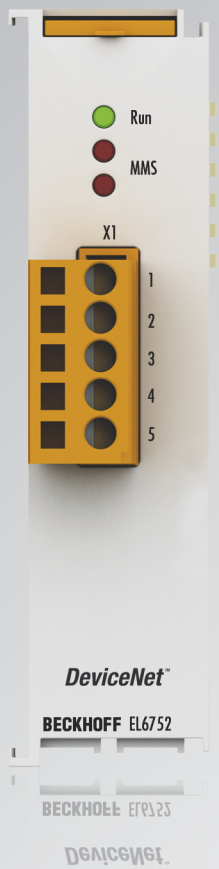
BECKHOFF New Automation Technology

Documentation | EN

EL6752

Master/Slave Terminal for DeviceNet®



2026-01-15 | Version: 2.5.0

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

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

Disclaimer

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

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

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

Trademarks

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

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

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

Copyright

Third-party brands

Trademarks of third parties may be used in this documentation. You can find the trademark notices here:
https://www.beckhoff.com/trademarks

1.2 Guide through documentation

NOTICE

Further components of documentation

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

Title	Description
EtherCAT System Documentation (PDF)	• System overview • EtherCAT basics • Cable redundancy • Hot Connect • EtherCAT devices configuration
Explosion Protection for Terminal Systems (PDF)	Notes on the use of the Beckhoff terminal systems in hazardous areas according to ATEX and IECEx
Infrastructure for EtherCAT/Ethernet (PDF)	Technical recommendations and notes for design, implementation and testing
Software Declarations I/O (PDF)	Open source software declarations for Beckhoff I/O components

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

- the “Documentation and Download” area of the respective product page,
- the Download finder,
- the Beckhoff Information System.

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

1.3 Safety instructions

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

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

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product

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

1.4 Documentation issue status

Version	Comment
2.5.0	• Update chapter "Technical data" • Update chapter "Object description and parameterization" • Update revision status • Structural update
2.4	• Chapter "Recommended mounting rail" removed • Update chapter "Technical data" • Chapter “IECEx - Special conditions” added • Update revision status
2.3	• Update chapter "Technical data" • Update chapter "Version identification of EtherCAT devices" • Structural update • Update notes • Update revision status • Chapter Disposal added
2.2	• Structural update
2.1	• Chapter "Explicit messages" amended • Update chapter “Technical data” • Structural update • Update revision status
2.0	• Migration • Structural update
1.4	• Addendum: Chapter “Configuration”: Changing DeviceNet® address and baud rate via ADS • Structural update
1.3	• Corrections to chapter "Technical data" • Addendum: chapter "Firmware status" • Structural update
1.2	• Corrections to chapter "Mounting and wiring"
1.1	• Corrections to chapter "Mounting and wiring"
1.0	• Corrections and addenda, first release
0.2	• Corrections and addenda
0.1	• Preliminary version for internal use

1.5 Version identification of EtherCAT devices

1.5.1 General notes on marking

Designation

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

- family key
- type
- version
- revision

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

Notes

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

1.5.2 Version identification of EL terminals

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

Structure of the serial number: **KK YY FF HH**	Example with serial number 12 06 3A 02:
KK - week of production (CW, calendar week)	12 - production week 12
YY - year of production	06 - production year 2006
FF - firmware version	3A - firmware version 3A
HH - hardware version	02 - hardware version 02

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

1.5.3 Beckhoff Identification Code (BIC)

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

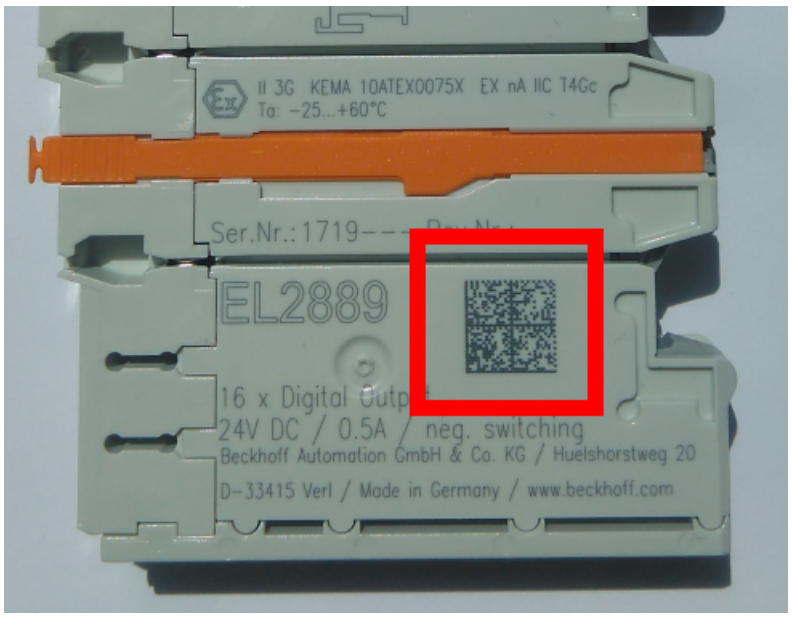


Fig. 2: BIC as data matrix code (DMC, code scheme ECC200)

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

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

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

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

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

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

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

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

Structure of the BIC

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

1P072222**SBTN**k4p562d7**1K**EL1809 **Q**1 **51S**678294

Accordingly as DMC:

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

BTN

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

NOTICE

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

1.5.4 Electronic access to the BIC (eBIC)

Electronic BIC (eBIC)

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

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

K-bus devices (IP20, IP67)

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

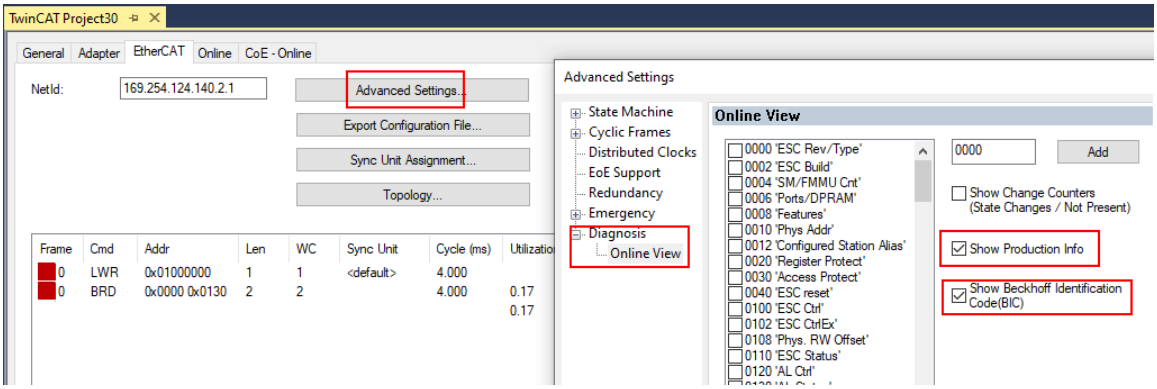
EtherCAT devices (IP20, IP67)

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

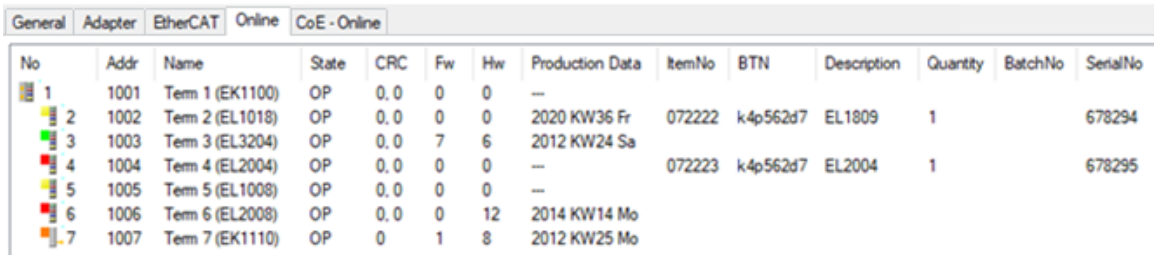
Beckhoff also stores the eBIC in the ESI-EEPROM. The eBIC was introduced into Beckhoff IO production (terminals, box modules) in 2020; as of 2023, implementation is largely complete.

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

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

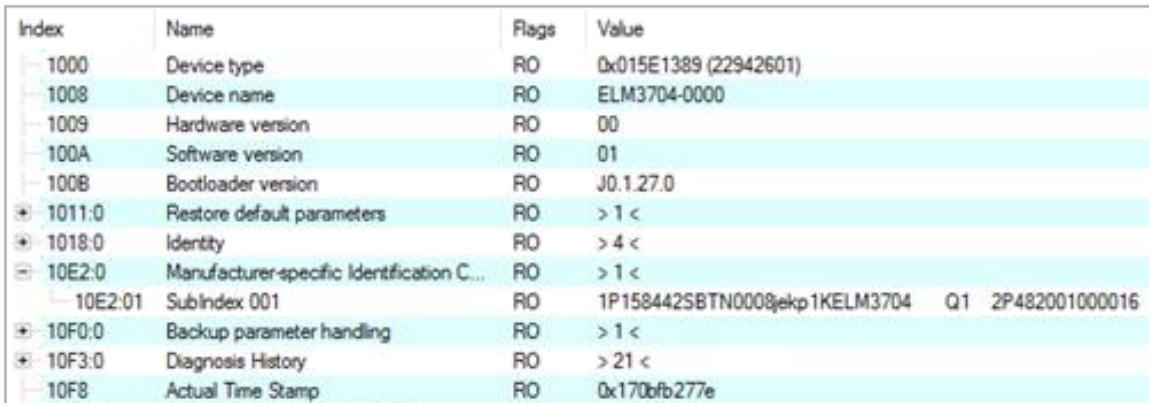


  - The BTN and its contents are then displayed:



  - Note: As shown in the figure, the production data HW version, FW version, and production date, which have been programmed since 2012, can also be displayed with "Show production info".
  - Access from the PLC: From TwinCAT 3.1. build 4024.24, the functions *FB_EcReadBIC* and *FB_EcReadBTN* for reading into the PLC are available in the Tc2_EtherCAT library from v3.3.19.0.
- EtherCAT devices with a CoE directory may also have the object 0x10E2:01 to display their own eBIC, which can also be easily accessed by the PLC:

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



Index	Name	Flags	Value
1000	Device type	RO	0x015E1389 (22942601)
1008	Device name	RO	ELM3704-0000
1009	Hardware version	RO	00
100A	Software version	RO	01
100B	Bootloader version	RO	J0.1.27.0
1011:0	Restore default parameters	RO	> 1 <
1018:0	Identity	RO	> 4 <
10E2:0	Manufacturer-specific Identification C...	RO	> 1 <
10E2:01	SubIndex 001	RO	1P158442SBTN0008jekp1KELM3704 Q1 2P482001000016
10F0:0	Backup parameter handling	RO	> 1 <
10F3:0	Diagnosis History	RO	> 21 <
10F8	Actual Time Stamp	RO	0x170bfb277e

  - The object 0x10E2 will be preferentially introduced into stock products in the course of necessary firmware revision.
  - From TwinCAT 3.1. build 4024.24, the functions *FB_EcCoEReadBIC* and *FB_EcCoEReadBTN* for reading into the PLC are available in the Tc2_EtherCAT library from v3.3.19.0
- The following auxiliary functions are available for processing the BIC/BTN data in the PLC in *Tc2_Utilities* as of TwinCAT 3.1 build 4024.24
  - F_SplitBIC: The function splits the Beckhoff Identification Code (BIC) sBICValue into its components using known identifiers and returns the recognized substrings in the ST_SplittedBIC structure as a return value
  - BIC_TO_BTN: The function extracts the BTN from the BIC and returns it as a return value
- Note: If there is further electronic processing, the BTN is to be handled as a string(8); the identifier "SBTN" is not part of the BTN.
- Technical background
  The new BIC information is written as an additional category in the ESI-EEPROM during device production. The structure of the ESI content is largely dictated by the ETG specifications, therefore the additional vendor-specific content is stored using a category in accordance with the ETG.2010. ID 03 tells all EtherCAT masters that they may not overwrite these data in the event of an update or restore the data after an ESI update.
  The structure follows the content of the BIC, see here. The EEPROM therefore requires approx. 50..200 bytes of memory.
- Special cases
  - If multiple hierarchically arranged ESCs are installed in a device, only the top-level ESC carries the eBIC information.
  - If multiple non-hierarchically arranged ESCs are installed in a device, all ESCs carry the eBIC information.
  - If the device consists of several sub-devices which each have their own identity, but only the top-level device is accessible via EtherCAT, the eBIC of the top-level device is located in the CoE object directory 0x10E2:01 and the eBICs of the sub-devices follow in 0x10E2:nn.

PROFIBUS, PROFINET, and DeviceNet® devices

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

2 Product overview

2.1 Introduction

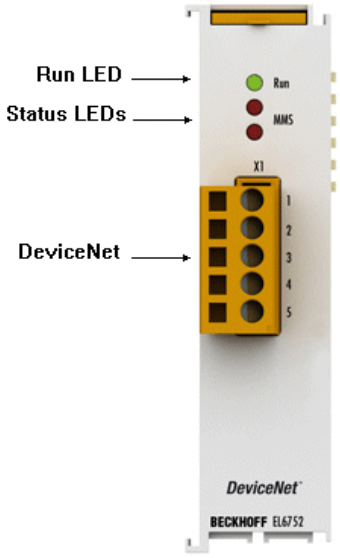


Fig. 4: EL6752

Master and slave terminals for DeviceNet®

The master and slave terminals for DeviceNet® correspond to the FC5201 PCI card from Beckhoff. Thanks to the connection via EtherCAT, no PCI slots are required in the PC. Within an EtherCAT terminal network, the terminal enables the integration of any DeviceNet® devices.
The EL6752 is optionally available in a master or slave version and has a powerful protocol implementation with many features:

- All I/O modes of the DeviceNet® are supported: polling, change of state, cyclic, strobed
- Unconnected message manager (UCMM)
- Powerful parameter and diagnostics interfaces
- Error management freely configurable for each bus device

A description of all functionalities and operating modes can be found in the chapter "Configuration [▶ 73]" and the corresponding subsections.

2.2 Technical data

Technical data	EL6752-0000	EL6752-0010
Bus / system	DeviceNet®	
Version	Master	Slave
Number of fieldbus channels	1	
Data transfer rate	125, 250 or 500 kbaud	
Bus interface	Open style 5-pin connector according to DeviceNet® specification, galvanically isolated; card comes with connector	
Bus devices	maximum 63 slaves	
Communication	DeviceNet® network master (scanner)	DeviceNet® - slave
Diagnostics	Status LEDs	
Power supply	via the E-bus	
Current consumption via E-bus	typ. 260 mA	
Electrical isolation	500 V (E-bus/CANopen)	
Configuration	with TwinCAT System Manager	
Weight	approx. 70 g	
Permissible ambient temperature range during operation	-25 °C ... +60 °C	
Permissible ambient temperature range during storage	-40 °C ... +85 °C	
Permissible relative air humidity	95%, no condensation	
Dimensions (W x H x D)	approx. 26 mm x 100 mm x 52 mm	
Installation [▶ 23]	on 35 mm mounting rail, conforms to EN 60715	
Vibration / shock resistance	conforms to EN 60068-2-6 / EN 60068-2-27	
EMC immunity / emission	conforms to EN 61000-6-2 / EN 61000-6-4	
Protection rating	IP20	
Installation position	variable	
Approvals / markings*	CE, UKCA, EAC, CCC cULus [▶ 22], ATEX [▶ 19], IECEx [▶ 20]	

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

Ex markings

Standard	Marking
ATEX	II 3 G Ex ec IIC T4 Gc
IECEx	Ex ec IIC T4 Gc

3 Basic DeviceNet® principles

Introduction to the system

DeviceNet® is an open system based on CAN. CAN was developed some years ago by R. Bosch for data transmission in motor vehicles. Millions of CAN chips are now in use. A disadvantage for application in automation is that CAN does not contain definitions for the application layer. CAN only defines the physical and data link layer.

DeviceNet® specifies a uniform application layer and this makes it possible to use the CAN protocol for industrial applications. ODVA (the Open DeviceNet® Vendor Association) is an independent association which supports manufacturers and users of the DeviceNet® system. ODVA ensures that all devices which conform to the specification can operate together in one system, regardless of their manufacturer. CAN's bit arbitration procedure makes it theoretically possible to operate communication networks using master/slave and multimaster access methods.
Further details can be found on the official website of the ODVA (http://www.odva.org).

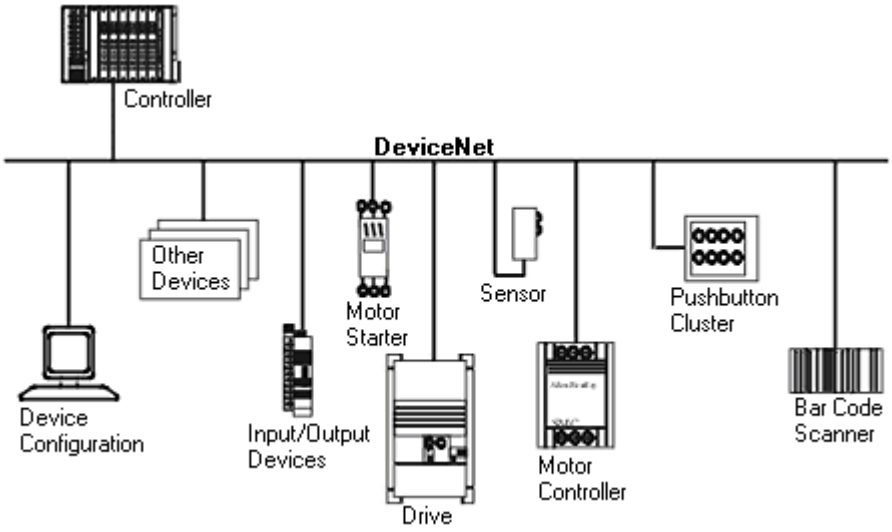


Fig. 5: Example of DeviceNet® in use

Bus cable

The bus cable consists of two pairs of shielded twisted-pair wiring, one for the data transfer and one for the power supply. The latter can carry currents of up to 8 amperes. The maximum possible length of a line depends essentially on the baud rate. If you choose the highest Baud rate (500 kbaud) you are restricted to lines of at most 100 m. With the lowest Baud rate (125 kbaud) you will be able to use cable with an overall length of 500 m. Refer to the chapter "Mounting and wiring [▶ 27]" for details

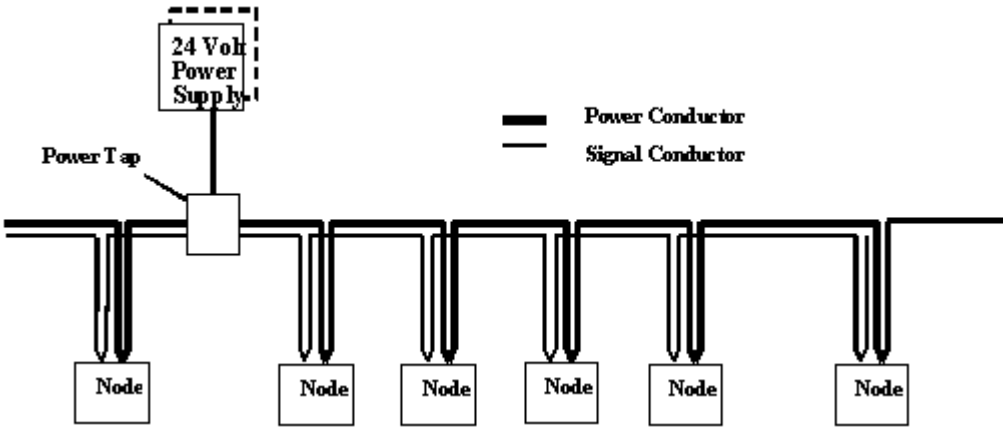


Fig. 6: Example of DeviceNet® cabling

4 Mounting and cabling

4.1 Instructions for ESD protection

NOTICE

Destruction of the devices by electrostatic discharge possible!

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

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

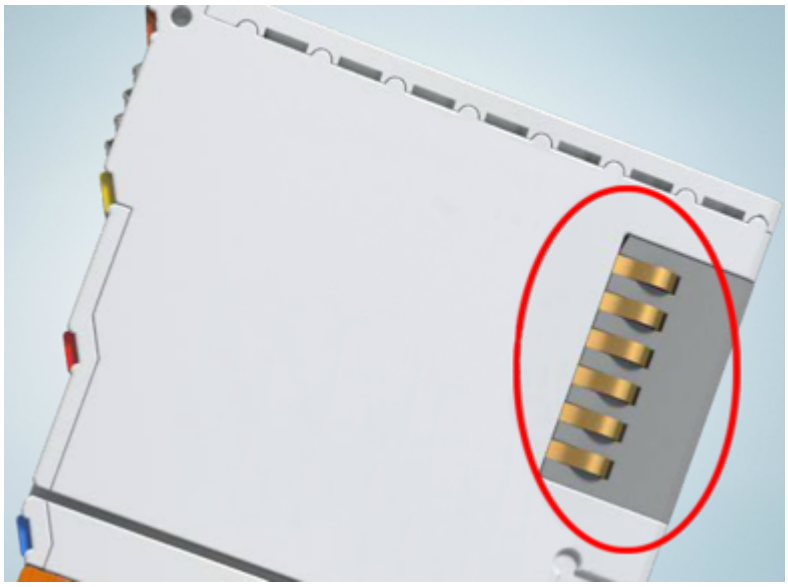


Fig. 7: Spring contacts of the Beckhoff I/O components

4.2 Explosion protection

4.2.1 ATEX - Special conditions (extended temperature range)

⚠ WARNING

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

- The certified components are to be installed in a suitable housing that guarantees a protection class of at least IP54 in accordance with EN 60079-7! The environmental conditions during use are thereby to be taken into account!
- If the temperatures during rated operation are higher than 70°C at the feed-in points of cables, lines or pipes, or higher than 80°C at the wire branching points, then cables must be selected whose temperature data correspond to the actual measured temperature values!
- Observe the permissible ambient temperature range of -25 to 60°C for the use of Beckhoff fieldbus components with extended temperature range (ET) in potentially explosive areas!
- Measures must be taken to protect against the rated operating voltage being exceeded by more than 40% due to short-term interference voltages!
- The individual terminals may only be unplugged or removed from the Bus Terminal system if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- The connections of the certified components may only be connected or disconnected if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- The fuses of the KL92xx/EL92xx power feed terminals may only be exchanged if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- Address selectors and ID switches may only be adjusted if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!

Standards

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

- EN IEC 60079-0:2018
- EN 60079-7:2015+A1:2018

Marking

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

II 3 G Ex ec IIC T4 Gc
KEMA 10 ATEX0075 X
DEKRA 22UKEX6024X
Ta: -25 ... + 60°C

4.2.2 IECEx - Special conditions

⚠ WARNING

Observe the special conditions for the intended use of Beckhoff fieldbus components in potentially explosive areas!

- For gas: The equipment shall be installed in a suitable enclosure providing a degree of protection of IP54 according to IEC 60079-7, taking into account the environmental conditions under which the equipment is used!
- The equipment shall only be used in an area of at least pollution degree 2, as defined in IEC 60664-1!
- Provisions shall be made to prevent the rated voltage from being exceeded by transient disturbances of more than 119 V!
- If the temperatures during rated operation are higher than 70°C at the feed-in points of cables, lines or pipes, or higher than 80°C at the wire branching points, then cables must be selected whose temperature data correspond to the actual measured temperature values!
- Observe the permissible ambient temperature range for the use of Beckhoff fieldbus components in potentially explosive areas!
- The individual terminals may only be unplugged or removed from the Bus Terminal system if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- The connections of the certified components may only be connected or disconnected if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- Address selectors and ID switches may only be adjusted if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- The front hatch of certified units may only be opened if the supply voltage has been switched off or a non-explosive atmosphere is ensured!

Standards

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

- IEC 60079-0:2017 (Edition 7.0)
- IEC 60079-7:2017 (Edition 5.1)

Marking

Beckhoff fieldbus components that are certified in accordance with IECEx for use in areas subject to an explosion hazard bear the following marking:

IECEx DEK 16.0078 X

Ex ec IIC T4 Gc

4.2.3 Continuative documentation for ATEX and IECEx

NOTICE	
	**Continuative documentation about explosion protection according to ATEX and IECEx** Pay also attention to the continuative documentation **Ex. Protection for Terminal Systems** Notes on the use of the Beckhoff terminal systems in hazardous areas according to ATEX and IECEx, that is available for download within the download area of your product on the Beckhoff homepage www.beckhoff.com!

4.3 UL notice

⚠ CAUTION	
c UL us	**Application** Beckhoff EtherCAT modules are intended for use with Beckhoff's UL Listed EtherCAT System only.

⚠ CAUTION	
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).

