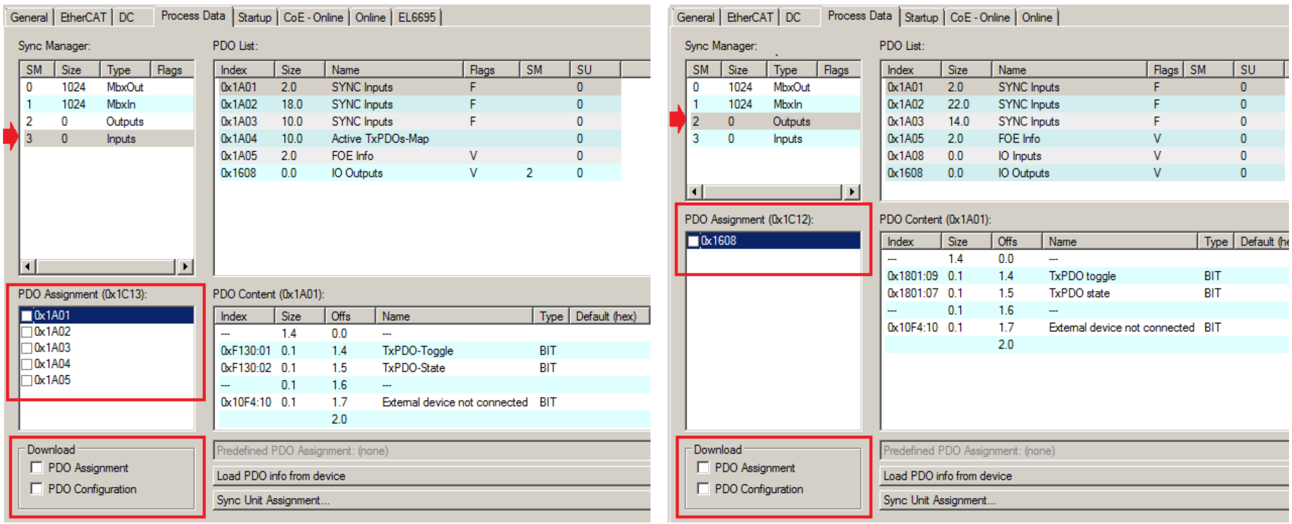
Mapping of CoE objects

0x1608 points to 0x7000, and 0x1A08 points to 0x6000. The objects 0x7000 and 0x6000 are mirrored on the respective other side of the terminal.

For example, if an object 0x7000:0E PD Outputs exists on the primary side, an object 0x6000:0E PD Inputs should be created on the secondary side for data exchange.

Procedure under TwinCAT

To create a selective mapping, the existing PDO configuration with its assignments should initially be disabled by unticking all options under outputs, inputs and downloads in the “Process Data” tab of the terminal.



Sync Manager:

SM	Size	Type	Flags
0	1024	MbxOut	
1	1024	MbxIn	
2	0	Outputs	
3	0	Inputs	

PDO List:

Index	Size	Name	Flags	SM	SU
0x1A01	2.0	SYNC Inputs	F		0
0x1A02	18.0	SYNC Inputs	F		0
0x1A03	10.0	SYNC Inputs	F		0
0x1A04	10.0	Active TxPDOs-Map			0
0x1A05	2.0	FOE Info	V		0
0x1608	0.0	IO Outputs	V	2	0

PDO Assignment (0x1C13):

- ☐ 0x1A01
- ☐ 0x1A02
- ☐ 0x1A03
- ☐ 0x1A04
- ☐ 0x1A05

PDO Content (0x1A01):

Index	Size	Offs	Name	Type	Default (hex)
---	1.4	0.0	---		
0xF130:01	0.1	1.4	TxPDO-Toggle	BIT	
0xF130:02	0.1	1.5	TxPDO-State	BIT	
---	0.1	1.6	---		
0x10F4:10	0.1	1.7	External device not connected	BIT	
		2.0			

Sync Manager:

SM	Size	Type	Flags
0	1024	MbxOut	
1	1024	MbxIn	
2	0	Outputs	
3	0	Inputs	

PDO List:

Index	Size	Name	Flags	SM	SU
0x1A01	2.0	SYNC Inputs	F		0
0x1A02	22.0	SYNC Inputs	F		0
0x1A03	14.0	SYNC Inputs	F		0
0x1A05	2.0	FOE Info	V		0
0x1A08	0.0	IO Inputs	V		0
0x1608	0.0	IO Outputs	V		0

PDO Assignment (0x1C12):

- ☐ 0x1608

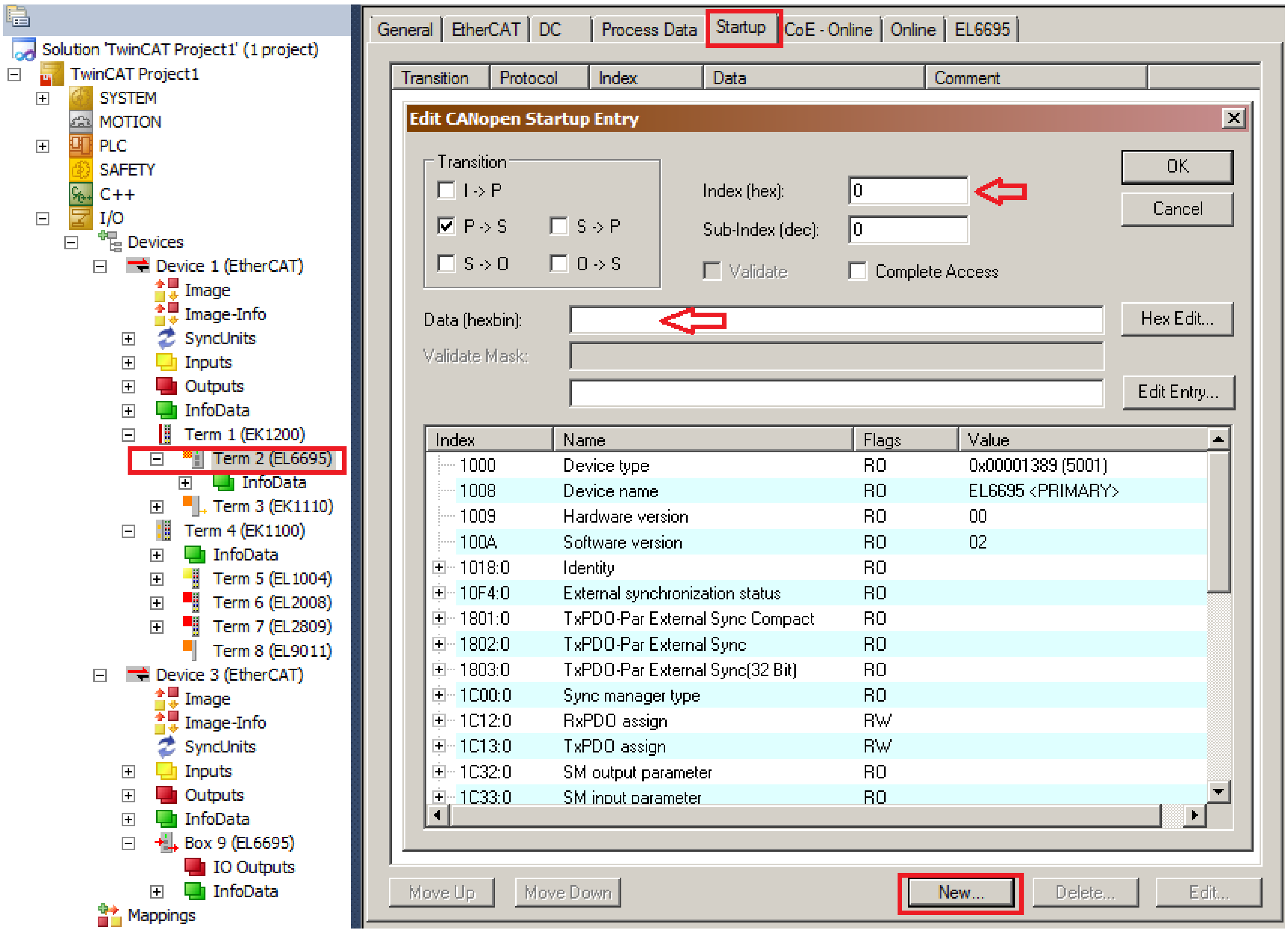
PDO Content (0x1A01):

Index	Size	Offs	Name	Type	Default (he
---	1.4	0.0	---		
0x1801:09	0.1	1.4	TxPDO toggle	BIT	
0x1801:07	0.1	1.5	TxPDO state	BIT	
---	0.1	1.6	---		
0x10F4:10	0.1	1.7	External device not connected	BIT	
		2.0			

Fig. 141: Process data in selective mapping

The existing entries in the Startup tab are then also deleted, and the terminal no longer has a process image. The process image is then redefined in two steps, as described below:

New entries have to be created in the Startup tab of the terminal. Select “New…” on the bottom right of this tab.



Transition
- ☐ I -> P
- ☑ P -> S ☐ S -> P
- ☐ S -> O ☐ O -> S

Index (hex): 0

Sub-Index (dec): 0

☐ Validate ☐ Complete Access

Data (hexbin):

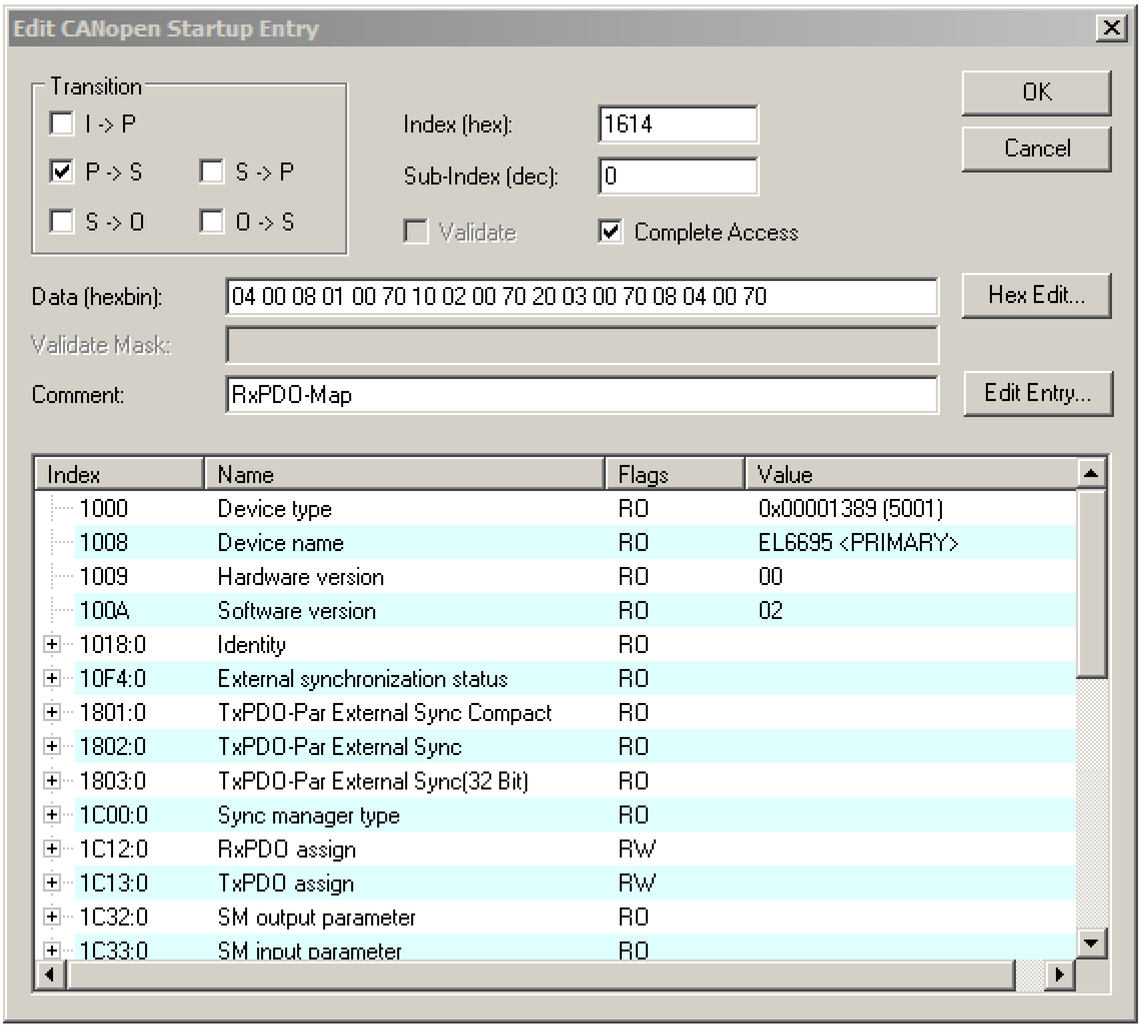
Validate Mask:

Index	Name	Flags	Value
1000	Device type	RO	0x00001389 (5001)
1008	Device name	RO	EL6695 <PRIMARY>
1009	Hardware version	RO	00
100A	Software version	RO	02
+ 1018:0	Identity	RO	
+ 10F4:0	External synchronization status	RO	
+ 1801:0	TxPDO-Par External Sync Compact	RO	
+ 1802:0	TxPDO-Par External Sync	RO	
+ 1803:0	TxPDO-Par External Sync(32 Bit)	RO	
+ 1C00:0	Sync manager type	RO	
+ 1C12:0	RxPDO assign	RW	
+ 1C13:0	TxPDO assign	RW	
+ 1C32:0	SM output parameter	RO	
+ 1C33:0	SM input parameter	RO	

Fig. 142: Creating new output variables (in this example: primary side of the EL6695)

As an example, a total of four output variables with sizes 8, 16, 32 and again 8 are created [recommended sequence]:

To this end a 0x1614 object (selectable from the range 0x1600 to 0x161F) is created in this example as follows (tick “Complete Access”; enter “IO RxPDO Map” as a comment, for example):



Transition
□ I -> P
☑ P -> S
□ S -> P
□ S -> O
□ O -> S

Index (hex): 1614

Sub-Index (dec): 0

□ Validate
☑ Complete Access

Data (hexbin): 04 00 08 01 00 70 10 02 00 70 20 03 00 70 08 04 00 70

Validate Mask:

Comment: RxPDO-Map

Index	Name	Flags	Value
1000	Device type	RO	0x00001389 (5001)
1008	Device name	RO	EL6695 <PRIMARY>
1009	Hardware version	RO	00
100A	Software version	RO	02
+ 1018:0	Identity	RO	
+ 10F4:0	External synchronization status	RO	
+ 1801:0	TxPDO-Par External Sync Compact	RO	
+ 1802:0	TxPDO-Par External Sync	RO	
+ 1803:0	TxPDO-Par External Sync(32 Bit)	RO	
+ 1C00:0	Sync manager type	RO	
+ 1C12:0	RxPDO assign	RW	
+ 1C13:0	TxPDO assign	RW	
+ 1C32:0	SM output parameter	RO	
+ 1C33:0	SM input parameter	RO	

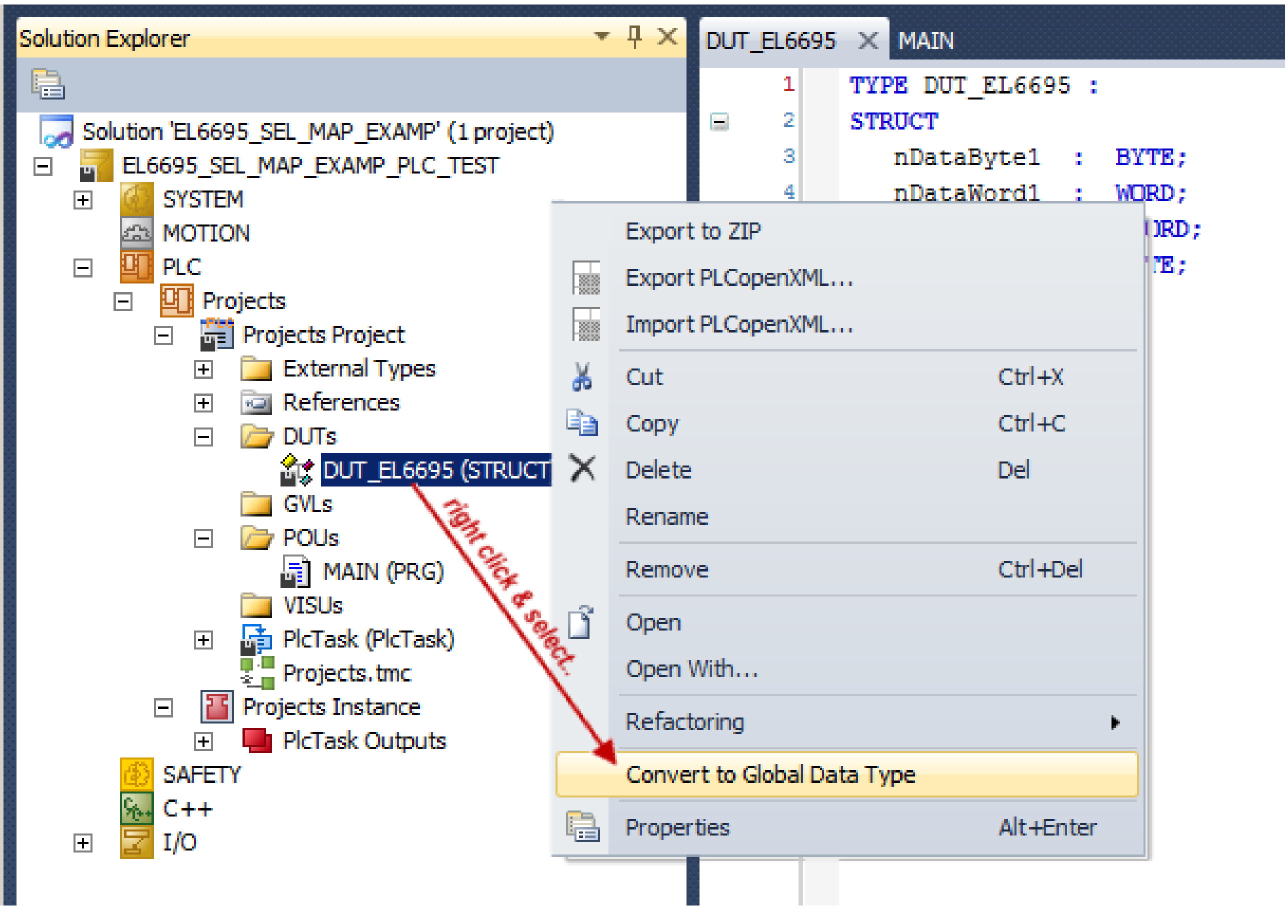
Fig. 143: Setting up the 0x1614 object

- Enter: **04 00 08 01 00 70 10 02 00 70 20 03 00 70 08 04 00 70**

Meaning of the entry data:

04 00	Subindex number
08 01 00 70	Object 7000, index 1, length 8
10 02 00 70	Object 7000, index 2, length 16
20 03 00 70	Object 7000, index 3, length 32
08 04 00 70	Object 7000, index 4, length 8

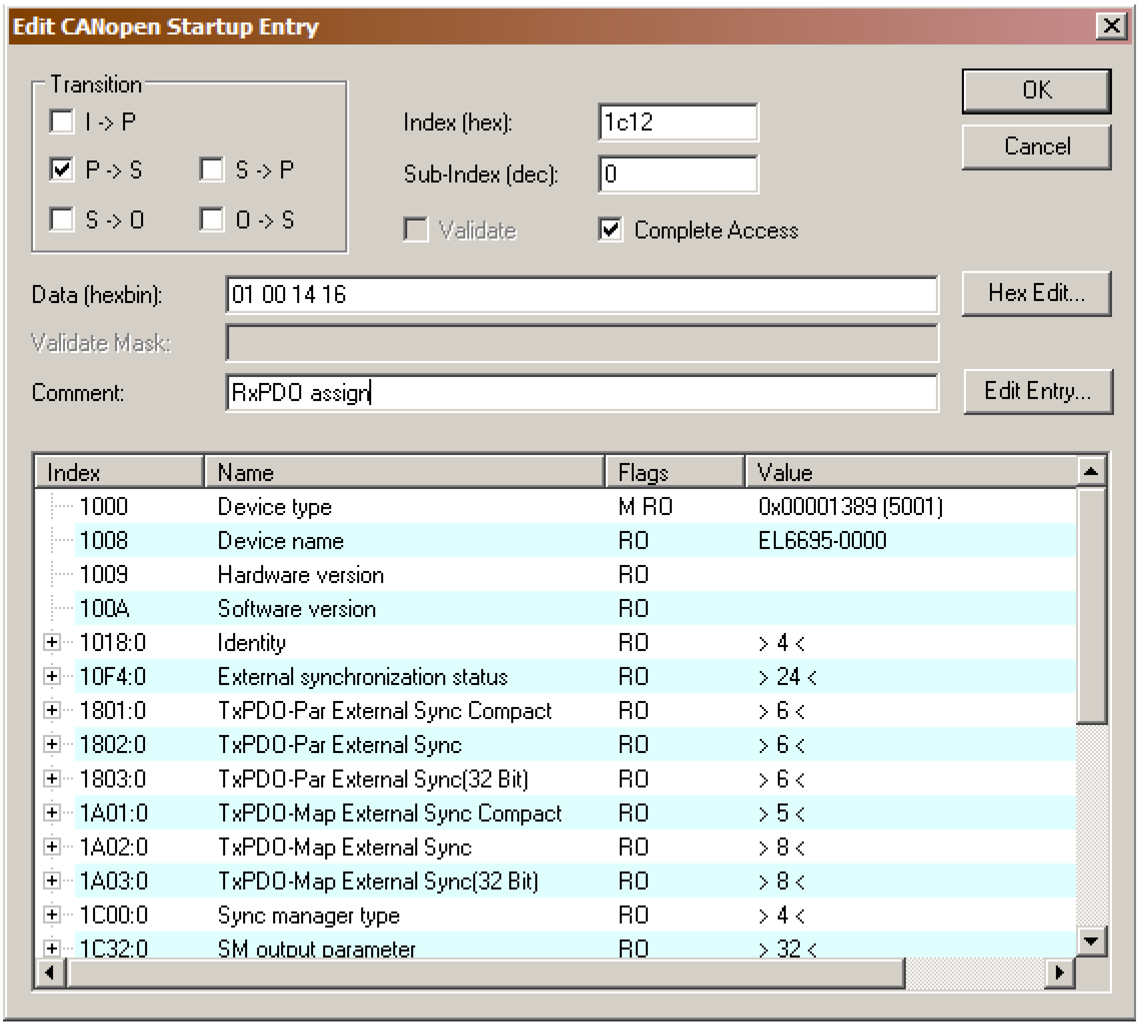
This can represent the data structure of a program object or structure, for example. In order to be able to link a structure variable with a PDO of the EL6695, it first has to be converted to a global data types under TC3:



```
TYPE DUT_EL6695 :
STRUCT
    nDataByte1   :  BYTE;
    nDataWord1   :  WORD;
```

Fig. 144: Conversion to a global data type

In the second step the following entries are made via the dialog “Edit CANopen Startup Entry”:



Edit CANopen Startup Entry

Transition
☐ I -> P
☑ P -> S ☐ S -> P
☐ S -> O ☐ O -> S

Index (hex): 1c12
Sub-Index (dec): 0
☐ Validate ☑ Complete Access

Data (hexbin): 01 00 14 16
Validate Mask:
Comment: RxPDO assign

Index	Name	Flags	Value
1000	Device type	M RO	0x00001389 (5001)
1008	Device name	RO	EL6695-0000
1009	Hardware version	RO	
100A	Software version	RO	
1018:0	Identity	RO	> 4 <
10F4:0	External synchronization status	RO	> 24 <
1801:0	TxPDO-Par External Sync Compact	RO	> 6 <
1802:0	TxPDO-Par External Sync	RO	> 6 <
1803:0	TxPDO-Par External Sync(32 Bit)	RO	> 6 <
1A01:0	TxPDO-Map External Sync Compact	RO	> 5 <
1A02:0	TxPDO-Map External Sync	RO	> 8 <
1A03:0	TxPDO-Map External Sync(32 Bit)	RO	> 8 <
1C00:0	Sync manager type	RO	> 4 <
1C32:0	SM output parameter	RO	> 32 <

Fig. 145: Creating object 0x1C12 for RxPDO

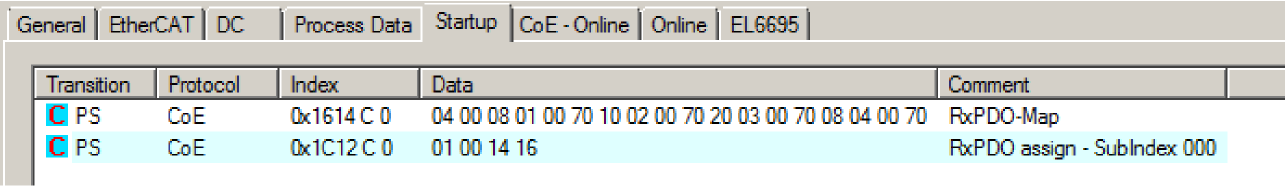
The CoE Startup entry to be created therefore is the index 0x1C12 with the subindex 0 (subindex for the start of the object). The tick at “Complete Acess” is set, so that the entered values are not interpreted as data type WORD, DWORD etc. The assignments should be made during the transition from PREOP to SAFEOP; only the tick “P → S” remains set (or is to be set) during the transition. Under “Comment” an explanatory comment can be added, which then appears in the CoE overview as the name of the object. The content of the CoE object is specified in “Data (hexbin)”.

The first two blocks define the number of subindices of the object (Hi/Lo byte switched: 01 00 → number of subindices = 1). This is followed by the content of the first subindex 01 with 14 16 → 0x1614.

- Enter: **01 00 14 16**

This is now one of the references described above, in this case to the already created “RxPDO Map” object (0x1614).

The [Startup] tab should now look as follows:



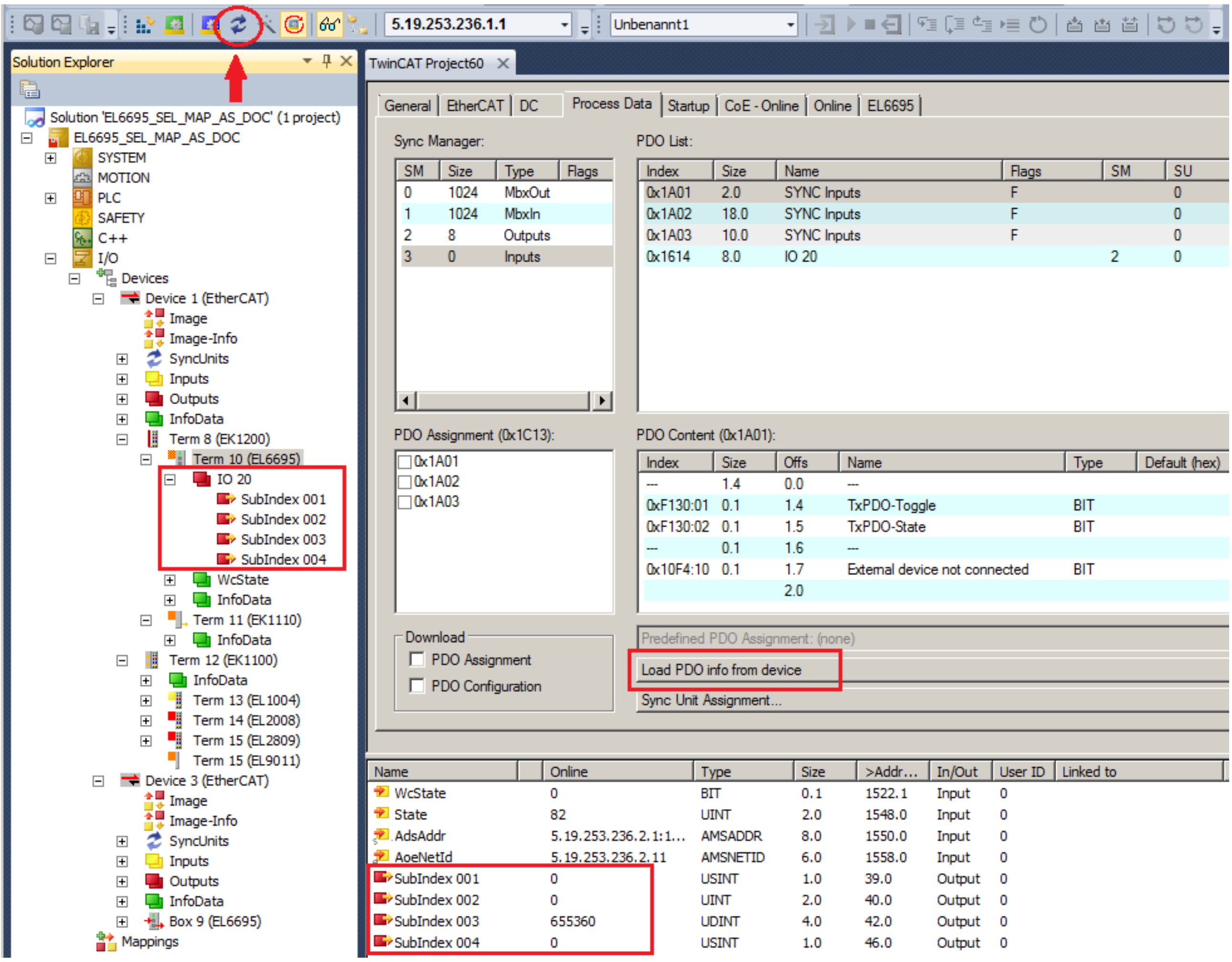
Transition	Protocol	Index	Data	Comment
C PS	CoE	0x1614 C 0	04 00 08 01 00 70 10 02 00 70 20 03 00 70 08 04 00 70	RxPDO-Map
C PS	CoE	0x1C12 C 0	01 00 14 16	RxPDO assign - SubIndex 000

Fig. 146: New PDO map and assignment objects on the primary side

It is important that “RxPDO assign” over object 0x1C12 is always at the end of the list of RxPDO Map entries. It may contain several references (as required). A data set would then look like this, for example: 06 00 03 16 1A 16 1E 16 08 16 04 16 .. etc.

Use the following steps to display the created variables in the configuration:

1. Selection in the menu: [TwinCAT] → [Reload Devices]
2. “Load PDO info from device” in the “Process Data” tab of the terminal
3. Display the CoE online objects via “Update List”



Sync Manager:

SM	Size	Type	Flags
0	1024	MbxOut	
1	1024	MbxIn	
2	8	Outputs	
3	0	Inputs	

PDO List:

Index	Size	Name	Flags	SM	SU
0x1A01	2.0	SYNC Inputs	F		0
0x1A02	18.0	SYNC Inputs	F		0
0x1A03	10.0	SYNC Inputs	F		0
0x1614	8.0	IO 20		2	0

PDO Assignment (0x1C13):

□ 0x1A01
□ 0x1A02
□ 0x1A03

PDO Content (0x1A01):

Index	Size	Offs	Name	Type	Default (hex)
---	1.4	0.0	---		
0xF130:01	0.1	1.4	TxPDO-Toggle	BIT	
0xF130:02	0.1	1.5	TxPDO-State	BIT	
---	0.1	1.6	---		
0x10F4:10	0.1	1.7	External device not connected	BIT	
		2.0			

Download
□ PDO Assignment
□ PDO Configuration

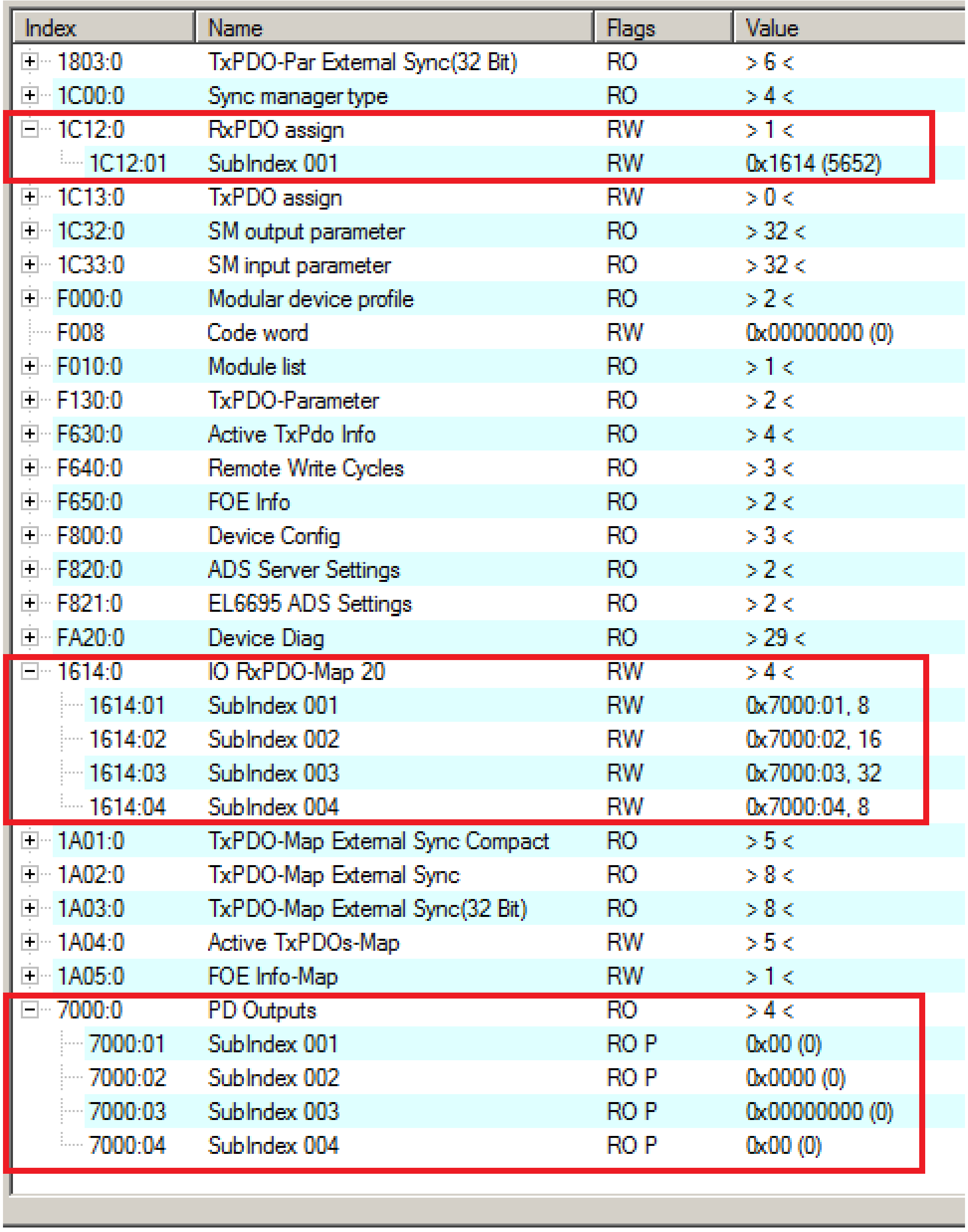
Predefined PDO Assignment: (none)

Load PDO info from device

Sync Unit Assignment...

Name	Online	Type	Size	>Addr...	In/Out	User ID	Linked to
WcState	0	BIT	0.1	1522.1	Input	0	
State	82	UINT	2.0	1548.0	Input	0	
AdsAddr	5.19.253.236.2.1:1...	AMSADDR	8.0	1550.0	Input	0	
AoeNetId	5.19.253.236.2.11	AMSNETID	6.0	1558.0	Input	0	
SubIndex 001	0	USINT	1.0	39.0	Output	0	
SubIndex 002	0	UINT	2.0	40.0	Output	0	
SubIndex 003	655360	UDINT	4.0	42.0	Output	0	
SubIndex 004	0	USINT	1.0	46.0	Output	0	

Fig. 147: Newly created process data:object 0x1608



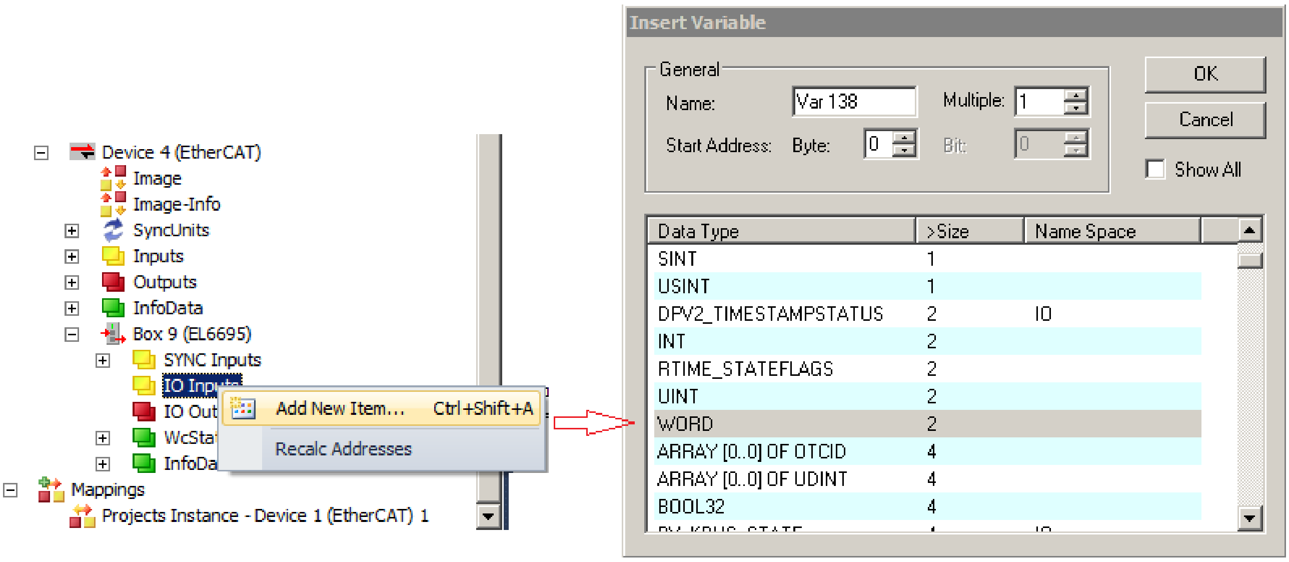
Index	Name	Flags	Value
1803:0	TxPDO-Par External Sync(32 Bit)	RO	> 6 <
1C00:0	Sync manager type	RO	> 4 <
1C12:0	RxPDO assign	RW	> 1 <
1C12:01	SubIndex 001	RW	0x1614 (5652)
1C13:0	TxPDO assign	RW	> 0 <
1C32:0	SM output parameter	RO	> 32 <
1C33:0	SM input parameter	RO	> 32 <
F000:0	Modular device profile	RO	> 2 <
F008	Code word	RW	0x00000000 (0)
F010:0	Module list	RO	> 1 <
F130:0	TxPDO-Parameter	RO	> 2 <
F630:0	Active TxPdo Info	RO	> 4 <
F640:0	Remote Write Cycles	RO	> 3 <
F650:0	FOE Info	RO	> 2 <
F800:0	Device Config	RO	> 3 <
F820:0	ADS Server Settings	RO	> 2 <
F821:0	EL6695 ADS Settings	RO	> 2 <
FA20:0	Device Diag	RO	> 29 <
1614:0	IO RxPDO-Map 20	RW	> 4 <
1614:01	SubIndex 001	RW	0x7000:01, 8
1614:02	SubIndex 002	RW	0x7000:02, 16
1614:03	SubIndex 003	RW	0x7000:03, 32
1614:04	SubIndex 004	RW	0x7000:04, 8
1A01:0	TxPDO-Map External Sync Compact	RO	> 5 <
1A02:0	TxPDO-Map External Sync	RO	> 8 <
1A03:0	TxPDO-Map External Sync(32 Bit)	RO	> 8 <
1A04:0	Active TxPDOs-Map	RW	> 5 <
1A05:0	FOE Info-Map	RW	> 1 <
7000:0	PD Outputs	RO	> 4 <
7000:01	SubIndex 001	RO P	0x00 (0)
7000:02	SubIndex 002	RO P	0x0000 (0)
7000:03	SubIndex 003	RO P	0x00000000 (0)
7000:04	SubIndex 004	RO P	0x00 (0)

Fig. 148: Display [CoE Online] the selectively created variables with the corresponding references

Creating the selective PDO variables

On the other side of the EL6695 bridge terminal, in this case the secondary side, selected “variables" can now be mirrored, if required. To this end, add the required variables in the “Solution Explorer”.

In this example, the primary side has a 1-2-4-1 byte structure, of which only the 2 byte and the 4 byte variable types ”WORD” and “DWORD” are to be provided as a subset of the whole process data, as shown:



General

Name: Var 138 Multiple: 1

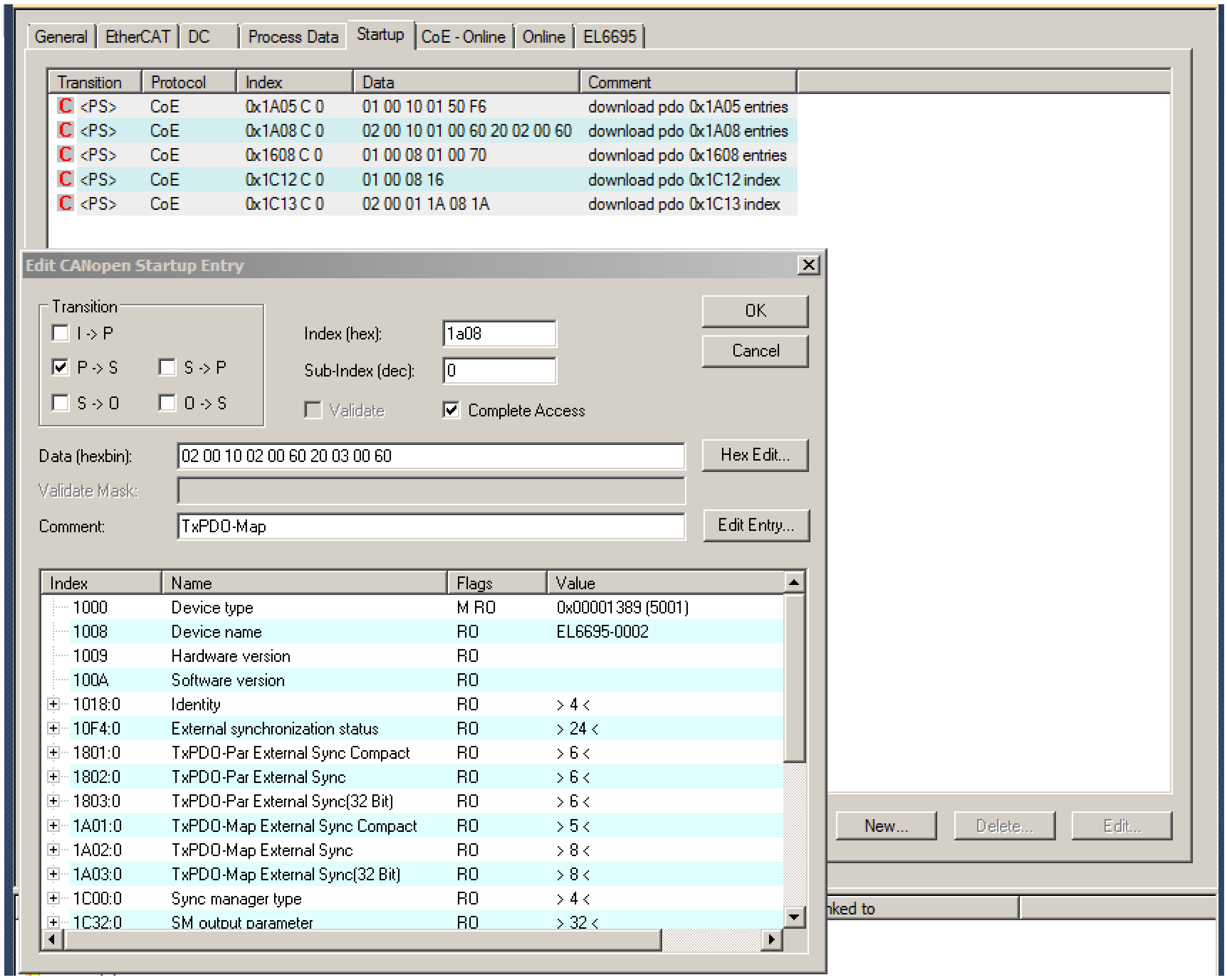
Start Address: Byte: 0 Bit: 0

Data Type	>Size	Name Space
SINT	1	
USINT	1	
DPV2_TIMESTAMPSTATUS	2	IO
INT	2	
RTIME_STATEFLAGS	2	
UINT	2	
WORD	2	
ARRAY [0..0] OF OTCID	4	
ARRAY [0..0] OF UDINT	4	
BOOL32	4	

Fig. 149: Adding selective PDO variables

The procedure for the DWORD variable is similar, as shown in the diagram.

Then complete the Startup entry as follows, to adapt the subindex values 02 for the WORD type and 03 for the DWORD type:



Transition	Protocol	Index	Data	Comment
C <PS>	CoE	0x1A05 C 0	01 00 10 01 50 F6	download pdo 0x1A05 entries
C <PS>	CoE	0x1A08 C 0	02 00 10 01 00 60 20 02 00 60	download pdo 0x1A08 entries
C <PS>	CoE	0x1608 C 0	01 00 08 01 00 70	download pdo 0x1608 entries
C <PS>	CoE	0x1C12 C 0	01 00 08 16	download pdo 0x1C12 index
C <PS>	CoE	0x1C13 C 0	02 00 01 1A 08 1A	download pdo 0x1C13 index

Transition

☐ I -> P

☑ P -> S ☐ S -> P

☐ S -> O ☐ O -> S

Index (hex): 1a08

Sub-Index (dec): 0

☐ Validate ☑ Complete Access

Data (hexbin): 02 00 10 02 00 60 20 03 00 60

Validate Mask:

Comment: TxPDO-Map

Index	Name	Flags	Value
1000	Device type	M RO	0x00001389 (5001)
1008	Device name	RO	EL6695-0002
1009	Hardware version	RO	
100A	Software version	RO	
1018:0	Identity	RO	> 4 <
10F4:0	External synchronization status	RO	> 24 <
1801:0	TxPDO-Par External Sync Compact	RO	> 6 <
1802:0	TxPDO-Par External Sync	RO	> 6 <
1803:0	TxPDO-Par External Sync(32 Bit)	RO	> 6 <
1A01:0	TxPDO-Map External Sync Compact	RO	> 5 <
1A02:0	TxPDO-Map External Sync	RO	> 8 <
1A03:0	TxPDO-Map External Sync(32 Bit)	RO	> 8 <
1C00:0	Sync manager type	RO	> 4 <
1C32:0	SM output parameter	RO	> 32 <

Fig. 150: Startup entry 0x1A08 for overwriting (for correcting reference index 01 → 02 and 02 → 03)

- Enter: **02 00 10 02 00 60 20 03 00 60**

Meaning of the entry data:

02 00	Subindex number
10 02 00 60	Object 6000, index 2, length 16
20 03 00 60	Object 6000, index 3, length 32

After a further “Reload” instruction and activation of the configuration, the selective PDO mapping is now available for use by the PLC programs. Note that, due to terminal-specific system characteristics, a special buffering mechanism is used for the real-time data exchange, so that the output variables always have to be configure with a byte at the start and a byte at the end (so called start/stop bytes). Links to PLC variables and therefore a data usage are not required for these “fillbyte” variables.

By an application of the EL6695 together with other EtherCAT masters the configuration procedure have to be adapted respectively.

Explanatory note regarding the use of start/stop bytes

For selective mapping in the output data, Beckhoff generally recommends to create a start byte at the start and a stop byte at the end. The technical reason lies in the internal memory management of the process data, where access to the last byte results in internal switching to the next EtherCAT buffer of the terminal; read access is then no longer available for this output package. If read access to the output buffer takes place selectively “from top to bottom”, this is not a problem. However, if the variables are read successively from different sections (which is probably the case in most applications) and the last byte is to be captured, the buffer is then locked, and no further data can be retrieved. An additional “empty” byte after the actual process data prevents inadvertent switching of the buffer.

The diagram illustrates that in many cases the start/stop byte wouldn't really be required:

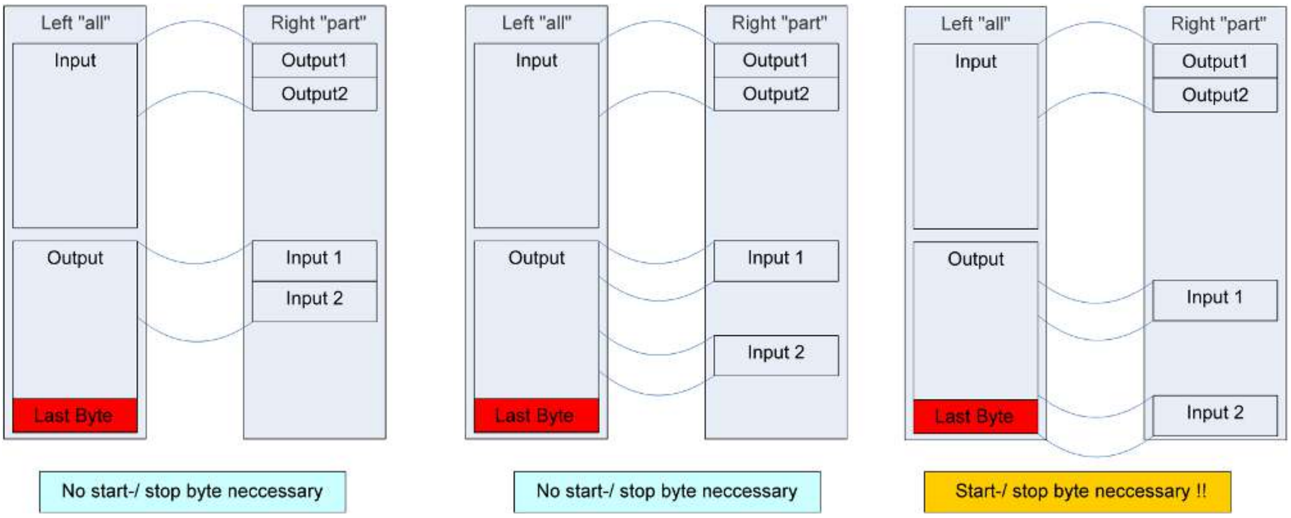


Fig. 151: Using the start/stop byte

Inserting start and stop byte

To avoid unexpected surprises during flexible utilization of the EL6695 over the service life of the application and any configuration changes, Beckhoff generally recommends to configure the output variables with a byte at the start and a byte at the end.

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.