⚠ CAUTION	
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:

4.4 Mounting and demounting - traction lever 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).

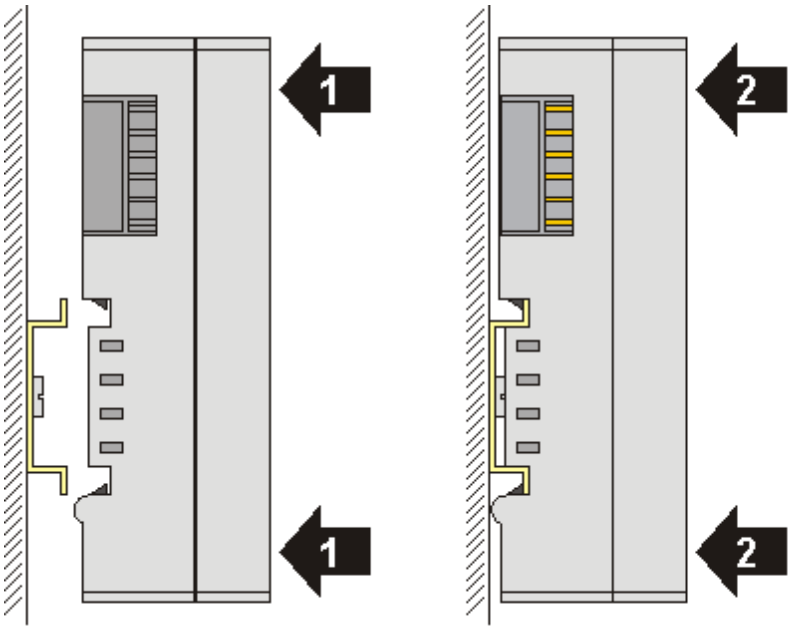
⚠ WARNING

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

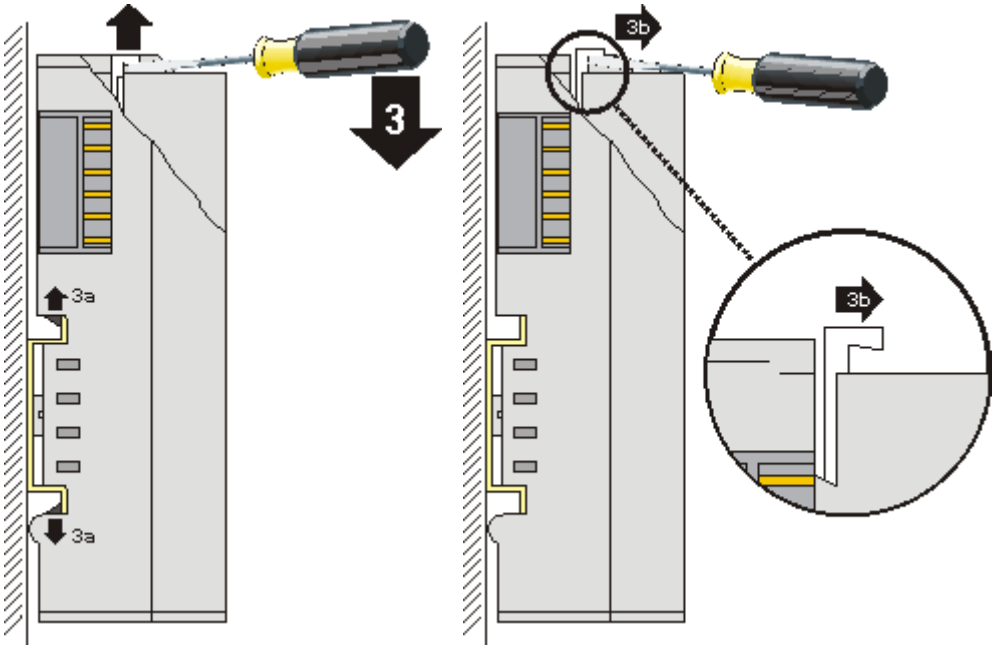
- Fit the mounting rail to the planned assembly location.



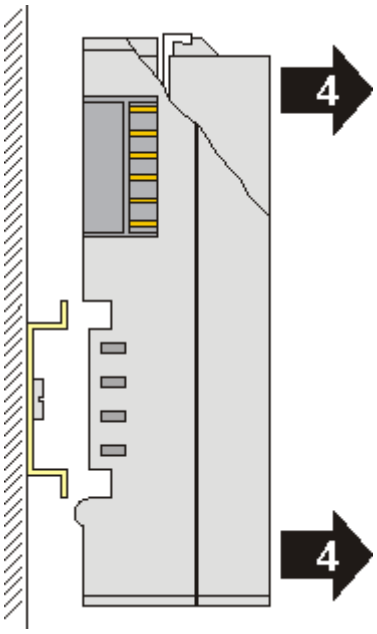
  and press (1) the terminal module against the mounting rail until it latches in place on the mounting rail (2).
- Attach the cables.

Demounting

- Remove all the cables. Thanks to the KM/EM connector, it is not necessary to remove all the cables separately for this, but for each KM/EM connector simply undo 2 screws so that you can pull them off (fixed wiring)!
- Lever the unlatching hook on the left-hand side of the terminal module upwards with a screwdriver (3). As you do this
  - an internal mechanism pulls the two latching lugs (3a) from the top hat rail back into the terminal module,
  - the unlatching hook moves forwards (3b) and engages



- In the case 32 and 64 channel terminal modules (KMxxx4 and KMxxx8 or EMxxx4 and EMxxx8) you now lever the second unlatching hook on the right-hand side of the terminal module upwards in the same way.
- Pull (4) the terminal module away from the mounting surface.



4.5 Mounting and demounting - top 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).

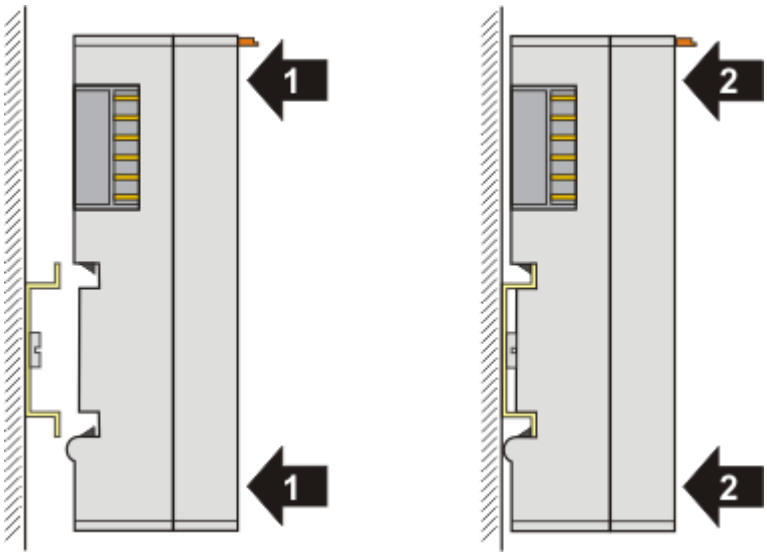
⚠ WARNING

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.

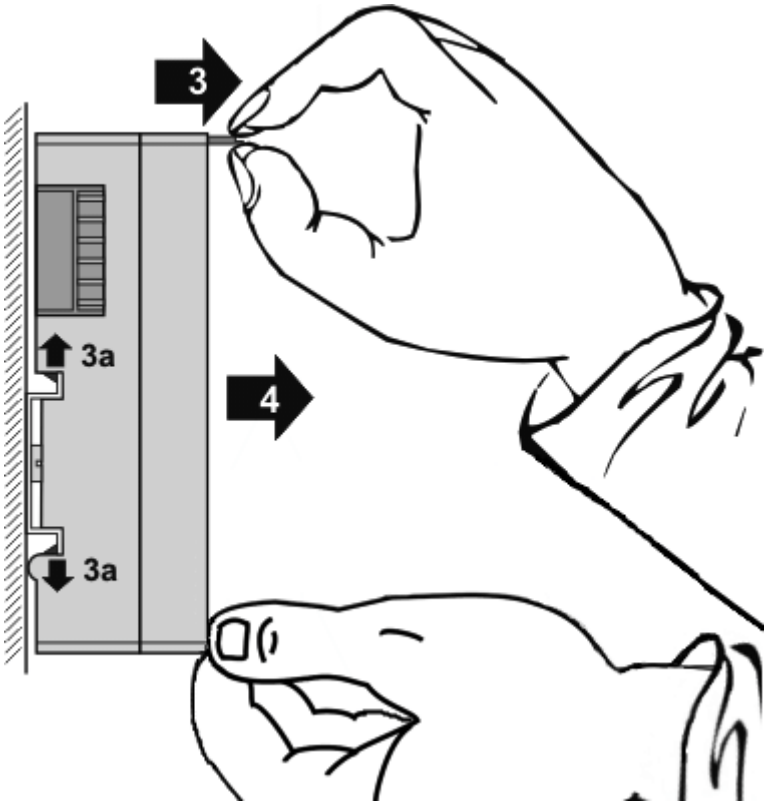


and press (1) the terminal module against the mounting rail until it latches in place on the mounting rail (2).

- 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.

4.6 Disposal

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

4.7 DeviceNet® wiring

4.7.1 CAN / DeviceNet® topology

CAN/DeviceNet is a 2-wire bus system, to which all participating devices are connected in parallel (i.e. using short drop lines) (Fig. *DeviceNet Topology*). The bus must be terminated at each end with a 120 (or 121) Ohm terminating resistor to prevent reflections. This is also necessary even if the cable lengths are very short!

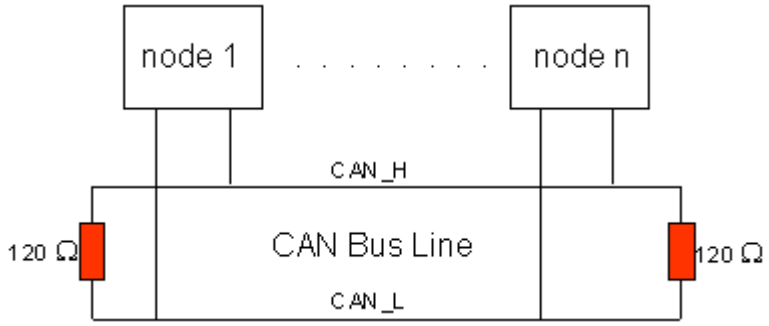


Fig. 8: DeviceNet topology

Since the CAN signals are represented on the bus as the difference between the two levels, the CAN leads are not very sensitive to incoming interference (EMI): Both leads are affected, so the interference has very little effect on the difference.

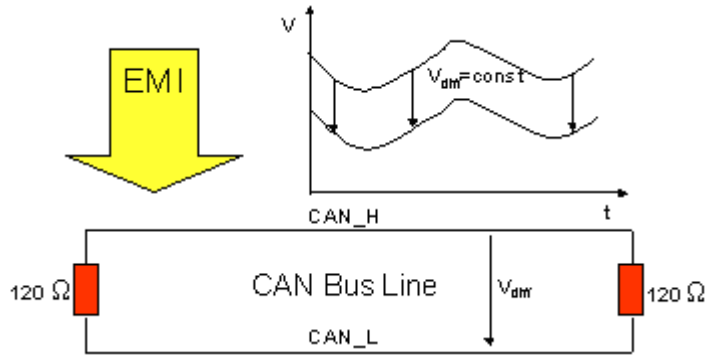


Fig. 9: Low interference through difference levels

4.7.2 Bus length

The maximum length of a CAN bus is primarily limited by the signal propagation delay. The multi-master bus access procedure (arbitration) requires signals to reach all the nodes at effectively the same time (before the sampling within a bit period). Since the signal propagation delays in the CAN connecting equipment (transceivers, opto-couplers, CAN controllers) are almost constant, the line length must be chosen in accordance with the baud rate:

Baud rate	Bus length
500 kbit/s	< 100 m
250 kbit/s	< 250 m
125 kbit/s	< 500 m

4.7.3 Drop lines

Drop lines must always be avoided as far as possible, since they inevitably cause reflections. The reflections caused by drop lines are not however usually critical, provided they have decayed fully before the sampling time. In the case of the bit timing settings selected in the Bus Couplers it can be assumed that this is the case, provided the following drop line lengths are not exceeded:

Baud rate	Drop line length	Total length of all drop lines
500 kbit/s	< 6 m	< 39 m
250 kbit/s	< 6 m	< 78 m
125 kbit/s	< 6 m	< 156 m

Drop lines must not be furnished with termination resistors (Fig. *Drop line topology*).

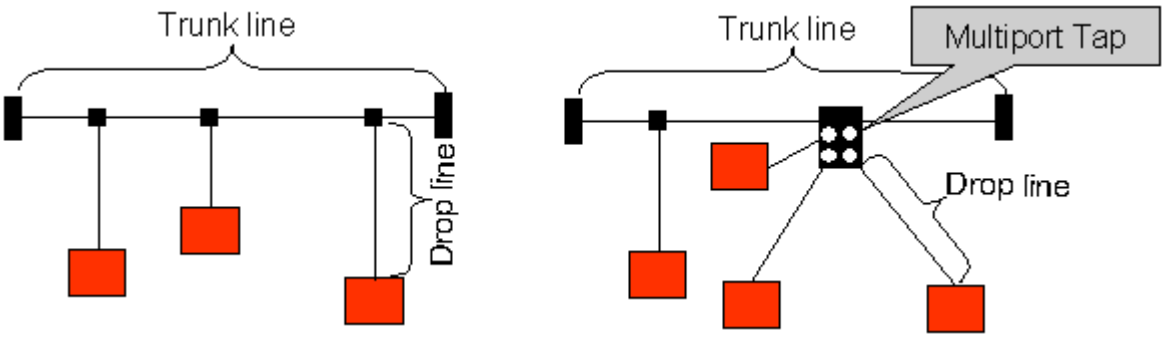


Fig. 10: Drop line topology

4.7.4 Star Hub (Multiport Tap)

Shorter drop line lengths must be maintained when passive distributors ("multiport taps"), such as the Beckhoff ZS5052-4500 Distributor Box. The following table indicates the maximum drop line lengths and the maximum length of the trunk line (without the drop lines):

Guide values

The following values are recommended by BECKHOFF.

Baud rate	Drop line length with multiport topology	Trunk line length (without drop lines)
500 kbit/s	< 1.2 m	< 66 m
250 kbit/s	< 2.4 m	< 120 m
125 kbit/s	< 4.8 m	< 310 m

4.7.5 CAN cable

Screened twisted-pair cables (2x2) with a characteristic impedance of between 108 and 132 Ohm is recommended for the CAN wiring. If the CAN transceiver's reference potential (CAN ground) is not to be connected, the second pair of conductors can be omitted. (This is only recommended for networks of small physical size with a common power supply for all the participating devices).

ZB5200 CAN/DeviceNet® Cable

The ZB5200 cable material corresponds to the DeviceNet® specification, and is also suitable for CANopen systems. The ready-made ZK1052-xxxx-xxxx bus cables for the Fieldbus Box modules are made from this cable material. It has the following specification:

- 2 x 2 x 0.34 mm² (AWG 22) twisted pairs
- double screened - braided screen with filler strand
- characteristic impedance (1 MHz): 126 ohm
- Conductor resistance 54 Ohm/km
- sheath: grey PVC, outside diameter 7.3 mm
- printed with "InterlinkBT DeviceNet Type 572" as well as UL and CSA ratings
- stranded wire colours correspond to the DeviceNet specification
- UL recognized AWM Type 2476 rating
- CSA AWM I/II A/B 80°C 300V FT1
- corresponds to the DeviceNet® "Thin Cable" specification

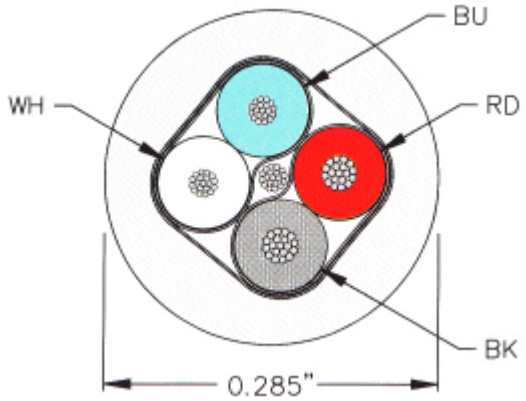


Fig. 11: DeviceNet® cable configuration

4.7.6 Shielding

The screen is to be connected over the entire length of the bus cable, and only galvanically grounded at one point, in order to avoid ground loops.
The design of the screening, in which HF interference is diverted through R/C elements to the mounting rail assumes that the rail is appropriately earthed and free from interference. If this is not the case, it is possible that HF interference will be transmitted from the mounting rail to the screen of the bus cable. In that case the screen should not be attached to the couplers - it should nevertheless still be fully connected through.

4.7.7 Cable colours and pin assignment

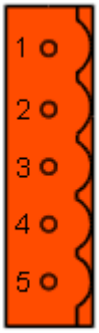


Fig. 12: Pin assignment (top view EL6752)

Suggested method of using the Beckhoff CAN cable on Bus Terminal and Fieldbus Box:

Pin	EL6752 assignment	ZB5200 cable color
1	V+ (24 V)	red
2	CAN High	white
3	Shield	Filler strand
4	CAN Low	blue
5	V-	black

4.8 Installation positions

NOTICE

Constraints regarding installation position and operating temperature range

- Please refer to the technical data of the device to ascertain whether any restrictions regarding the installation position and/or the operating temperature range have been specified.
- When installing devices with increased heat dissipation, ensure that there is sufficient space above and below the devices during operation to guarantee adequate ventilation!

The installation positions and their names for mounting devices on mounting rails are specified below. The illustration of the devices in the following figures is an example.
The following applies to all installation positions: The reference direction "down" (see arrow) is the acceleration of gravity.

Horizontal installation (standard installation)

The mounting rail is mounted horizontally on a vertical mounting plate.
The connection level of the devices points to the front.

The devices are ventilated from below, which enables optimum cooling of the electronics through convection. This is therefore also the recommended installation position.

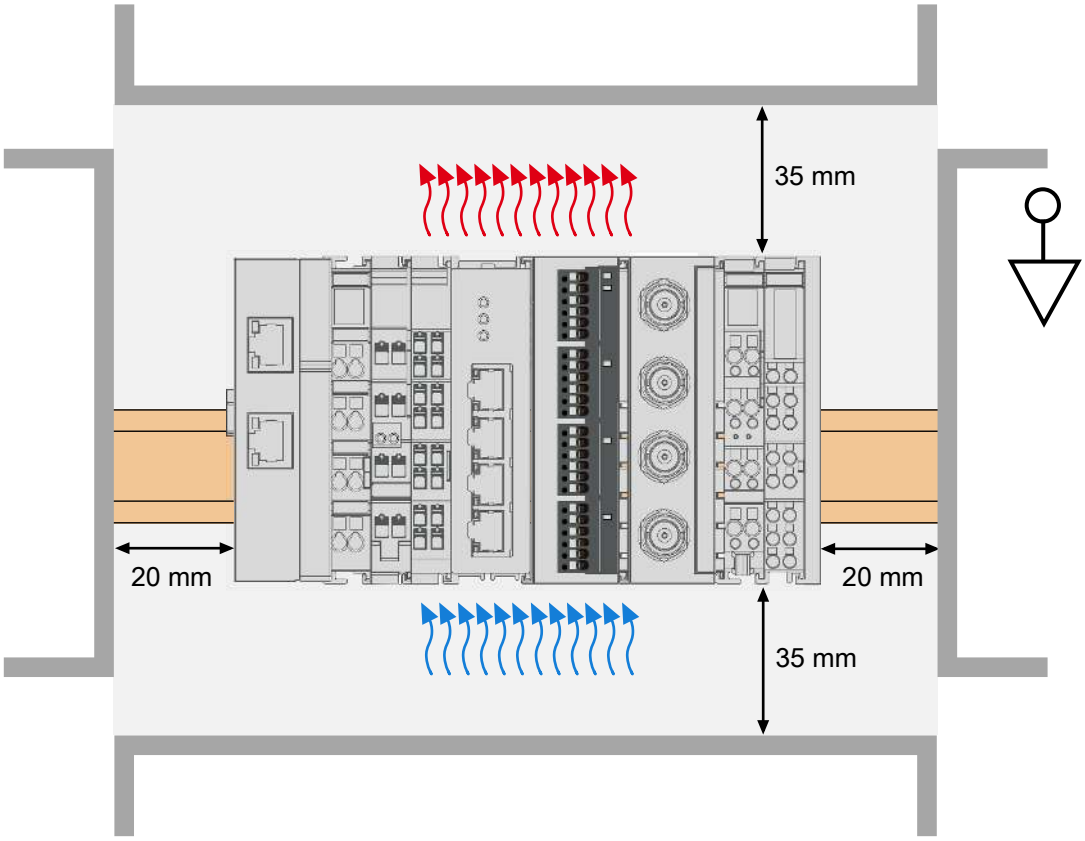


Fig. 13: Recommended minimum distances for standard installation position

NOTICE

Compliance with the minimum distances

Compliance with the minimum distances shown in the figure "Recommended minimum distances for standard installation position" is strongly recommended in all installation positions.

Vertical installation

The mounting rail is mounted vertically on a vertical mounting plate.
The connection level of the devices points to the front.
The devices can be arranged as follows:

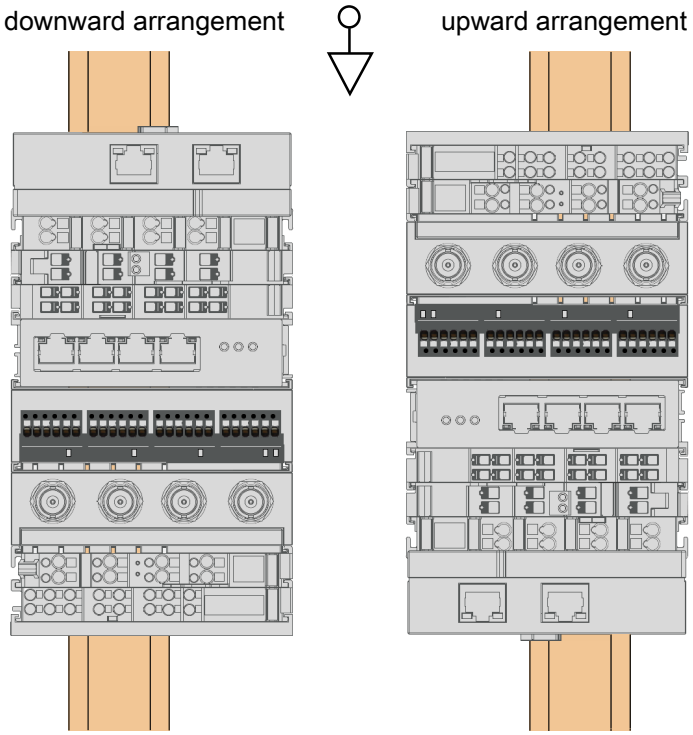


Fig. 14: Installation position Vertical, downward arrangement (left) / upward arrangement (right)

Flat installation

In the flat installation position, the mounting rail is laid on a horizontal mounting plate.
The connection level of the devices points upwards.

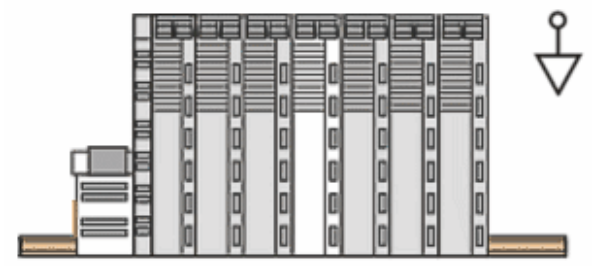


Fig. 15: Flat installation position

NOTICE

Danger of sliding off the mounting rail

Particularly in the "vertical" installation position, but also in other installation positions under corresponding mechanical load, the terminal segment may move on the mounting rail. These can lead to undesirable malfunctions.

- If this risk exists, secure the terminal segment with appropriate locking devices, e.g. by clamping it to the mounting rail.

NOTICE

Compliance with the minimum distances

Compliance with the minimum distances shown in the figure "Recommended minimum distances for standard installation position" is strongly recommended in all installation positions.

Installation positions with ZB8610 fan cartridge

If the cooling should or must be increased for the intended application, the ZB8610 fan cartridge can be mounted on the underside of the device. In the horizontal installation position, the devices are ventilated from bottom to top by the fan cartridge. The optimum cooling is further enhanced by convection ventilation (see following figure).

The fan cartridge can be used in any installation position.

Further information on operation with and without a fan can be found in the technical data for the device (e.g. derating, information on installation positions, etc.).

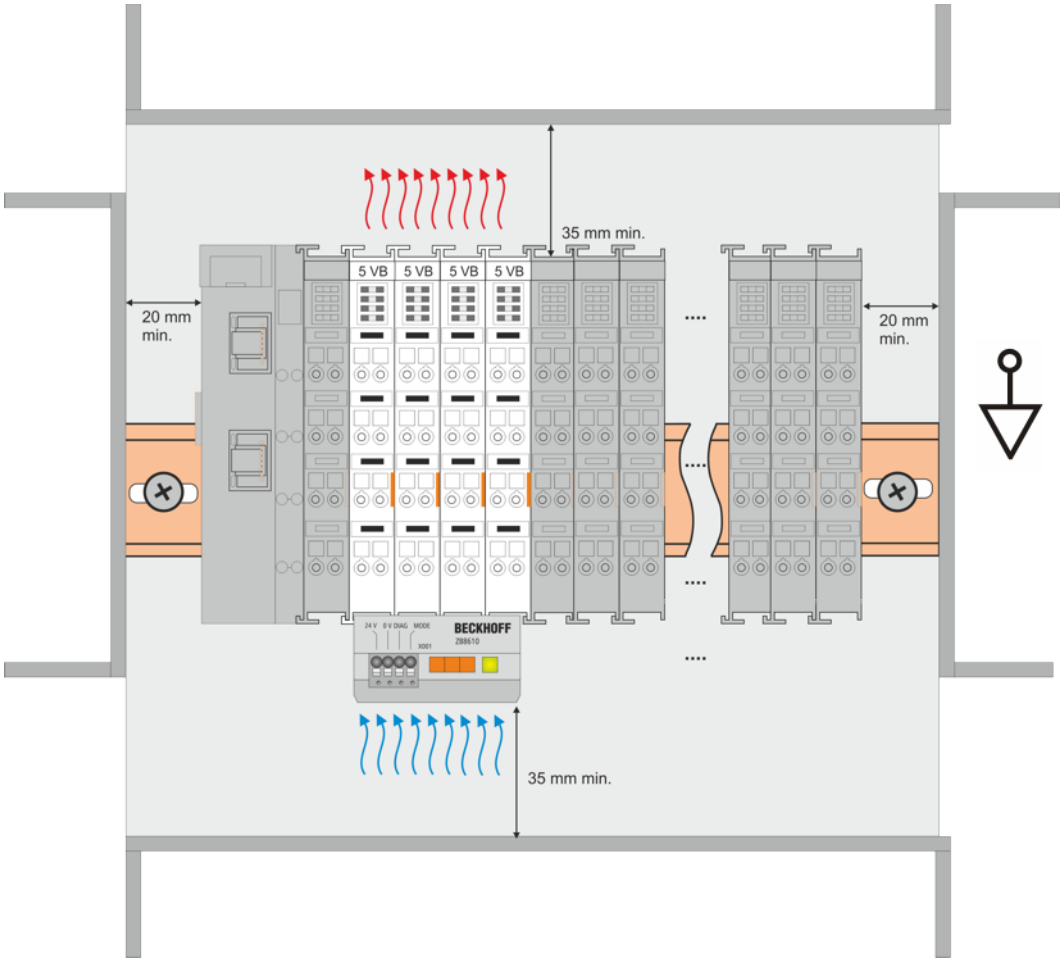


Fig. 16: Recommended minimum distances for operation with fan, using horizontal installation position as an example

NOTICE

Compliance with the minimum distances

Compliance with the minimum distances as shown in the figure "Recommended minimum distances for operation with fan" is strongly recommended.