→ example programs selective mapping available on request:

https://infosys.beckhoff.com/content/1033/el6695/Resources/zip/1421824651.zip

https://infosys.beckhoff.com/content/1033/el6695/Resources/zip/1421826315.zip

6.4.4 FSoE transfer

FSoE transfer is realized as cyclic process data transfer and can be implemented in two different ways:

- Create an ESI with safety modules/slots and load in EmulationMode, see there
- Configure the safety connection for an “External Device”
- Notice: The EL6695 is not a safety-related device requiring licensing, but “only” an element of the transmission link. If the ETG FSoE test is to be run against a safety device that hides “behind” the EL6695, the PDO configuration of the EL6695 should, of course, be such that the FSoE ETG test recognizes the EL6695 as a “Safety” device.

For further information please contact Beckhoff support, providing pertinent data (TwinCAT version, required performance values, cycle time, data quantities etc.)

6.5 Local mailbox protocols

Acyclic data, which are processed locally by the EL6695 and are not forwarded. Usually transported via the EtherCAT mailbox.

6.5.1 CoE - Can over EtherCAT

CoE - Can over EtherCAT

Processing: local, no forwarding

Within a PLC program (ST), data can be read from the object directory of an EtherCAT slave via the function block **FB_EcCoeSdoRead** and SDO (service data object) access. Use the parameters nSubIndex and nIndex to selected the object to be read. This block is part of the (standard) library Tc2_EtherCAT (see TwinCAT 3 Engineering → PLC → Libraries → TwinCAT 3 PLC Lib: Tc2_EtherCAT → CoE under http://infosys.beckhoff.com).

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.

→ example program CoE available on request:

https://infosys.beckhoff.com/content/1033/el6695/Resources/zip/1421141259.zip

6.6 Mailbox protocols with transfer capability

6.6.1 AoE application during CoE access

AoE application during CoE access

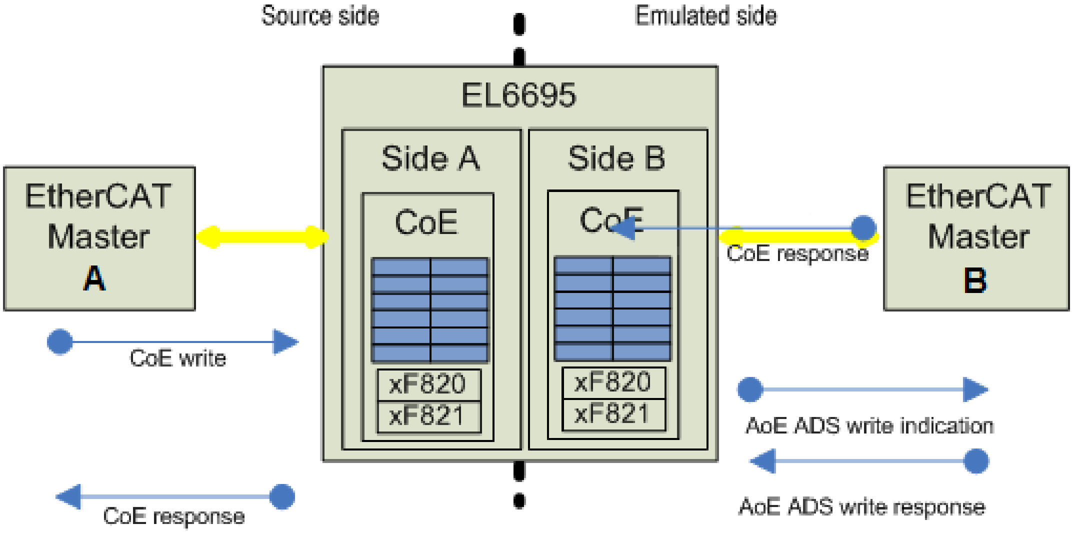
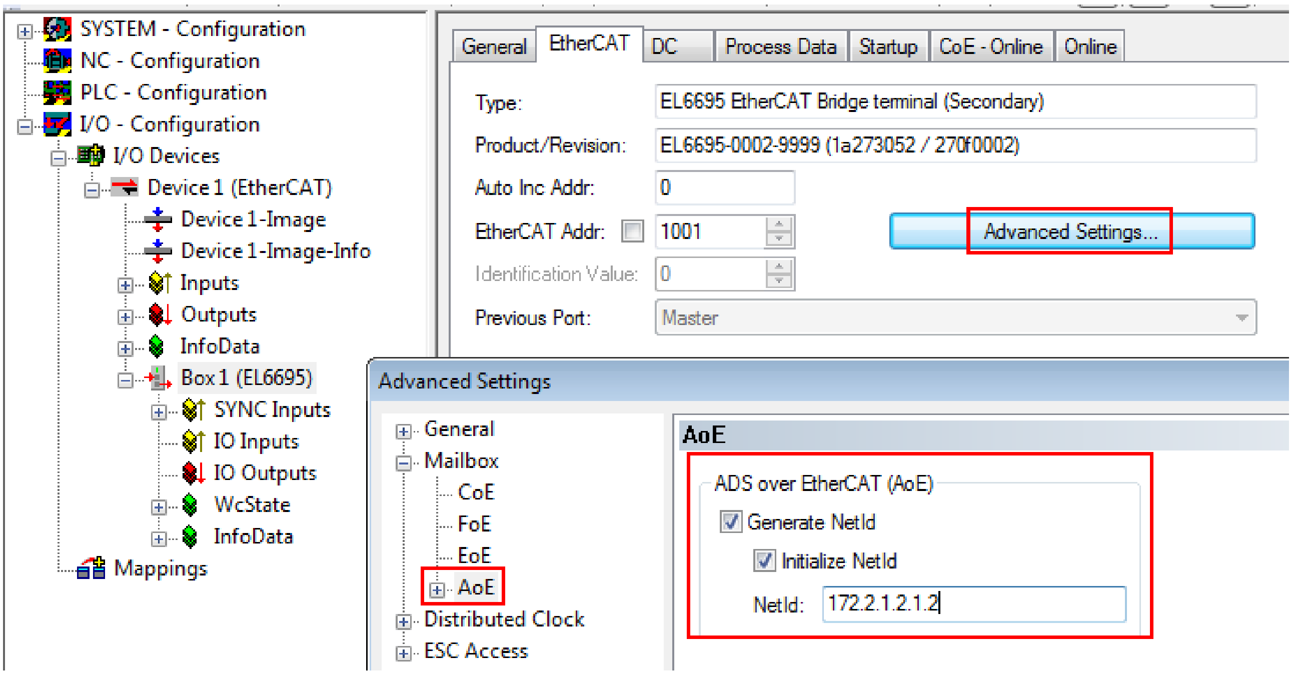


Fig. 152: Diagram showing a CoE write access with operation via AoE ADS

- The EL6695 has a CoE directory on both sides, whereby at least one side is declared user-specific. As a rule, in this mode the EL6695 does not establish connections between CoE A and CoE B. The two controllers are therefore not able to exchange parameters acyclically via CoE, and write access on one side cannot inform the master on the other side. Therefore, the following data exchange mechanism is implemented in the EL6695 via Beckhoff ADS:
  - Assumption: at least one side is a TwinCAT system or a EtherCAT master, which supports Beckhoff AoE (ADS over EtherCAT). In this example master B is such a system, master A only supports CoE.
  - A write/read access by master A to CoE A is transferred to CoE B.
  - The EL6695 reads the target Ams-NetID from CoE B 0xF820 and sends an AoE ADS write/read indication to the specified address, where a service has to pick up and process this indication. In the TwinCAT PLC this may be an FB ADSWRITEIND/ADSREADIND from the TcSystem.lib, for example.
  - From the PLC then use ADSWRITERES/ADSREADRES to send the response back to the EL6695. Use the own AMS Net ID as target address from CoE A 0xF821. During startup the EtherCAT master should load this into the EL6695 as AoE Init Cmd.
  - The EL6695 can now return the CoE write/read response to master A
  - The transfer time should be taken into account for this process.
  - This AoE mechanism operates in the CoE areas where AMS NetIDs are stored.



Type: EL6695 EtherCAT Bridge terminal (Secondary)

Product/Revision: EL6695-0002-9999 (1a273052 / 270f0002)

Auto Inc Addr: 0

EtherCAT Addr: □ 1001

Identification Value: 0

Previous Port: Master

AoE

ADS over EtherCAT (AoE)

☑ Generate NetId

☑ Initialize NetId

NetId: 172.2.1.2.1.2

Fig. 153: Adoption of the NetID

Storing of AMS addresses required

If a partial object directory is used, in which parts of the OD are exchanged by other devices and added to the OD, corresponding AMS addresses have to be stored for the respective (additional) objects (via object 0xF820 [▶ 112]).

- Mode “partial object directory”:
  - Parts of the OD are downloaded and inserted into the OD.
- Mode “partial object directory with mirrored objects”:
  - Parts of the OD are loaded into the EL6695 and inserted into the OD. In addition, mirrored objects are created in the remote OD. This mode enables a direct data link between CoE A and CoE B, without the AoE mechanism referred to above. However, this mode only covers the parameter range 0x8nnn:
  - CoE A 0x8000 – 0x87FF is mirrored to CoE B 0x8800 – 0x8FFF
  - CoE B 0x8000 – 0x87FF is mirrored to CoE A 0x8800 – 0x8FFF
  - The 0x88xy objects are therefore “Read Only” (RO)

Modified CoE objects

● The original name is stored as string “Beckhoff EL6695” in CoE object 0x10E1 “OEM Device Name”. This cannot be changed.

● The OD also defines the process data via 0x1C13 and following. The process data length on the source side and the emulated side must be the same!

● The OD is stored power failure-proof in the EL6695, maximum size 128 kbytes per side.

● The source side must be online/present and have defined the behavior of the EL6695, before the emulated side of master B can be accessed. If master B starts first, the EL6695 responds with PreOp_ERR to indicate that the process data are not defined. Once master A has then been started and the EL6695 defined, master B must run a "clear ERR" and restart side B in OP. The behavior is configurable via CoE object 0xF800:02 bit 1 (MASK_RUN_PRI_UNC) and bit 2 (MASK_RUN_SCND_UNC) (see Annex).

- The mode “complete”, “partial”, “mirrored” is set via the content of the *.coe file in the EL6695

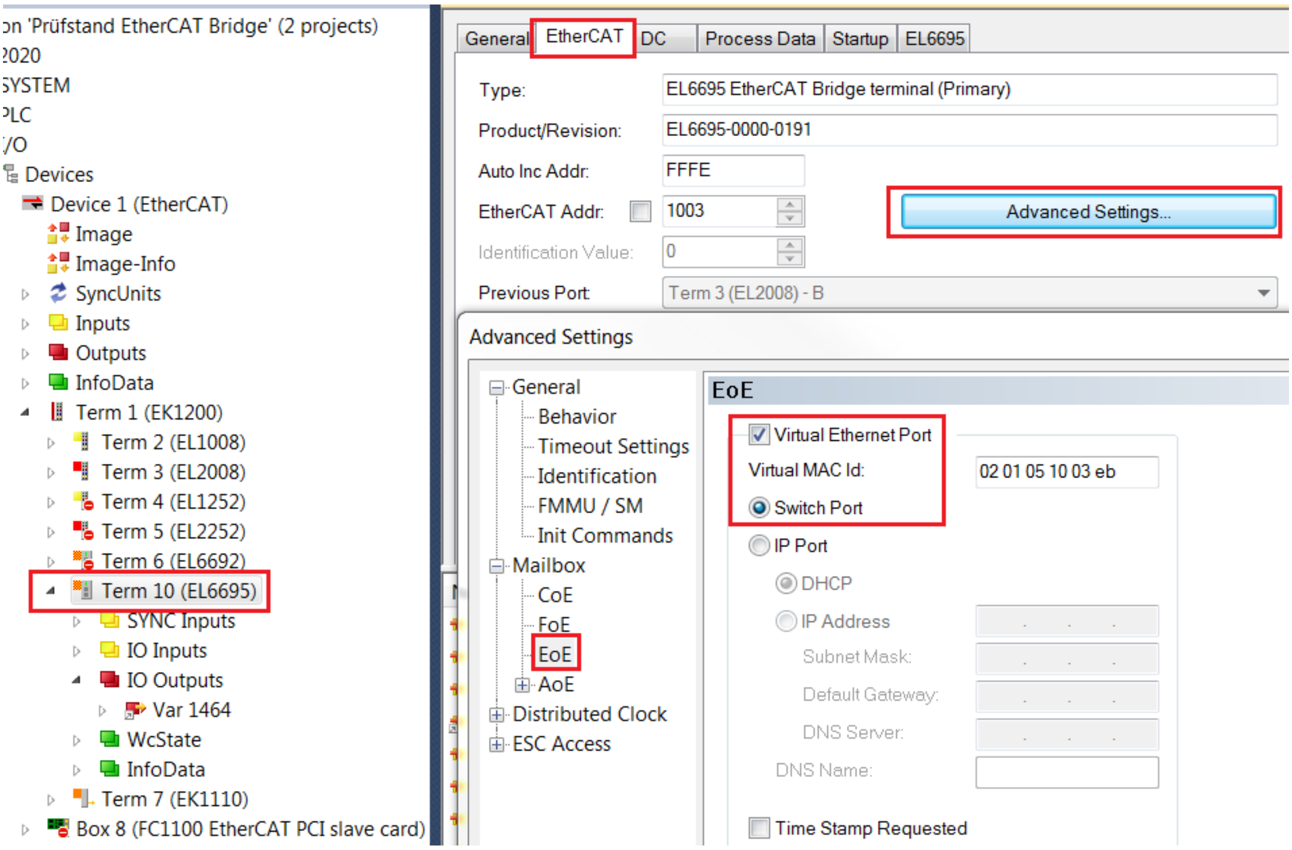
6.6.2 EoE - Ethernet over EtherCAT

Processing: Forwarding

Support for different mailbox sizes on the primary/secondary side

Since TwinCAT and the connected EtherCAT environment act as a virtual network card, Windows handles the routing of IP frames (e.g. 192.168.2.1) to the EL6692, in order to transport the frames to the EtherCAT system on the other side. Basic principles: see documentation for EL6601/EL6614.

To enable EoE in the EL6695, tick the Virtual Ethernet Port option and select Switch Port in the EtherCAT tab of the terminal in Advanced Settings under EoE.



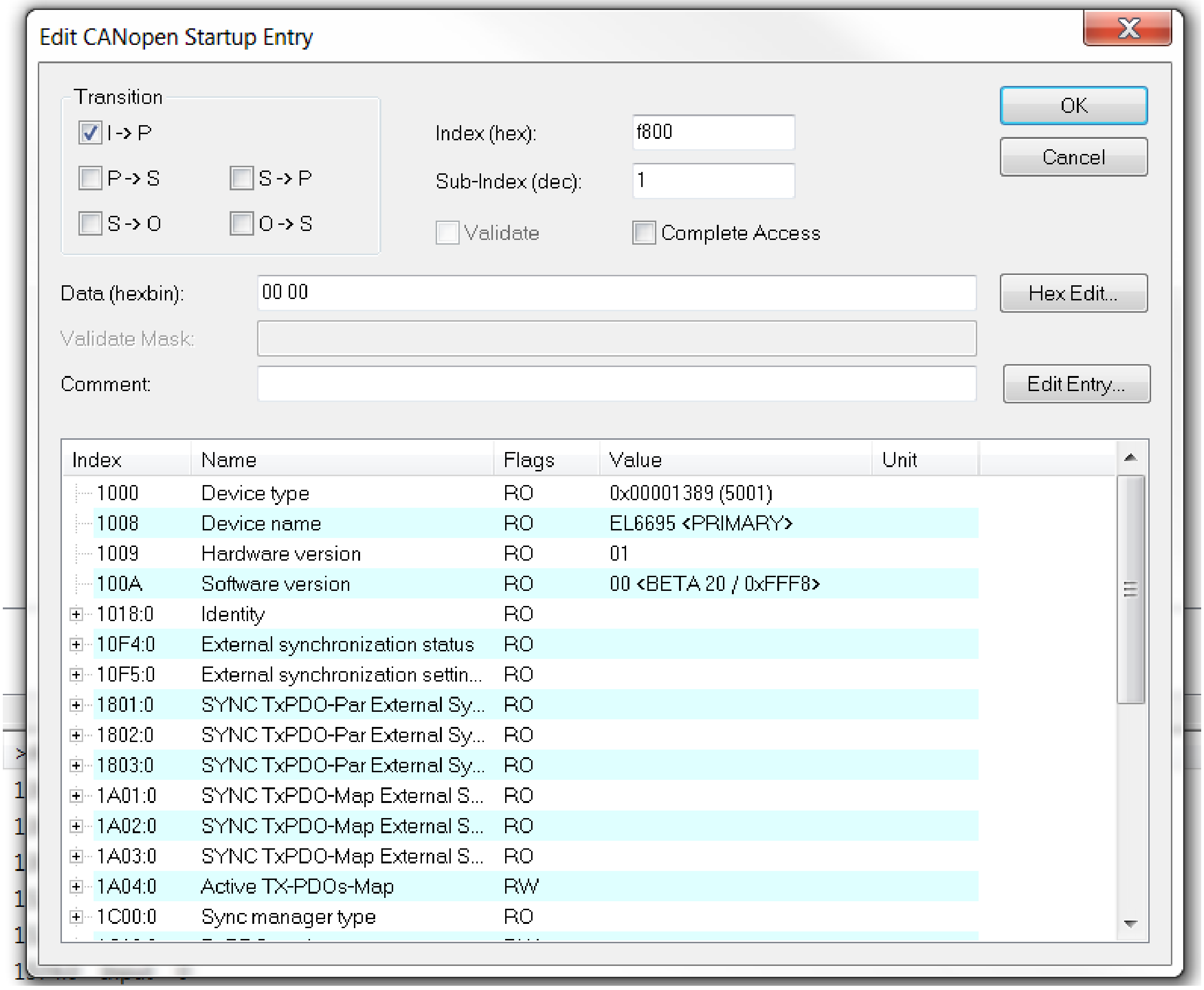
Type: EL6695 EtherCAT Bridge terminal (Primary)
Product/Revision: EL6695-0000-0191
Auto Inc Addr: FFFE
EtherCAT Addr: □ 1003
Identification Value: 0
Previous Port: Term 3 (EL2008) - B

Advanced Settings – EoE:
☑ Virtual Ethernet Port
Virtual MAC Id: 02 01 05 10 03 eb
● Switch Port
○ IP Port
○ DHCP
○ IP Address
Subnet Mask:
Default Gateway:
DNS Server:
DNS Name:
□ Time Stamp Requested

Fig. 154: EoE configuration

Activation of EoE via object 0xF800 (Device Config)

If required, check the activation of EoE via the CoE object of terminal 0xF800 (Device Config). If bit 14 is set in the entry 0xF800:01 (Config 1) (e.g. the value 0x4000 is entered), EoE mailbox protocols are blocked (see Annex). For activation set the bit to zero, or the complete 16-bit value can be set to 0x0000. Alternatively, this can be done via a startup entry (see diagram [▸ 141]).



Edit CANopen Startup Entry

Transition
☑ I -> P
□ P -> S □ S -> P
□ S -> O □ O -> S

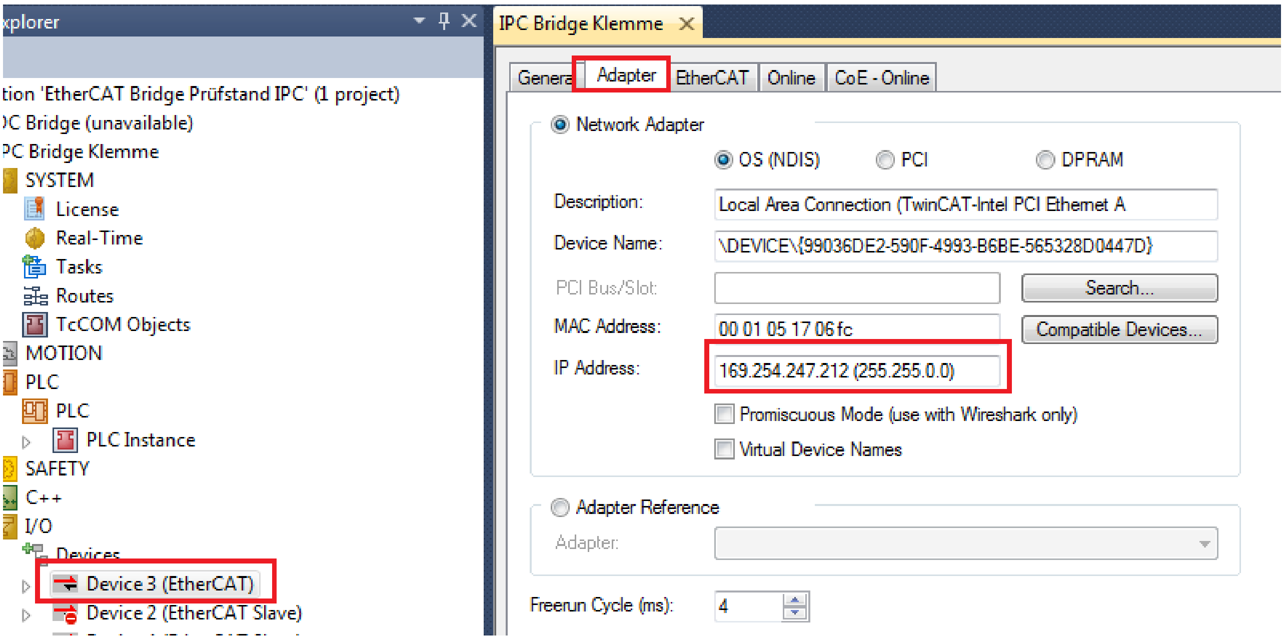
Index (hex): f800
Sub-Index (dec): 1
□ Validate □ Complete Access

Data (hexbin): 00 00
Validate Mask:
Comment:

Index	Name	Flags	Value	Unit
1000	Device type	RO	0x00001389 (5001)	
1008	Device name	RO	EL6695 <PRIMARY>	
1009	Hardware version	RO	01	
100A	Software version	RO	00 <BETA 20 / 0xFFF8>	
1018:0	Identity	RO		
10F4:0	External synchronization status	RO		
10F5:0	External synchronization settin...	RO		
1801:0	SYNC TxPDO-Par External Sy...	RO		
1802:0	SYNC TxPDO-Par External Sy...	RO		
1803:0	SYNC TxPDO-Par External Sy...	RO		
1A01:0	SYNC TxPDO-Map External S...	RO		
1A02:0	SYNC TxPDO-Map External S...	RO		
1A03:0	SYNC TxPDO-Map External S...	RO		
1A04:0	Active TX-PDOs-Map	RW		
1C00:0	Sync manager type	RO		

Fig. 155: Verifying the correct value (0x0000) of object 0xF800: EoE enabled

To ensure the EoE communication is working, the IP addresses of both EtherCAT masters have to be in the same subnet.



xplorer

tion 'EtherCAT Bridge Prüfstand IPC' (1 project)
)C Bridge (unavailable)
PC Bridge Klemme
SYSTEM
License
Real-Time
Tasks
Routes
TcCOM Objects
MOTION
PLC
PLC
PLC Instance
SAFETY
C++
I/O
Devices
Device 3 (EtherCAT)
Device 2 (EtherCAT Slave)

IPC Bridge Klemme

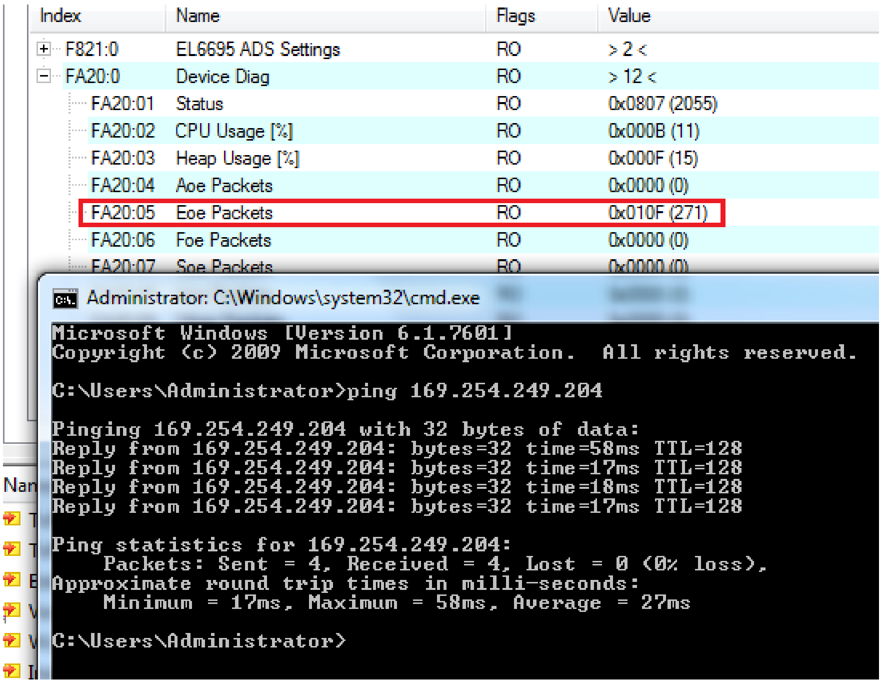
● Network Adapter
● OS (NDIS) ○ PCI ○ DPRAM
Description: Local Area Connection (TwinCAT-Intel PCI Ethernet A
Device Name: \DEVICE\{99036DE2-590F-4993-B6BE-565328D0447D}
PCI Bus/Slot:
MAC Address: 00 01 05 17 06 fc
IP Address: 169.254.247.212 (255.255.0.0)
□ Promiscuous Mode (use with Wireshark only)
□ Virtual Device Names
○ Adapter Reference
Adapter:
Freerun Cycle (ms): 4

Fig. 156: Checking the EoE IP address

In the example, master 1 has the IP 169.254.247.212 and subnet 255.255.0.0.

Master 2 can have the IP 169.254.249.204 but also has to be in subnet 255.255.0.0.

Master 2 can then be contacted by issuing a “ping” command in the command line of master 1.



Index	Name	Flags	Value
F821:0	EL6695 ADS Settings	RO	> 2 <
FA20:0	Device Diag	RO	> 12 <
FA20:01	Status	RO	0x0807 (2055)
FA20:02	CPU Usage [%]	RO	0x000B (11)
FA20:03	Heap Usage [%]	RO	0x000F (15)
FA20:04	Aoe Packets	RO	0x0000 (0)
FA20:05	Eoe Packets	RO	0x010F (271)
FA20:06	Foe Packets	RO	0x0000 (0)
FA20:07	Soe Packets	RO	0x0000 (0)

Administrator: C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Administrator>ping 169.254.249.204

Pinging 169.254.249.204 with 32 bytes of data:
Reply from 169.254.249.204: bytes=32 time=58ms TTL=128
Reply from 169.254.249.204: bytes=32 time=17ms TTL=128
Reply from 169.254.249.204: bytes=32 time=18ms TTL=128
Reply from 169.254.249.204: bytes=32 time=17ms TTL=128

Ping statistics for 169.254.249.204:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 17ms, Maximum = 58ms, Average = 27ms

C:\Users\Administrator>
```

Fig. 157: Executing “ping” from master 1 to master 2 via the command line

In the CoE objects of the EL6695, the EoE packets are then incremented in object 0xFA20:05. In this example, a ping command is used to send four packets.

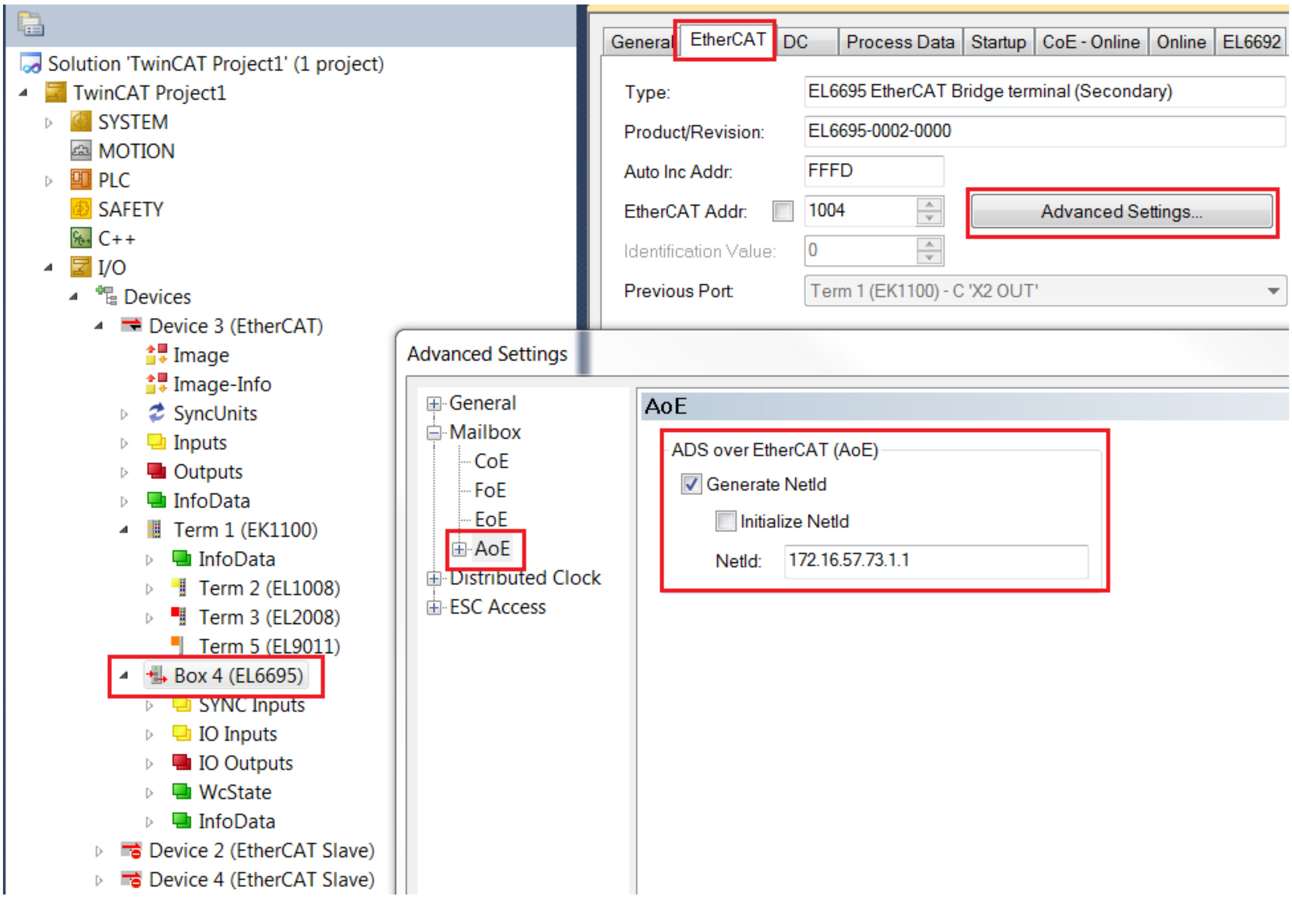
6.6.3 AoE - ADS over EtherCAT

AoE - ADS over EtherCAT

Processing: Forwarding

Support for different mailbox sizes on the primary/secondary side

ADS (Automation Device Specification) is a protocol developed and disclosed by Beckhoff for data exchange between hardware- or software-based devices. The structure of these ADS telegrams can be viewed in the Beckhoff Information System or the ETG standards. The EL6695 can transport ADS telegrams directly to the other side via AoE (ADS over EtherCAT), without underlying IP channel. To this end, communication via AoE must be enabled on both sides.



Type: EL6695 EtherCAT Bridge terminal (Secondary)
Product/Revision: EL6695-0002-0000
Auto Inc Addr: FFFD
EtherCAT Addr: □ 1004
Identification Value: 0
Previous Port: Term 1 (EK1100) - C 'X2 OUT'

AoE
ADS over EtherCAT (AoE)
☑ Generate NetId
□ Initialize NetId
NetId: 172.16.57.73.1.1

Fig. 158: Settings for ADS frame routing

To notify TwinCAT how or via which channel the ADS frames are to be routed, the address of the respective other system should be set as NetID. The example above is from the perspective of PC2.

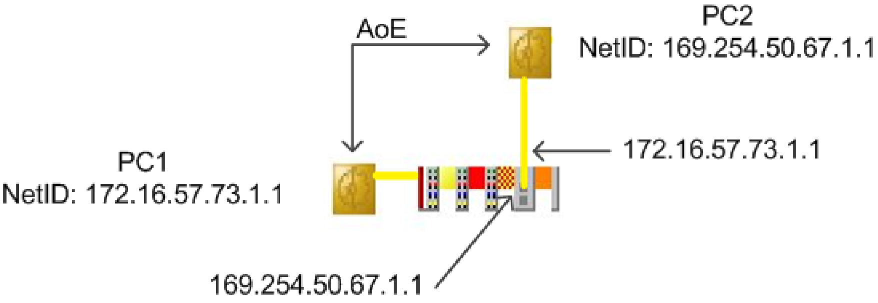


Fig. 159: Setting the NetID for AoE – ADS over EtherCAT

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.

→ example program available on request:

https://infosys.beckhoff.com/content/1033/el6695/Resources/zip/1421139595.zip

6.6.4 FoE – Filetransfer over EtherCAT

FoE – Filetransfer over EtherCAT

Essentially, this is to enable data access via two referenced memory areas of the respective participating masters (PLCs). The data can be transported from the primary side (E-bus side) to secondary side (RJ45 connection) or vice versa as a type of stream.

The maximum size per memory is 32768 bytes.

Processing in the EL6695 depends on the EtherCAT status:

- BOOT: FoE is used for the local FW update
- all other states (INIT, PreOP, SafeOP, OP): depending on the CoE password
  - no password: FoE forward to the remote site
  - with password: local saving depending on the password. Special functions are thus served.

FoE request / no password

The FoE request must be accepted and served by the opposite side. The TwinCAT versions (3.10 b4014 and 2.11 b2245) do not support this. The mailbox size must be identical on both sides.

Buffered FoE operating mode

Different mailbox sizes are permissible if the setting of [MASK_BUFFER_FOE] = bit 8 in the object 0xF800 (Device Config), subindex 02.

6.6.4.1 Requirements for FoE under TwinCAT

Programming in TwinCAT

For programming it is necessary to integrate the library “Tc2_EtherCAT” with function blocks for accessing EtherCAT master and slave devices ("Tc2_.." is usable for TwinCAT 3). The following function blocks contained in it enable the following accesses for data or file transport:

- FUNCTION_BLOCK FB_EcFoeOpen (opens the communication port)
- FUNCTION_BLOCK FB_EcFoeAccess (writing/reading of data)
- FUNCTION_BLOCK FB_EcFoeClose (closes the communication port)

Further details are available on the Beckhoff information page (infosys.beckhoff.com).

Note: the FoE transfer cannot check itself for a regular file end and thus complete data transfer. It is recommended to transmit the file size or the file completeness via other channels, e.g. verifiable meta information in the file itself.

An FoE data exchange via the EL6695 follows the following pattern:

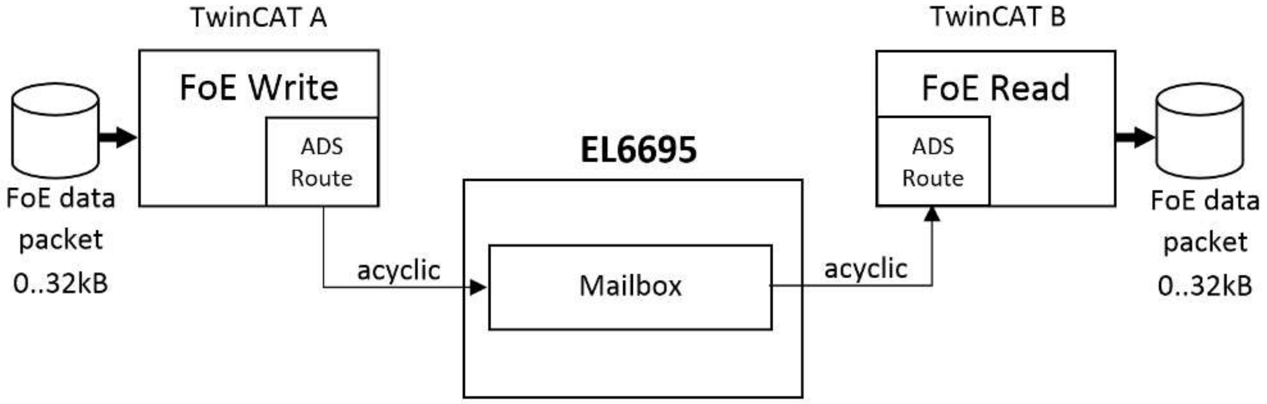


Fig. 160: FoE data exchange via the EL6695

- FoE data sets, each up to max. 32 kB in size, can be transmitted via the EL6695
- The acyclic EtherCAT communication via the EL6695 mailbox (default 1024 bytes) is used
- The EL6695 serves only as a data buffer without any FoE processing of its own

6.6.4.2 Configuration example

Configuration example

The functionality of FoE will now be illustrated on the basis of a programming example. The following illustration [▶ 146] shows the physical structure with a CX2040 Embedded PC incl. CX2100-0014 power supply unit and downstream EL2809, EL1004 und EL6695, where the Ethernet connector X001 of the CX PC is connected to the “upper” RJ45 connector X1 of the EL6695 bridge terminal:

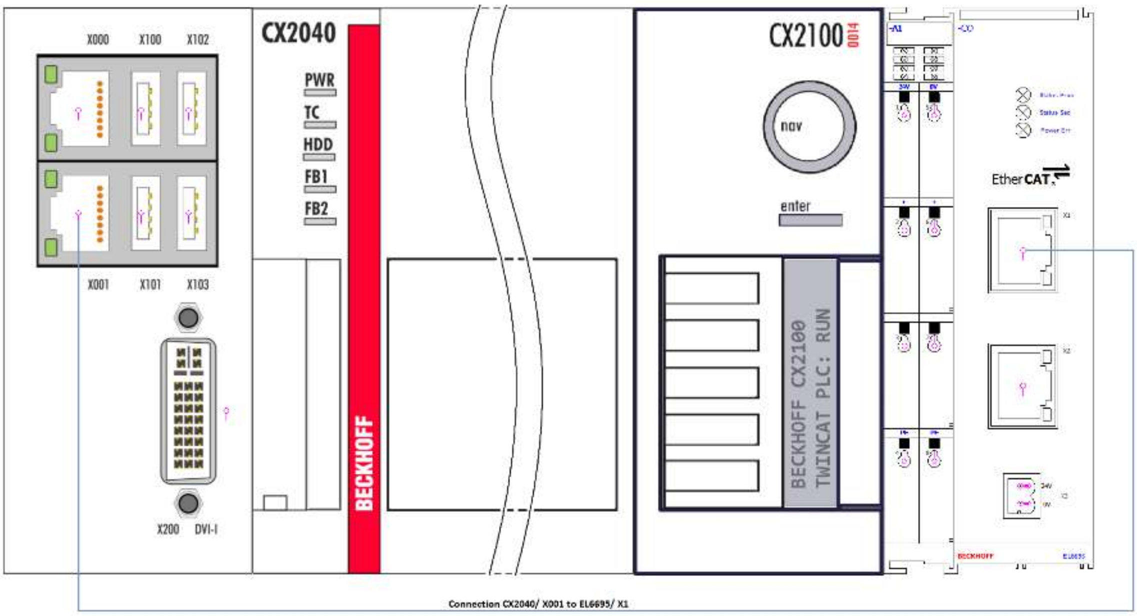
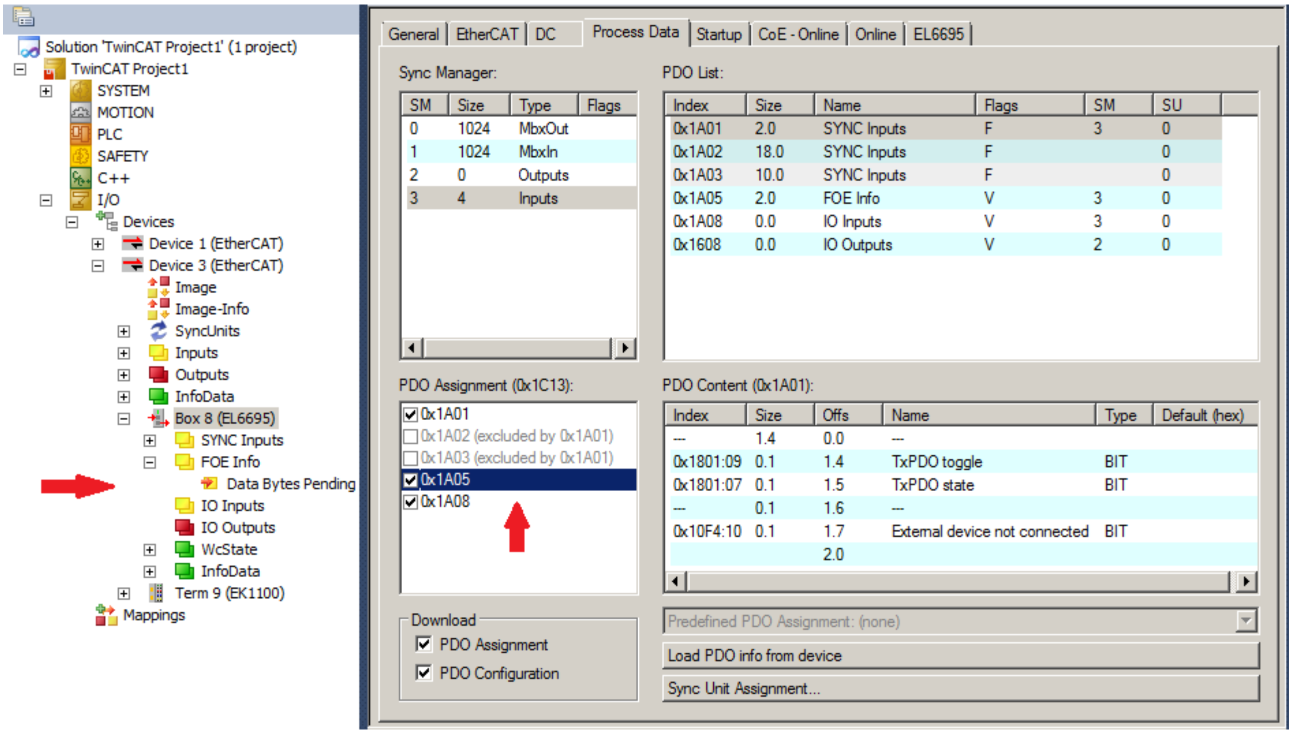


Fig. 161: Configuration with CX2040 including CX2100-0014 power supply unit and EL6695

Furthermore, the CoE object 0x1A05 is to be added in the System Manager so that it is possible on the receiving side to detect, on the basis of the info object “Data Bytes Pending”, whether data or a file have been received. To do this the respective checkbox must be activated in the process data. This is shown in the illustration only for the secondary side, which represents the receiving side in this example. Furthermore, the openly accessible variable “ScndFoeBytesToRead” contained in the TwinCAT sample program is to be linked with “Data Bytes Pending”.