4.9 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)

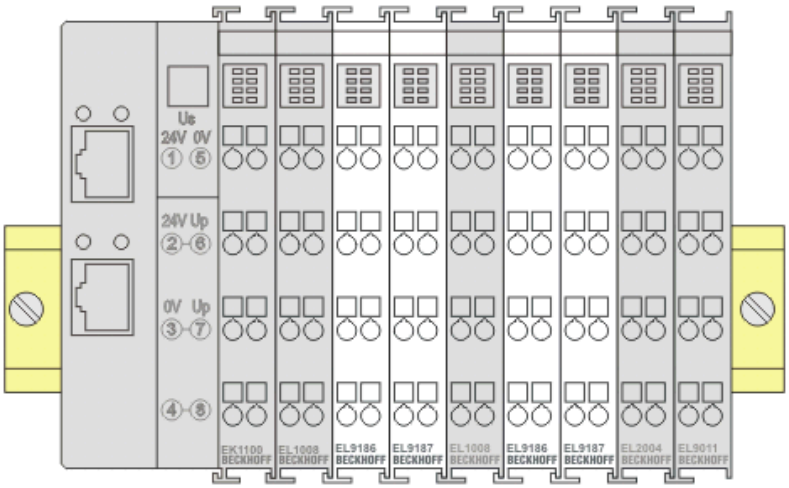


Fig. 17: Correct positioning

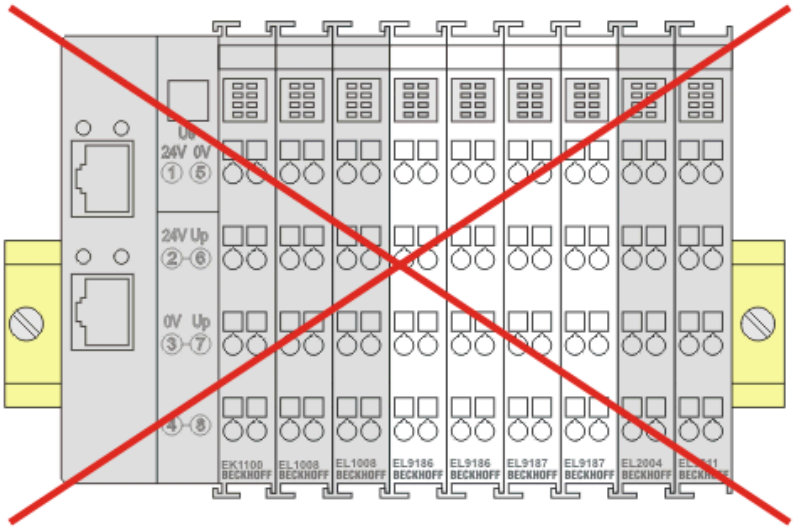


Fig. 18: Incorrect positioning

5 DeviceNet® communication

5.1 DeviceNet® Introduction

Fig. 19: DeviceNet®

DeviceNet® is an open system based on CAN. CAN was developed some years ago by R. Bosch for data transmission in motor vehicles. Millions of CAN chips are now in use. A disadvantage for application in automation is that CAN does not contain definitions for the application layer. CAN only defines the physical and data link layer.

DeviceNet® specifies a uniform application layer and this makes it possible to use the CAN protocol for industrial applications. ODVA (the Open DeviceNet Vendor Association) is an independent association which supports manufacturers and users of the DeviceNet® system. ODVA ensures that all devices which conform to the specification can operate together in one system, regardless of their manufacturer.

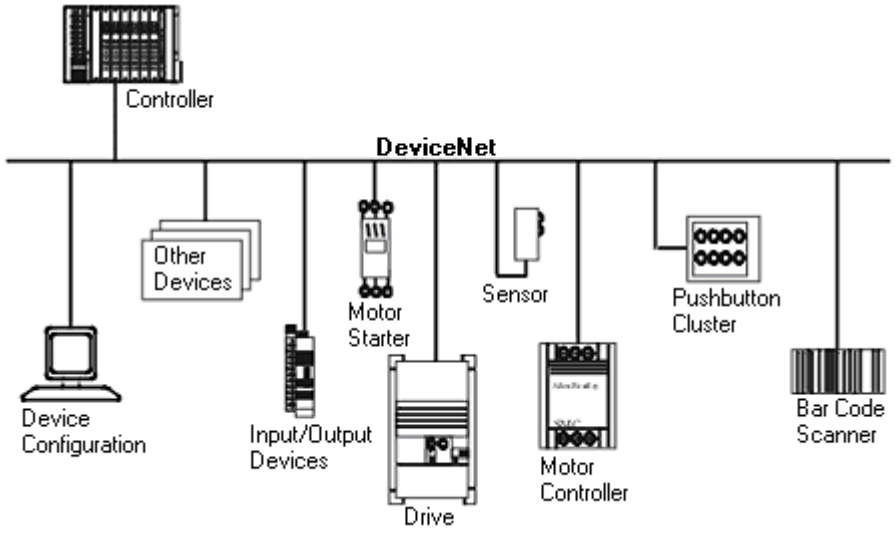


Fig. 20: Example of DeviceNet® in use

DeviceNet® is a sensor/actuator bus system. It is internationally standardised (EN50325) and is based on CAN (Controller Area Network). DeviceNet® supports a number of communication types for the input and output data:

- Polling: The master module ("scanner") sends the output data cyclically to the assigned devices and receives the input data in an answer telegram.
- Change-of-State: Telegrams are sent as soon as their contents have changed.
- Cyclic: The modules send the data automatically after a cycle time has elapsed.
- Strobed: The scanner requests the input data using a broadcast telegram to all the devices.

The DeviceNet® devices support all I/O communication types.

The DeviceNet® devices are parameterized via acyclical services (explicit messaging).

The effective utilization of the bus bandwidth allows DeviceNet®, particularly in Change-of-State mode, to achieve short system reaction times in spite of the relatively low data rates. The BECKHOFF DeviceNet® devices have a powerful implementation of the protocol. Through active participation in the ODVA's technical committees, BECKHOFF are contributing to the further development of this bus system, and has in this way itself gathered profound DeviceNet® expertise.

Configuration

The node address is set in the range from 0 to 63 using two decimally coded rotary switches. The data transfer rate set at the DeviceNet® scanner is automatically recognized by the DeviceNet® Box (auto baud rate). "Electronic Data Sheets" (EDS files) for DeviceNet® configuration tools are available for download from the Beckhoff internet site (http://www.beckhoff.de), and on the BECKHOFF product CDs. Special I/O parameters that are not covered by the DeviceNet® standard can be set via the KS2000 software (serial connection) or via acyclical explicit messages.

Diagnostics

The extensive diagnostic functions of the BECKHOFF DeviceNet® devices allow rapid fault localisation. The diagnostic messages are transmitted over the bus and collated by the master. The status of the network connection, the device status, the status of the inputs and outputs and of the power supply are displayed by LEDs.

Data transfer rates

Three data transfer rates from 125 kbaud to 500 kbaud are available for different bus lengths. The effective utilization of the bus bandwidth allows DeviceNet® to achieve short system reaction times at relatively low data rates.

Topology

DeviceNet® is based on a linear topology. The number of devices participating in each network is logically limited by DeviceNet® to 64, but physically the present generation of drivers allows up to 64 nodes in one network segment. The maximum possible size of the network for any particular data rate is limited by the signal propagation delay required on the bus medium. For 500 kbaud, for instance, the network may extend 100 m, whereas at 125 kbaud the network may reach up to 500 m. At low data rates the size of the network can be increased by repeaters, which also allow the construction of tree structures.

Bus access procedures

CAN utilizes the Carrier Sense Multiple Access (CSMA) procedure, i.e. all participating devices have the same right of access to the bus and may access it as soon as it is free (multi-master bus access). The exchange of messages is thus not device-oriented but message-oriented. This means that every message is unambiguously marked with a prioritized identifier. In order to avoid collisions on the bus when messages are sent by different devices, a bit-wise bus arbitration is carried out at the start of the data transmission. The bus arbitration assigns bus bandwidth to the messages in the sequence of their priority. At the end of the arbitration phase only one bus device occupies the bus, collisions are avoided and the bandwidth is optimally exploited.

Configuration and parameterization

The TwinCAT System Manager allows all the DeviceNet® parameters to be set conveniently. An "eds" file (electronic data sheet) is available on the BECKHOFF website (http://www.beckhoff.de) for the parameterization of BECKHOFF DeviceNet® devices using configuration tools from other manufacturers.

5.2 Explicit messages

Program example „ExplMessageEditor“: https://infosys.beckhoff.com/content/1033/el6752/Resources/5979571979.zip

With the following ADS commands you can use EL6752 to send explicit messages.

```
GET_ATTRIBUTE_SINGLE via ADSRead Data Transfer
SET_ATTRIBUTE_SINGLE via ADSWrite Data Transfer
COMMON SERVICE via ADSReadWrite Data Transfer
```

For the ADS NetID and the port, the values from the system manager are to be used.

```
(*
GET_ATTRIBUTE_SINGLE via ADSRead Data Transfer

IDXGRP: Index GroupNumber = Object Class
IDXOFFS: Index OffsetNumber = (Object Instance *. 0x100) + Attribute Id
LEN: Read Data Lengths in Bytes
DESTADDR: Address of DataBuffer to read with the Get-Attribute Single Service
*)
```

```
fbADSRead(
NETID:= ADSNetId,
PORT:= ADSPort,
IDXGRP:= IGrp_ADSRead,
IDXOFFS:= IOff_ADSRead,
LEN:= ADSReadLen,
DESTADDR:= ADR(GetAttributeData[0]),
READ:= ADSReadCommand,
TMOUT:= T#5s,
BUSY=> ADSReadBusy,
ERR=> ADSReadErr,
ERRID=> ADSReadErrID);
(*
```

```
COMMON SERVICE via ADSReadWrite Data Transfer

IDXGRP: Index GroupNumber = Object Class
IDXOFFS: Index OffsetNumber = (Object Instance *. 0x100) + Service Id
WRITELEN: Write Data Lengths in Bytes
READLEN: Read Data Lengths in Bytes
SRCADDR: Address of DataBuffer to write
DESTADDR: Address of DataBuffer to read
*)
```

```
fbADSReadWrite(
NETID:= ADSNetId,
PORT:= ADSPort,
IDXGRP:= Grp_ADSReadWrite,
IDXOFFS:= IOff_ADSReadWrite,
WRITELEN:= ADSReadWriteWriteLen,
READLEN:= ADSReadWriteReadLen,
SRCADDR:= ADR(CommonServiceWriteData[0]),
DESTADDR:= ADR(CommonServiceReadData[0]),
WRTRD:= ADSReadWriteCommand,
TMOUT:= T#5s,
BUSY=> ADSReadWriteBusy,
ERR=> ADSReadWriteErr,
ERRID=> ADSReadWriteErrID);
```

and

```
(*
SET_ATTRIBUTE_SINGLE via ADSWrite Data Transfer
IDXGRP: Index GroupNumber = Object Class
IDXOFFS: Index OffsetNumber = (Object Instance *. 0x100) + Attribute Id
LEN: Write Data Lengths in Bytes
SRCADDR: Address of DataBuffer to write with the Set-Attribute Single Service
*)
```

```
fbADSWrite(
NETID:= ADSNetId,
PORT:= ADSPort,
IDXGRP:= IGrp_ADSWrite,
```

```
IDXOFFS:= IOff_ADSWrite,
LEN:= ADSWriteLen,
SRCADDR:= ADR(SetAttributeWriteData[0]),
WRITE:= ADSWriteCommand,
TMOUT:= T#5s,
BUSY=> ADSWriteBusy,
ERR=> ADSWriteErr,
ERRID=> ADSWriteErrID);
```

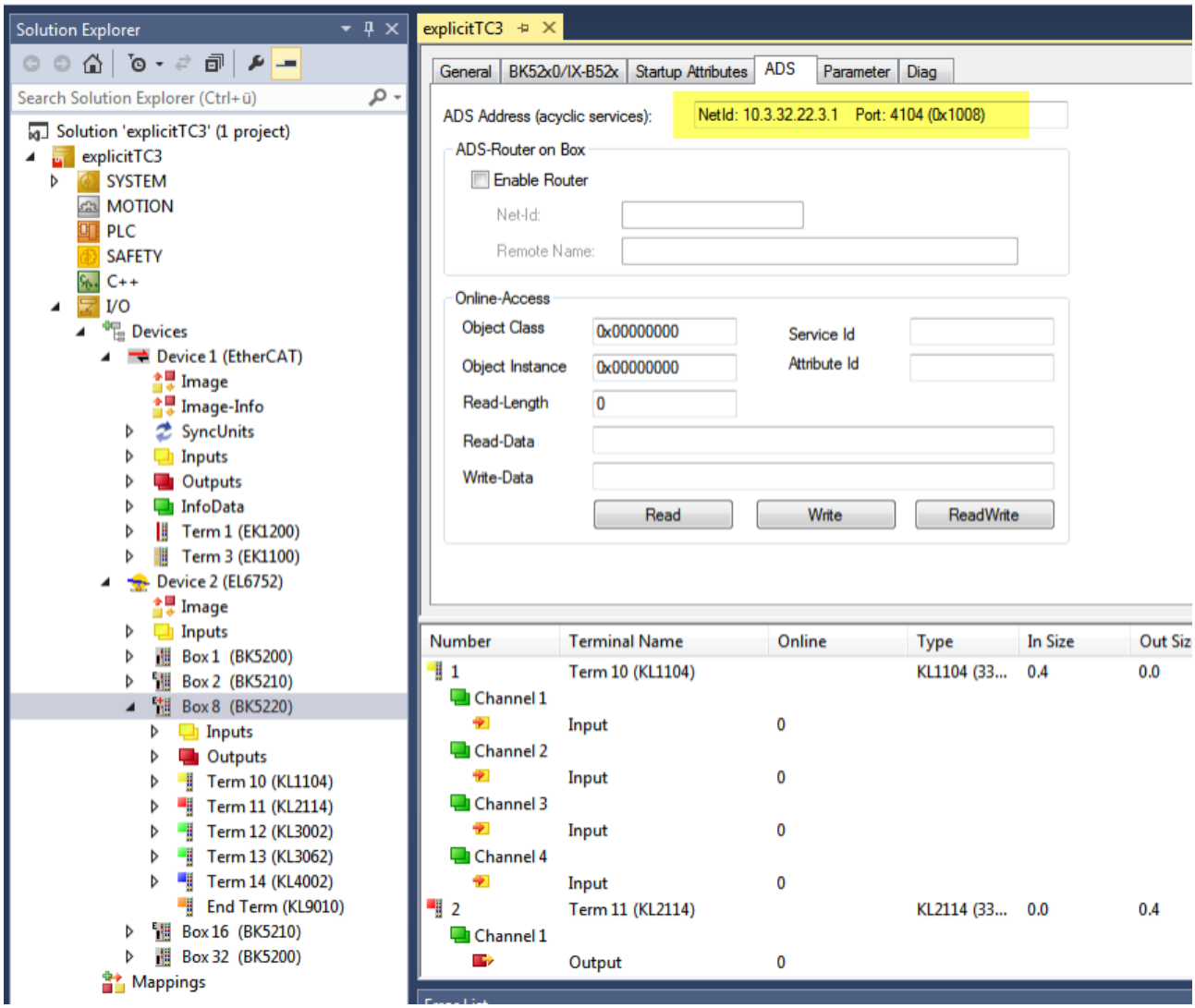


Fig. 21: Using ADS NetID and Port from System Manager

6 Parameterization and commissioning

6.1 CoE Interface

General description

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

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

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

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

The value ranges are

- Index: 0x0000 …0xFFFF ($0...65535_{dec}$)
- Subindex: 0x00…0xFF ($0...255_{dec}$)

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

The relevant ranges for EtherCAT fieldbus users are:

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

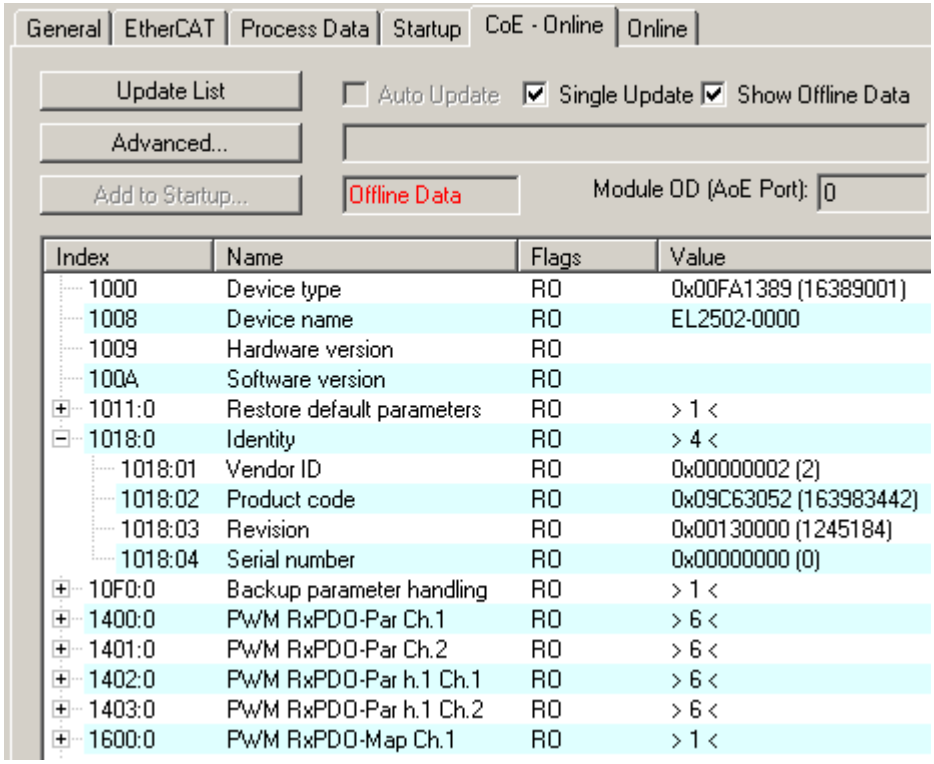
Other important ranges are:

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

Availability

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

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



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. 22: “CoE Online” tab

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

NOTICE

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

When using/manipulating the CoE parameters observe the general CoE notes in chapter "CoE interface" of the EtherCAT system documentation:

- Keep a startup list if components have to be replaced,
- Distinction between online/offline dictionary,
- Existence of current XML description (download from the Beckhoff website),
- "CoE-Reload" for resetting the changes
- Program access during operation via PLC (see TwinCAT 3 | PLC Library: "Tc2_EtherCAT" and Example program R/W CoE)

Data management and function “NoCoeStorage”

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

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

Data management

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

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

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

Startup list

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

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

Recommended approach for manual modification of CoE parameters

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

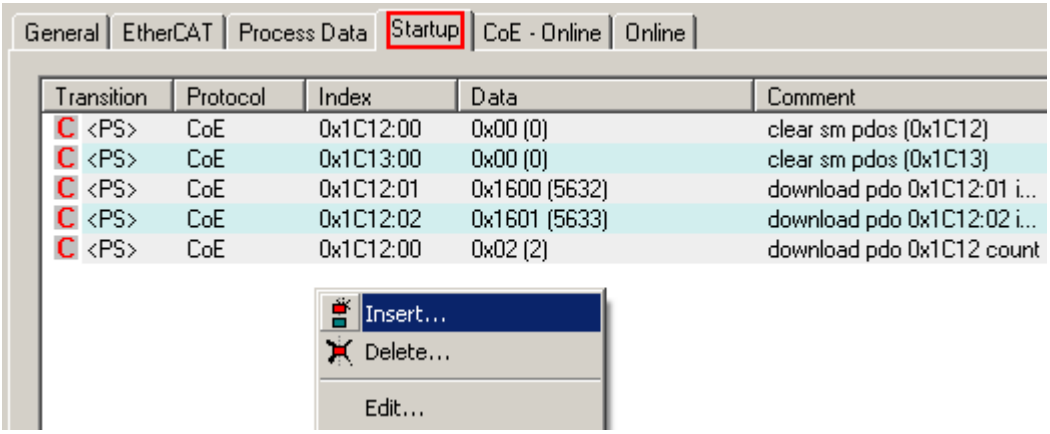


Fig. 23: Startup list in the TwinCAT System Manager

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

Online / offline list

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

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

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

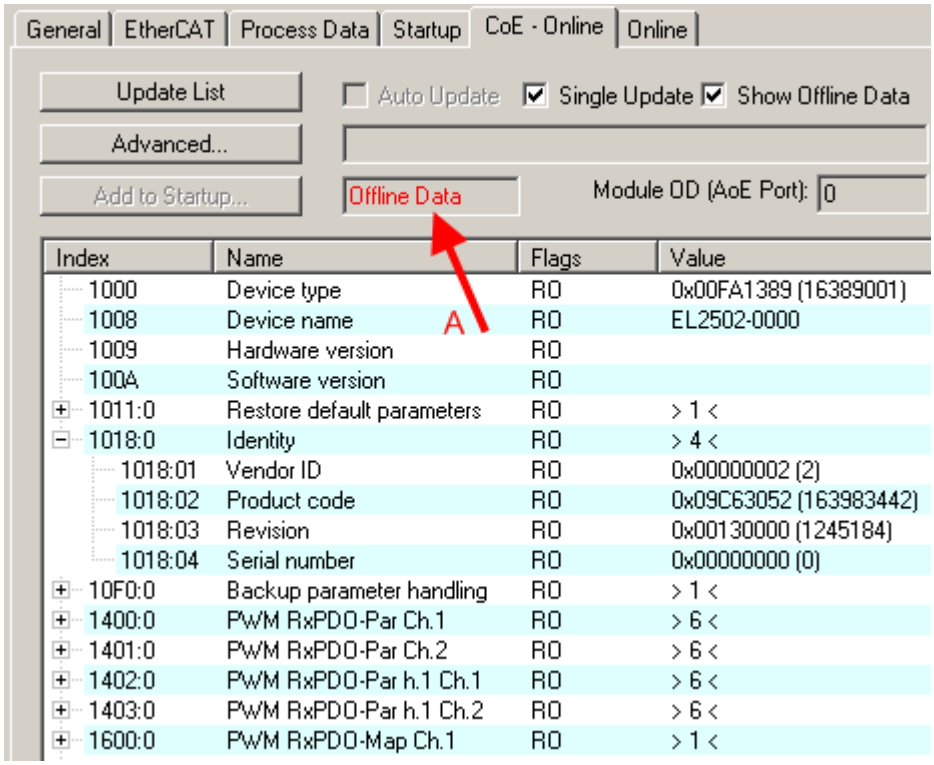


Fig. 24: Offline list

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

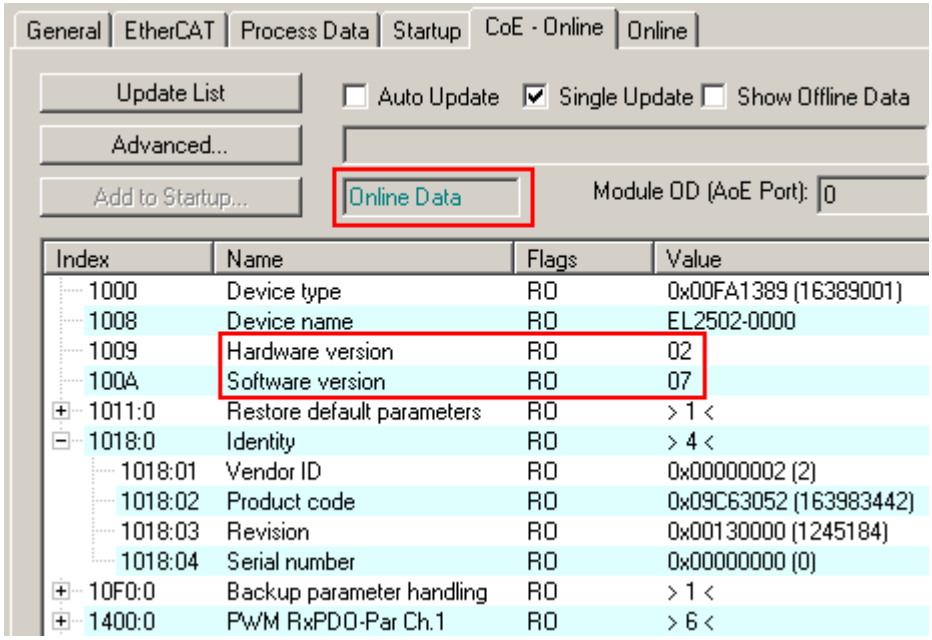


Fig. 25: Online list

Channel-based order

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

In the CoE system 16 indices, each with 255 subindices, are generally sufficient for representing all channel parameters. The channel-based order is therefore arranged in 16_{dec} or 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.

BackUp objects and checksum 0x10F0:01

The following object types, among others, are defined for the CoE parameter/object directory with regard to information retention, but they do not all have to appear in every device at the same time:

- Vendor objects
  - Are stored persistently (fail-safe) in the device.
  - Technically with the property ReadWrite (RW).
  - Can only be modified/deleted if the corresponding vendor password is known.
  - Used for vendor-specific adjustment or identity data.
- BackUp objects
  - These are objects that are stored persistently in the device, even after changes.
  - Technically with the property ReadWrite (RW).
  - They can be modified/deleted at any time via CoE access from the EtherCAT master

- In particular, they are reset to their initial state, which is stored permanently in the firmware, by selecting "Restore Default Parameters" (see chapter "Restoring the delivery state"). Since this reset to a previous value resembles the restoration of a backup, these are referred to as “BackUp objects”.
- They are used for regular function parameters of the device that determine the behavior.

- BackUp objects with write protection option.
  - The same applies as for the BackUp objects.
  - In addition, the user can activate write protection for these objects using a code word in xF009, thereby preventing accidental changes. Details can be found in the device documentation of the devices that contain these objects.
- Volatile objects
  - These are objects that are not stored persistently in the device.
  - They are used to display internal information (process data, states, temperatures ...) and are available as ReadOnly (RO) or as function parameters (ReadWrite). However, the latter must be written by the EtherCAT master for each PowerOn if they are to have a value other than the default value.

The device indicates a 16-bit CRC in the 32-bit object 0x10F0:01 CheckSum, subindex 01 of Backup Parameter Handling by means of the current state of the so-called BackUp objects:

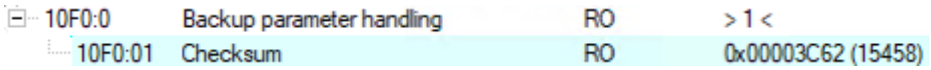


Fig. 26: CoE index 10F0

If a BackUp object is changed, the firmware calculates a new checksum accordingly. This can be used to detect changes to the BackUp objects.

Note: The initial value of the checksum may vary depending on the firmware version, as function extensions can add additional objects to the CRC’s detection range.

6.2 General notes for setting the watchdog

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

The EtherCAT slave controller features two watchdogs:

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

Their times are individually parameterized in TwinCAT as follows:

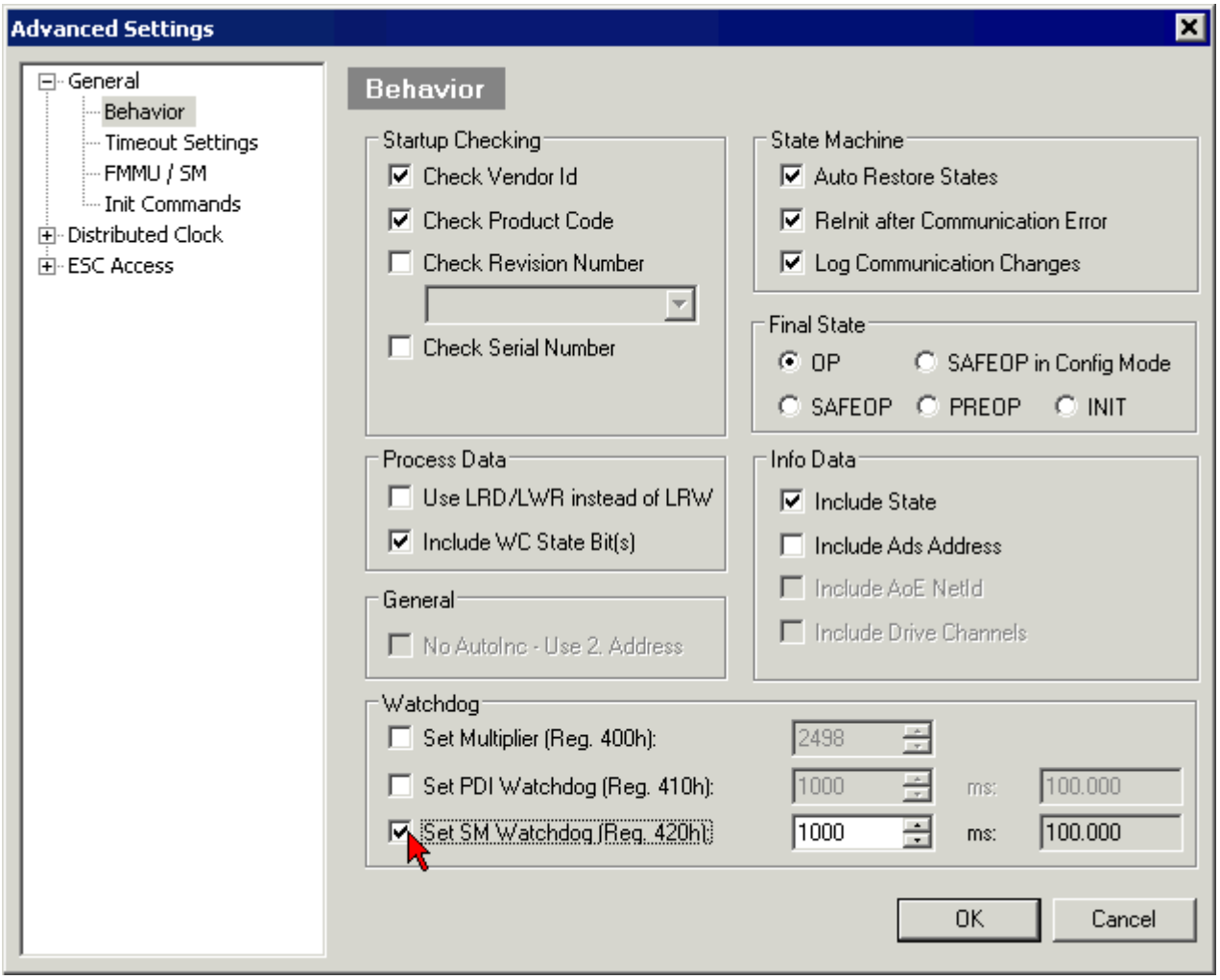


Fig. 27: eEtherCAT tab -> Advanced Settings -> Behavior -> Watchdog

Notes:

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

SM watchdog (SyncManager Watchdog)

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

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

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

PDI watchdog (Process Data Watchdog)

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

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

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

Calculation

Watchdog time = [1/25 MHz * (Watchdog multiplier + 2)] * SM/PDI watchdog

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

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

⚠ CAUTION

Undefined state possible!

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

⚠ CAUTION

Damage of devices and undefined state possible!

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

6.3 EtherCAT State Machine

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

A distinction is made between the following states:

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

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

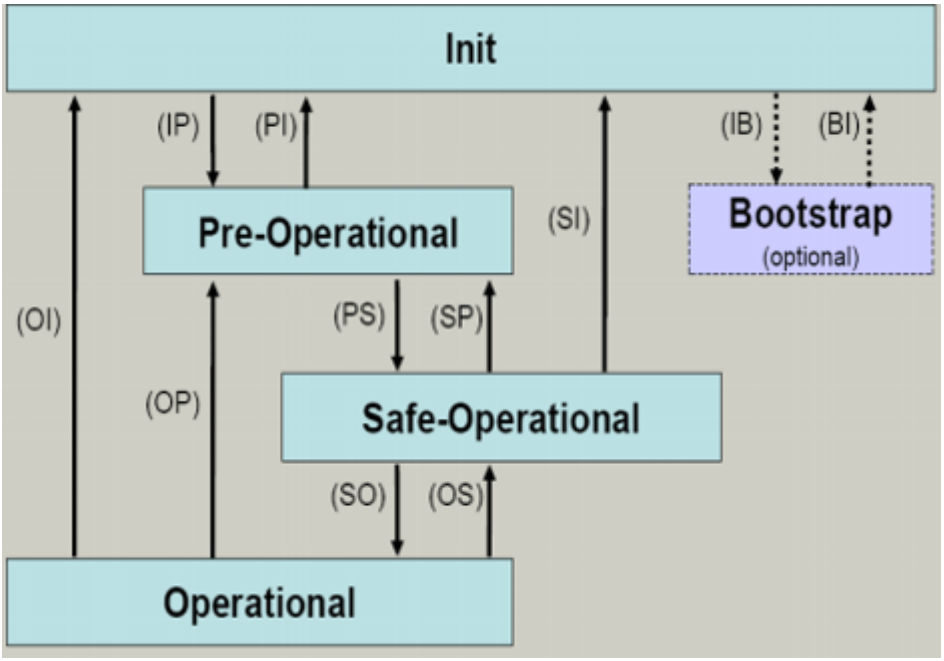


Fig. 28: States of the EtherCAT State Machine

Init

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

Pre-Operational (Pre-Op)

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

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

Safe-Operational (Safe-Op)

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

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

Outputs in SAFEOP state

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

Operational (Op)

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

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

Boot

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

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

6.4 TwinCAT System Manager

The TwinCAT System Manager tool is used for the configuration of the EL6752 DeviceNet® master/slave terminal. The System Manager provides a representation of the number of programs of the TwinCat PLC systems, the configuration of the axis control and of the connected I/O channels as a structure, and organizes the mapping of the data traffic.

Fig. 29: TwinCAT System Manager logo

For applications without TwinCAT PLC or NC, the TwinCAT System Manager Tool configures the programming interfaces for a wide range of application programs:

- ActiveX control (ADS-OCX) for e.g. Visual Basic, Visual C++, Delphi, etc.
- DLL interface (ADS-DLL) for e.g. Visual C++ projects
- Script interface (ADS script DLL) for e.g. VBScript, JScript, etc.

System Manager – Features

- Bit-wise association of server process images and I/O channels
- Standard data formats such as arrays and structures
- User defined data formats
- Continuous variable linking
- Drag and Drop
- Import and export at all levels

Configuration by means of the TwinCAT System Manager

The procedure and the configuration facilities in the System Manager are described below.

EL6752 DeviceNet® master terminal [▶ 49]
EL6752-0010 - DeviceNet® slave terminal [▶ 52]

EL6752 DeviceNet® master terminal

Append device

The terminal can be appended to the I/O configuration either using the "Device search" routine in the TwinCAT System Manager or by manually selecting the "DeviceNet® Master EL6752, EtherCAT" from the possible DeviceNet® devices (Fig. *Appending the device "DeviceNet slave EL6752, EtherCAT"*). A right-click brings up the following context menu for selection:

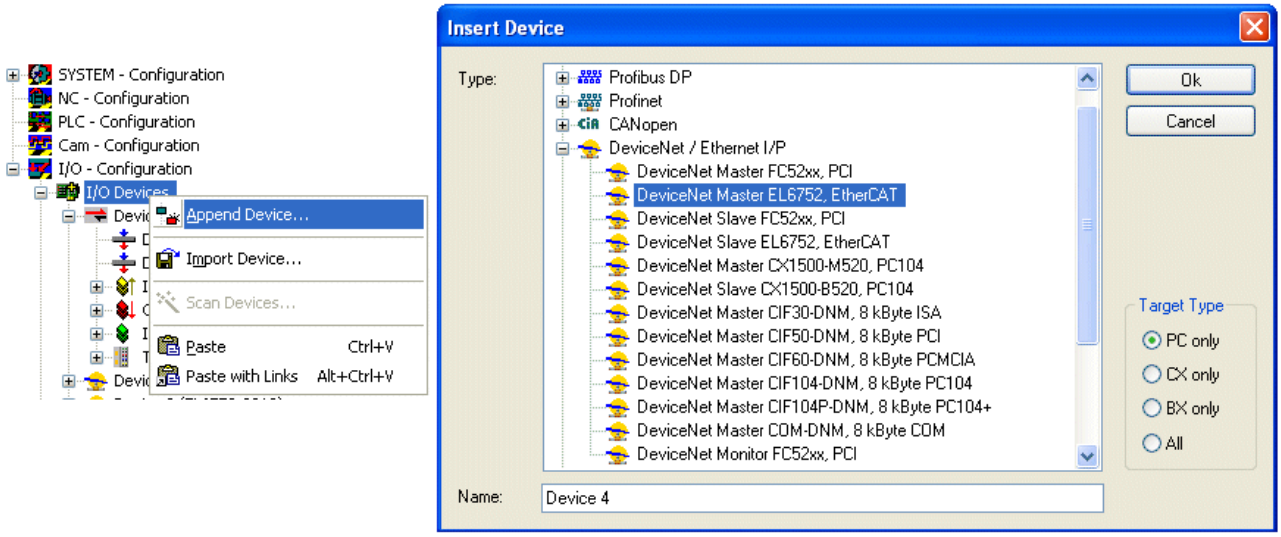


Fig. 30: Appending the device "DeviceNet master EL6752, EtherCAT"

"EL6752" tab

Click on the "Device EL6752" in the TwinCAT tree and then on the EL6752 tab:

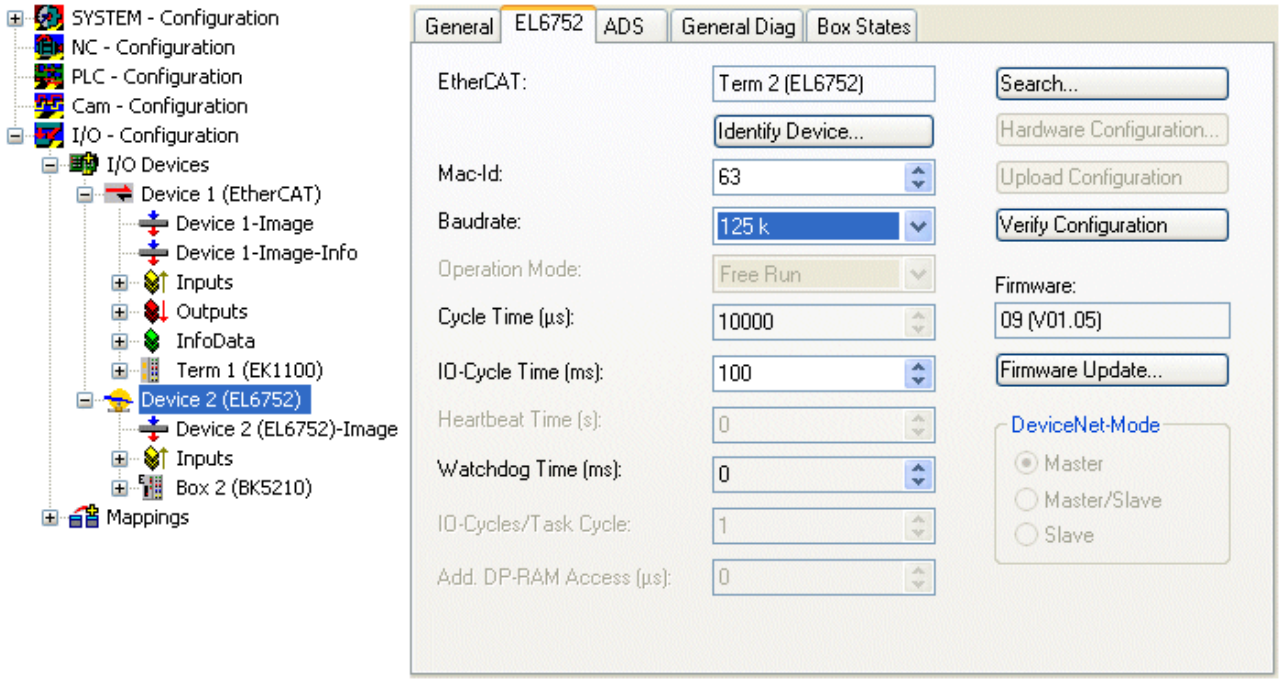


Fig. 31: "EL6752" tab

EtherCAT

Terminal ID in the terminal network.

MAC-ID

Each DeviceNet® device - master included - requires a unique station number referred to as MAC ID (Medium Access Identifier) - value range: 0...63.

Baud rate

Baud rate setting: 125 kbaud, 250 kbaud or 500 kbaud

Cycle time

Displays the cycle time of the corresponding highest priority task. The display is updated when the mapping is generated.

IO-Cycle Time

Setting of the cycle time for the I/O connections. This value is the standard value for newly inserted boxes.

Watchdog time

Time until triggering of the watchdog

Search...

This function searches for all existing channels of the EL6752 and the desired one can be selected.

Check configuration

In preparation.

Firmware

Shows the current firmware version of the EL6752.

Firmware Update...

Updates the EL6752 firmware. Attention: The TwinCAT System must be stopped for this function.

“ADS” tab

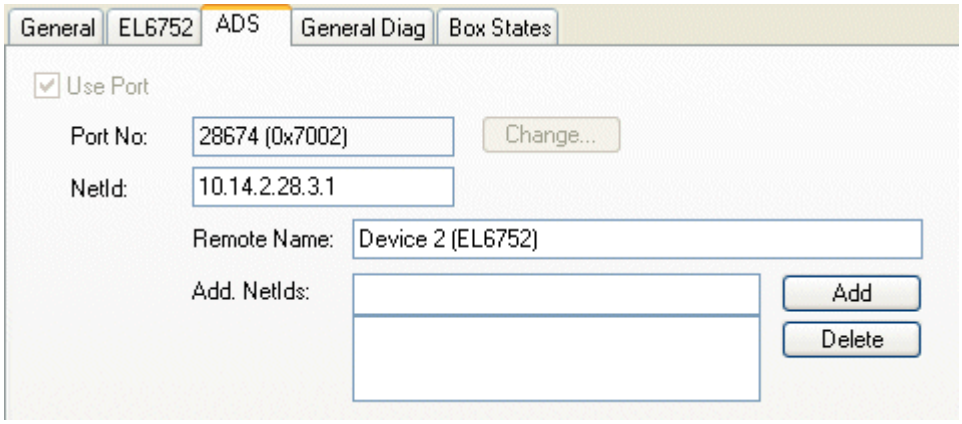


Fig. 32: “ADS” tab

The EL6752 is an ADS device with its own net ID, which can be changed here. All ADS services (diagnostics, acyclical communication) associated with the EL6752 device must address the card via this NetID.

"Box States" tab

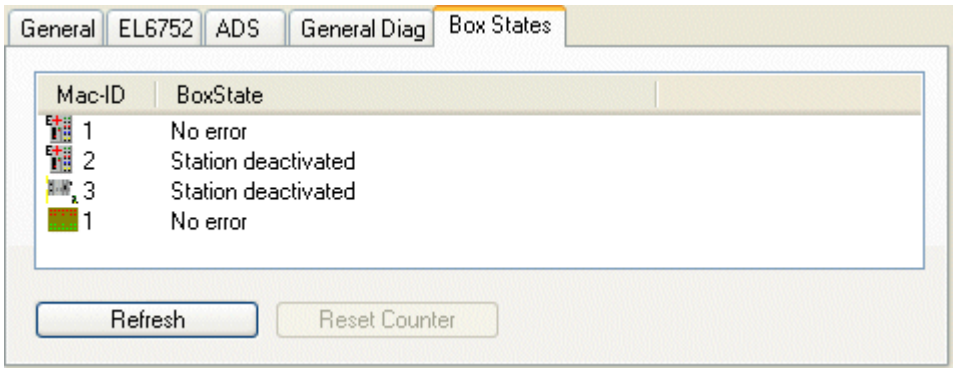


Fig. 33: "Box states" tab

Displays an overview of all current box statuses.

EL6752-0010 – DeviceNet® slave terminal

In the system configuration tree structure right-click on I/O Devices and “Append device” to open the selection list of supported fieldbus cards.
Select EL6752-0010 CANopenSlave. TwinCAT searches for the terminal and displays the memory addresses and slots it finds. Select the required address and confirm.

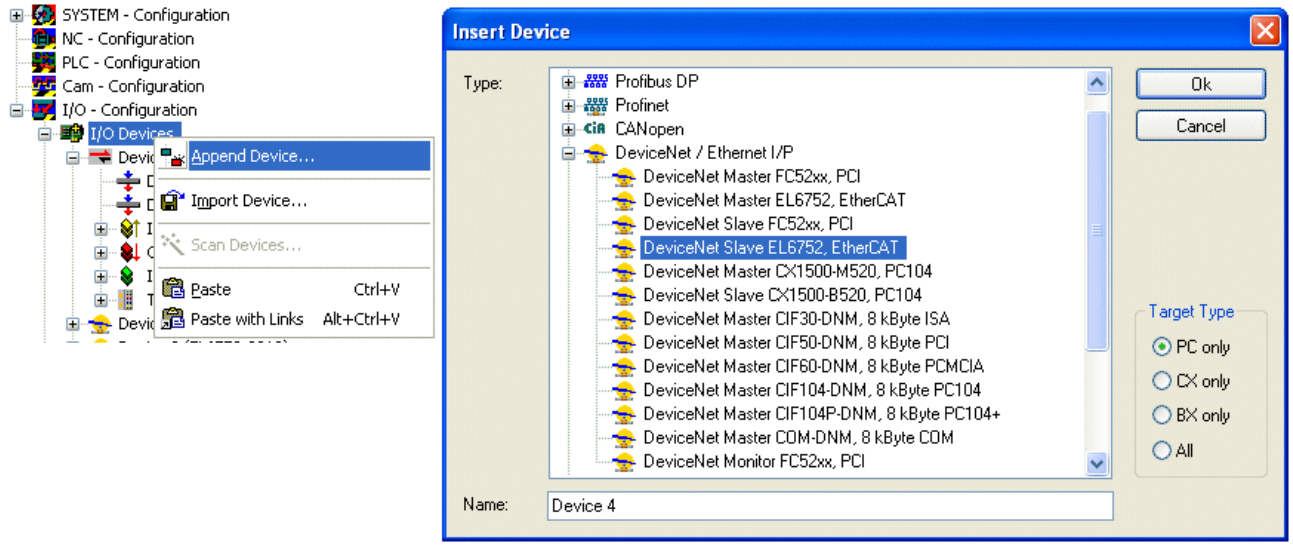


Fig. 34: Appending the device "DeviceNet slave EL6752, EtherCAT"

Right-click on "Device (EL6752-0010)" to insert the box for the EL6752-0010:

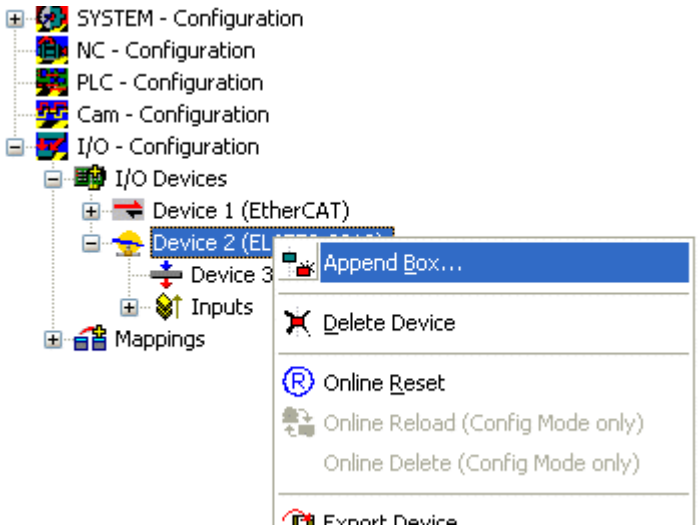


Fig. 35: Appending the box "DeviceNet slave EL6752, EtherCAT"

Selecting the I/O device for the EL6752-0010 in the tree structure opens a dialog with various configuration options:

“EL6752-0010” tab

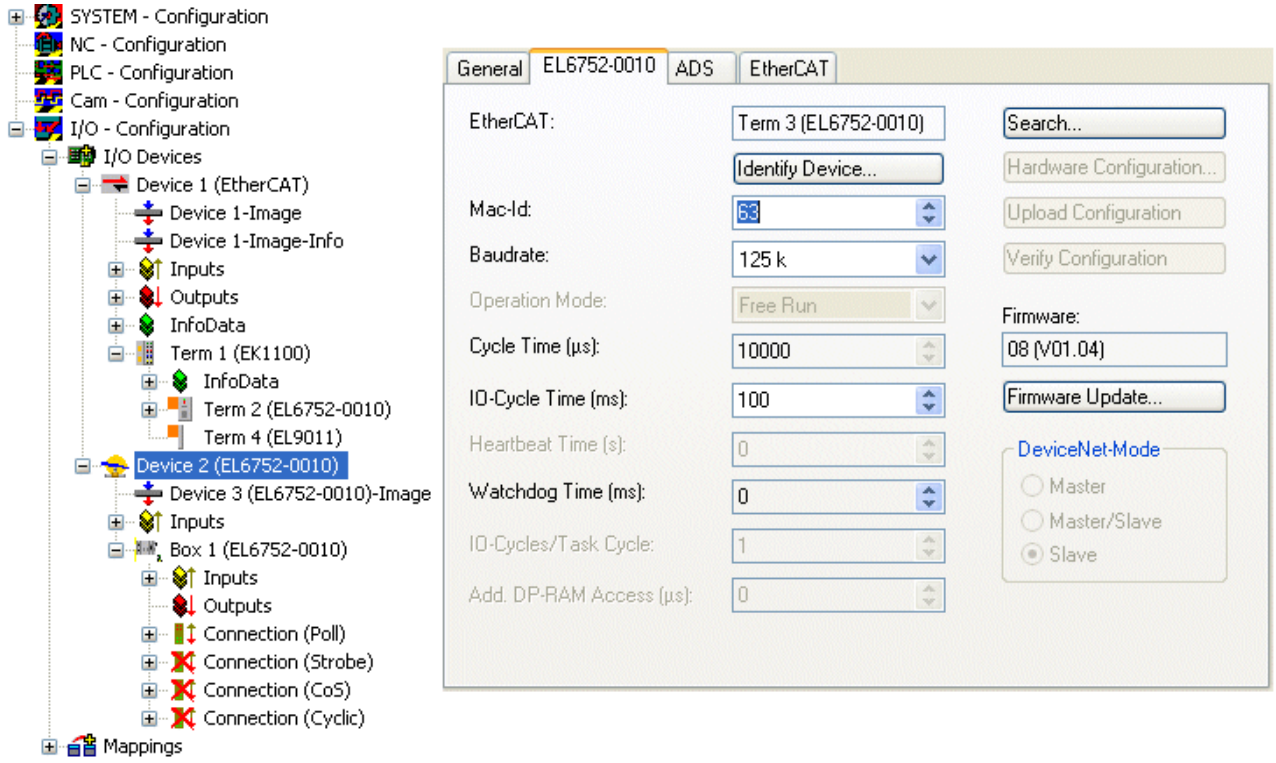


Fig. 36: “EL6752-0010” tab

EtherCAT

Terminal ID in the terminal network.

MAC-ID

Each DeviceNet® device requires a unique station number referred to as MAC ID (Medium Access Identifier) - value range: 0...63.

Baud rate

The baud rate is set here.

Cycle time

Displays the cycle time of the corresponding highest priority task. The display is updated when the mapping is generated. The network variables are updated with the cycle of this task.

Watchdog time

Time until the watchdog is triggered

Search...

Searches for all available EL6752-0010 channels, from which the required channel can be selected. In the case of an FC5102 both channels A and B appear. These behave in logical terms like two FC5101 cards.

Firmware

Displays the current EL6752-0010 firmware version.

Firmware Update...

Updates the EL6752-0010 firmware. Attention: The TwinCAT System must be stopped for this function.

“ADS” tab

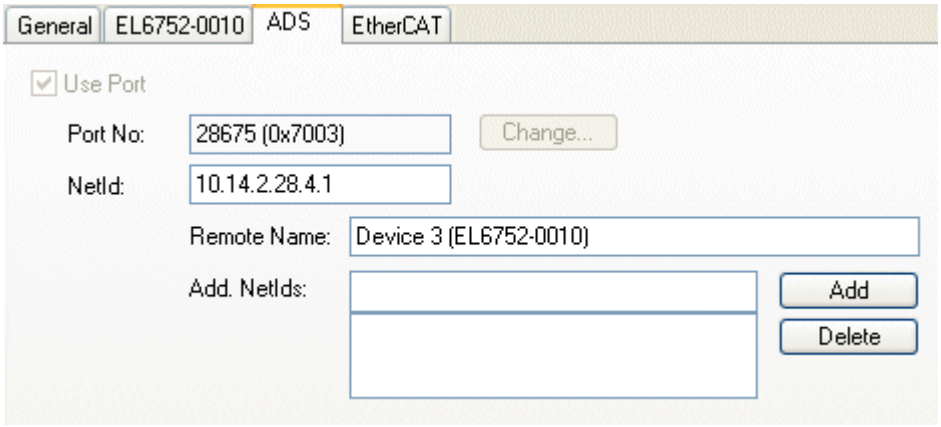


Fig. 37: “ADS” tab

The EL6752-0010 is an ADS device with its own net ID, which can be changed here. All ADS services (diagnostics, acyclic communication) associated with the EL6752-0010 device must address the card via this NetID. Additional ADS Net IDs can be entered for addressing subordinate ADS devices (e.g. an additional fieldbus card in the same PC) via the card.

"(Online) DPRAM" tab

Read access to the DPRAM of the card is provided for diagnostic purposes.