```
Solution 'TwinCAT Project1' (1 project)
⊟ TwinCAT Project1
    ⊞ SYSTEM
      MOTION
      PLC
      SAFETY
      C++
    ⊟ I/O
        ⊟ Devices
            ⊞ Device 1 (EtherCAT)
            ⊟ Device 3 (EtherCAT)
                  Image
                  Image-Info
                ⊞ SyncUnits
                ⊞ Inputs
                ⊞ Outputs
                ⊞ InfoData
                ⊟ Box 8 (EL6695)
                    ⊞ SYNC Inputs
                    ⊟ FOE Info
                          Data Bytes Pending
                      IO Inputs
                      IO Outputs
                    ⊞ WcState
                    ⊞ InfoData
                ⊞ Term 9 (EK1100)
          Mappings
```

Sync Manager:

SM	Size	Type	Flags
0	1024	MbxOut	
1	1024	MbxIn	
2	0	Outputs	
3	4	Inputs	

PDO List:

Index	Size	Name	Flags	SM	SU
0x1A01	2.0	SYNC Inputs	F	3	0
0x1A02	18.0	SYNC Inputs	F		0
0x1A03	10.0	SYNC Inputs	F		0
0x1A05	2.0	FOE Info	V	3	0
0x1A08	0.0	IO Inputs	V	3	0
0x1608	0.0	IO Outputs	V	2	0

PDO Assignment (0x1C13):

☑ 0x1A01
☐ 0x1A02 (excluded by 0x1A01)
☐ 0x1A03 (excluded by 0x1A01)
☑ 0x1A05
☑ 0x1A08

PDO Content (0x1A01):

Index	Size	Offs	Name	Type	Default (hex)
---	1.4	0.0	---		
0x1801:09	0.1	1.4	TxPDO toggle	BIT	
0x1801:07	0.1	1.5	TxPDO state	BIT	
---	0.1	1.6	---		
0x10F4:10	0.1	1.7	External device not connected	BIT	
		2.0			

Download
☑ PDO Assignment
☑ PDO Configuration

Predefined PDO Assignment: (none)

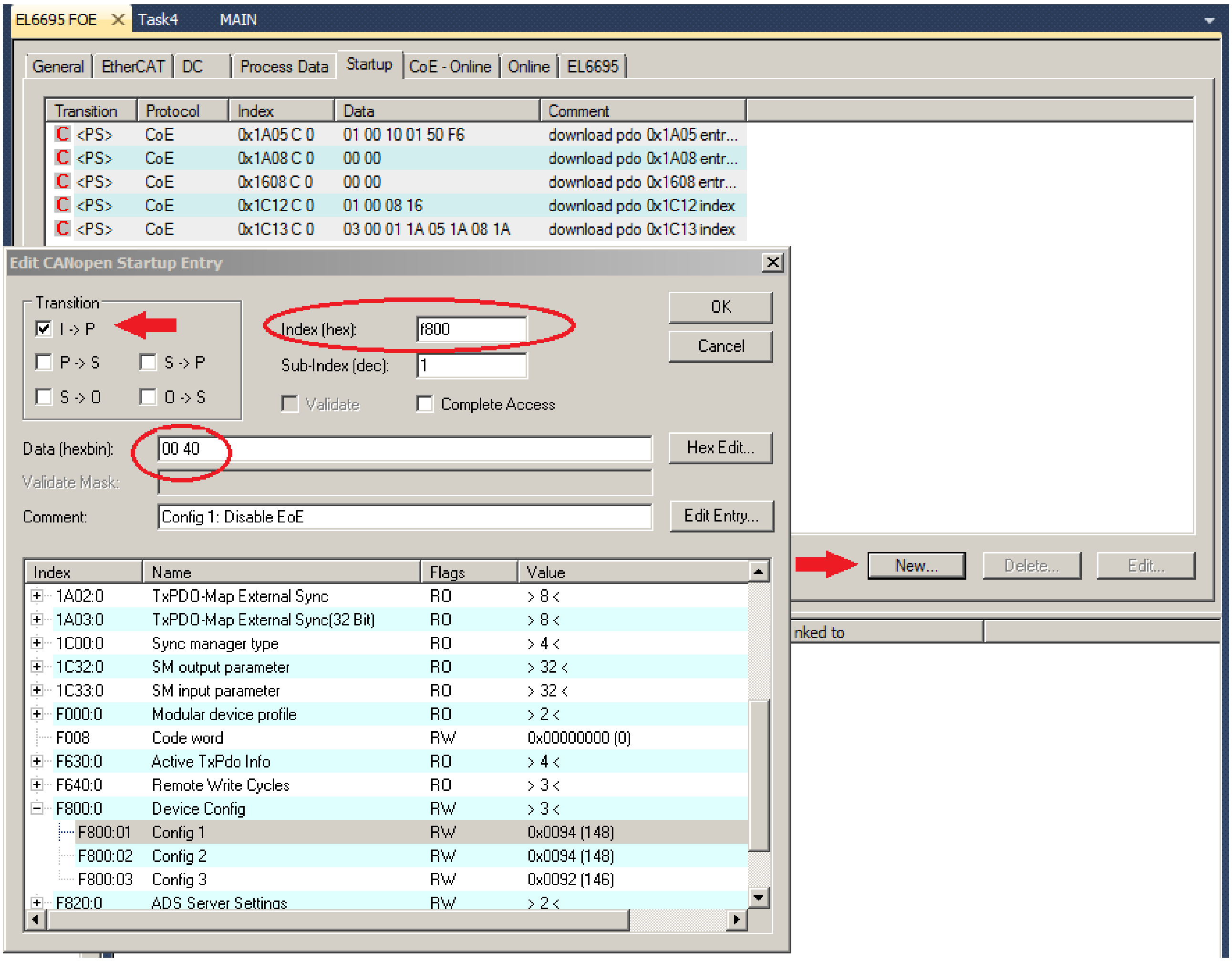
Fig. 162: Addition of 0x1A05 (Data Bytes Pending) on the secondary side

Init – startup configuration

In accordance with the configuration definition of the object 0xF800 (see CoE/Parameter directory – profile-specific objects), two values have to be entered in 0xF800 for the transition from the Init state to the PreOp state (I → P):

- The value 0x4000 in 0xF800:01 (to deactivate EoE)
- The value 0x0100 in 0xF800:02 (special mode to reserve an FoE buffer memory on the EL6695)

The procedure is shown in the following under the user interface of the System Manager (TC3.1)“Startup” tab on the primary *or* secondary side marked terminal or box:



Transition	Protocol	Index	Data	Comment
C <PS>	CoE	0x1A05 C 0	01 00 10 01 50 F6	download pdo 0x1A05 entr...
C <PS>	CoE	0x1A08 C 0	00 00	download pdo 0x1A08 entr...
C <PS>	CoE	0x1608 C 0	00 00	download pdo 0x1608 entr...
C <PS>	CoE	0x1C12 C 0	01 00 08 16	download pdo 0x1C12 index
C <PS>	CoE	0x1C13 C 0	03 00 01 1A 05 1A 08 1A	download pdo 0x1C13 index

☑ I -> P
□ P -> S □ S -> P
□ S -> O □ O -> S

Index (hex): f800

Sub-Index (dec): 1

□ Validate □ Complete Access

Data (hexbin): 00 40

Validate Mask:

Comment: Config 1: Disable EoE

Index	Name	Flags	Value
+ 1A02:0	TxPDO-Map External Sync	RO	> 8 <
+ 1A03:0	TxPDO-Map External Sync(32 Bit)	RO	> 8 <
+ 1C00:0	Sync manager type	RO	> 4 <
+ 1C32:0	SM output parameter	RO	> 32 <
+ 1C33:0	SM input parameter	RO	> 32 <
+ F000:0	Modular device profile	RO	> 2 <
F008	Code word	RW	0x00000000 (0)
+ F630:0	Active TxPdo Info	RO	> 4 <
+ F640:0	Remote Write Cycles	RO	> 3 <
− F800:0	Device Config	RW	> 3 <
F800:01	Config 1	RW	0x0094 (148)
F800:02	Config 2	RW	0x0094 (148)
F800:03	Config 3	RW	0x0092 (146)
+ F820:0	ADS Server Settings	RW	> 2 <

Fig. 163: Entry of the value 0x4000 in the device configuration 0xF800:01 (comment = Config 1: Disable EoE)

The value 0x4000 (type 00 40) is to be entered here for the object 0xF800, subindex 01 in order to deactivate EoE. This is necessary only due to the use in this example of the connection of the primary and secondary side of the terminal, since both sides are connected to an EtherCAT master.

Blocking the EoE protocol under FoE

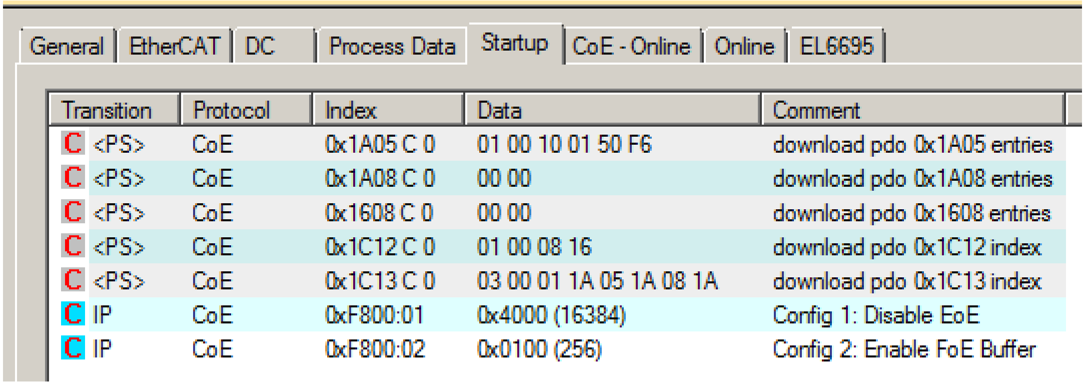
If the primary and secondary sides are connected in one EtherCAT line, the EoE protocol must be blocked by setting bit 14 (0x4000) in object 0xF800:01 on the terminal, since otherwise the terminal will be blocked (due to repeated ARP Ethernet requests).

Furthermore, the value 0x0100 (type 00 01) must be entered for 0xF800:02 (Comment = Config 2: Enable FoE Buffer). It doesn’t matter whether this is done on the primary or the secondary side of the EL6695 bridge terminal, since these settings are always also adopted on the respective other side.

Changing the Device Config 0xF800

The object 0xF800 is configurable on both the primary and secondary side and is always adopted by the respective other side.

Care must be taken that the correct transition is selected: P → S must be off, I → P must be activated. Once this has been done it should look like the following illustration:



Transition	Protocol	Index	Data	Comment
C <PS>	CoE	0x1A05 C 0	01 00 10 01 50 F6	download pdo 0x1A05 entries
C <PS>	CoE	0x1A08 C 0	00 00	download pdo 0x1A08 entries
C <PS>	CoE	0x1608 C 0	00 00	download pdo 0x1608 entries
C <PS>	CoE	0x1C12 C 0	01 00 08 16	download pdo 0x1C12 index
C <PS>	CoE	0x1C13 C 0	03 00 01 1A 05 1A 08 1A	download pdo 0x1C13 index
C IP	CoE	0xF800:01	0x4000 (16384)	Config 1: Disable EoE
C IP	CoE	0xF800:02	0x0100 (256)	Config 2: Enable FoE Buffer

Fig. 164: Entry of both values in the device configuration 0xF800

Explanations regarding the program example

The program example in the attachment, which is intended to illustrate an FoE data transfer, is based on the state diagram for the write and read access illustrated in the following.

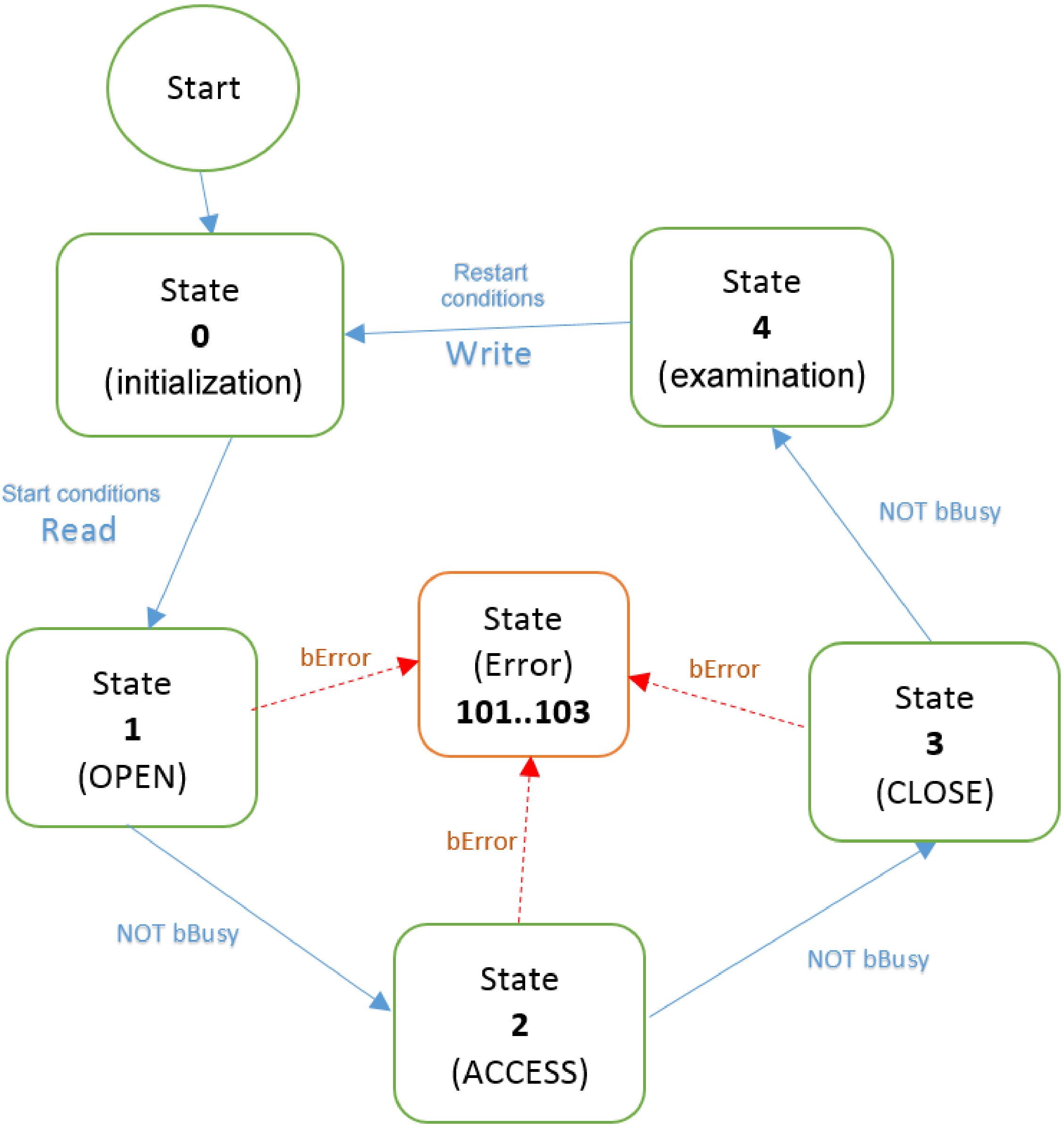


Fig. 165: State diagram for the example program: writing and reading random data by “OPEN”, “ACCESS” and “CLOSE”

The **bEnabled** flag defined in the program is intended for the control of the program and is used for the start condition or restart condition. On account of the additional declaration AT%I*, this flag can be linked with a “real” input of an input terminal in order to control the program “from the outside”, e.g. with a connected button (up +).

- Start condition Read: **bEnabled** must be TRUE (e.g. actuate externally connected button); in addition, “ScndFoeBytesToRead” is checked for incoming data and must be > 0.
- Restart condition Write: **bEnabled** must be FALSE again (e.g. release externally connected button again).

In addition, the **bEnabled** flag is tested for TRUE within state 1 of the state machine for writing the data, whereby the write process is then only started by an OPEN.

On account of the additional declaration AT%Q*, **bDataEqual** can be linked with an output variable of a terminal that provides digital outputs. By this one can see at the end whether the correct data transport has taken place.

The individual states **0** to **4** are programmed as follows as **iWrState**:

- **[iWrState =0]:** execution of initializations:
    - Set FALSE → **bDataEqual**
    - Generate random data in **aWrBuffer**
    - Preparation of the call of the function block for OPEN: Set FALSE → **bExecute**
    - Set next state: 1 → **iWrState**
- **[iWrState =1]:** Query the input variable **bEnabled** as to whether to start calling the function block for OPEN. If **bEnabled** = TRUE, the OPEN function block is called with the following parameters:
    - EC_MasterNetId_Wr → **sNetId**
    - EL6695_WR_EcAddr → **nPort**
    - 16#12345678 → **dwPass**
    - 1 → **eMode** (identifier for writing)
    - TRUE → **bExecute**
    - T#10S → **tTimeout**
    - This call is repeated until a query of the **bBusy** flag returns FALSE; the following then takes place:
    - Set next state: 2 → **iWrState;** if **bBusy** = TRUE, then no change of state takes place. The handling of the error cases by the **bError** flag of the OPEN function block (for read/write) is not considered in this example or for any other function block (ACCESS, CLOSE for read/write).
    - Preparation of the call of the function block for ACCESS: Set FALSE → **bExecute**
- **[iWrState =2]:** The ACCESS function block is called with the following parameters:
    - fbWrFoeOpen.hFoe → **hFoe**
    - ADR(aWrBuffer) →**pBuffer**
    - TRUE → **bExecute**
    - T#14S → **tTimeout**
    - This call is repeated until a query of the **bBusy** flag returns FALSE; the following then takes place:
    - Set next state: 3 → **iWrState**
    - Preparation of the call of the function block for CLOSE: Set FALSE → **bExecute**
- **[iWrState =3]:** The CLOSE function block is called with the following parameters:
    - fbWrFoeOpen.hFoe → **hFoe**
    - TRUE → **bExecute**
    - T#14S → **tTimeout**
    - This call is repeated until a query of the **bBusy** flag returns FALSE; the following then takes place:
    - Set next state: 4 → **iWrState**
- **[iWrState =4]:** in this state the input variable **bEnabled** is merely queried as to whether it is FALSE again. The program then begins again with the start state:
    - If bEnabled = FALSE, then set the next state: 0 → **iWrState**

At the next start new random numbers are generated again, which should once again correspond to the read values when compared. The **bDataEqual** flag is provided for and can indicate this, as we shall see in the following.

The state diagram for the read access looks similar in principle and has in state **4** a program section for the data comparison of the written values with the read values.

The individual states 0 to 4 are programmed as follows as **iRdState**:

- **[iRdState =0]:** in this state the input variable **bEnabled** is queried as to whether it is TRUE. With that the program begins with the querying of the input variable “ScndFoeBytesToRead”, which is linked to [FoE Info].Data Bytes Pending (object 0x1A05), in order to determine whether data are “present” on the receiving side.
    - Query of **ScndFoeBytesToRead** if **bDataEqual** = TRUE
    - Set next state: 1 →**iRdState** if **ScndFoeBytesToRead** = TRUE

    - Set to prepare the call of the function block for OPEN: set FALSE → **bExecute** if **ScndFoeBytesToRead** = TRUE
- **[iRdState =1]:** Query the input variable **bEnabled** as to whether to start calling the function block for OPEN. If **bEnabled** = TRUE, the OPEN function block is called with the following parameters:
    - EC_MasterNetId_Rd → **sNetId**
    - EL6695_RD_EcAddr → **nPort**
    - 16#12345678 → **dwPass**
    - 0 → **eMode** (identifier for reading)
    - TRUE → **bExecute**
    - T#10S → **tTimeout**
    - This call is repeated until a query of the **bBusy** flag returns FALSE; the following then takes place:
    - Set next state: 2 → **iRdState**
    - Preparation of the call of the function block for ACCESS: Set FALSE → **bExecute**
- **[iRdState =2]:** The ACCESS function block is called with the following parameters:
    - fbRdFoeOpen.hFoe → **hFoe**
    - ADR(aRdBuffer) →**pBuffer**
    - TRUE → **bExecute**
    - T#14S → **tTimeout**
    - This call is repeated until a query of the **bBusy** flag returns FALSE; the following then takes place:
    - Set next state: 3 → **iRdState**