Box EL6752-0010 slave

A box "EL6752-0010 (DeviceNet slave)" is created automatically. Further parameters have to be set:

Box EL6752-0010 tab:

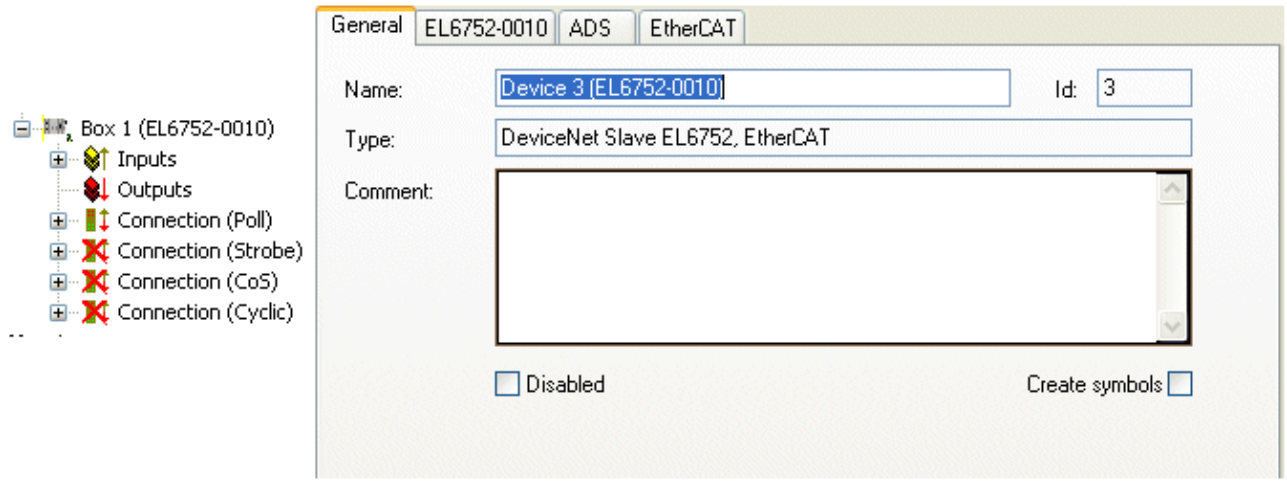


Fig. 38: "General" tab, Box EL6752-0010

DeviceNet® IO modes

The EL6752-0010 supports the DeviceNet® modes cyclic polling, change of state / cyclic and bit strobe. The IO modes can be selected according to the DeviceNet® specification.

The DeviceNet® IO mode cyclic polling is the default selection for the EL6752-0010:

IO mode	Input data length / bytes	Output data length / bytes
Polling	0 - 255	0 - 255
Change of State	0 - 255	0 - 255
Cyclic	0 - 255	0 - 255
Bit strobe	1 bit	0-8

Polling / Change of State (COS) / Cyclic

The cyclic polling mode is characterized by cyclic polling of the IO data by the master. The change of state mode is characterized by event-oriented sending of IO data. In cyclic mode the IO data are sent cyclically based on the communication parameters configured by the master. Since the communication settings are specified through the master no further settings are possible. Further information on the modes can be found in section DeviceNet® Communication. The settings are identical for these modes.

The input and output data lengths are pre-initialized to 8 byte each:

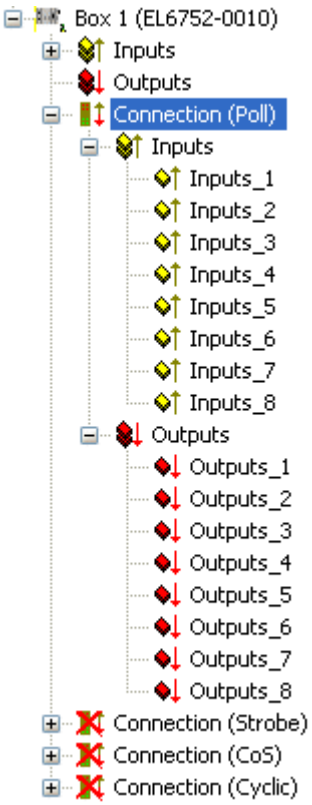


Fig. 39: Pre-initialized input and output data lengths in polling mode

According to needs and the application, further input or output data can be appended by right-clicking (Fig. Adding further variables). Any data type can be selected:

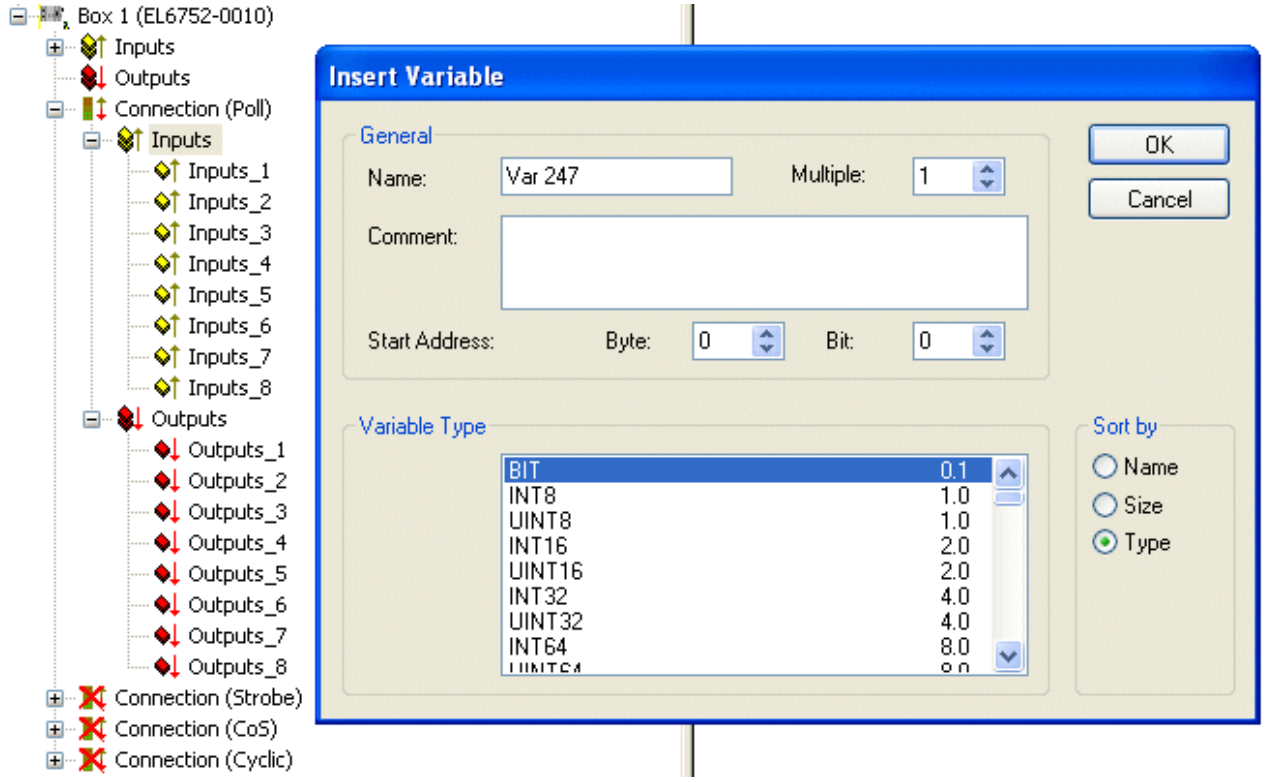


Fig. 40: Adding further variables

The data length is converted to a byte stream according to the DeviceNet® specification and displayed in the tab for the corresponding connection:

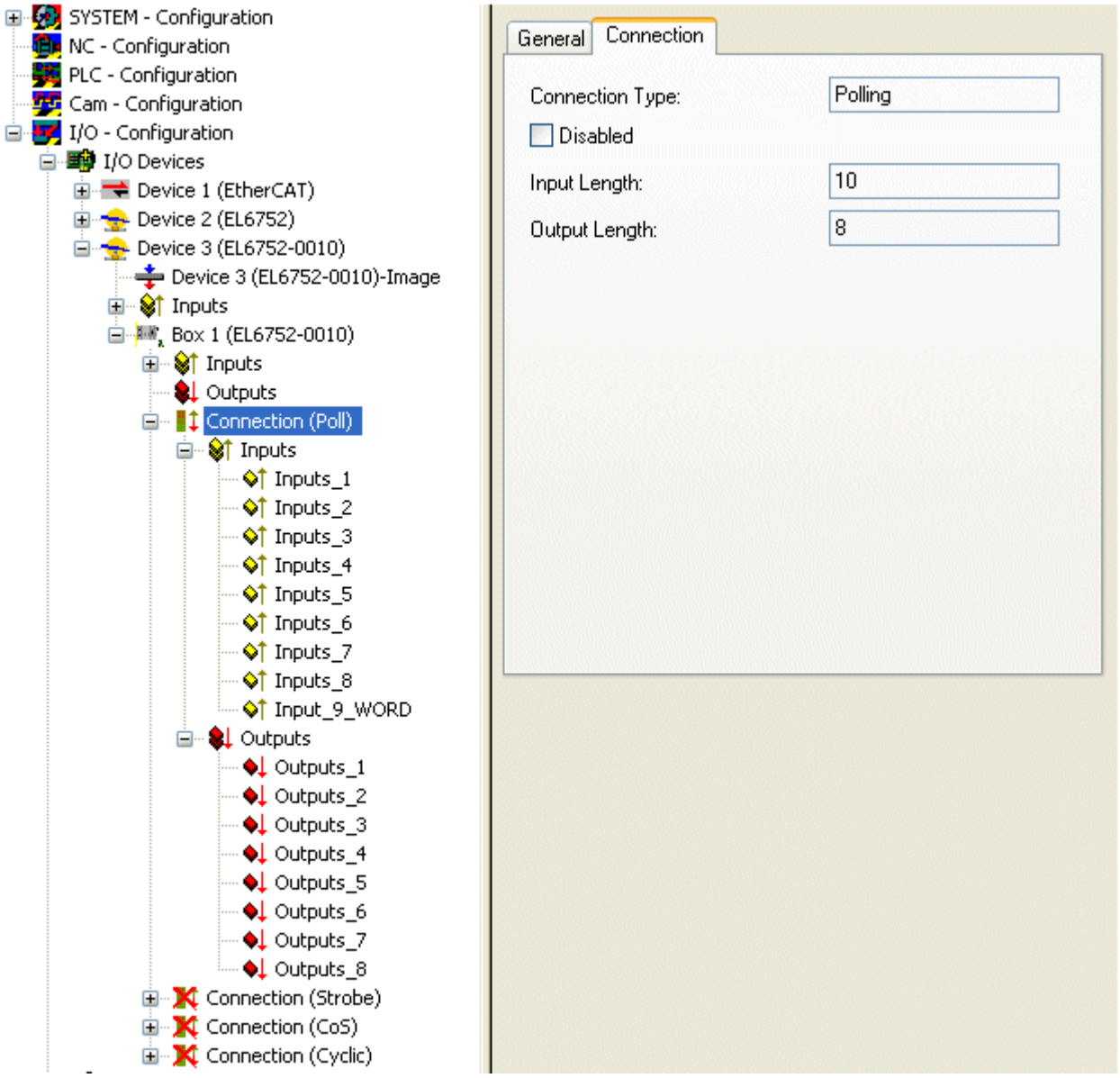


Fig. 41: "Connection" tab showing connection type "Polling" and input and output parameters

Maximum output data length

The maximum data length per data direction is 255 bytes.

The indicated input and output data lengths must be configured for the corresponding DeviceNet® master.

Bit strobe

The IO mode bit strobe involves an 8-byte command from the master to the slaves. For each possible address/MAC ID (DeviceNet address space: 64) 1 bit of user data is allocated. The maximum length of the response message from the slave is 8 bytes. It is sent to the master immediately when the bit strobe command is received.

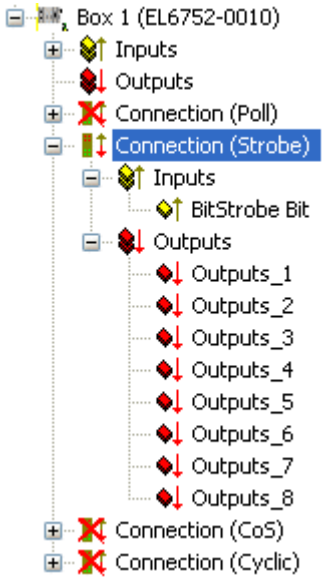


Fig. 42: Display of output parameters in the TwinCAT tree for connection type "Bit strobe"

Maximum output data length

The maximum output data length is 8 bytes. The input data length is fixed.

Since the communication settings are specified through the master no further settings are possible.

6.5 Beckhoff DeviceNet® Bus Coupler

The Bus Coupler BK52xx and the IPxxx-B520 Fieldbus Box are used in the **DeviceNet®** bus. The specific properties which distinguish them from other Bus Couplers and/or fieldbus box modules are then described below.

Types	Description
BK5210	Economy Bus Coupler
BK5220	Economy + Bus Coupler
LC5200	Low-Cost Bus Coupler
BK5250	Compact Bus Coupler
BC5250	Compact Bus Terminal Controller with 48 kbyte program memory
BX5200	BX Bus Terminal Controller with 256 kbyte program memory
IPxxxx-B520	Fieldbus compact box: DeviceNet® input/output module in protection class IP67

The following tabs are used for parameterization:

- "BK52x0" tab [▶ 59]
- "Startup Attributes" tab [▶ 61]
- "ADS" tab [▶ 62]
- "Parameters" tab [▶ 63]
- "Diag" tab [▶ 63]

"BK52x0" tab

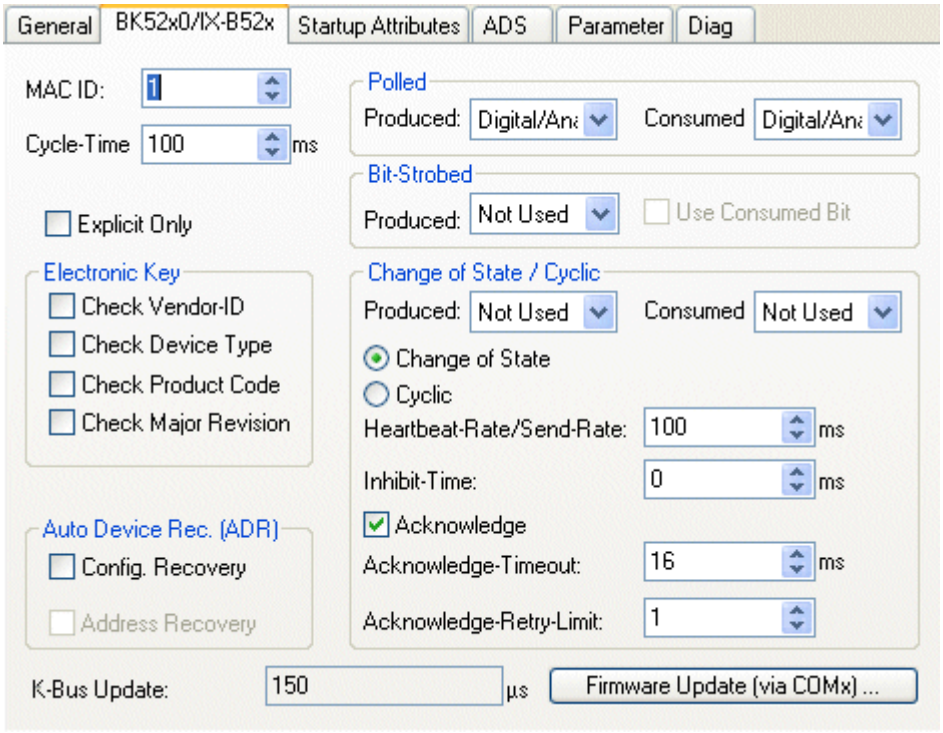


Fig. 43: "BK52x0" tab

MAC-ID

Sets the MAC ID, i.e. the device address of the DeviceNet® device (between 0 and 63). This value must comply with the value set at the Bus Coupler and/or at the compact box.

Cycle time

Sets the cycle time of the IO connection polling and bit strobe. The value is used as “Expected Packet Rate” attribute of the “Connection Object” according to the DeviceNet® specification.

Electronic Key

Serves to check the devices within the network at the system StartUp. The electronic key is read from the devices at every system StartUp and compared with the saved configuration.

Polled

Produced/Consumed
Activation of the “Polling” mode, cyclic writing and reading of IO data. Setting of the data content of the data transmitted via the polled IO connections. You can choose from digital data, analog data or both. The selection depends upon the BK52xx terminal arrangement.

Bit-Strobed

Produced/Consumed
Activation of the “Bit Strobe” operating mode. A broadcast message requests all nodes to send their bit strobe message (up to 7 bytes input or status data). Setting of the data content of the data transmitted via the bit-strobed IO connections. You can choose between digital data or diagnostic data.

Change of State / Cyclic

Produced/Consumed
Setting of the data content of the data transmitted via the change of state/cyclical IO connections. You can choose from digital data, analog data or both. The selection depends upon the BK52xx terminal arrangement.

Change of State / Cyclic
Selection of the required operating mode.

Heartbeat-Rate / Send-Rate
In the “Change of State” mode the heartbeat rate gives the cycle time of the cyclical send of the lower-level (i.e. in addition to the event driven) IO data. In “Cyclic” mode the send rate specifies the cycle time with which the IO data are sent.

Inhibit-Time
Delay time in “Change of State” mode; after a change of state IO data are sent after the specified time at the earliest.

Acknowledge Timeout
Time before the retransmission in the event of faulty acknowledgement of a change of state / cyclical message.

Acknowldege Retry Limit
Maximum number of retransmissions until IO connection goes into error mode.

K-Bus update

Calculates the expected time required for a full update of the terminal bus (depends on the connected terminals).

Auto Device Replacement (ADR)

Not supported.

“Startup Attributes” tab

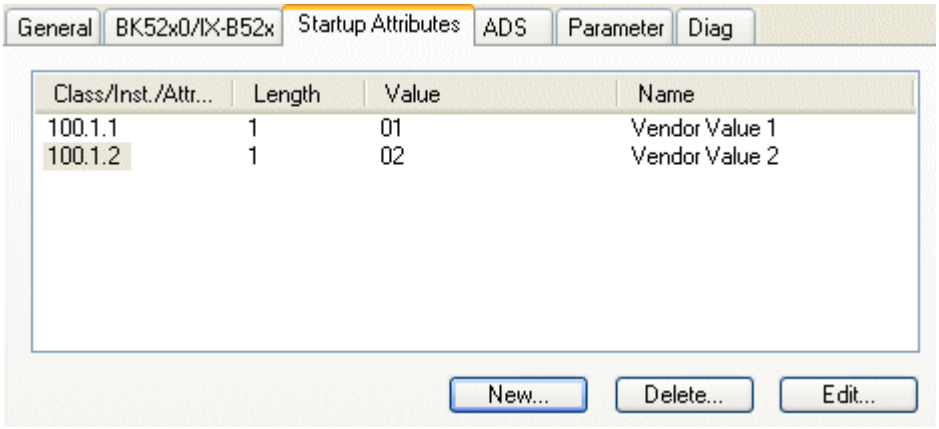


Fig. 44: “Startup Attributes” tab

The startup attributes are sent to the slave before the cyclic data exchange. The messages are sent before the actual IO data traffic.

Use the “New” or “Edit” button for configuration:

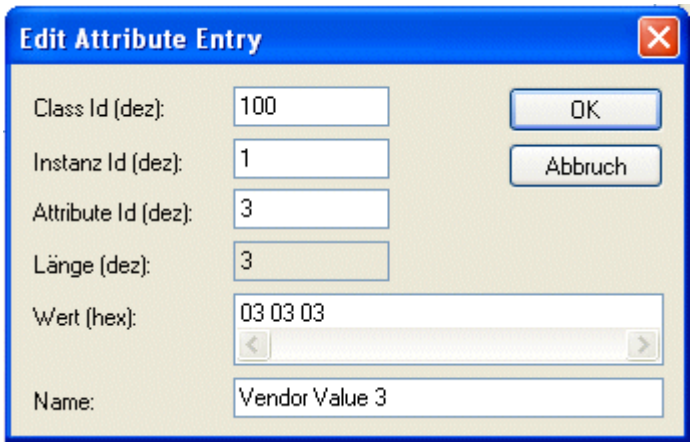


Fig. 45: Edit an attribute entry

The attributes are initialized via Class/Instance/Attributes. Note the “Value” specification in hexadecimal form.

“ADS” tab

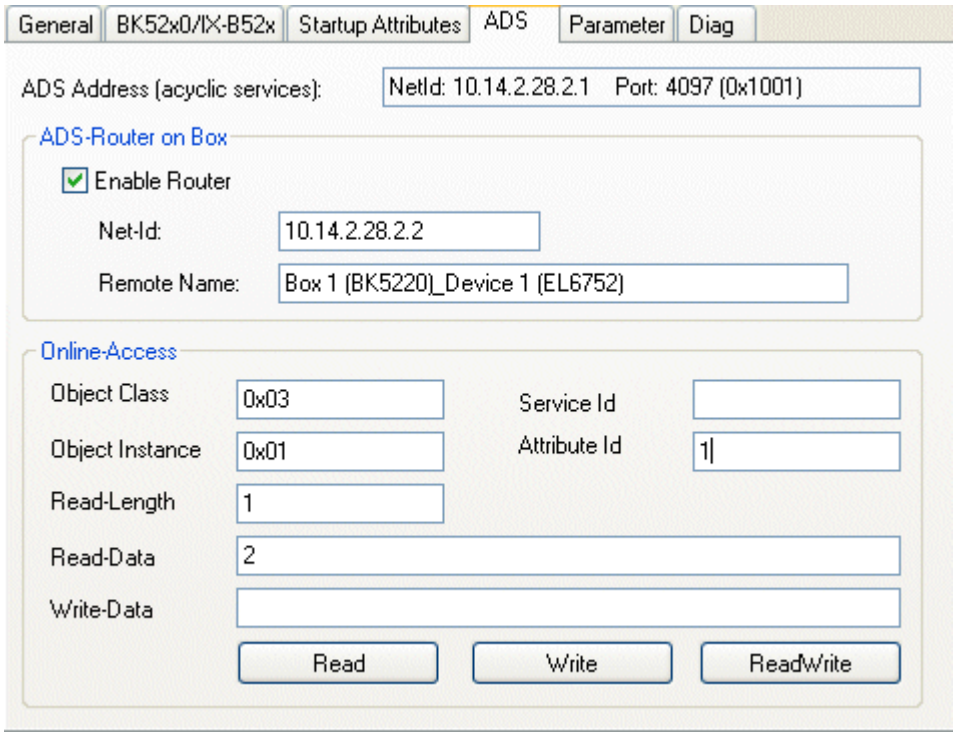


Fig. 46: “ADS” tab

The node (Bus Coupler) is assigned an ADS port to enable writing and reading of DeviceNet® objects at runtime (e.g. from the PLC). It can be changed if required. A detailed description of explicit messages can be found in section “DeviceNet Communication” under “Explicit Messages”.

DeviceNet® objects can be accessed via Online Access. To this end the DeviceNet®-specific information such as Class/Instance/Attributes has to be entered.

Read
Reading of an object attribute via DeviceNet “Get_Attribute_Single” service. A service ID is not required.

Write
Writing of an object attribute via DeviceNet “Set_Attribute_Single” service. A service ID is not required.

Read/Write
Executing any DeviceNet® service. Specification of the service ID is required.

“Parameter” tab

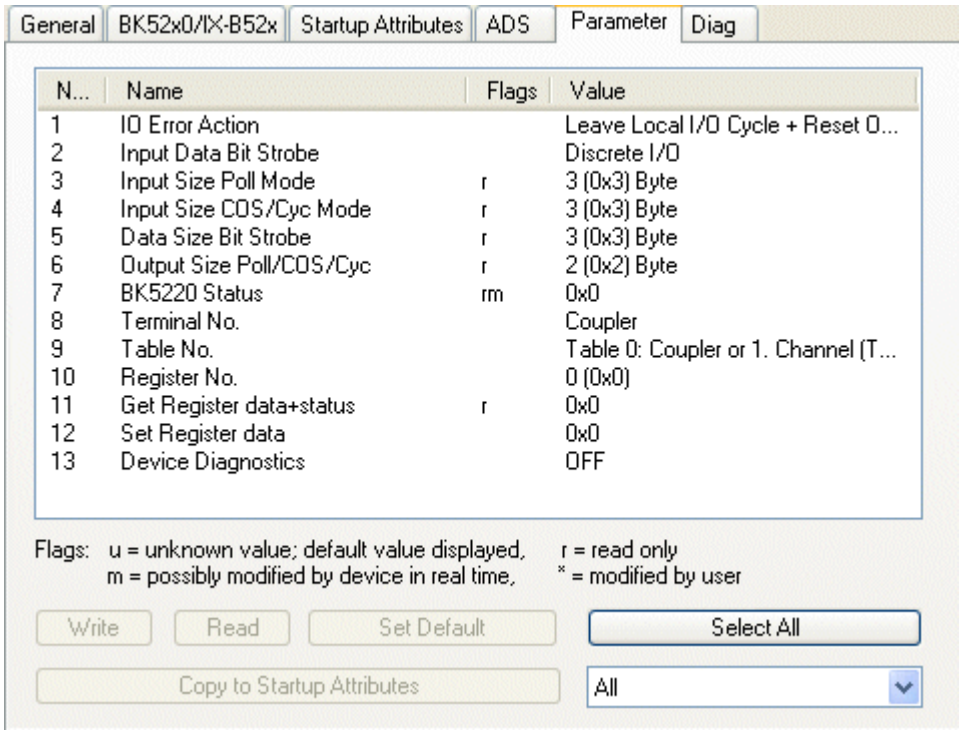


Fig. 47: “Parameter” tab

The parameters read from the EDS file are shown under the “Parameters” tab. Parameters can be read, written and entered in the list of the startup parameters.

“Diag” tab

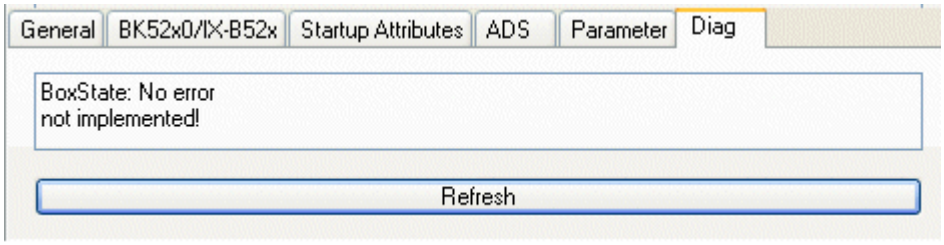


Fig. 48: "Diag" tab

The “Diag” tab indicates the state of the box. No further diagnostic options are available.

Also see about this

- Explicit messages [▶ 37]

6.6 General DeviceNet® device

DeviceNet® devices are integrated as general DeviceNet® devices.

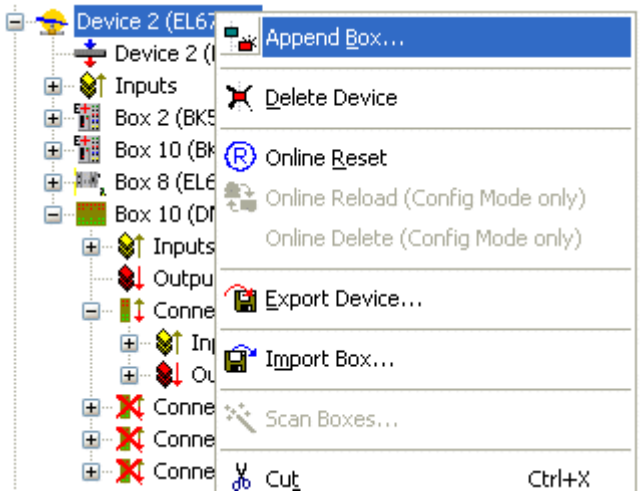


Fig. 49: Adding a DeviceNet® device (I/O Devices-> Device n (EL6752)->right-click-> Append Box...)

6.6.1 Integrating a DeviceNet® device with EDS file

If an EDS file is available for the DeviceNet® to be integrated, it must be copied into the **..TwinCAT/IO/DeviceNet®** directory.

Subsequently the device appears under the "Append Box" selection (see fig. *Adding a DeviceNet® device (I/O Devices -> Device n (EL6752) -> right-click -> Append Box ...)* with the manufacturer ID:

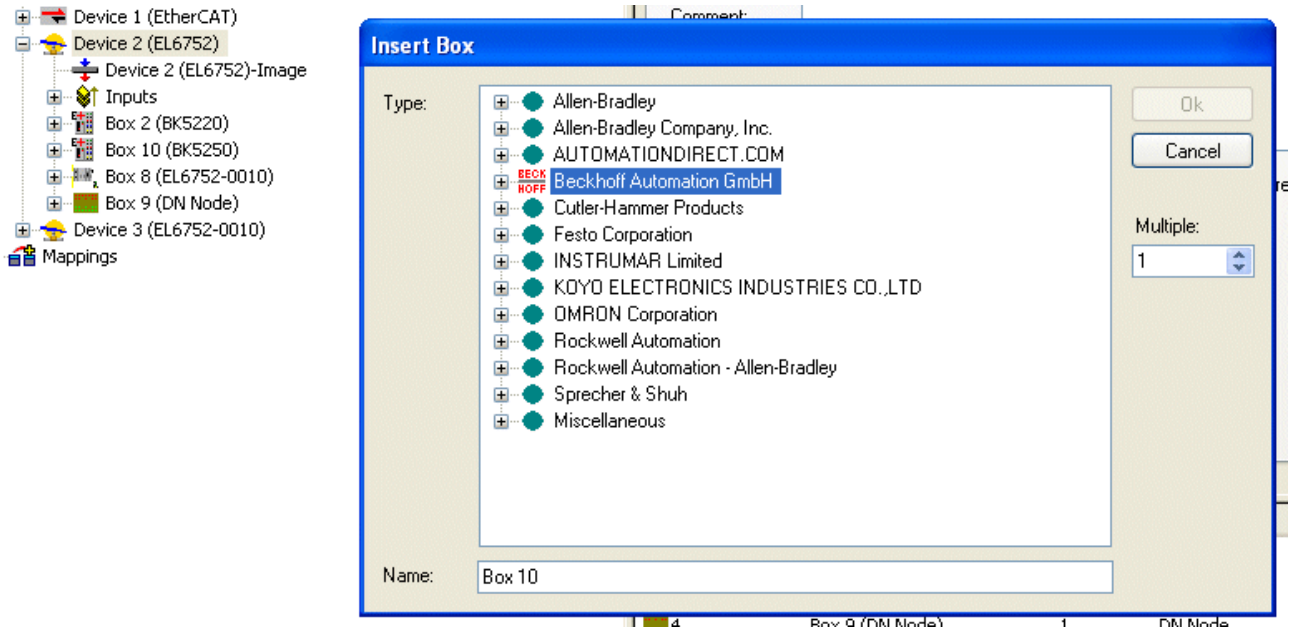


Fig. 50: Adding a box with the manufacturer ID

Alternatively a DeviceNet® device with EDS file can be integrated via the "Miscellaneous" option:

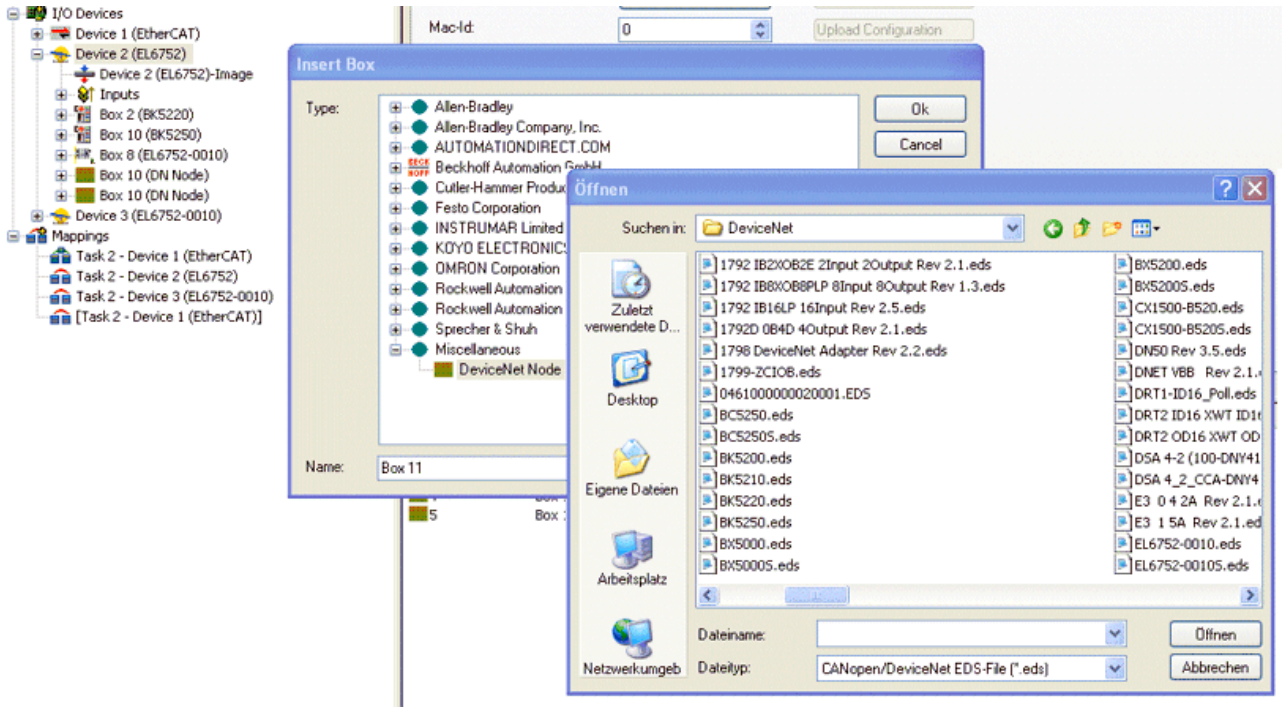


Fig. 51: Adding a box without EDS file

Depending on the information contained in the EDS file a DeviceNet® node will appear with or without Parameters tab.

The IO mode and the corresponding data lengths are specified in the EDS file.

6.6.2 Integrating a DeviceNet® device without EDS file

A DeviceNet® device without EDS file can be integrated via the "Miscellaneous" option:

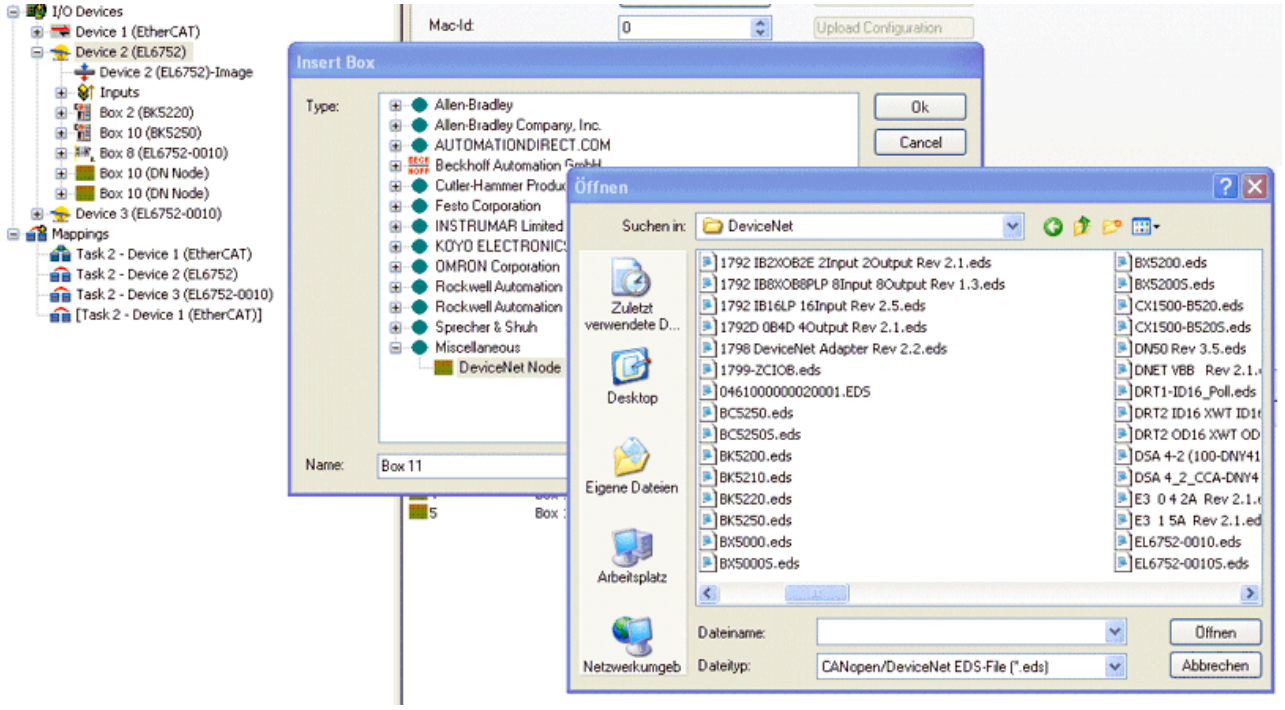


Fig. 52: Adding a box without EDS file (click "Cancel")

Terminate EDS file selection with "Cancel". A general DeviceNet® device is created.

Selection of the IO mode and the general configuration must then be carried out manually.

DeviceNet® IO modes

For DeviceNet® devices the EL6752 supports the DeviceNet® modes cyclic polling, change of state / cyclic and bit strobe. The IO modes can be selected according to the DeviceNet® specification.

The DeviceNet® IO mode cyclic polling is the default selection for the EL6752:

IO mode	Input data length / bytes	Output data length / bytes
Polling	0 - 255	0 - 255
Change of State	0 - 255	0 - 255
Cyclic	0 - 255	0 - 255
Bit strobe	1 bit	0-8
Total of all IO data	max. xxx bytes	max. xxx bytes

polling / change of state (COS) / cyclic

The cyclic polling mode is characterized by cyclic polling of the IO data by the master. The change of state mode is characterized by event-oriented sending of IO data. In cyclic mode the IO data are sent cyclically based on the communication parameters configured by the master. The settings are identical for these modes.

The input and output data lengths must be supplemented according to the device configuration:

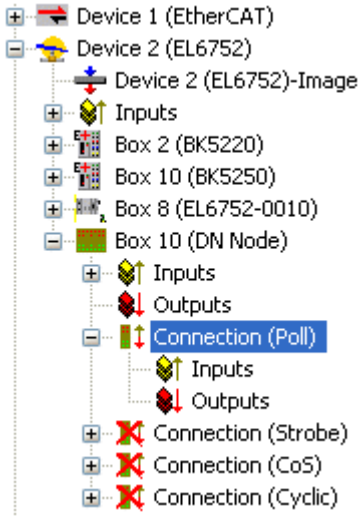


Fig. 53: Supplementing the input and output data

Input or output data must be appended depending on the device configuration. Any data type can be selected:

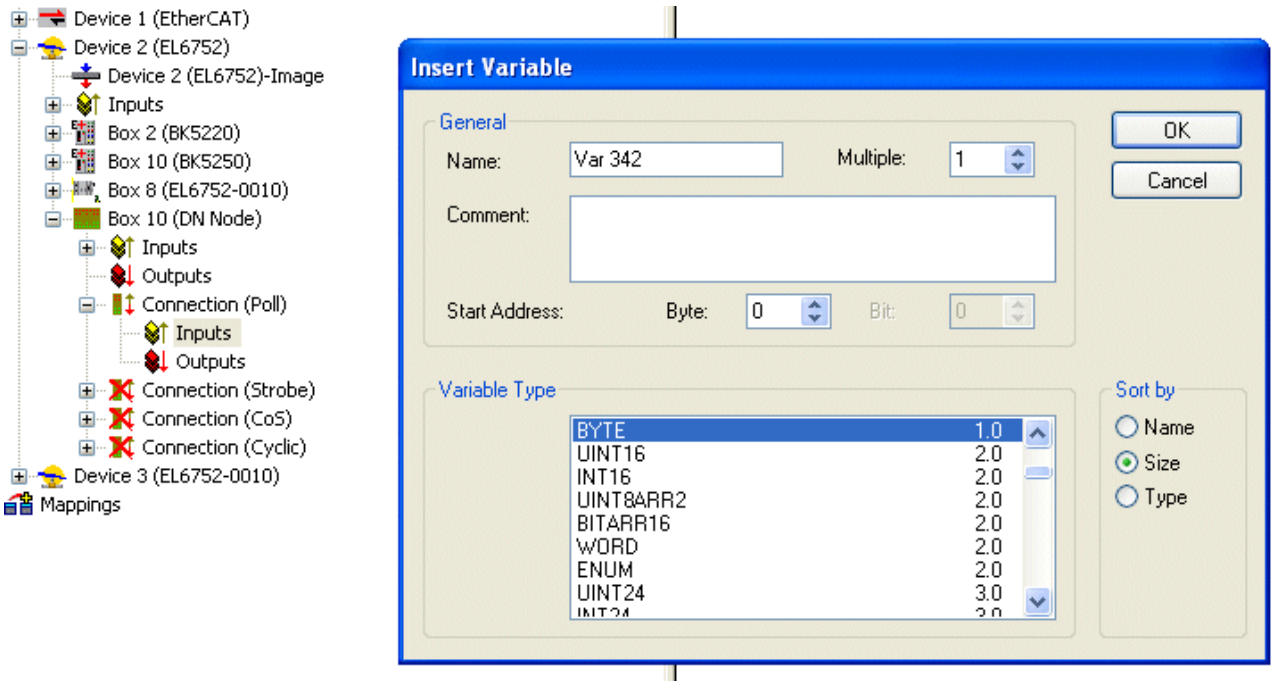


Fig. 54: Add Variables

The data length is converted to a byte stream according to the DeviceNet® specification and displayed in the tab for the corresponding connection:

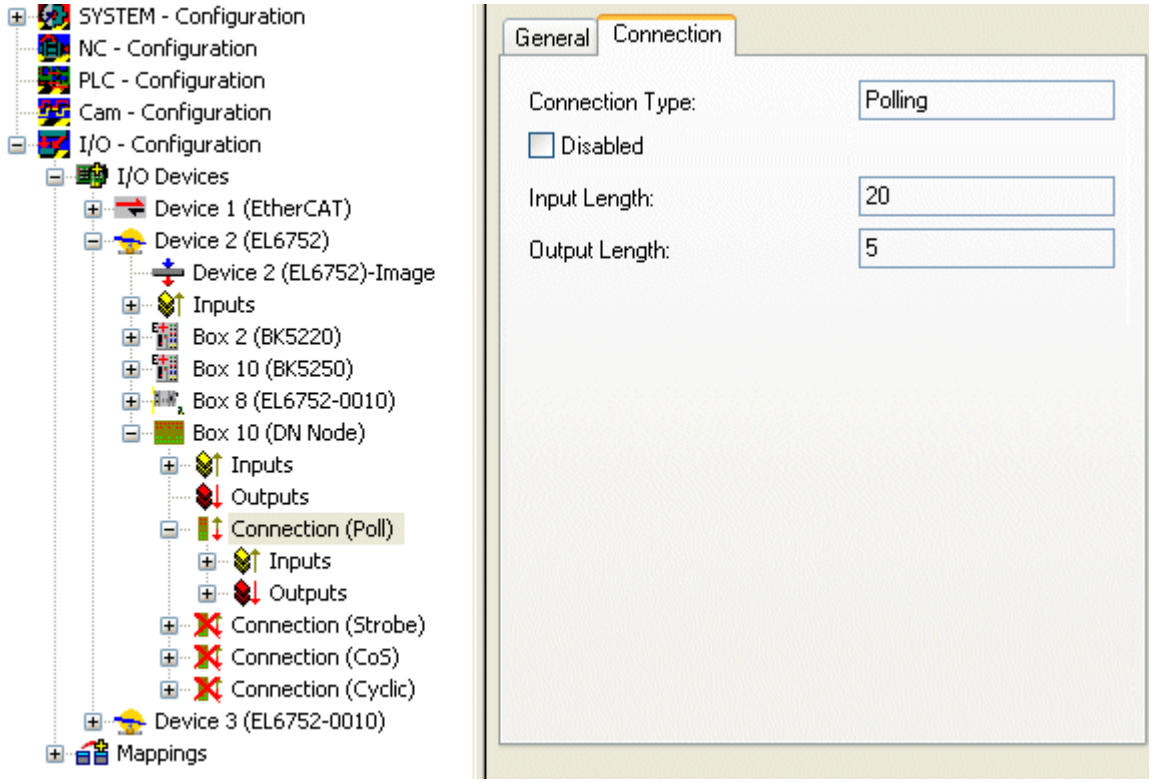


Fig. 55: "Connection" tab showing connection type "Polling" and input and output parameters

Maximum data length

The maximum data length per data direction is 255 bytes.

Bit strobe

The IO mode bit strobe involves an 8-byte command from the master to the slaves. For each possible address/MAC ID (DeviceNet address space: 64) 1 bit of user data is allocated. The maximum length of the response message from the slave is 8 bytes. It is sent to the master immediately when the bit strobe command is received.

After selection of the bit strobe mode the input data must be configured accordingly. Any data type can be selected (see polling/ COS / cyclic). The data length is converted to a byte stream according to the DeviceNet® specification and displayed in the tab for the bit strobe connection:

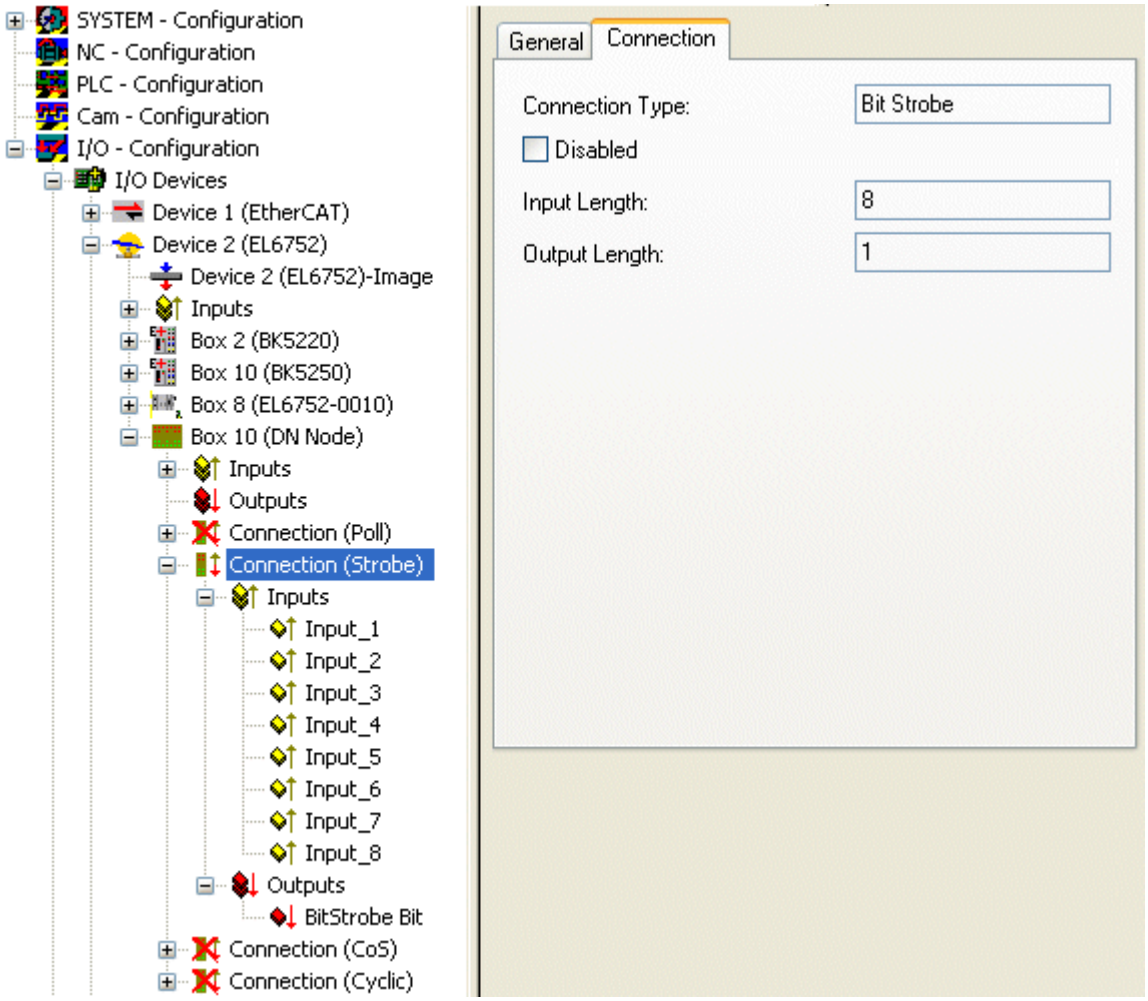


Fig. 56: "Connection" tab showing connection type "Bit Strobe" and input and output parameters

Maximum data length

The maximum input data length is 8 bytes. The output data length is fixed.

Since the communication settings are specified through the master no further settings are possible.

6.6.3 Parameterization of a DeviceNet® device

The DeviceNet devices are parameterized with the following tabs:

- "DeviceNet Node" tab [▶ 69]
- "Startup Attributes" tab [▶ 70]
- "ADS" tab [▶ 71]
- "Parameter" tab [▶ 72]
- "Diag" tab [▶ 72]

"DeviceNet Node" tab

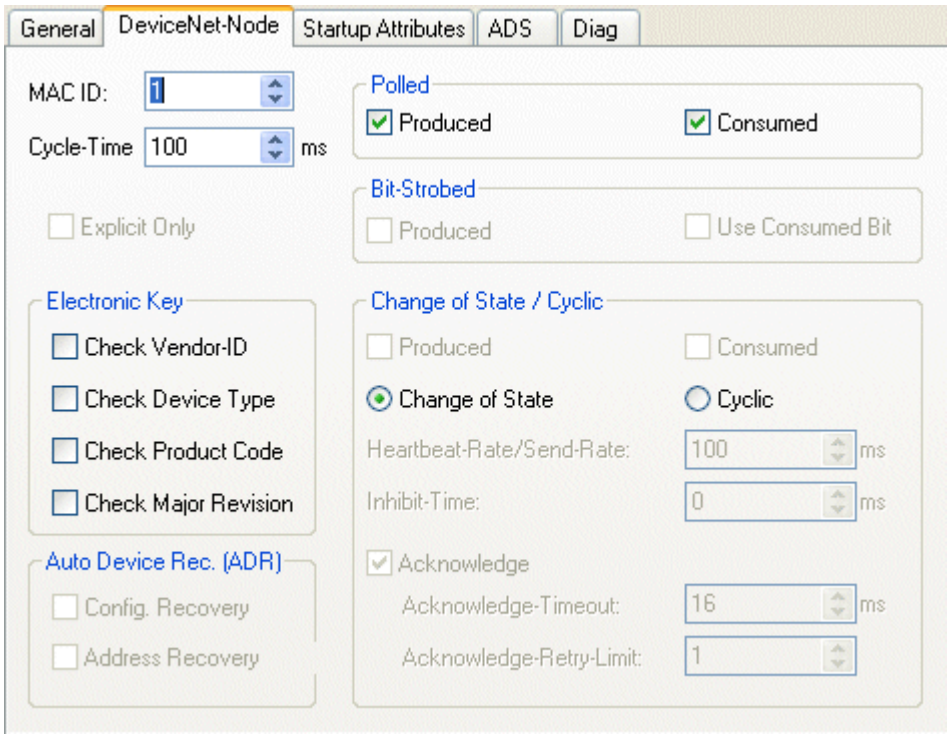


Fig. 57: "DeviceNet Node" tab

MAC-ID

Sets the MAC ID, i.e. the device address of the DeviceNet® device (between 0 and 63). This value must comply with the value set at the Bus Coupler and/or at the compact box.

Cycle time

Sets the cycle time of the IO connection polling and bit strobe. The value is used as “Expected Packet Rate” attribute of the “Connection Objects” according to the DeviceNet® specification.

Electronic Key

Serves to check the devices within the network at the system StartUp. The electronic key is read from the devices at every system StartUp and compared with the saved configuration.

Polled

Produced/Consumed
Activation of the “Polling” mode, cyclic writing and reading of IO data. Setting of the data content of the data transmitted via the polled IO connections. You can choose from digital data, analog data or both. The selection depends upon the BK52xx terminal arrangement.

Bit-Strobed

Produced/Consumed
Activation of the “Bit Strobe” operating mode. A broadcast message requests all nodes to send their bit strobe message (up to 7 bytes input or status data). Setting of the data content of the data transmitted via the bit-strobed IO connections. You can choose between digital data or diagnostic data.

Change of State / Cyclic

Produced/Consumed
Setting of the data content of the data transmitted via the change of state/cyclical IO connections. You can choose from digital data, analog data or both. The selection depends upon the BK52xx terminal arrangement.

Change of State / Cyclic
Selection of the required operating mode.

Heartbeat-Rate / Send-Rate
In the “Change of State” mode the heartbeat rate gives the cycle time of the cyclical send of the lower-level (i.e. in addition to the event driven) IO data. In “Cyclic” mode the send rate specifies the cycle time with which the IO data are sent.

Inhibit Time
Delay time in “Change of State” mode; after a change of state IO data are sent after the specified time at the earliest.