    - Preparation of the call of the function block for CLOSE: Set FALSE → **bExecute**
- **[iRdState =3]:** The CLOSE function block is called with the following parameters:
    - fbRdFoeOpen.hFoe → **hFoe**
    - TRUE → **bExecute**
    - T#14S → **tTimeout**
    - This call is repeated until a query of the **bBusy** flag returns FALSE; the following then takes place:
    - Set next state: 4 → **iRdState**
- **[iRdState =4]:** In this state the check of both memory areas takes place at the end and the **bDataEqual** flag is set to TRUE if both memory areas are identical. It is recommended to make this flag "visible" as an output variable so that, for example, it can be switched to an output via an output terminal.
    - Set next state: 0 → **iRdState**

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.

→ Download example program FoE:

https://infosys.beckhoff.com/content/1033/el6695/Resources/zip/1421822987.zip

Preparations for starting the sample programs (tnzip file / TwinCAT 3)

- Click on the download button to save the Zip archive locally on your hard disk, then unzip the *.tnzip archive file in a temporary folder.

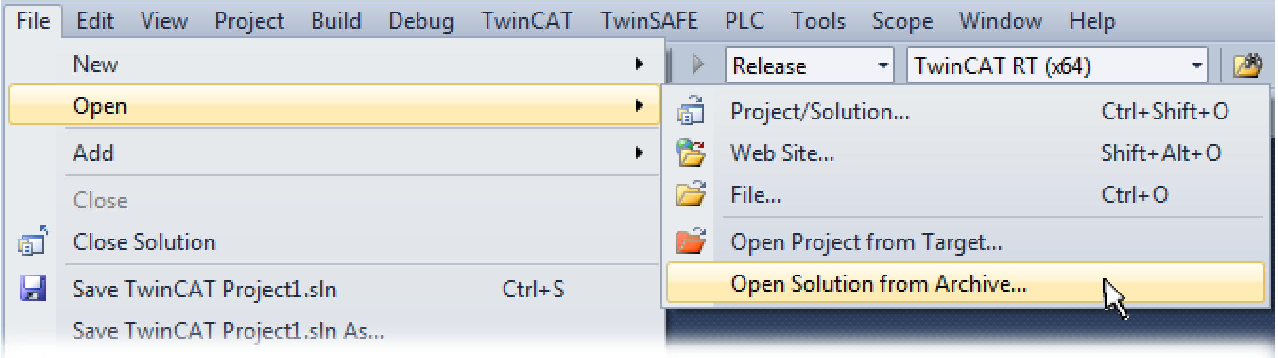
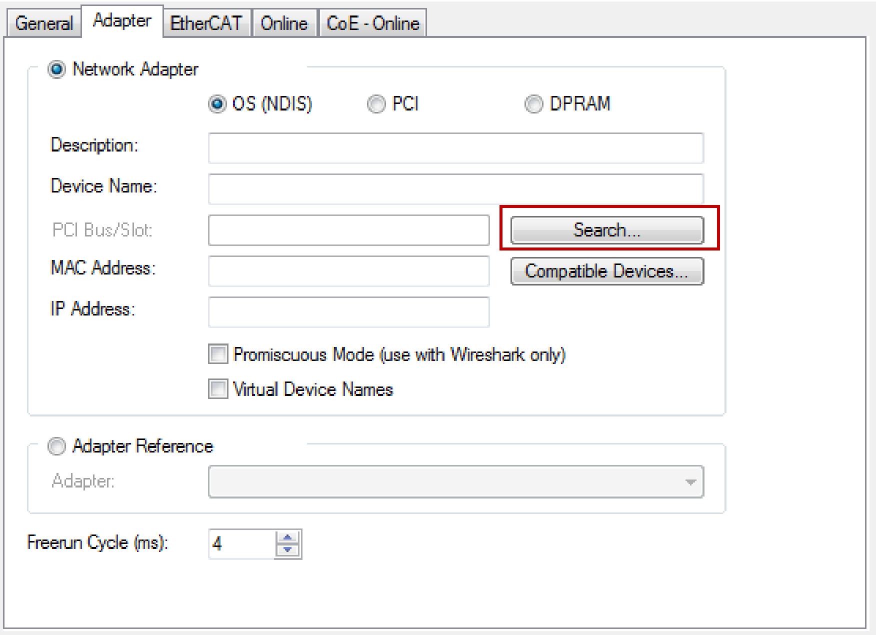


Fig. 166: Opening the *. tnzip archive

- Select the .tnzip file (sample program).
- A further selection window opens. Select the destination directory for storing the project.
- For a description of the general PLC commissioning procedure and starting the program please refer to the terminal documentation or the EtherCAT system documentation.
- The EtherCAT device of the example should usually be declared your present system. After selection of the EtherCAT device in the “Solutionexplorer” select the “Adapter” tab and click on “Search...”:



● Network Adapter

● OS (NDIS) ○ PCI ○ DPRAM

Description:

Device Name:

PCI Bus/Slot:

MAC Address:

IP Address:

□ Promiscuous Mode (use with Wireshark only)

□ Virtual Device Names

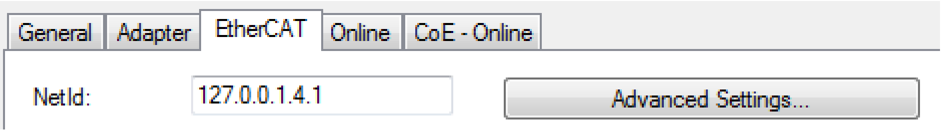
○ Adapter Reference

Adapter:

Freerun Cycle (ms): 4

Fig. 167: Search of the existing HW configuration for the EtherCAT configuration of the example

- Checking NetId: the “EtherCAT” tab of the EtherCAT device shows the configured NetId:



NetId: 127.0.0.1.4.1

The first 4 numbers have to be identical with the project NetId of the target system. The project NetId can be viewed within the TwinCAT environment above, where a pull down menu can be opened to choose a target system (by clicking right in the text field). The number blocks are placed in brackets there next to each computer name of a target system.

- Modify the NetId: By right clicking on “EtherCAT device” within the solution explorer a context menu opens where “Change NetId...” have to be selected. The first four numbers of the NetId of the target computer have to be entered; the both last values are 4.1 usually.
Example:
    - NetId of project: myComputer (123.45.67.89.1.1)
    - Entry via „Change NetId...“: 123.45.67.89.4.1

6.6.4.3 Sample: FoE data throughput

A sample of an FoE throughput determination using an EL6695 is shown below. The values should be regarded as an example of this layout and as coarse guiding values. The real achievable throughput in the respective application must be determined otherwise if necessary.

The TwinCAT program used for that is not part of this documentation; it is based on the specified FoE sample program [▸ 152].

The data throughput is defined as the ratio of data quantity to the time required (data quantity per unit of time). With the FoE access blocks already described, a given data quantity is written to or read from the EL6695. The EL6695 is designed to establish communication between two different TwinCAT systems. However, it is also possible with the terminal to establish a connection between the primary and secondary side within a TwinCAT system, for example if the connection of its secondary side via X1 originates from output X2 of the EK1100 coupler. There are then also two independent EtherCAT segments in a TwinCAT system (cf. Example configuration [▸ 146]).

The time measurement takes place indirectly via the number of task cycles required for each completely executed reading access (i.e. the number of bytes written is the same as the number read) via the FB_EcFoeAccess function block (see Notes on the program example [▸ 149]). If data are pending, the "DataBytesPending" PDO of the side of the terminal to be read is > 0 and marks the start of the read access. The end of the reading access is determined from the known written data quantity (with two independent systems the file size must be known).

With a task cycle time of 1 ms, the data throughput is simply calculated as follows:
{data quantity [bytes]} / {number of task cycles} = throughput in [kB/s]

Measurement sequence

- the test environment determines the number of task cycles required by the terminal for a reading access. A data set of a different size with different data packet sizes written to the terminal was repeatedly and completely read out again. The number of task cycles was thereby counted for each individual reading access.
- The FoE throughput of the test program without the intermediately connected EL6695 was also determined. This results in the "virtual" time requirement of the test program.
- the actual number of task cycles required by the terminal (and thus the data throughput) then results from the difference between the required "real" task cycles (with EL6695) and the virtual task cycles.
- The entire throughput test encompasses several thousand individual tests with different data quantities and data packet sizes.

Configuration in the sample:

- C6015 + EK1100 + EL6695 (primary side) + EL9011
- EK1100 + EL6695-0002 (secondary side)

Key data for the sample:

- Cycle time = 1 ms
- The test run encompassed all FoE and packet sizes. Since the throughput does not change significantly outside of defined key ranges, the following limits were chosen for the compressed graphic illustration in the following:
- Data quantity transmitted by FoE (FoE file size): from 1167 to 32767 bytes in 50 steps, with 632 bytes per step
- Data packets (i.e. packets into which the FoE file is divided): from 30 to 470 bytes in 40 steps, with 11 bytes per step

Result:

Data packet size (incremented in steps) = X-axis

Data quantity (incremented in steps) = Y-axis

Number of task cycles required = Z-axis; equivalent to Z(X, Y)

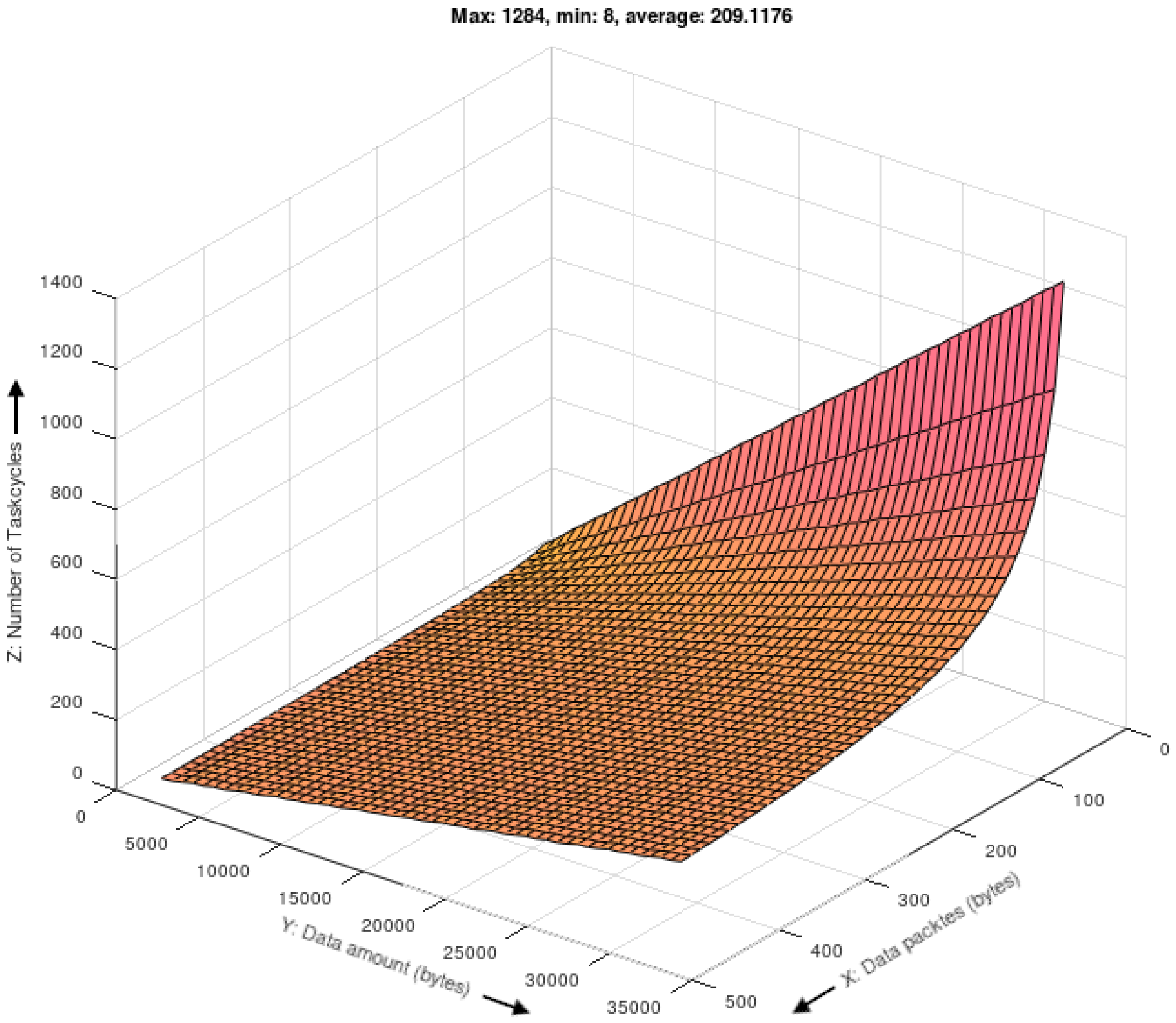


Fig. 168: FoE throughput test with EL6695: Number of task cycles depending on data packet size and data quantity

However, this also includes the system-related virtual values on account of the sequential data access (see above). If these are also recorded accordingly and subtracted from the individual values of the Z-axis for each X/Y test point, the result is a quasi linear diagram curve that is no longer dependent on the data packet size:

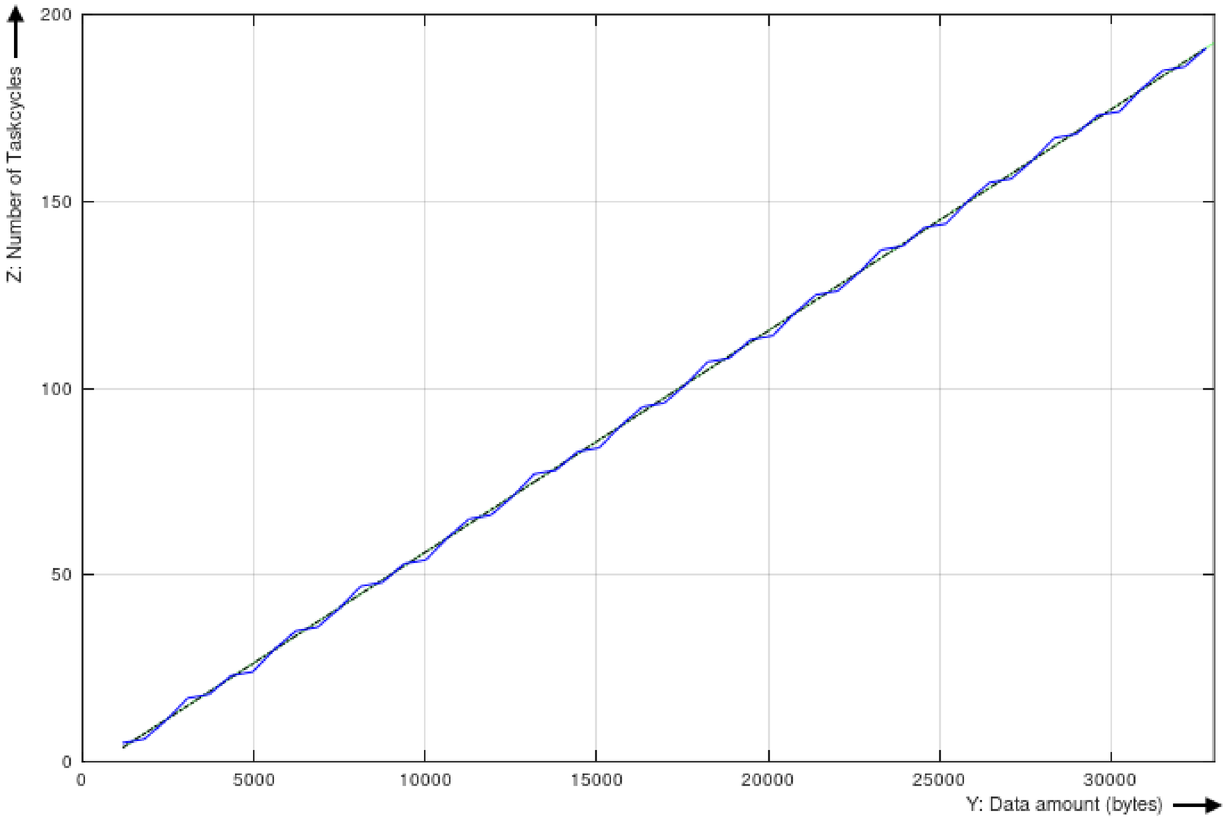


Fig. 169: About the example FoE data throughput of the EL6695

In the linear approximation this FoE example results in a throughput of about **171 kB/s**.

6.6.5 VoE - vendor-specific protocol over EtherCAT

VoE enables implementation of a user-specific protocol, so that the internal mailbox can be used for a special, newly defined data transport.

For further information please contact Beckhoff support, providing pertinent data (TwinCAT version, required performance values, cycle time, data quantities etc.)

6.6.6 SoE - Servo Drive Profile over EtherCAT

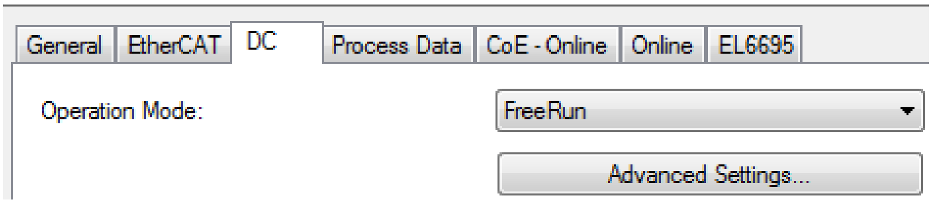
For further information please contact Beckhoff support, providing pertinent data (TwinCAT version, required performance values, cycle time, data quantities etc.)

6.7 Distributed Clocks

DC support in the EtherCAT master

The distributed clock unit of the EL6695 has the DC system time, but no sync/latch unit. The corresponding initialization routine is supported by TwinCAT 3 from b4018.4, TwinCAT 2 from b2248.

The EL6695 configuration can be viewed under the tab. For the data exchange it has the default setting “no synchronization” (“FreeRun”).

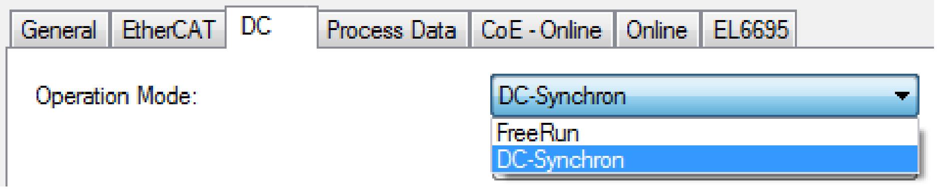


Operation Mode: FreeRun

Fig. 170: Distributed Clocks: no synchronization

Since the EL6695 contains two complete EtherCAT slaves, the two distributed clock units are basically independent of each other. The EL6695 supports two DC synchronization mechanisms:

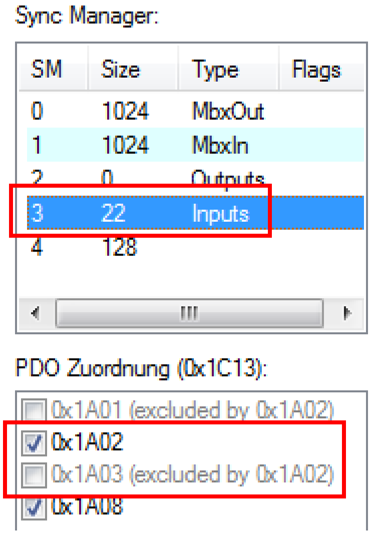
- Like its predecessor (EL6692) it can make the internal/external timestamp information available to the sync slave side, thereby offering the EtherCAT master stored there the option to adjust its real-time/ EtherCAT cycle. Both control directions are possible.
    - Application under TwinCAT: Set the EL6695 to DC mode, so that TwinCAT can use it as reference clock



Operation Mode: DC-Synchron

Fig. 171: Distributed Clocks: synchronization

    - Show and activate a timestamp PDO (0x1A02 for 64 bits or 0x1A03 for 32 bits). TwinCAT thus detects that this terminal can be used as an external reference clock and reads in the time stamps.



Sync Manager:

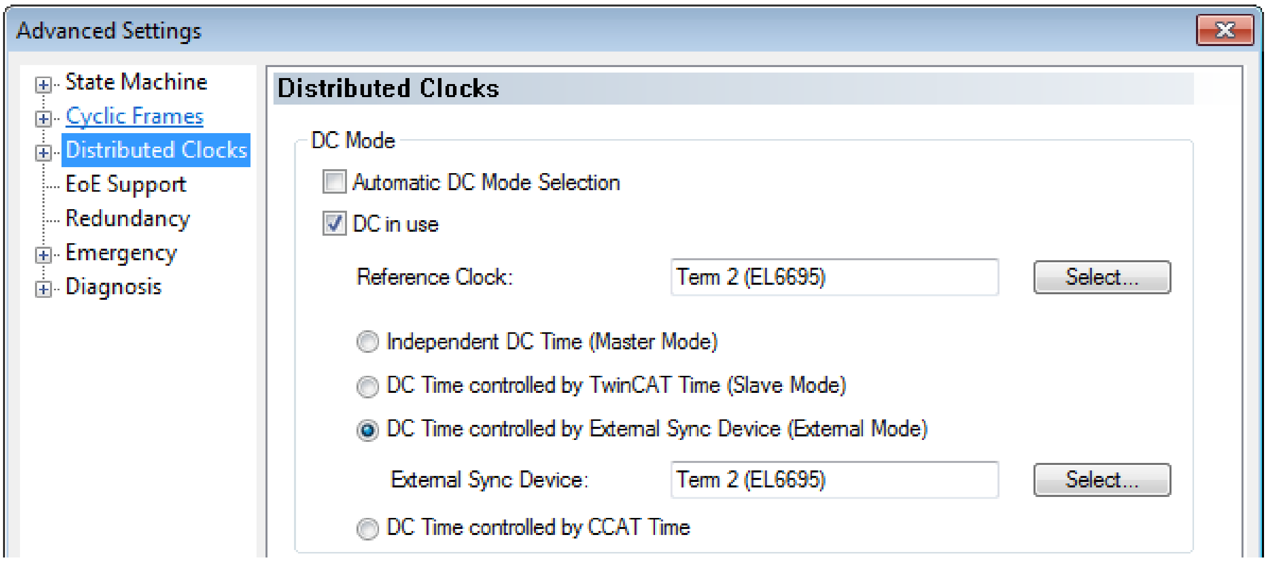
SM	Size	Type	Flags
0	1024	MbxOut	
1	1024	MbxIn	
2	0	Outputs	
3	22	Inputs	
4	128		

PDO Zuordnung (0x1C13):

□ 0x1A01 (excluded by 0x1A02)
☑ 0x1A02
□ 0x1A03 (excluded by 0x1A02)
☑ 0x1A08

Fig. 172: Show timestamp PDO 0x1A02 or 0x1A03

    - The external reference clock can then be selected in the “EtherCAT DC master settings”:



Distributed Clocks

DC Mode

☐ Automatic DC Mode Selection

☑ DC in use

Reference Clock: Term 2 (EL6695)

○ Independent DC Time (Master Mode)

○ DC Time controlled by TwinCAT Time (Slave Mode)

● DC Time controlled by External Sync Device (External Mode)

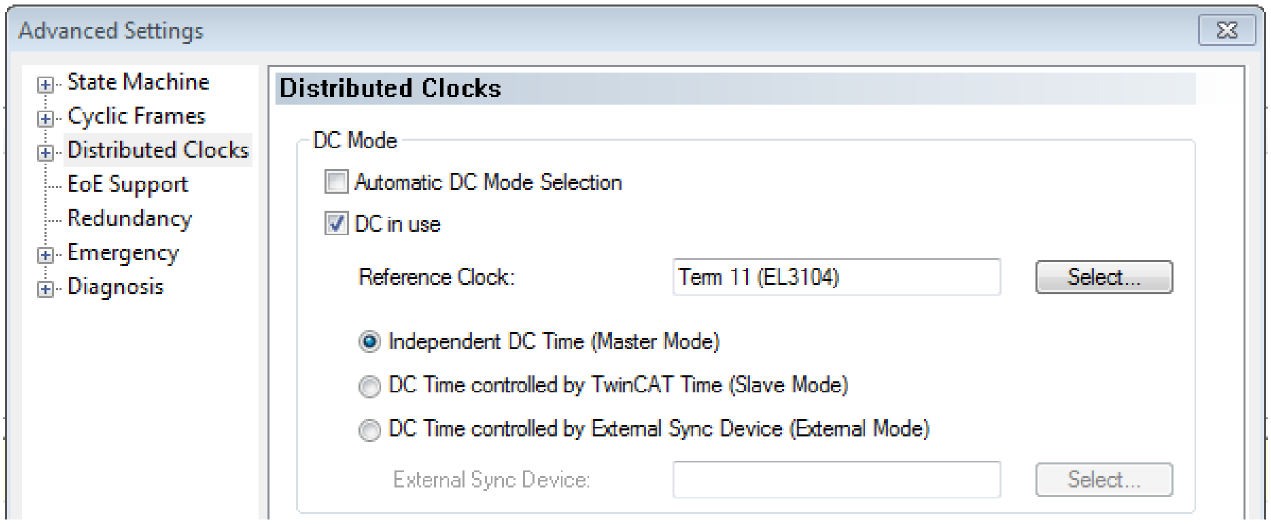
External Sync Device: Term 2 (EL6695)

○ DC Time controlled by CCAT Time

Fig. 173: Distributed Clocks: Selecting the external reference clock (on the primary side in this example)

This setting is only required on the “SyncSlave” side.

On the “SyncMaster” side an EL3104 (DC-capable) may be entered, for example:



Distributed Clocks

DC Mode

☐ Automatic DC Mode Selection

☑ DC in use

Reference Clock: Term 11 (EL3104)

● Independent DC Time (Master Mode)

○ DC Time controlled by TwinCAT Time (Slave Mode)

○ DC Time controlled by External Sync Device (External Mode)

External Sync Device:

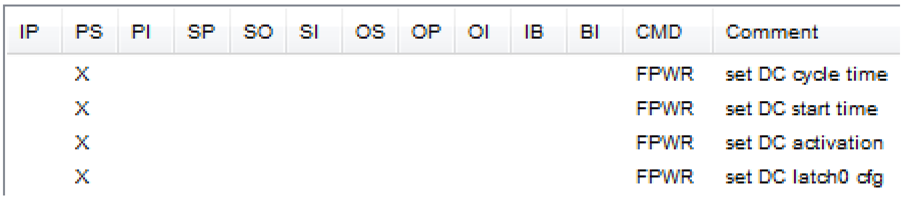
Fig. 174: Distributed Clocks: Selecting the EL3104 as “SyncMaster” (on the secondary side in this example)

- If the EL6695 is the first DC-capable slave in both EtherCAT systems, a direct DC coupling of the two bridge sides can be enabled without having to notify the EtherCAT masters. In this case, both EtherCAT systems/masters follow the EL6695 time, without the need for special configuration. However, a constant offset remains between the two system times.