Acknowledge Timeout
Time before the retransmission in the event of faulty acknowledgement of a change of state / cyclical message.

Acknowledge Retry Limit
Maximum number of re-sends until IO connection goes into error mode.

K-Bus update

Calculates the expected time required for a full update of the terminal bus (depends on the connected terminals).

Auto Device Replacement (ADR)

Not supported.

“Startup Attributes” tab

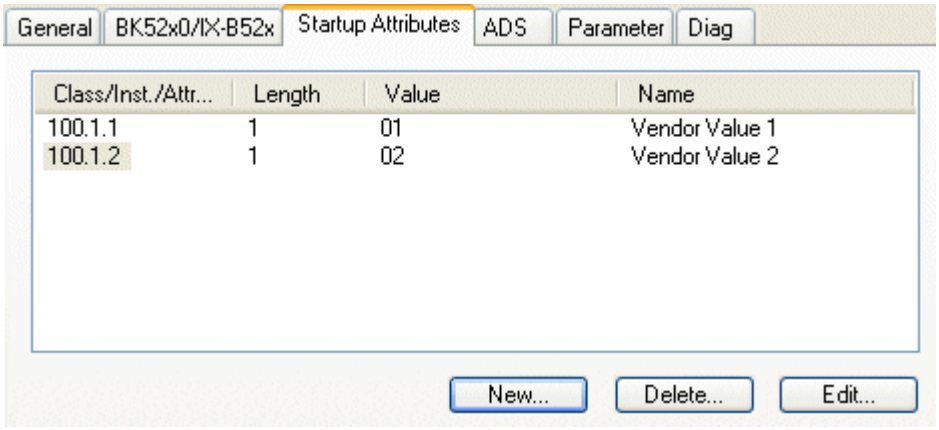


Fig. 58: “Startup Attributes” tab

The startup attributes are sent to the slave before the cyclic data exchange. The messages are sent before the actual IO data traffic.

Use the “New” or “Edit” button for configuration:

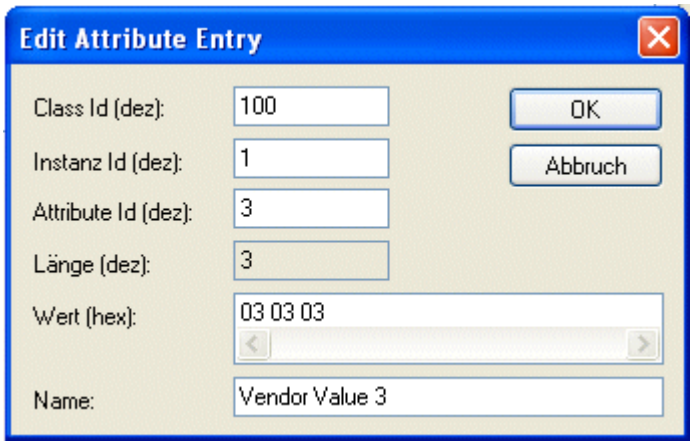


Fig. 59: Edit an attribute entry

The attributes are initialized via Class/Instance/Attributes. Note the “Value” specification in hexadecimal form.

“ADS” tab

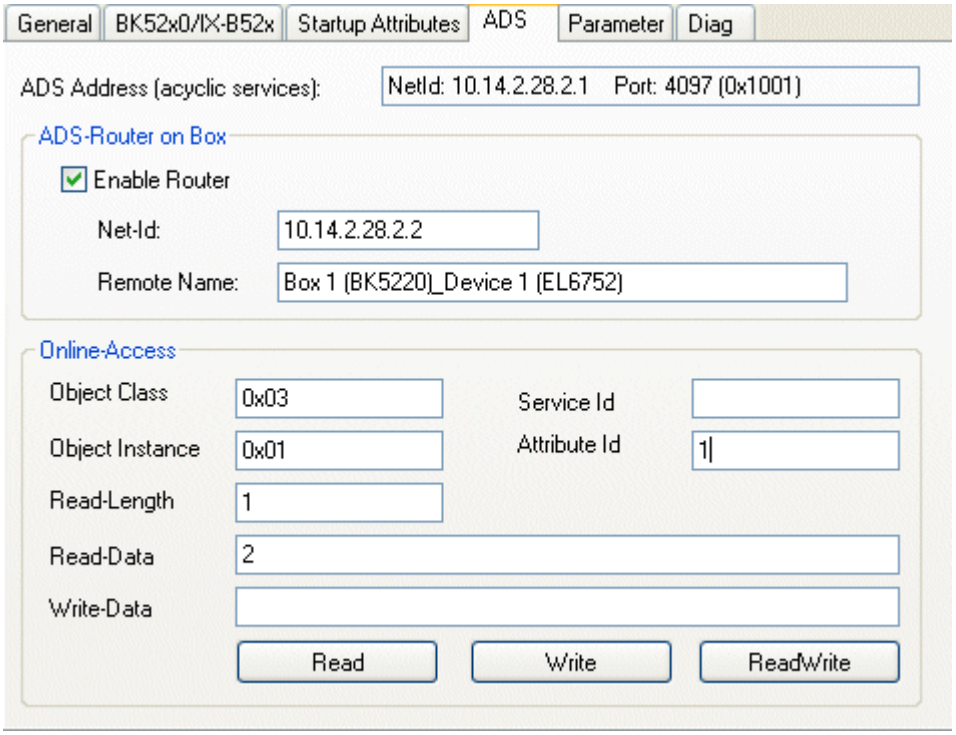


Fig. 60: “ADS” tab

The node (Bus Coupler) is assigned an ADS port to enable writing and reading of DeviceNet® objects at runtime (e.g. from the PLC). It can be changed if required. A detailed description of explicit messages can be found in section “DeviceNet Communication” under “Explicit Messages”.

DeviceNet® objects can be accessed via Online Access. To this end the DeviceNet-specific information such as Class/Instance/Attributes has to be entered.

Read
Reading of an object attribute via DeviceNet “Get_Attribute_Single” service. A service ID is not required.

Write
Writing of an object attribute via DeviceNet “Set_Attribute_Single” service. A service ID is not required.

Read / Write
Executing any DeviceNet® service. Specification of the service ID is required.

“Parameter” tab

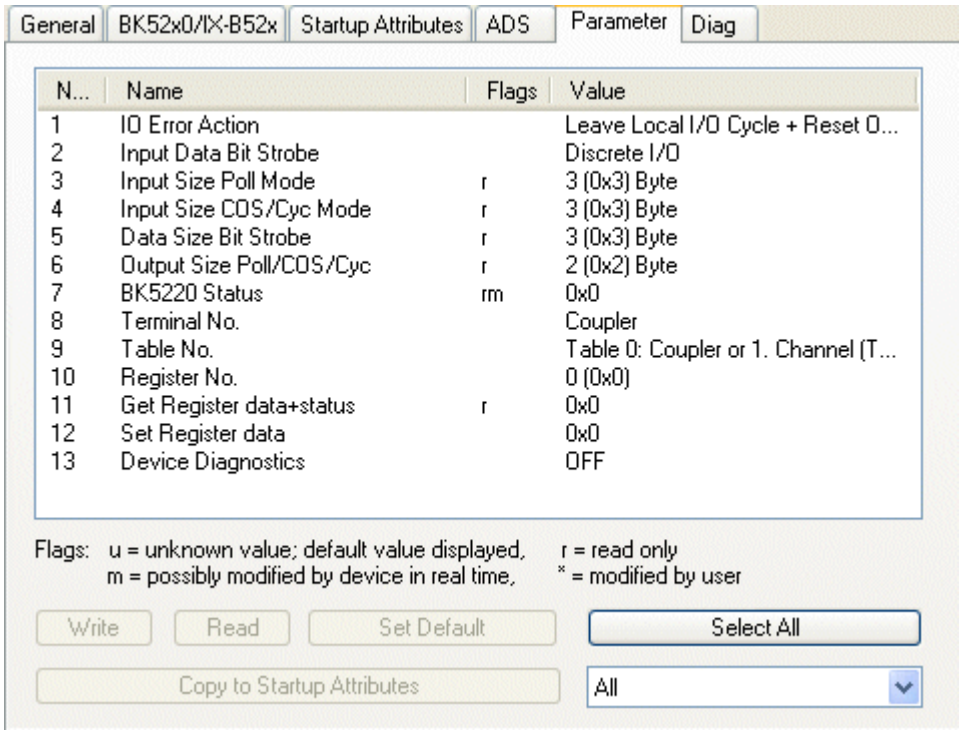


Fig. 61: “Parameter” tab

The parameters read from the EDS file are shown under the “Parameters” tab. Parameters can be read, written and entered in the list of the startup parameters.

“Diag” tab

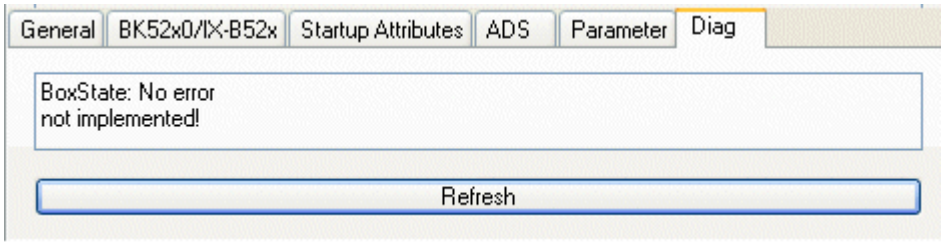


Fig. 62: "Diag" tab

The “Diag” tab indicates the state of the box. No further diagnostic options are available.

6.7 EtherCAT description

6.7.1 Introduction

The DeviceNet® functionality and configuration options can be changed and parameterized depending on the different EtherCAT states.

EtherCAT states

The EtherCAT states (INIT, PREOP, SAFEOP, OP) have the following meaning according to the fieldbus-specific functions:

EtherCAT state	Meaning
INIT	Fieldbus not running
PREOP	Load fieldbus configuration
SAFEOP	Fieldbus cyclic operation, safe state. Inputs are read, outputs are not written
OP	Fieldbus cyclic operation. Inputs are read, outputs are written

The procedure and the configuration options are described below

6.7.1.1 EL6752 DeviceNet® master configuration

The DeviceNet® master and the associated DeviceNet® slaves are configured in EtherCAT state PREOP. The DeviceNet® master parameters are written via the EtherCAT object 0xF800, the slave parameters are written via the EtherCAT objects from 0x80n0 [▶ 80], see section EtherCAT Object Description.

The EtherCAT states are mapped to DeviceNet® as follows:

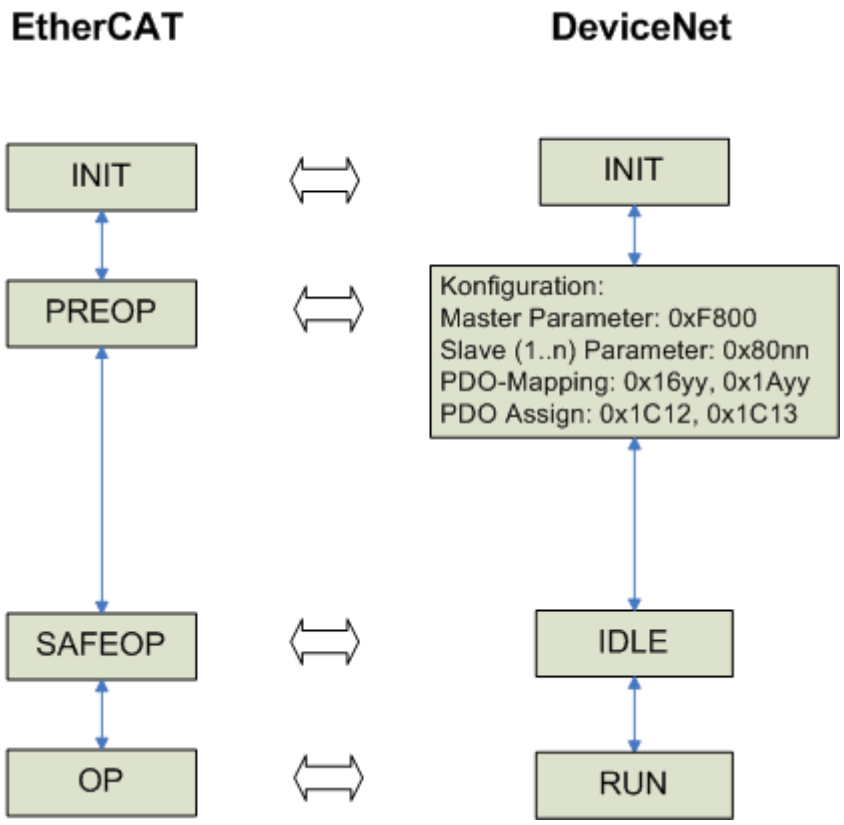


Fig. 63: EtherCAT states in mapping on EL6752-0000

The EtherCAT PDO mapping (EtherCAT objects 0x16yy, 0x1Ayy) and the PDO assignment (EtherCAT objects 0x1C12, 0x1C13) can be read once the DeviceNet® master parameters and the DeviceNet® slave parameters have been written. The associated process image is then generated.

Once the DeviceNet® master parameters have been written via the EtherCAT object 0xF800 [▶ 93], the DeviceNet® master registers itself in the network and carries out the Duplicate MAC ID check.

Starting the fieldbus

During the EtherCAT state transition from PREOP to SAFEOP the DeviceNet® master starts the data communication with the slaves and allocates the configured operating modes. In EtherCAT state SAFEOP the DeviceNet® master is in IDLE mode. During the EtherCAT state transition from SAFEOP to OP the DeviceNet® master switches to RUN mode.

Loading a new configuration

A new DeviceNet® configuration can only be loaded through an EtherCAT state transition to IDLE or PREOP. The DeviceNet® master parameters and DeviceNet® slave parameters then have to be written again.

6.7.1.2 EL6752-0010 DeviceNet® slave configuration

The DeviceNet® slaves are configured in EtherCAT state PREOP. The general DeviceNet® slave parameters are written via the EtherCAT object 0xF800, the slave configuration data, i.e. the communication features and the IO configuration are written via the EtherCAT object 0x8000 [▶ 96], see section EtherCAT Object Description.

The EtherCAT states are mapped to DeviceNet® as follows:

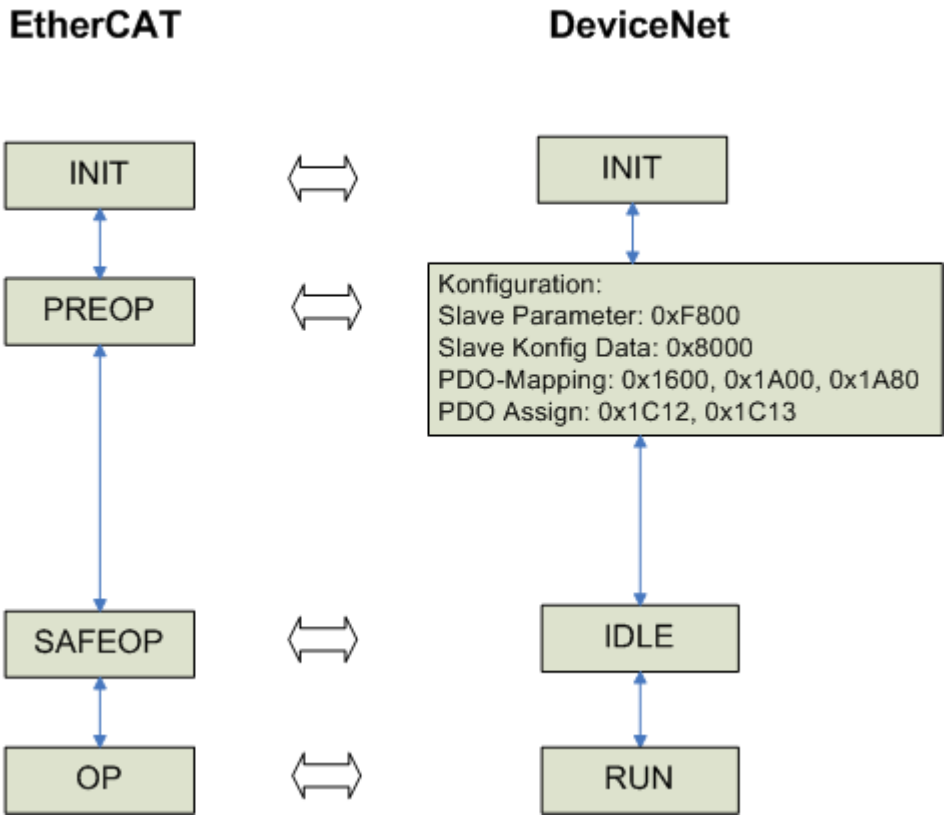


Fig. 64: EtherCAT states in mapping on EL6752-0010

The EtherCAT PDO mapping (EtherCAT objects 0x1600, 0x1A00, 0x1A80) and the PDO assignment (EtherCAT objects 0x1C12 [▶ 99], 0x1C13 [▶ 99]) can be read once the DeviceNet® slave parameters and the DeviceNet® slave configuration data have been written. The associated process image is then generated.

Once the DeviceNet® slave parameters have been written via the EtherCAT object 0xF800, the DeviceNet® slave registers itself in the network and carries out the Duplicate MAC ID check.

Starting the fieldbus

During the EtherCAT state transition from PREOP to SAFEOP the DeviceNet® slave starts the data communication, i.e. it is now ready for communication with a DeviceNet® master. In EtherCAT state SAFEOP the DeviceNet® slave is in IDLE mode. During the EtherCAT state transition from SAFEOP to OP the DeviceNet® slave switches to RUN mode.

Loading a new configuration

A new DeviceNet® configuration can only be loaded through an EtherCAT state transition to IDLE or PREOP. The DeviceNet® slave parameters and DeviceNet® slave configuration data then have to be written again.

6.7.1.3 EL6752-0010 - Changing the DeviceNet® address and baud rate using ADS

The DeviceNet address (MACId) and the baud rate of the EL6752-0010 DeviceNet Slave terminal can be set using an ADS command in addition to the familiar functions as already described in the chapter "Configuration with the TwinCAT System Manager [▶ 53]"

ADS command

Setting the MAC-ID and the baud rate using ADS

```
IDXGRP=0x1F480
Index Offset 0x00

LEN=6

DATA[0]=0x45
DATA[1]=0x23
DATA[2]=MACId (0 _ 63)
DATA[3]=0
DATA[4]=Baudrate (1=500k, 2=250k, 3=125k)
DATA[5]=0

Ams Net Id: die der EL6752
Ams Port: 200
```

After writing the command the terminal must be switched once to INIT and then back to OP. The set data can be read in the object 0xF800 Index 1 (MAC ID) and Index 2 (baud rate).

Command taking the example of the TwinCAT AMS ADS Viewer

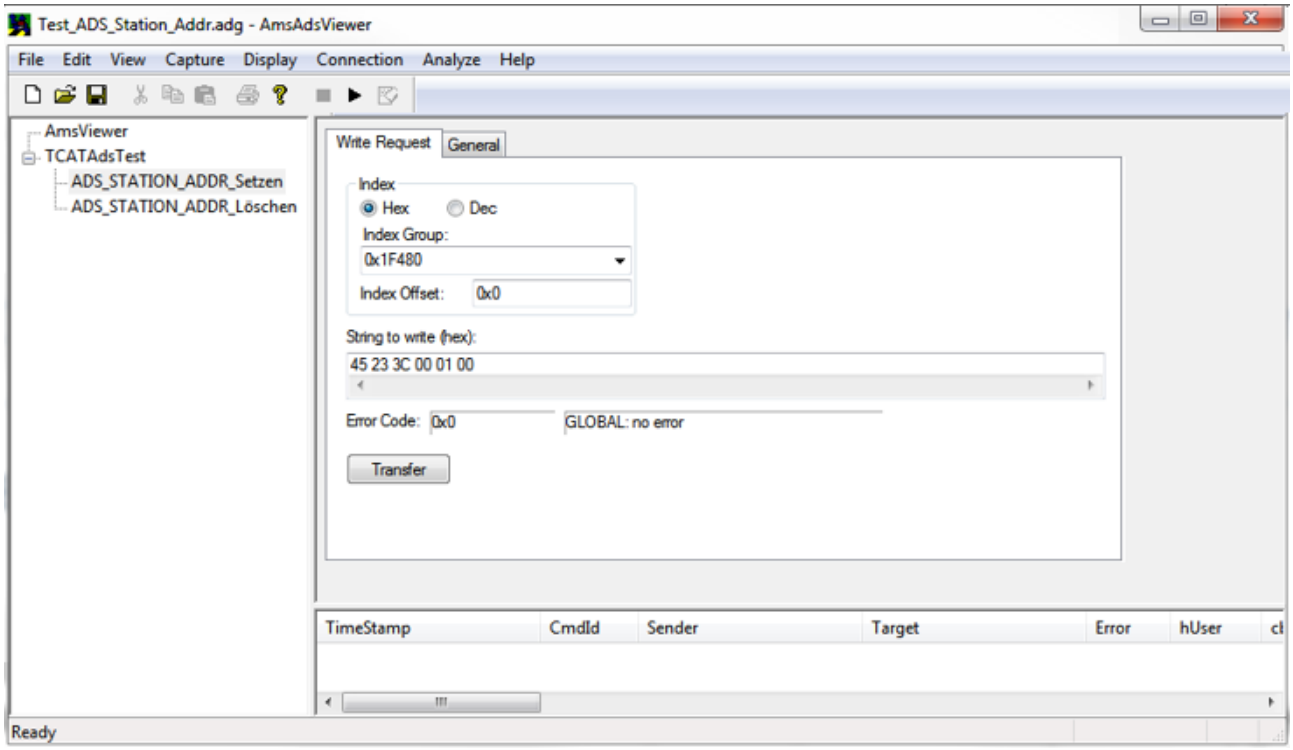


Fig. 65: ADS command with the data 3C - MACId (60dec) and 01 - baud rate (500k)

Reset

Once the MAC ID and the baud rate have been set using the ADS command, the terminal stores the information persistently. Once these data have been written, the entries in the objects 0x8000:01, 0xF800:01 and 0xF800:02 are ignored! ! This concerns the start-up commands, which are then ignored by the terminal.

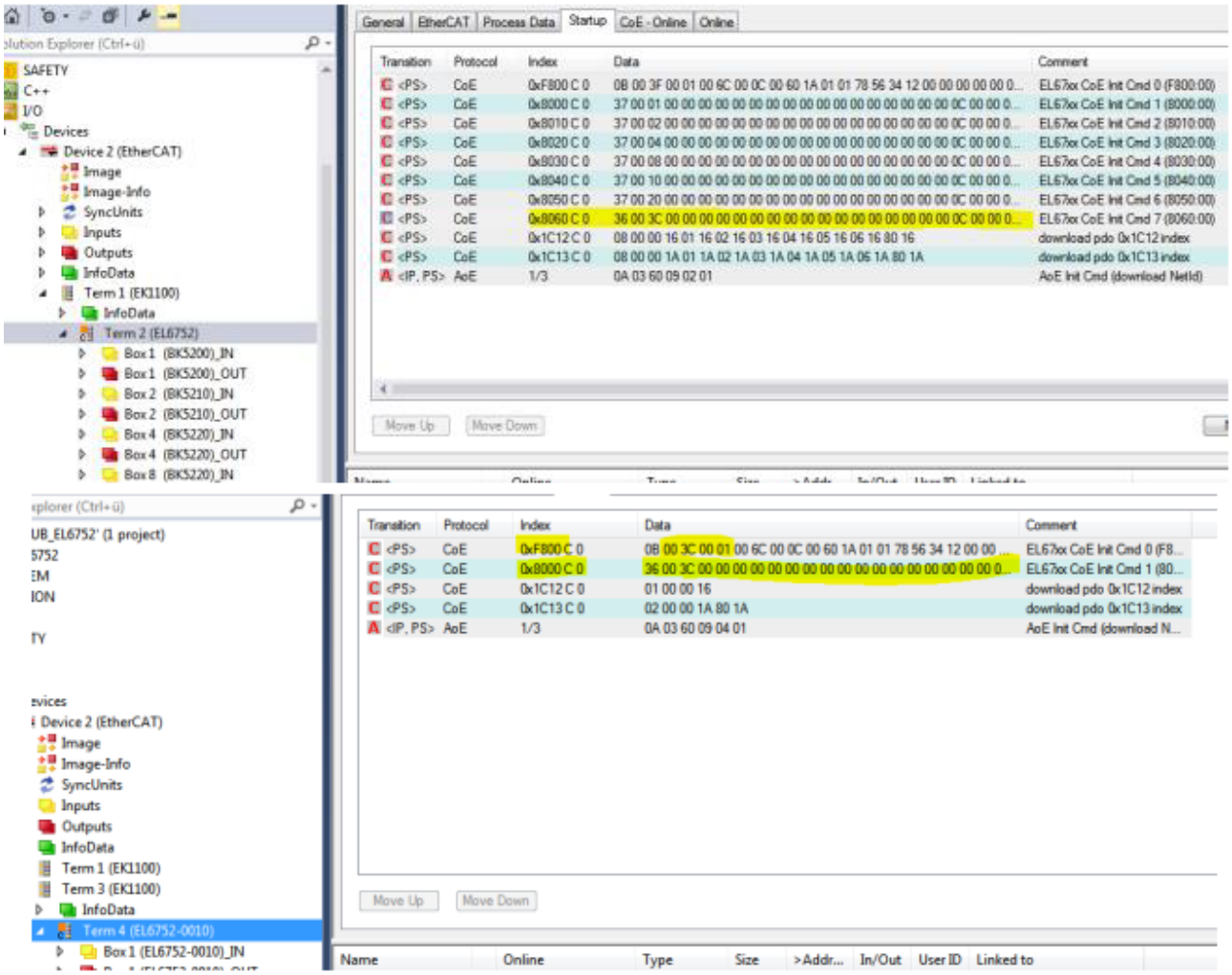


Fig. 66: Example of start-up CMD (0x8000:01; 0xF800:01 and 0xF800:02) that are ignored by the slave terminal after successfully setting the MACId and baud rate using ADS

ADS command (reset)

```
IDXGRP=0x1F480
Index Offset 0x00

LEN=6

DATA[0]=0
DATA[1]=0
DATA[2]=0
DATA[3]=0
DATA[4]=0
DATA[5]=0


Ams Net Id: die der EL6752
Ams Port: 200
```

In this way the data can be permanently deleted again and the terminal behaves as in the delivery condition.