  - Both EtherCAT masters set 0x0920 DC system time offset from their side. The EL6695 accepts this and ensures frequency synchronicity for both ESC is the EL6695.

Local control procedure not supported

Unlike the EL6692, the EL6695 does not support the local control procedure “ControlValue for DC MasterClock”. EtherCAT masters should implement the timestamp procedure referred to above.

Since the EL6695 does not use the time information of the distributed clock internally but only forwards it to the other side, the use of special DC registers in the ESC is not required and not supported. In many EtherCAT devices with DC support, the master writes to DC registers 0x09A0, 0x0990, 0x0980, 0x09A8 during the status transition from PreOp to SafeOp. The EL6695 does not have these registers, which is why a write attempt during startup results in error messages from the EtherCAT master, e.g. “Init Cmd failed: set DC cycle time”.



IP	PS	PI	SP	SO	SI	OS	OP	OI	IB	BI	CMD	Comment
	X										FPWR	set DC cycle time
	X										FPWR	set DC start time
	X										FPWR	set DC activation
	X										FPWR	set DC latch0 cfg

Fig. 175: Distributed Clocks: Error Messages

With the EL6695, the EtherCAT master should therefore avoid writing to these registers, or it does not evaluate the WcState of the write command. In the EL6695-ESI this DC feature is identified through the flag *TimeLoopControlOnly* = *TRUE*, which the EtherCAT master can follow.

6.8 Online Scan

In the EL6695, 128 kbytes of memory are available on each side for any file that can be written and read via FoE.

This can be used for saving the emulation ESI to ensure that a master receives the complete ESI (EtherCAT slave information) and can use it in its configuration

Addendum: Example

6.9 EL6692 CompatibilityMode

The EL6695 features a so-called EL6692 compatible mode.

These modes operate as in the familiar EL6692. It enables the higher-performance EL6695 to be operated with EtherCAT masters and application software designed for the EL6692 interfaces. In these modes primarily the same default process data are offered.

6.9.1 Module operation with PDO mapping and assignment

This module operation is basically the standard mode, also for the new EL 6695. It is described in section "Symmetric PDO mapping".

6.10 EL6695 performance modes

6.10.1 Basic principles of PDO mapping

obsolete

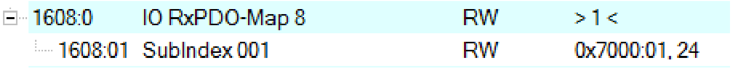
Also see about this

- Symmetric PDO mapping [▶ 121]

6.10.2 Without Init commands

This mode is primarily intended for external masters, which are used solely for process data transfer. The checkboxes for automatic PDO assignment and mapping in the Process Data tab must not be ticked (see section Selective PDO mapping [▶ 127]).

Variables can then be declared on the respective terminal side. If, for example, three byte output variables are declared and the device is restarted (under TwinCAT: Reload Devices), the structure of the PD output object can be seen in the respective CoE object 0x1608. It has a length of 24 bits with three byte variables.



1608:0	IO RxPDO-Map 8	RW	> 1 <
1608:01	SubIndex 001	RW	0x7000:01, 24

Fig. 176: 24 bit CoE Object 0x1608

On the other side of the terminal, assignment and mapping should also be disabled for the data exchange, to ensure that the terminal can interpret the data and does not expect three entries with 8 bits each, but an entry with 24 bits. Moreover, the process data length on the remote side must match exactly. In this example the mapping for CoE object 0x1A01 should be disabled on the remote side (see section Symmetric PDO mapping [▶ 121]), since otherwise a different quantity of input data is assumed and the data exchange would not work.

6.10.3 Object description download

EtherCAT communication devices feature a CoE directory as an overview for objects and parameters for internal function and communication. The EL6695 bridge terminal always has a corresponding object directory (OD) in its initial state, which applies to the EL6695.

Through complete or partial modification of the object directory (OD) it is possible to define a user-specific device on the EL6695 terminal. Via the extension in the TwinCAT System Manager a complete or partial OD can be created as required and saved as a *.coe file, which is an internal Beckhoff file format.

6.10.4 Device emulation

During device emulation, and EtherCAT slave that differs from the bridge terminal is emulated on the bridge side. Physically, the EL6695 bridge terminal remains the EtherCAT slave and always be reset to this original state.

The EtherCAT slave to be emulated is loaded on the selected EL6695 side in the form of the ESI. The EL6695 takes over this data and to the outside then emulates the slave in the form of identity data, PDO and CoE. However, this only applies to the formal representation, without (temporal) behavior, for which the firmware of the EL6695 would have to be modified. The “behavior” of the emulated slave has to be mapped via the other side and the data supplied to/from it.

Since the slave represented in this way also has to pass the EtherCAT ETG conformance test in EmulationMode, the user must ensure that only valid/certified/certifiable ESI are loaded to the EL6695 in EmulationMode. Therefore the ETG rules ETG1000.6 (Mapping) and ETG2000 (ESI Specification) have to be respected.

The EmulationMode can be used on one side or both sides.

Extension (user interface in TwinCAT)

The so-called extension (user interface in TwinCAT) is no longer available in the emulated slave.

For further information please contact Beckhoff support, providing pertinent data (TwinCAT version, required performance values, cycle time, data quantities etc.)

6.11 Applications specific variables definition

As long the terminal EL6695 don’t delivers process data by default configuration, they have to be set by the configurator depending on the specific application requirements. If the used EtherCAT Master does not support variable PDO mapping, the user may create an own ESI device description file to be applied by the specific configurator.

Technical

As the following documents for that, the ETG rules ETG1000.6 (Mapping) and ETG2000 (ESI Specification) have to be respected.

Legal

In addition the device descriptions have to pass the ETG ConformanceTest; the respective user/creator must take care about approval by the ETG of those descriptions to be published.

7 Appendix

7.1 EtherCAT AL Status Codes

For detailed information please refer to the EtherCAT system description.

7.2 Firmware compatibility

Beckhoff EtherCAT devices are delivered with the latest available firmware version. Compatibility of firmware and hardware is mandatory; not every combination ensures compatibility. The overview below shows the hardware versions on which a firmware can be operated.

Note

- It is recommended to use the newest possible firmware for the respective hardware.
- Beckhoff is not under any obligation to provide customers with free firmware updates for delivered products.

NOTE

Risk of damage to the device

Pay attention to the instructions for firmware updates on the separate page [▶ 163].

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!

EL6695

Hardware (HW)	Firmware (FW)	Revision no.	Release date
02 – 08*	03	Primary: EL6695-0000-0001 Secondary: EL6695-0002-0001	2015/02
	04	Primary: EL6695-0000-0002 Secondary: EL6695-0002-0002	2015/06
		Primary: EL6695-0000-0003 Secondary: EL6695-0002-0003	2015/06
	05	Primary: EL6695-0000-0004 Secondary: EL6695-0002-0004	2015/07
	06		2015/08
	07	Primary: EL6695-0000-0005 Secondary: EL6695-0002-0005	2017/10
	08*		2018/04

*) This is the current compatible firmware/hardware version at the time of the preparing this documentation. Check on the Beckhoff web page whether more up-to-date documentation is available.

7.3 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.

NOTE

Only use TwinCAT 3 software!

A firmware update of Beckhoff IO devices must only be performed with a TwinCAT 3 installation. It is recommended to build as up-to-date as possible, available for free download on the Beckhoff website https://www.beckhoff.com/en-us/.

To update the firmware, TwinCAT can be operated in the so-called FreeRun mode, a paid license is not required.

The device to be updated can usually remain in the installation location, but TwinCAT has to be operated in the FreeRun. Please make sure that EtherCAT communication is trouble-free (no LostFrames etc.).

Other EtherCAT master software, such as the EtherCAT Configurator, should not be used, as they may not support the complexities of updating firmware, EEPROM and other device components.

Storage locations

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

- 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.

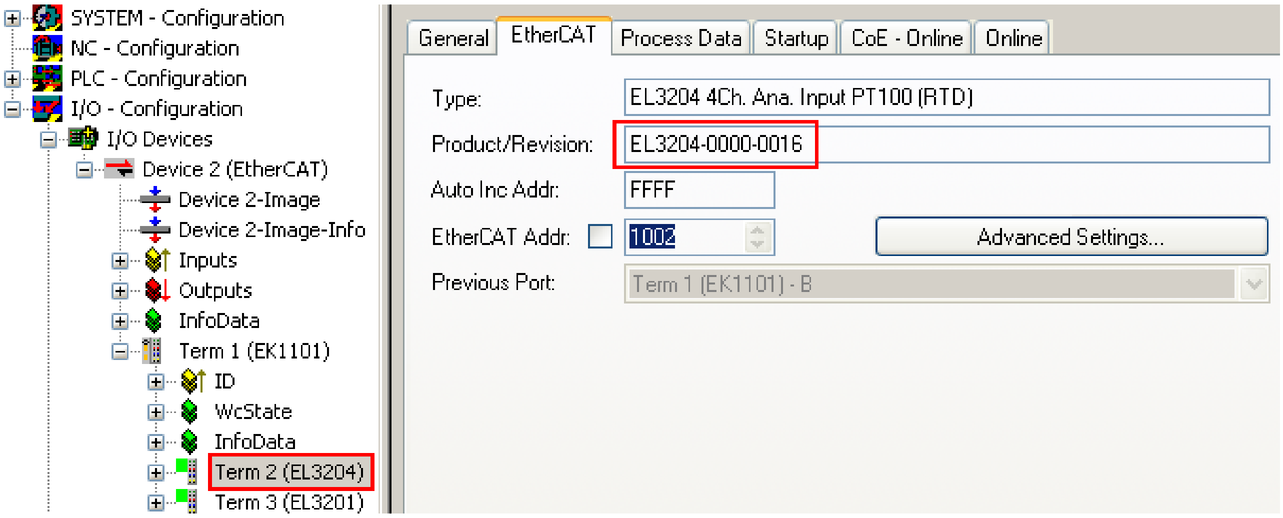
7.3.1 Device description ESI file/XML

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:



Type: EL3204 4Ch. Ana. Input PT100 (RTD)

Product/Revision: EL3204-0000-0016

Auto Inc Addr: FFFF

EtherCAT Addr: □ 1002

Previous Port: Term 1 (EK1101) - B

Fig. 177: 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:

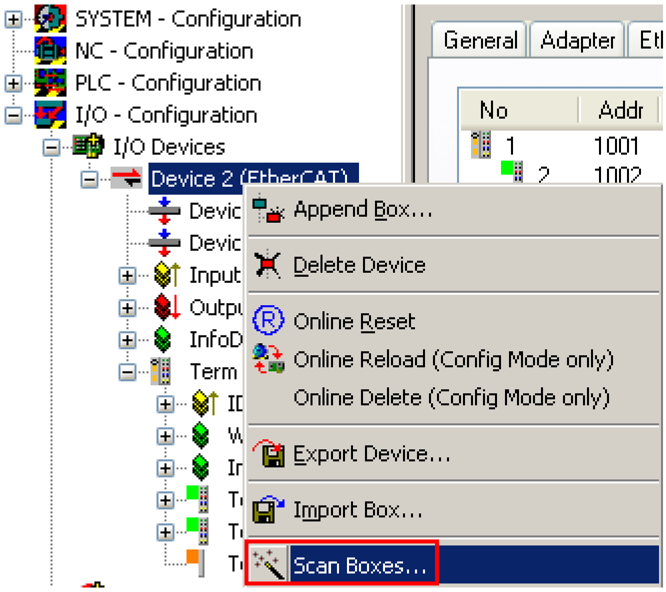
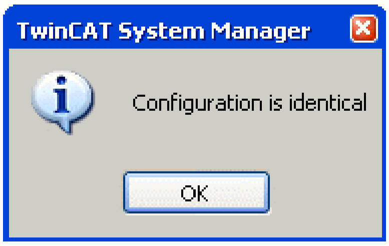


Fig. 178: Scan the subordinate field by right-clicking on the EtherCAT device

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


TwinCAT System Manager

Configuration is identical

Fig. 179: Configuration is identical

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

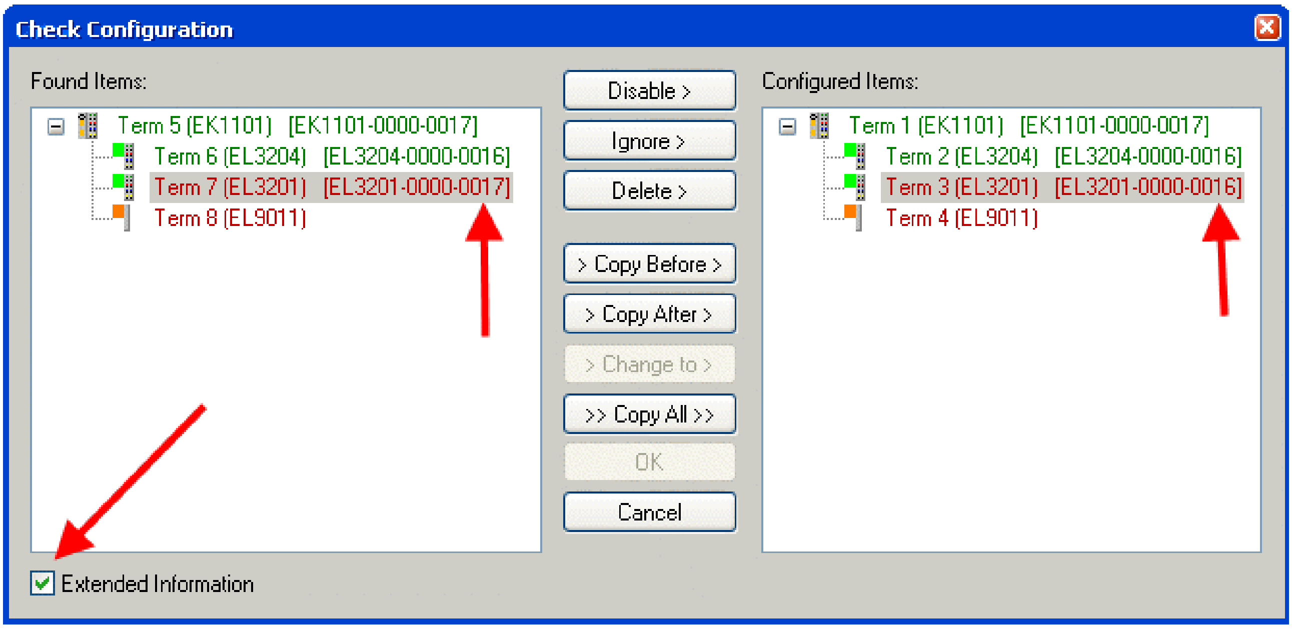

Check Configuration
Found Items:
Term 5 (EK1101) [EK1101-0000-0017]
Term 6 (EL3204) [EL3204-0000-0016]
Term 7 (EL3201) [EL3201-0000-0017]
Term 8 (EL9011)
Disable >
Ignore >
Delete >
> Copy Before >
> Copy After >
> Change to >
>> Copy All >>
OK
Cancel
Configured Items:
Term 1 (EK1101) [EK1101-0000-0017]
Term 2 (EL3204) [EL3204-0000-0016]
Term 3 (EL3201) [EL3201-0000-0016]
Term 4 (EL9011)
☑ Extended Information

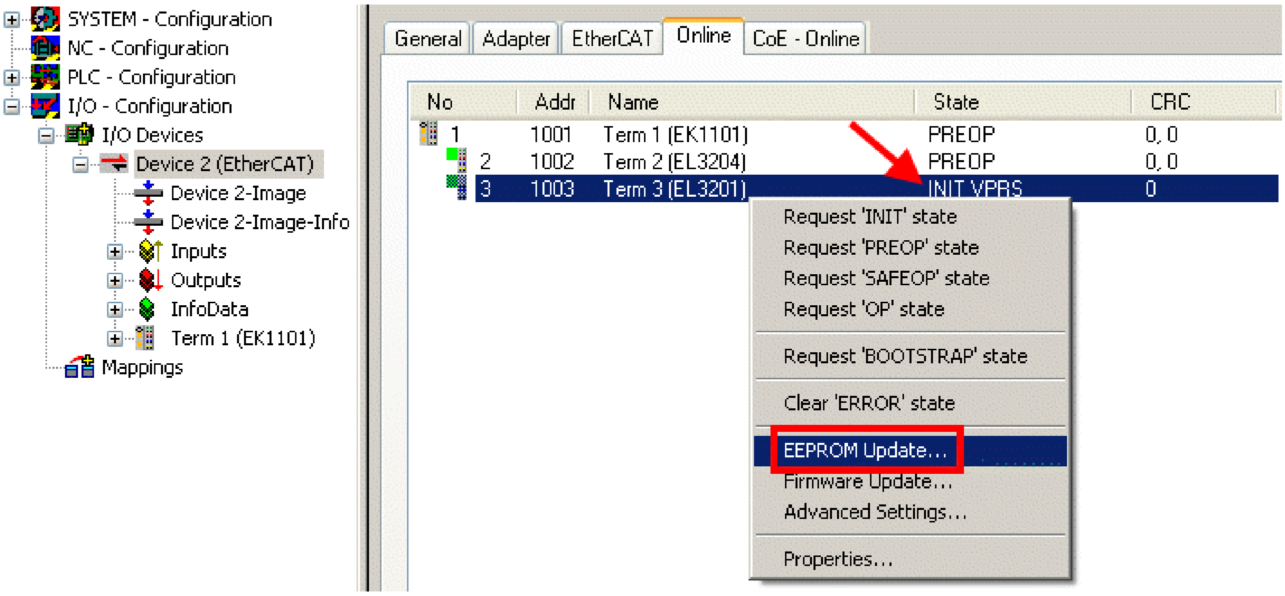
Fig. 180: Change dialog

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:

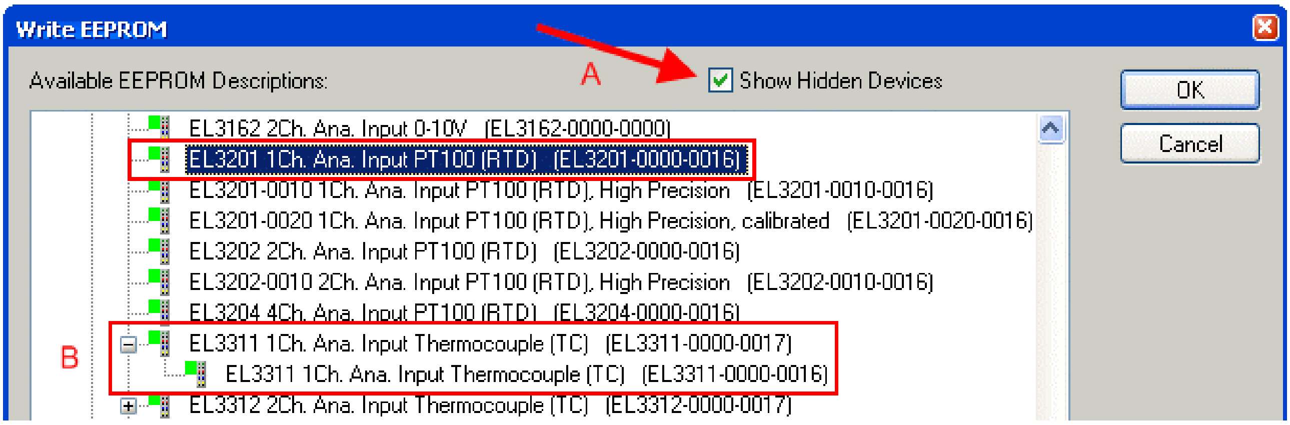
- Trouble-free EtherCAT communication must be established with the slave.
- The state of the slave is irrelevant.
- Right-clicking on the slave in the online display opens the *EEPROM Update* dialog, Fig. *EEPROM Update*



No	Addr	Name	State	CRC
1	1001	Term 1 (EK1101)	PREOP	0, 0
2	1002	Term 2 (EL3204)	PREOP	0, 0
3	1003	Term 3 (EL3201)	INIT VPRS	0

Fig. 181: 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.



Available EEPROM Descriptions: ☑ Show Hidden Devices

EL3162 2Ch. Ana. Input 0-10V (EL3162-0000-0000)
EL3201 1Ch. Ana. Input PT100 (RTD) (EL3201-0000-0016)
EL3201-0010 1Ch. Ana. Input PT100 (RTD), High Precision (EL3201-0010-0016)
EL3201-0020 1Ch. Ana. Input PT100 (RTD), High Precision, calibrated (EL3201-0020-0016)
EL3202 2Ch. Ana. Input PT100 (RTD) (EL3202-0000-0016)
EL3202-0010 2Ch. Ana. Input PT100 (RTD), High Precision (EL3202-0010-0016)
EL3204 4Ch. Ana. Input PT100 (RTD) (EL3204-0000-0016)
EL3311 1Ch. Ana. Input Thermocouple (TC) (EL3311-0000-0017)
EL3311 1Ch. Ana. Input Thermocouple (TC) (EL3311-0000-0016)
EL3312 2Ch. Ana. Input Thermocouple (TC) (EL3312-0000-0017)

Fig. 182: Selecting the new ESI

A progress bar in the System Manager shows the progress. Data are first written, then verified.

The change only takes effect after a restart.

Most EtherCAT devices read a modified ESI description immediately or after startup from the INIT. Some communication settings such as distributed clocks are only read during power-on. The EtherCAT slave therefore has to be switched off briefly in order for the change to take effect.

7.3.2 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

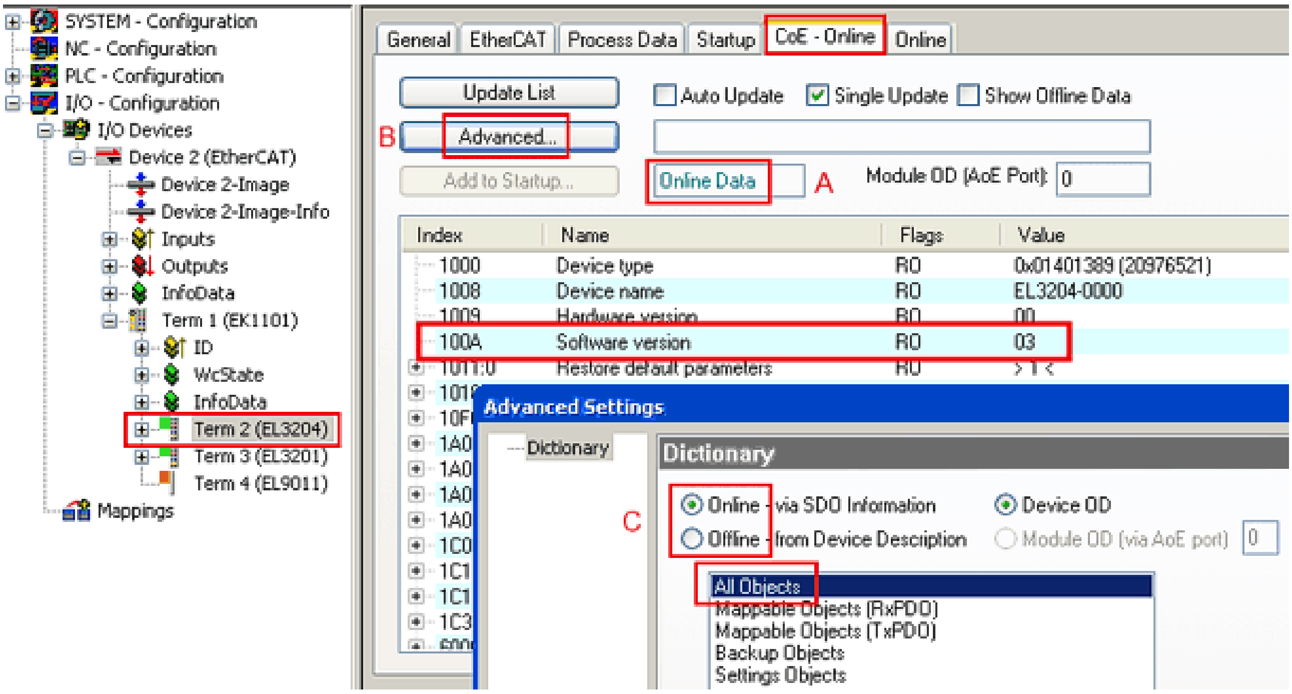
The TwinCAT System Manager shows the version of the controller firmware if the master can access the slave online. Click on the E-Bus Terminal whose controller firmware you want to check (in the example terminal 2 (EL3204)) and select the tab *CoE Online* (CAN over EtherCAT).

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.



☐ Auto Update ☑ Single Update ☐ Show Offline Data

Online Data

Module OD (AoE Port): 0

Index	Name	Flags	Value
1000	Device type	RO	0x01401389 (20976521)
1008	Device name	RO	EL3204-0000
1009	Hardware version	RO	00
100A	Software version	RO	03
1011:0	Restore default parameters	RO	> 1 <

Dictionary

● Online - via SDO Information
○ Offline - from Device Description
● Device OD
○ Module OD (via AoE port) 0

Fig. 183: Display of EL3204 firmware version

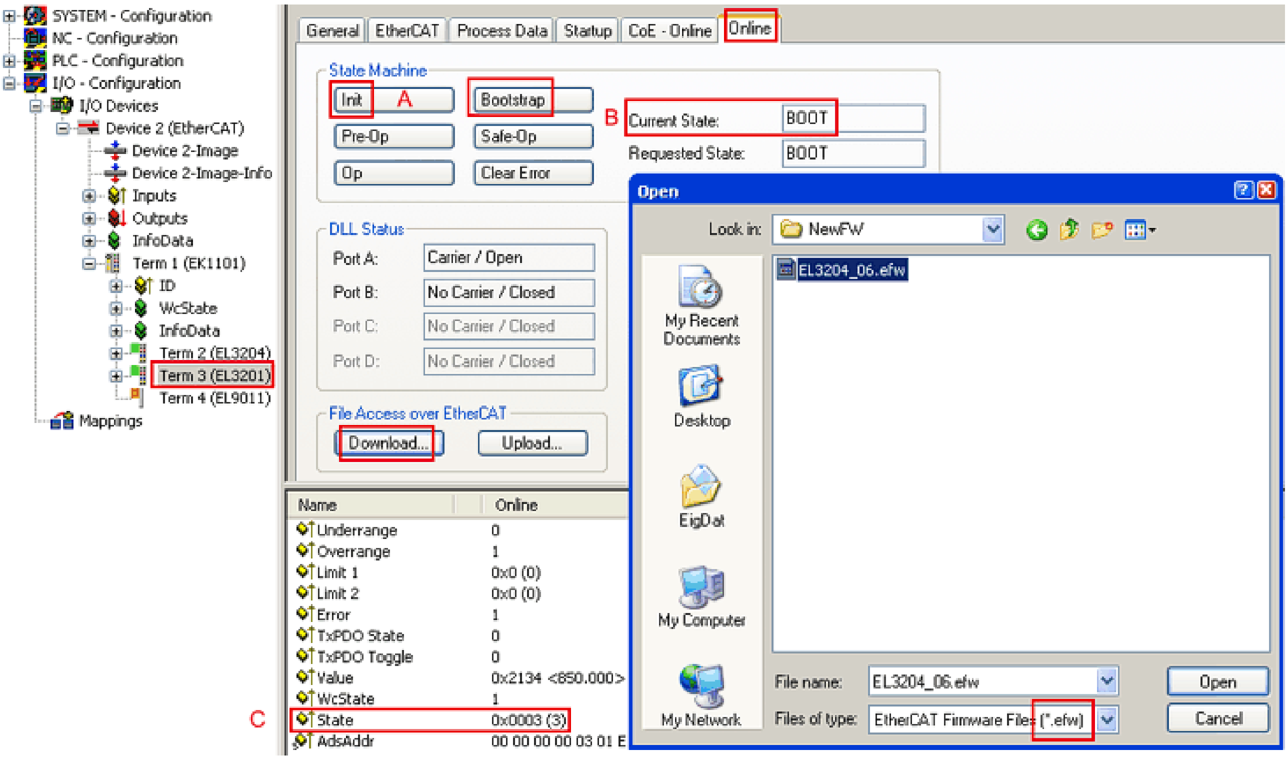
In (A) TwinCAT 2.11 shows that the Online CoE directory is currently displayed. If this is not the case, the Online directory can be loaded via the *Online* option in Advanced Settings (B) and double-clicking on *AllObjects*.

7.3.3 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.*



Current State: BOOT

Requested State: BOOT

Port A: Carrier / Open

Port B: No Carrier / Closed

Port C: No Carrier / Closed

Port D: No Carrier / Closed

Name	Online
Underrange	0
Overrange	1
Limit 1	0x0 (0)
Limit 2	0x0 (0)
Error	1
TxPDO State	0
TxPDO Toggle	0
Value	0x2134 <850.000>
WcState	1
State	0x0003 (3)
AdsAddr	00 00 00 00 03 01 E

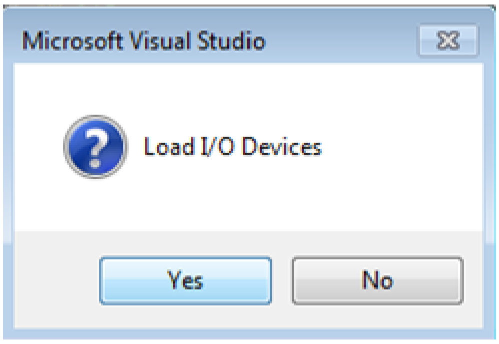
File name: EL3204_06.efw

Files of type: EtherCAT Firmware Files (*.efw)

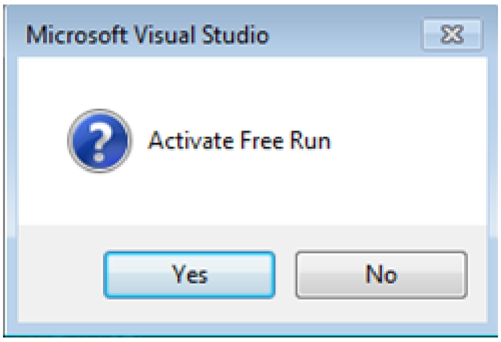
Fig. 184: Firmware Update

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.

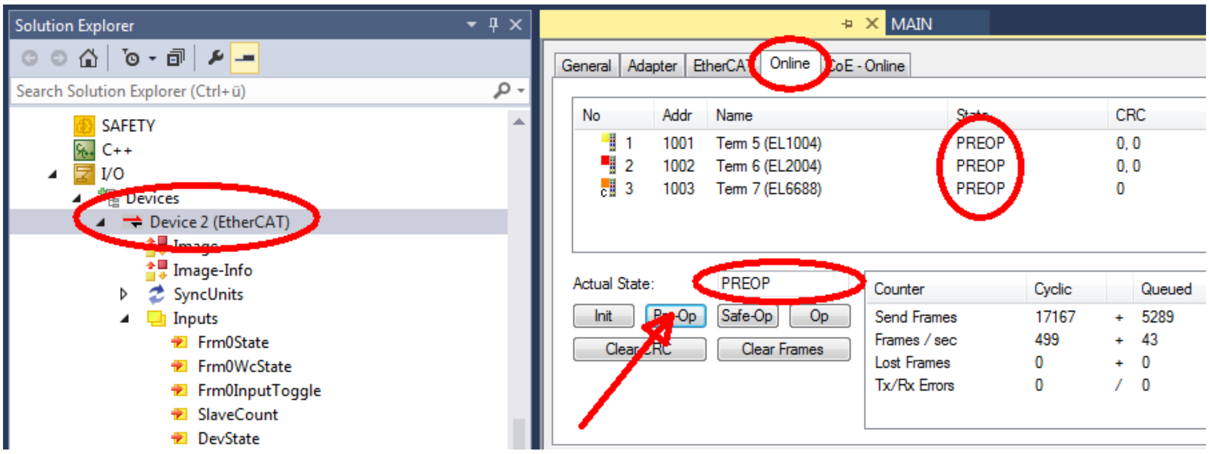

Microsoft Visual Studio

Load I/O Devices


Microsoft Visual Studio

Activate Free Run

- Switch EtherCAT Master to PreOP

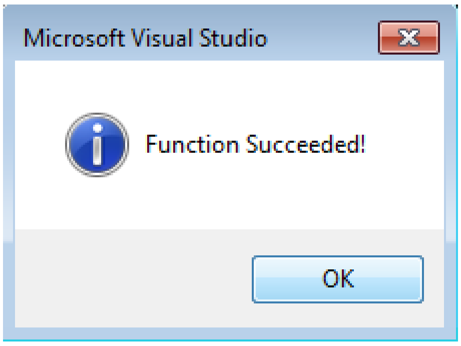


No	Addr	Name	State	CRC
1	1001	Term 5 (EL1004)	PREOP	0, 0
2	1002	Term 6 (EL2004)	PREOP	0, 0
3	1003	Term 7 (EL6688)	PREOP	0

Actual State: PREOP

Counter	Cyclic		Queued
Send Frames	17167	+	5289
Frames / sec	499	+	43
Lost Frames	0	+	0
Tx/Rx Errors	0	/	0

- 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.


Microsoft Visual Studio

Function Succeeded!

- After the download switch to INIT, then PreOP
- Switch off the slave briefly (don't pull under voltage!)
- Check within CoE 0x100A, if the FW status was correctly overtaken.

7.3.4 FPGA firmware *.rbf

If an FPGA chip deals with the EtherCAT communication an update may be accomplished via an *.rbf file.

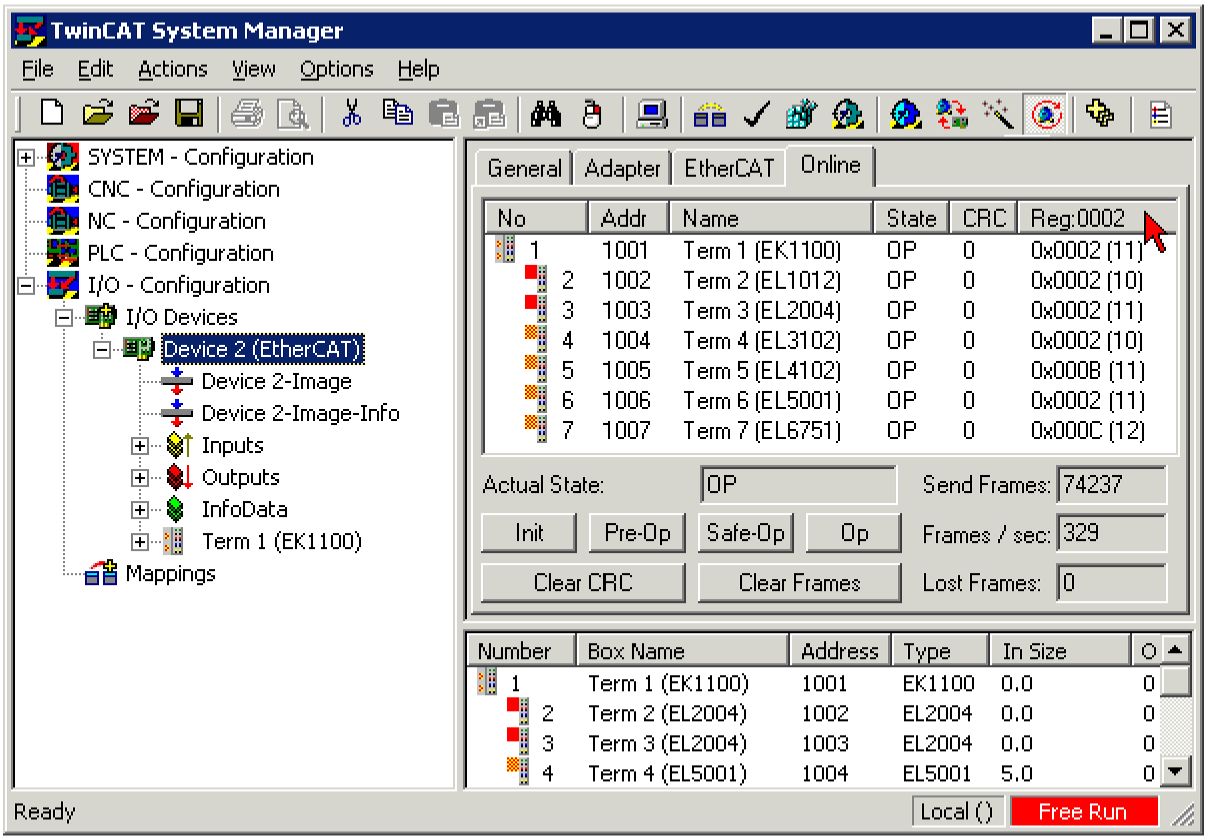
- Controller firmware for processing I/O signals
- FPGA firmware for EtherCAT communication (only for terminals with FPGA)

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.



No	Addr	Name	State	CRC	Reg:0002
1	1001	Term 1 (EK1100)	OP	0	0x0002 (11)
2	1002	Term 2 (EL1012)	OP	0	0x0002 (10)
3	1003	Term 3 (EL2004)	OP	0	0x0002 (11)
4	1004	Term 4 (EL3102)	OP	0	0x0002 (10)
5	1005	Term 5 (EL4102)	OP	0	0x000B (11)
6	1006	Term 6 (EL5001)	OP	0	0x0002 (11)
7	1007	Term 7 (EL6751)	OP	0	0x000C (12)

Actual State: OP

Send Frames: 74237

Frames / sec: 329

Lost Frames: 0

Number	Box Name	Address	Type	In Size	O
1	Term 1 (EK1100)	1001	EK1100	0.0	0
2	Term 2 (EL2004)	1002	EL2004	0.0	0
3	Term 3 (EL2004)	1003	EL2004	0.0	0
4	Term 4 (EL5001)	1004	EL5001	5.0	0

Fig. 185: FPGA firmware version definition

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

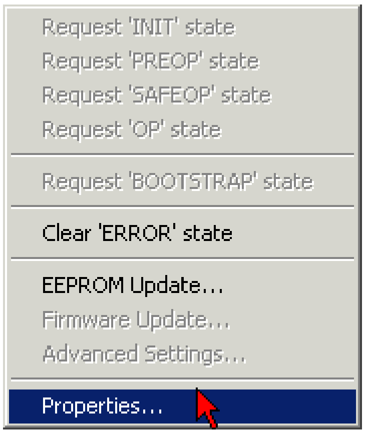
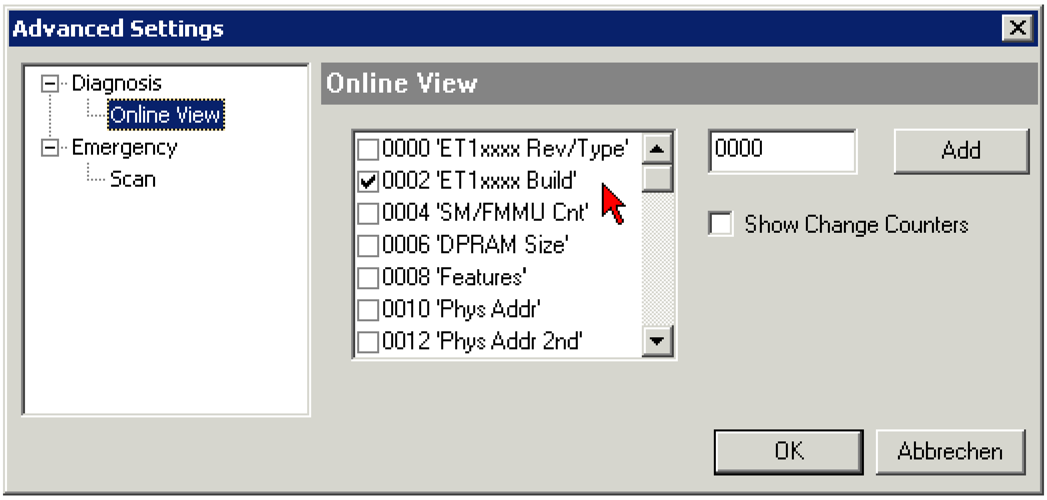


Fig. 186: Context menu *Properties*

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.



Advanced Settings

- Diagnosis
  - Online View
- Emergency
  - Scan

Online View

□ 0000 'ET1xxxx Rev/Type'
☑ 0002 'ET1xxxx Build'
□ 0004 'SM/FMMU Cnt'
□ 0006 'DPRAM Size'
□ 0008 'Features'
□ 0010 'Phys Addr'
□ 0012 'Phys Addr 2nd'

0000

□ Show Change Counters

Fig. 187: 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.

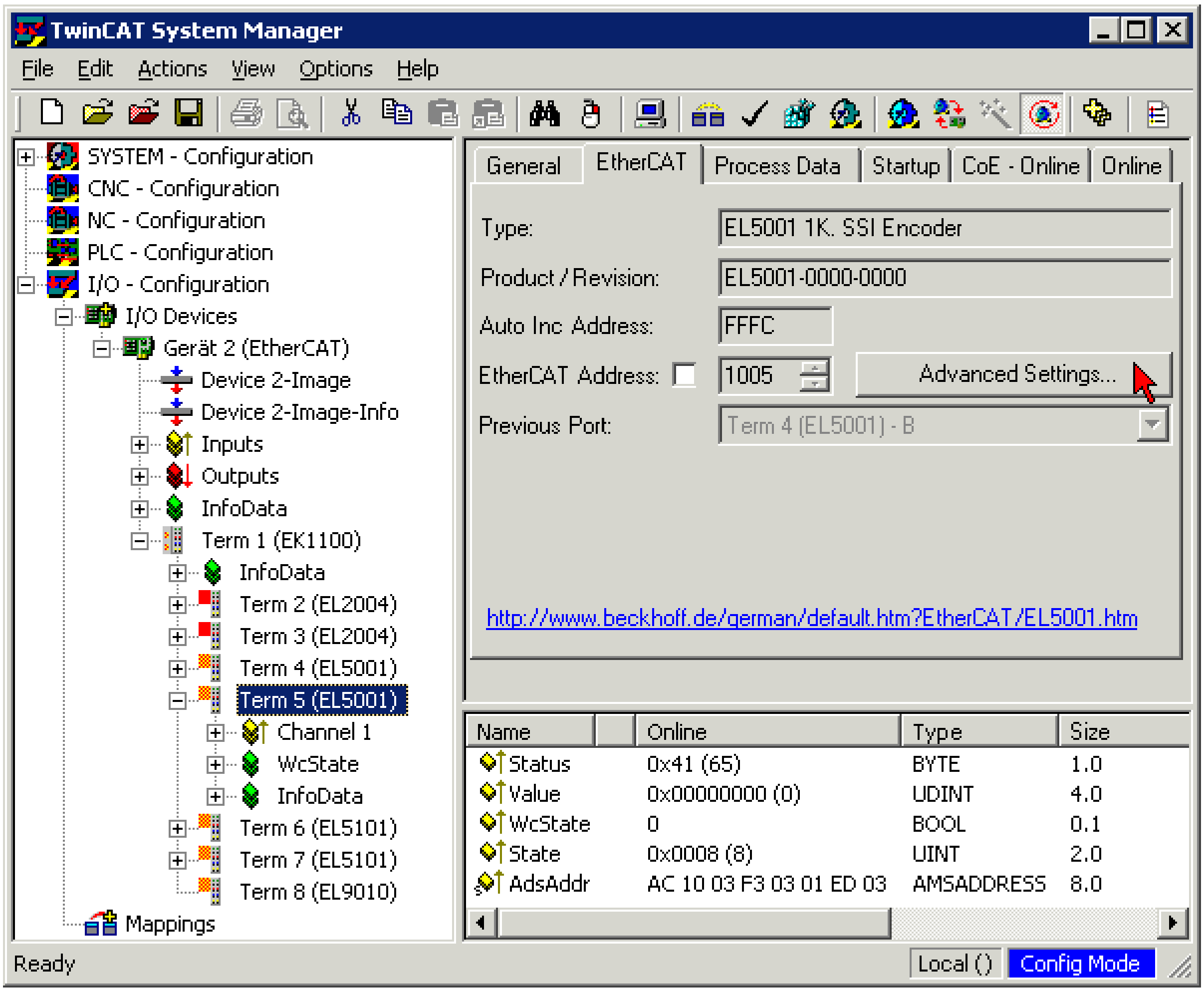
Older firmware versions can only be updated by the manufacturer!

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:



Type: EL5001 1K. SSI Encoder

Product / Revision: EL5001-0000-0000

Auto Inc Address: FFFC

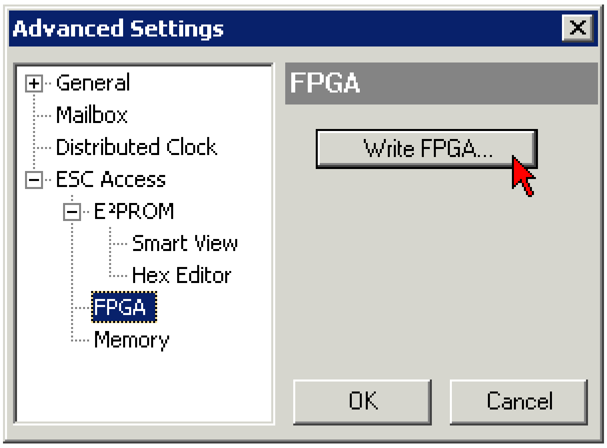
EtherCAT Address: □ 1005

Previous Port: Term 4 (EL5001) - B

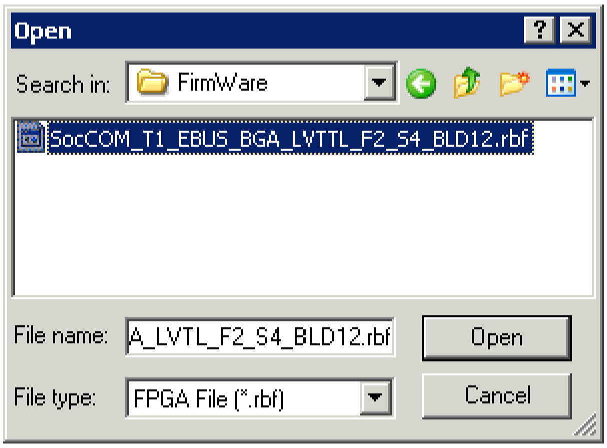
http://www.beckhoff.de/german/default.htm?EtherCAT/EL5001.htm

Name	Online	Type	Size
Status	0x41 (65)	BYTE	1.0
Value	0x00000000 (0)	UDINT	4.0
WcState	0	BOOL	0.1
State	0x0008 (8)	UINT	2.0
AdsAddr	AC 10 03 F3 03 01 ED 03	AMSADDRESS	8.0

- 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:



Search in: FirmWare

SocCOM_T1_EBUS_BGA_LVTTL_F2_S4_BLD12.rbf

File name: A_LVTL_F2_S4_BLD12.rbf

File type: FPGA File (*.rbf)

- 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

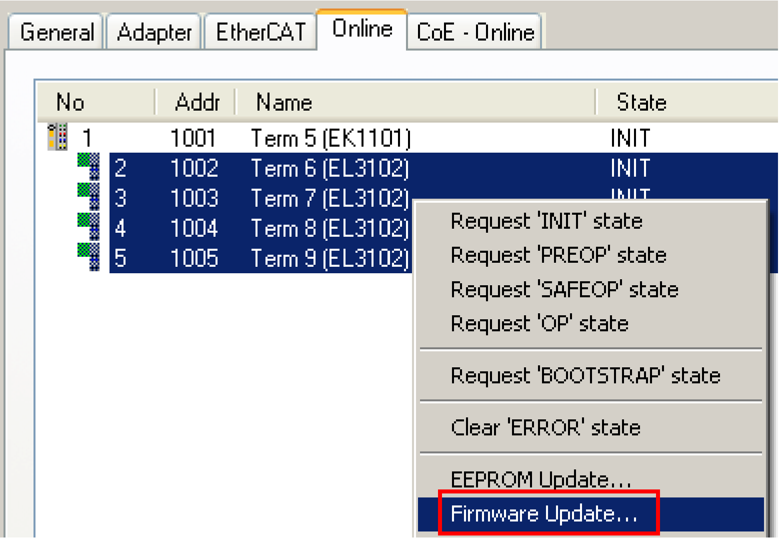
NOTE

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!

7.3.5 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.



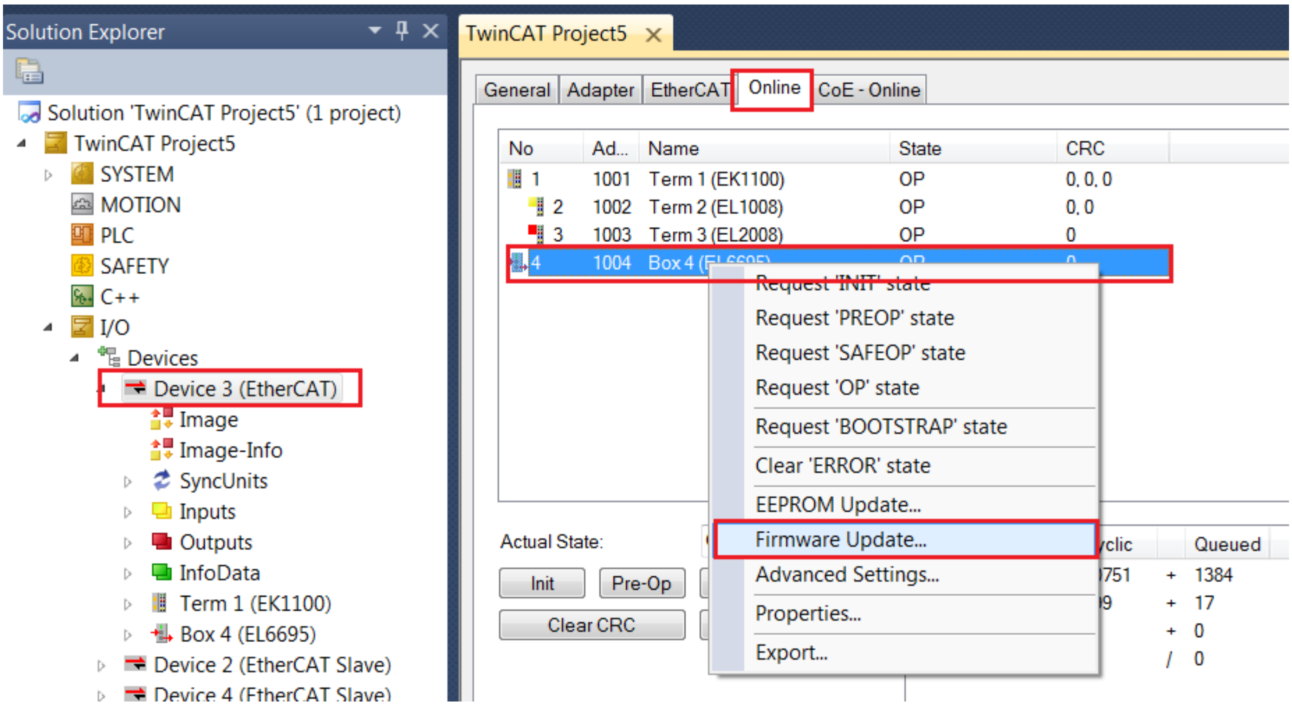
No	Addr	Name	State
1	1001	Term 5 (EK1101)	INIT
2	1002	Term 6 (EL3102)	INIT
3	1003	Term 7 (EL3102)	INIT
4	1004	Term 8 (EL3102)	
5	1005	Term 9 (EL3102)	

Fig. 188: Multiple selection and firmware update

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

7.4 Addition to firmware update

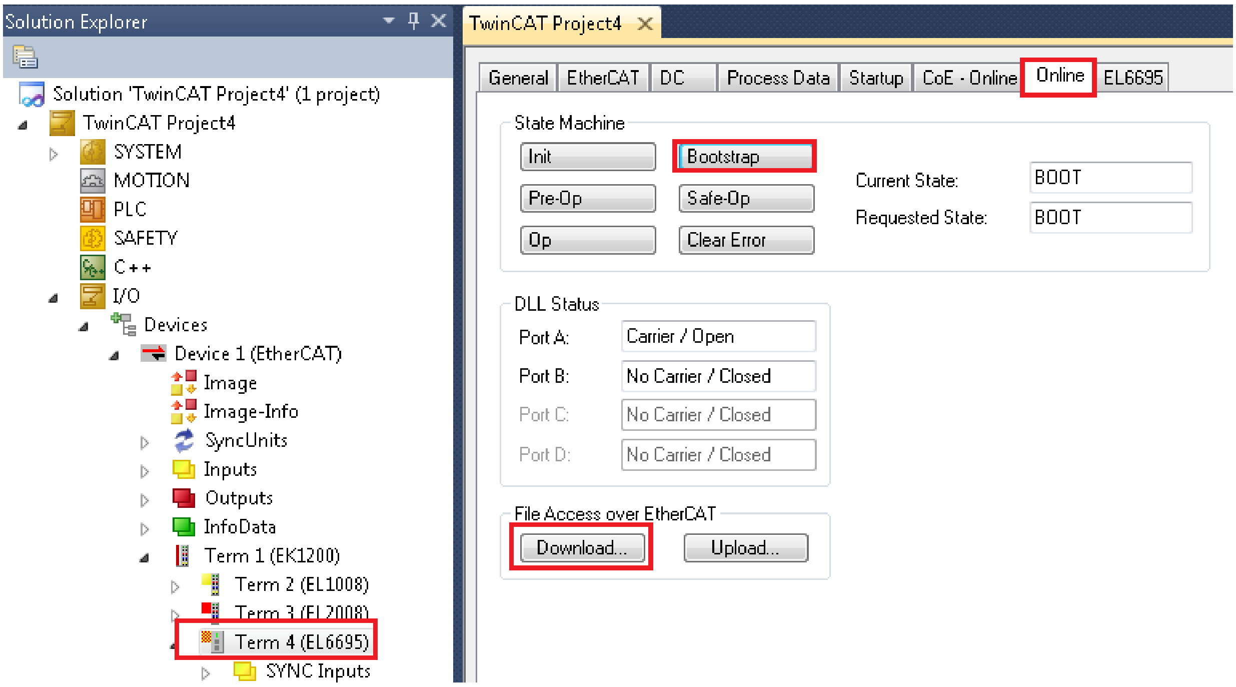
A firmware update can take place in two ways: one option is by selecting the device (EL6695) from the list via the Online tab, then select “Firmware Update..” in the drop-down menu. A *.efw file can then be loaded via the corresponding path; a password is not required, but can optionally be issued. Both terminal sides should then be set to INIT state, before they are set to OP state again.



No	Ad...	Name	State	CRC
1	1001	Term 1 (EK1100)	OP	0, 0, 0
2	1002	Term 2 (EL1008)	OP	0, 0
3	1003	Term 3 (EL2008)	OP	0
4	1004	Box 4 (EL6695)	OP	0

Fig. 189: Firmware update via the Online tab

The second option is directly via the terminal and the Online tab. To this end set the terminal to state BOOT on the selected side.



Current State: BOOT

Requested State: BOOT

DLL Status

Port A: Carrier / Open

Port B: No Carrier / Closed

Port C: No Carrier / Closed

Port D: No Carrier / Closed

Fig. 190: Setting the BOOT state

The *.efw file can then be downloaded to the terminal via the Download button under “File Access over EtherCAT”. Then set the terminal to INIT and back to OP. The other side also has to be set to INIT and back to OP.

NOTE

Note for FW update (beta FW)

- **Before** the update ensure that the EtherCAT bus operates without lost frames or lost link.
- If several EL6695 terminals are present in the system, an FW update may only be carried out for one EL6695 terminal at a time. Otherwise the device would be irreversibly damaged, and the terminal would have to be replaced by the Beckhoff service.
- A power supply failure must be avoided during the update.

7.5 Diagnostics

Diagnostic LEDs

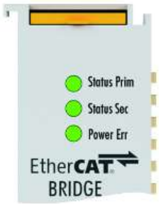


Fig. 191: The operating state is indicated via diagnostic LEDs

LED	Meaning	
Run Prim Run Sec (operating state of the respective slave side)	off	State of the EtherCAT state machine: **INIT** = initialization of the terminal or BOOTSTRAP = function for firmware updates of the terminal
Run Prim Run Sec (operating state of the respective slave side) Power/Error (supply state)	green flashing	State of the EtherCAT state machine: **PREOP** = function for mailbox communication and different standard-settings set
	green single flash	State of the EtherCAT state machine: **SAFEOP** = verification of the sync manager channels and the distributed clocks. Outputs remain in safe state
	green on	State of the EtherCAT state machine: **OP** = normal operating state; mailbox and process data communication is possible
	off	No supply
Power/Error (supply state)	orange	Supply only from one side, primary or secondary
	flashing red	FW update failed, send terminal to service
	green	Supply present on both sides, 24 V input is used

7.6 Restoring the delivery state

The terminal EL6695 supports two reset methods:

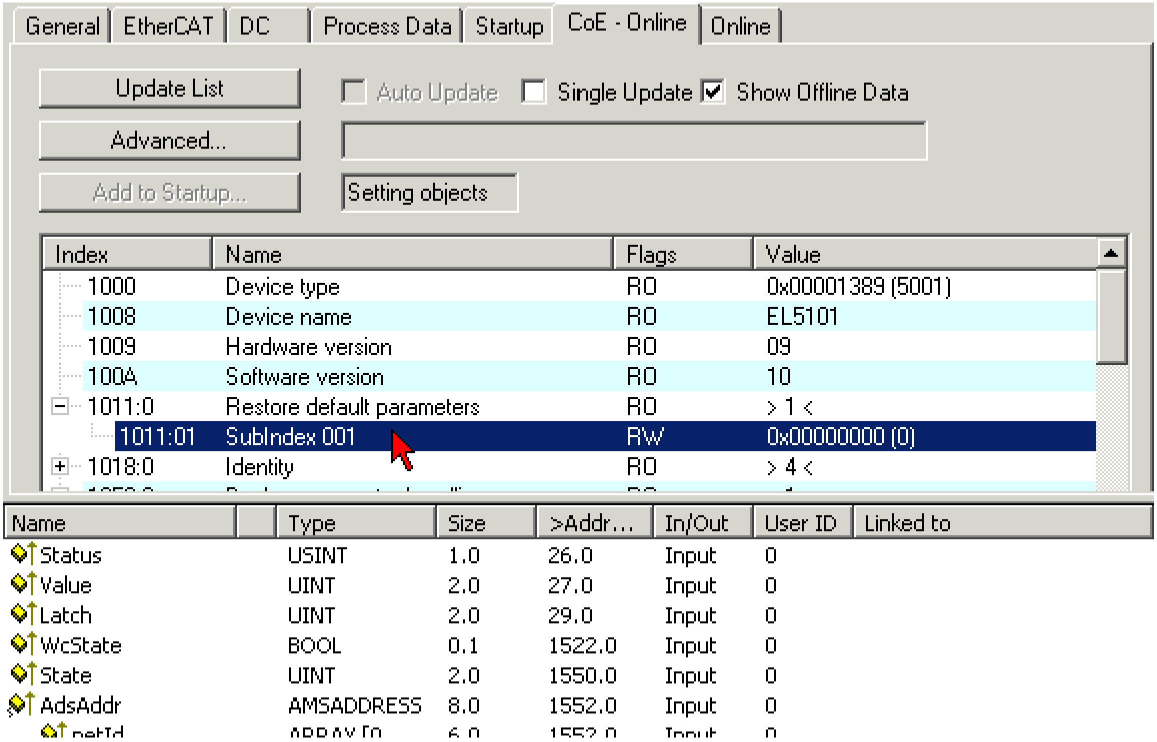
- By command
- Via the CoE-Object 0x1011 (up to FW07)

7.6.1 Common device reset

The settings stored inside the terminal EL6695 can be reset by putting the code 0x33336695 into the CoE Object 0xF008.

7.6.2 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*)



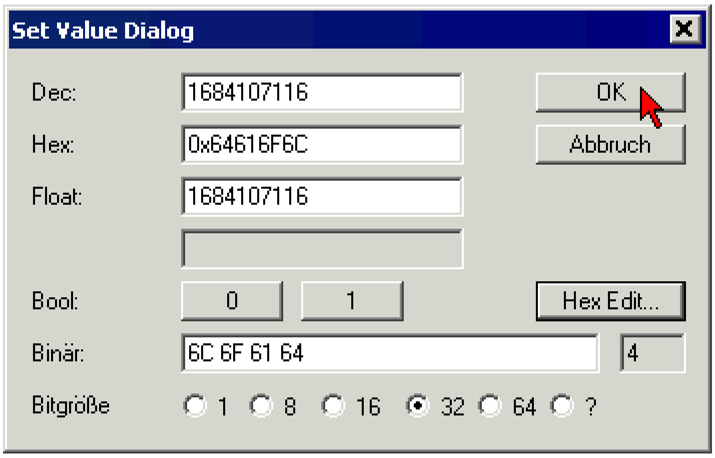
□ Auto Update □ Single Update ☑ Show Offline Data

Index	Name	Flags	Value
1000	Device type	RO	0x00001389 (5001)
1008	Device name	RO	EL5101
1009	Hardware version	RO	09
100A	Software version	RO	10
1011:0	Restore default parameters	RO	> 1 <
1011:01	SubIndex 001	RW	0x00000000 (0)
1018:0	Identity	RO	> 4 <

Name	Type	Size	>Addr...	In/Out	User ID	Linked to
Status	USINT	1.0	26.0	Input	0	
Value	UINT	2.0	27.0	Input	0	
Latch	UINT	2.0	29.0	Input	0	
WcState	BOOL	0.1	1522.0	Input	0	
State	UINT	2.0	1550.0	Input	0	
AdsAddr	AMSADDRESS	8.0	1552.0	Input	0	

Fig. 192: 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.



Dec: 1684107116

Hex: 0x64616F6C

Float: 1684107116

Bool: 0 1

Binär: 6C 6F 61 64 4

Bitgröße ○ 1 ○ 8 ○ 16 ● 32 ○ 64 ○ ?

Fig. 193: 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.

7.7 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: https://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963 157
Fax:	+49 5246 963 9157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963 460
Fax:	+49 5246 963 479
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963 0
Fax:	+49 5246 963 198
e-mail:	info@beckhoff.com
web:	https://www.beckhoff.com

List of illustrations

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com