Reset command taking the example of the TwinCAT AMS ADS Viewer

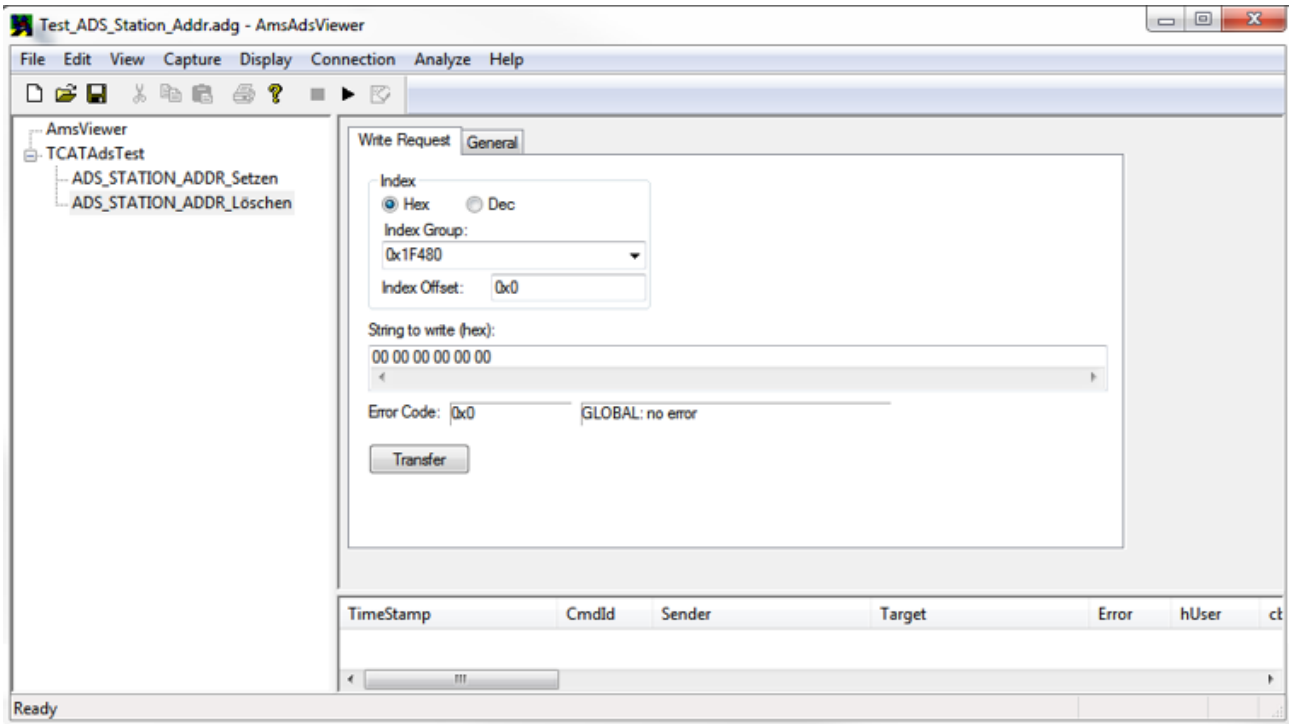


Fig. 67: Resetting the persistent data for MAC ID and baud rate

6.7.2 Object description and parameterization

6.7.2.1 DeviceNet® master - EL6752

EtherCAT XML Device Description

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

Parameterization via the CoE list (CAN over EtherCAT)

The EtherCAT device is parameterized via the CoE-Online tab (double-click on the respective object) or via the Process Data tab (allocation of PDOs). Please note the following general CoE notes [▶ 39] when using/manipulating the CoE parameters:

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

Introduction

The CoE overview contains objects for different intended applications:

- Objects required for parameterization during commissioning
- Objects for indicating internal settings (may be fixed)

The parameterization and the objects required for normal operation will be presented first of all below. All further objects that are not needed for the normal application case can be found in the lower section of the table.

6.7.2.1.1 Objects for the parameterization

Index 8000-803E Configuration Data Slave 1

Index (hex)	Name	Meaning	Data type	Flags	Default
8000+n*16:0	Configuration Data Slave 1	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x21 (33_{dec})
(8000+n*16):01	MAC ID	DeviceNet device address (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):02	ProductName	Product name	OCTET-STRING[32]	RW	{0}
(8000+n*16):03	Device Type	Device type (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):04	Vendor ID	Vendor ID (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):05	Product Code	Product code (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):06	Revision Number	Revision number (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):07	Serial Number	Serial number (see DeviceNet specification)	UINT32	RW	0x00000000 (0_{dec})
(8000+n*16):08	Network Flags	Reserved for AMS via DeviceNet	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):09	Network Port	Reserved for AMS via DeviceNet	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):0A	Network Segment Address	Reserved for AMS via DeviceNet	OCTET-STRING[6]	RW	{0}
(8000+n*16):0B	Reserved	reserved	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):0C	Electronic Key	Electronic key bit mask: Bit 0: Check Vendor Id Bit 1: Check DeviceType Bit 2: Check Product Code Bit 3: Check Revision Bit 4: reserved(0) Bit 5: reserved(0) Bit 6: reserved(0) Bit 7: reserved(0)	UINT16	RW	0x0100 (256_{dec})
(8000+n*16):0D	Allocation Choice	DeviceNet mode selection (see DeviceNet specification) Bit 0: reserved (0) Bit1: Polled Bit2: Bit-Strobed Bit3: reserved (0) Bit4: Change of State Bit5: Cyclic Bit6: Acknowledge Suppression Bit7: reserved(0)	UINT16	RW	0x0100 (256_{dec})
(8000+n*16):0E	Expected Packet Rate - Poll	Timing parameter for the poll connection (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):0F	Expected Packet Rate - Bit Strobe	Timing parameter for the bit strobe connection (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):10	Expected Packet Rate - COS/Cyclic	Timing parameter for the COS/cyclic connection (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):11	Produced Data Size - Poll	Data length in poll mode	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):12	Produced Data Size - Bit Strobe	Data length in bit strobe mode	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):13	Produced Data Size - COS/Cyclic	Data length in Change of State / Cyclic mode	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):14	Reserved	reserved	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):15	Reserved	reserved	UINT16	RW	0x0000 (0_{dec})

Index (hex)	Name	Meaning	Data type	Flags	Default
(8000+n*16):16	Reserved	reserved	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):17	Consumed Data Size - Poll	Data length in poll mode	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):18	Condumed Data Size - Bit Strobe	Data length in bit strobe mode	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):19	Consumed Data Size - COS/Cyclic	Data length in Change of State / Cyclic mode	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):1A	Reserved	reserved	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):1B	Reserved	reserved	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):1C	Reserved	reserved	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):1D	Inhibit Time	Timing parameter for the COS/cyclic connection (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):1E	Acknowledge Timer	Timing parameter for the COS/cyclic connection (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):1F	Acknowledge Retry Limit	Timing parameter for the COS/cyclic connection (see DeviceNet specification)	UINT16	RW	0x0100 (256_{dec})
(8000+n*16):20	Reserved	reserved	UINT16	RW	0x0000 (0_{dec})
(8000+n*16):21	Reserved	reserved	UINT16	RW	0x0000 (0_{dec})

Index 8001-803F StartUp Attributes Slave 1

Index (hex)	Name	Meaning	Data type	Flags	Default
8001+n*16:0	StartUp Attributes Slave 1	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x01 (1_{dec})
(8001+n*16):01	Start Up Attribute.000		OCTET-STRING[32]	RW	{0}

6.7.2.1.2 Objects for internal settings

6.7.2.1.2.1 Standard objects (0x1000-0x1FFF)

Index 1000 Device type

Index (hex)	Name	Meaning	Data type	Flags	Default
1000:0	Device type	Device type of the EtherCAT slave: The Lo-Word contains the CoE profile used (5001). The Hi-Word contains the module profile according to the modular device profile.	UINT32	RO	0x14501389 (340792201_{dec})

Index 1008 Device name

Index (hex)	Name	Meaning	Data type	Flags	Default
1008:0	Device name	Device name of the EtherCAT slave	STRING	RO	EL6752

Index 1009 Hardware version

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

Index 100A Software version

Index (hex)	Name	Meaning	Data type	Flags	Default
100A:0	Software version	Firmware version of the EtherCAT slave	STRING	RO	00

Index 1018 Identity

Index (hex)	Name	Meaning	Data type	Flags	Default
1018:0	Identity	Information for identifying the slave	UINT8	RO	0x04 (4_{dec})
1018:01	Vendor ID	Vendor ID of the EtherCAT slave	UINT32	RO	0x00000002 (2_{dec})
1018:02	Product code	Product code of the EtherCAT slave	UINT32	RO	0x1A603052 (442511442_{dec})
1018:03	Revision	Revision number of the EtherCAT slave; the low word (bit 0-15) indicates the special terminal number, the high word (bit 16-31) refers to the device description	UINT32	RO	0x00100000 (1048576_{dec})
1018:04	Serial number	Serial number of the EtherCAT slave; the low byte (bit 0-7) of the low word contains the year of production, the high byte (bit 8-15) of the low word contains the week of production, the high word (bit 16-31) is 0	UINT32	RO	0x00000000 (0_{dec})

Index 10E2 Manufacturer-specific Identification Code

Index (hex)	Name	Meaning	Data type	Flags	Default
10E2:0	Manufacturer-specific Identification Code	Manufacturer-specific Identification Code	UINT8	RO	0x01 (1_{dec})
10E2:01	SubIndex 001	reserved	STRING	RO	

Index 1600-163E DNS RxPDO-Map Slave 0

Index (hex)	Name	Meaning	Data type	Flags	Default
1600+n:0	DNS RxPDO-Map Slave 0	PDO Mapping RxPDO 1 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1600+n):01					
...					
(1600+n):FF					

Index 1650-168E DNS RxPDO-Map Slave 5

Index (hex)	Name	Meaning	Data type	Flags	Default
1650+n:0	DNS RxPDO-Map Slave 5	PDO Mapping RxPDO 81 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1650+n):01					
...					
(1650+n):FF					

Index 16A0-16DE DNS RxPDO-Map Slave 10

Index (hex)	Name	Meaning	Data type	Flags	Default
16A0+n:0	DNS RxPDO-Map Slave 10	PDO Mapping RxPDO 161 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(16A0+n):01					
...					
(16A0+n):FF					

Index 16F0-172E DNS RxPDO-Map Slave 15

Index (hex)	Name	Meaning	Data type	Flags	Default
16F0+n:0	DNS RxPDO-Map Slave 15	PDO Mapping RxPDO 241 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(16F0+n):01					
...					
(16F0+n):FF					

Index 1740-177E DNS RxPDO-Map Slave 20

Index (hex)	Name	Meaning	Data type	Flags	Default
1740+n:0	DNS RxPDO-Map Slave 20	PDO Mapping RxPDO 321 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1740+n):01					
...					
(1740+n):FF					

Index 1790-17CE DNS RxPDO-Map Slave 25

Index (hex)	Name	Meaning	Data type	Flags	Default
1790+n:0	DNS RxPDO-Map Slave 25	PDO Mapping RxPDO 401 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1790+n):01					
...					
(1790+n):FF					

Index 17E0-181E DNS RxPDO-Map Slave 30

Index (hex)	Name	Meaning	Data type	Flags	Default
17E0+n:0	DNS RxPDO-Map Slave 30	PDO Mapping RxPDO 481 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(17E0+n):01					
...					
(17E0+n):FF					

Index 1830-186E DNS RxPDO-Map Slave 35

Index (hex)	Name	Meaning	Data type	Flags	Default
1830+n:0	DNS RxPDO-Map Slave 35	PDO parameter TxPDO 49 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1830+n):01					
...					
(1830+n):FF					

Index 1880-18BE DNS RxPDO-Map Slave 40

Index (hex)	Name	Meaning	Data type	Flags	Default
1880+n:0	DNS RxPDO-Map Slave 40	PDO parameter TxPDO 129 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1880+n):01					
...					
(1880+n):FF					

Index 18D0-190E DNS RxPDO-Map Slave 45

Index (hex)	Name	Meaning	Data type	Flags	Default
18D0+n:0	DNS RxPDO-Map Slave 45	PDO parameter TxPDO 209 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(18D0+n):01					
...					
(18D0+n):FF					

Index 1920-195E DNS RxPDO-Map Slave 50

Index (hex)	Name	Meaning	Data type	Flags	Default
1920+n:0	DNS RxPDO-Map Slave 50	PDO parameter TxPDO 289 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1920+n):01					
...					
(1920+n):FF					

Index 1970-19AE DNS RxPDO-Map Slave 55

Index (hex)	Name	Meaning	Data type	Flags	Default
1970+n:0	DNS RxPDO-Map Slave 55	PDO parameter TxPDO 369 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1970+n):01					
...					
(1970+n):FF					

Index 19C0-19FE DNS RxPDO-Map Slave 60

Index (hex)	Name	Meaning	Data type	Flags	Default
19C0+n:0	DNS RxPDO-Map Slave 60	PDO parameter TxPDO 449 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(19C0+n):01					
...					
(19C0+n):FF					

Index 1AB0-1AEE DNS TxPDO-Map Slave 11

Index (hex)	Name	Meaning	Data type	Flags	Default
1AB0+n:0	DNS TxPDO-Map Slave 11	PDO Mapping TxPDO 177 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1AB0+n):01					
...					
(1AB0+n):FF					

Index 1B00-1B3E DNS TxPDO-Map Slave 16

Index (hex)	Name	Meaning	Data type	Flags	Default
1B00+n:0	DNS TxPDO-Map Slave 16	PDO Mapping TxPDO 257 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1B00+n):01					
...					

Index (hex)	Name	Meaning	Data type	Flags	Default
(1B00+n):FF					

Index 1B50-1B8E DNS TxPDO-Map Slave 21

Index (hex)	Name	Meaning	Data type	Flags	Default
1B50+n:0	DNS TxPDO-Map Slave 21	PDO Mapping TxPDO 337 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1B50+n):01					
...					
(1B50+n):FF					

Index 1BA0-1BDE DNS TxPDO-Map Slave 26

Index (hex)	Name	Meaning	Data type	Flags	Default
1BA0+n:0	DNS TxPDO-Map Slave 26	PDO Mapping TxPDO 417 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1BA0+n):01					
...					
(1BA0+n):FF					

Index 1BF0-1C2E DNS TxPDO-Map Slave 31

Index (hex)	Name	Meaning	Data type	Flags	Default
1BF0+n:0	DNS TxPDO-Map Slave 31	PDO Mapping TxPDO 497 (one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1BF0+n):01					
...					
(1BF0+n):FF					

Index 1C00 Sync manager type

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

Index 1C32 SM output parameter

Index (hex)	Name	Meaning	Data type	Flags	Default
1C32:0	SM output parameter	Synchronization parameters for the outputs	UINT8	RO	0x20 (32_{dec})
1C32:01	Sync mode	Current sync mode: • 0: Free Run • 1: Synchron with SM 2 Event • 2: DC-Mode - Synchron with SYNC0 Event 3: DC-Mode - Synchron with SYNC1 Event	UINT16	RW	0x0000 (0_{dec})
1C32:02	Cycle time	Cycle time (in ns): • Free Run: cycle time of the local timer • Synchron with SM 2 Event: cycle time of the master 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	RO	0x00000000 (0_{dec})
1C32:04	Sync modes supported	Sync modes supported: • Bit 0 = 1: Free Run is supported	UINT16	RO	0x4001 (16385_{dec})

Index (hex)	Name	Meaning	Data type	Flags	Default
		• Bit 1 = 1: Synchron with SM 2 Event is supported • Bit 2-3 = 01: DC-Mode is supported • Bit 4-5 = 10: Output Shift with SYNC1 Event (DC Mode only) Bit 14 = 1: dynamic times (measurement through writing of 1C32:08)			
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})
1C32:07	Minimum delay time		UINT32	RO	0x00000000 (0_{dec})
1C32:08	Get Cycle Time	• 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, 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})
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 intervals between SYNC0 and SYNC1 events that are 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 1C33 SM input parameter

Index (hex)	Name	Meaning	Data type	Flags	Default
1C33:0	SM input parameter	Synchronization parameters for the inputs	UINT8	RO	0x20 (32_{dec})
1C33:01	Sync mode	Current sync mode: • 0: Free Run • 1: Synchron with SM 3 Event (no outputs available) • 2: DC - Synchron with SYNC0 Event • 3: DC - Synchron with SYNC1 Event 34: Synchron 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, DC Mode only)	UINT32	RO	0x00000000 (0_{dec})
1C33:04	Sync modes supported	Sync modes supported: • Bit 0: Free Run is supported • Bit 1: Synchron with SM 2 Event is supported (outputs available) • Bit 1: Synchron 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	0x4001 (16385_{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 the inputs being available for the master (in ns, DC Mode only)	UINT32	RO	0x00000000 (0_{dec})

Index (hex)	Name	Meaning	Data type	Flags	Default
1C33:07	Minimum delay time		UINT32	RO	0x00000000 (0_{dec})
1C33:08	Get Cycle Time	as 1C32:08	UINT16	RW	0x0000 (0_{dec})
1C33:09	Maximum delay time	Time between SYNC1 event and reading of the inputs (in ns, DC Mode only)	UINT32	RO	0x00000000 (0_{dec})
1C33:0B	SM event missed counter	as 1C32:11	UINT16	RO	0x0000 (0_{dec})
1C33:0C	Cycle exceeded counter	as 1C32:12	UINT16	RO	0x0000 (0_{dec})
1C33:0D	Shift too short counter	as 1C32:13	UINT16	RO	0x0000 (0_{dec})
1C33:20	Sync error	as 1C32:32	BOOLEAN	RO	0x00 (0_{dec})

Index 1C40-1C7E DNS TxPDO-Map Slave 36

Index (hex)	Name	Meaning	Data type	Flags	Default
1C40+n:0	DNS TxPDO-Map Slave 36	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1C40+n):01					
...					
(1C40+n):FF					

Index 1C50-1C8E DNS TxPDO-Map Slave 37

Index (hex)	Name	Meaning	Data type	Flags	Default
1C50+n:0	DNS TxPDO-Map Slave 37	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1C50+n):01					
...					
(1C50+n):FF					

Index 1C60-1C9E DNS TxPDO-Map Slave 38

Index (hex)	Name	Meaning	Data type	Flags	Default
1C60+n:0	DNS TxPDO-Map Slave 38	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1C60+n):01					
...					
(1C60+n):FF					

Index 1C70-1CAE DNS TxPDO-Map Slave 39

Index (hex)	Name	Meaning	Data type	Flags	Default
1C70+n:0	DNS TxPDO-Map Slave 39	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1C70+n):01					
...					
(1C70+n):FF					

Index 1C80-1CBE DNS TxPDO-Map Slave 40

Index (hex)	Name	Meaning	Data type	Flags	Default
1C80+n:0	DNS TxPDO-Map Slave 40	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1C80+n):01					
...					
(1C80+n):FF					

Index 1C90-1CCE DNS TxPDO-Map Slave 41

Index (hex)	Name	Meaning	Data type	Flags	Default
1C90+n:0	DNS TxPDO-Map Slave 41	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1C90+n):01					
...					
(1C90+n):FF					

Index 1CA0-1CDE DNS TxPDO-Map Slave 42

Index (hex)	Name	Meaning	Data type	Flags	Default
1CA0+n:0	DNS TxPDO-Map Slave 42	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1CA0+n):01					
...					
(1CA0+n):FF					

Index 1CB0-1CEE DNS TxPDO-Map Slave 43

Index (hex)	Name	Meaning	Data type	Flags	Default
1CB0+n:0	DNS TxPDO-Map Slave 43	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1CB0+n):01					
...					
(1CB0+n):FF					

Index 1CC0-1CFE DNS TxPDO-Map Slave 44

Index (hex)	Name	Meaning	Data type	Flags	Default
1CC0+n:0	DNS TxPDO-Map Slave 44	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1CC0+n):01					
...					
(1CC0+n):FF					

Index 1CD0-1D0E DNS TxPDO-Map Slave 45

Index (hex)	Name	Meaning	Data type	Flags	Default
1CD0+n:0	DNS TxPDO-Map Slave 45	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1CD0+n):01					
...					
(1CD0+n):FF					

Index 1CE0-1D1E DNS TxPDO-Map Slave 46

Index (hex)	Name	Meaning	Data type	Flags	Default
1CE0+n:0	DNS TxPDO-Map Slave 46	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1CE0+n):01					
...					
(1CE0+n):FF					

Index 1CF0-1D2E DNS TxPDO-Map Slave 47

Index (hex)	Name	Meaning	Data type	Flags	Default
1CF0+n:0	DNS TxPDO-Map Slave 47	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1CF0+n):01					
...					
(1CF0+n):FF					

Index 1D00-1D3E DNS TxPDO-Map Slave 48

Index (hex)	Name	Meaning	Data type	Flags	Default
1D00+n:0	DNS TxPDO-Map Slave 48	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1D00+n):01					
...					
(1D00+n):FF					

Index 1D10-1D4E DNS TxPDO-Map Slave 49

Index (hex)	Name	Meaning	Data type	Flags	Default
1D10+n:0	DNS TxPDO-Map Slave 49	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1D10+n):01					
...					
(1D10+n):FF					

Index 1D20-1D5E DNS TxPDO-Map Slave 50

Index (hex)	Name	Meaning	Data type	Flags	Default
1D20+n:0	DNS TxPDO-Map Slave 50	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1D20+n):01					
...					
(1D20+n):FF					

Index 1D30-1D6E DNS TxPDO-Map Slave 51

Index (hex)	Name	Meaning	Data type	Flags	Default
1D30+n:0	DNS TxPDO-Map Slave 51	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1D30+n):01					
...					
(1D30+n):FF					

Index 1D40-1D7E DNS TxPDO-Map Slave 52

Index (hex)	Name	Meaning	Data type	Flags	Default
1D40+n:0	DNS TxPDO-Map Slave 52	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1D40+n):01					
...					
(1D40+n):FF					

Index 1D50-1D8E DNS TxPDO-Map Slave 53

Index (hex)	Name	Meaning	Data type	Flags	Default
1D50+n:0	DNS TxPDO-Map Slave 53	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})

Index (hex)	Name	Meaning	Data type	Flags	Default
(1D50+n):01					
...					
(1D50+n):FF					

Index 1D60-1D9E DNS TxPDO-Map Slave 54

Index (hex)	Name	Meaning	Data type	Flags	Default
1D60+n:0	DNS TxPDO-Map Slave 54	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1D60+n):01					
...					
(1D60+n):FF					

Index 1D70-1DAE DNS TxPDO-Map Slave 55

Index (hex)	Name	Meaning	Data type	Flags	Default
1D70+n:0	DNS TxPDO-Map Slave 55	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1D70+n):01					
...					
(1D70+n):FF					

Index 1D80-1DBE DNS TxPDO-Map Slave 56

Index (hex)	Name	Meaning	Data type	Flags	Default
1D80+n:0	DNS TxPDO-Map Slave 56	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1D80+n):01					
...					
(1D80+n):FF					

Index 1D90-1DCE DNS TxPDO-Map Slave 57

Index (hex)	Name	Meaning	Data type	Flags	Default
1D90+n:0	DNS TxPDO-Map Slave 57	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1D90+n):01					
...					
(1D90+n):FF					

Index 1DA0-1DDE DNS TxPDO-Map Slave 58

Index (hex)	Name	Meaning	Data type	Flags	Default
1DA0+n:0	DNS TxPDO-Map Slave 58	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1DA0+n):01					
...					
(1DA0+n):FF					

Index 1DB0-1DEE DNS TxPDO-Map Slave 59

Index (hex)	Name	Meaning	Data type	Flags	Default
1DB0+n:0	DNS TxPDO-Map Slave 59	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1DB0+n):01					
...					

Index (hex)	Name	Meaning	Data type	Flags	Default
(1DB0+n):FF					

Index 1DC0-1DFE DNS TxPDO-Map Slave 60

Index (hex)	Name	Meaning	Data type	Flags	Default
1DC0+n:0	DNS TxPDO-Map Slave 60	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1DC0+n):01					
...					
(1DC0+n):FF					

Index 1DD0-1E0E DNS TxPDO-Map Slave 61

Index (hex)	Name	Meaning	Data type	Flags	Default
1DD0+n:0	DNS TxPDO-Map Slave 61	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1DD0+n):01					
...					
(1DD0+n):FF					

Index 1DE0-1E1E DNS TxPDO-Map Slave 62

Index (hex)	Name	Meaning	Data type	Flags	Default
1DE0+n:0	DNS TxPDO-Map Slave 62	(one object per module (0 <= n < maximum number of modules))	UINT8	RW	0x00 (0_{dec})
(1DE0+n):01					
...					
(1DE0+n):FF					

Index F008 Code word

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

6.7.2.1.2.2 Profile-specific objects (0x6000-0xFFFF)

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

Index 6000-603E Produced Data Slave 1

Index (hex)	Name	Meaning	Data type	Flags	Default
6000+n*16:0	Produced Data Slave 1	(one object per module (0 <= n < maximum number of modules))	UINT8	RO	0x01 (1_{dec})
(6000+n*16):01					
...					
(6000+n*16):FF					

Index 7000-703E Consumed Data Slave 1

Index (hex)	Name	Meaning	Data type	Flags	Default
7000+n*16:0	Consumed Data Slave 1	(one object per module (0 <= n < maximum number of modules))	UINT8	RO	0x01 (1_{dec})
(7000+n*16):01					
...					
(7000+n*16):FF					

Index A000-A03E Status Data Slave 1

Index (hex)	Name	Meaning	Data type	Flags	Default
A000+n*16:0	Status Data Slave 1	(one object per module (0 <= n < maximum number of modules))	UINT8	RO	0x02 (2_{dec})
(A000+n*16):01					
...					
(A000+n*16):FF					

Index F000 Modular device profile

Index (hex)	Name	Meaning	Data type	Flags	Default
F000:0	Modular Device Profile	General information for the Modular Device Profile	UINT8	RO	0x02 (2_{dec})
F000:01	Index distance	Index distance of the objects of the individual channels	UINT16	RO	0x0010 (16_{dec})
F000:02	Maximum number of modules	Number of channels	UINT16	RO	0x003F (63_{dec})
F000:03	General configuration		UINT32	RO	0x7FFFFFFF (2147483647_{dec})

Index F010 Module list

Index (hex)	Name	Meaning	Data type	Flags	Default
F010:0	Module list	List of the DeviceNet slaves connected to the EL6752.	UINT8	RW	0x00 (0_{dec})
F010:01		Product code of the first DeviceNet slave	UINT16	RO	0x00 (0_{dec})
...					
F010:3F					

Index F100 DeviceNet status

Index (hex)	Name	Meaning		Data type	Flags	Default
F100:0	DeviceNet status	Max. Subindex		UINT8	RO	0x10 (16_{dec})
F100:01	Communication status	permitted values:		UINT8	RO	0x00 (0_{dec})
		0	No error			
		1	Node deactivated			
		2	Node not exist, Timeout			

Index (hex)	Name	Meaning		Data type	Flags	Default
		18	Node ready			
		49	Idle Mode, Idle Messages received			
F100:10	TxPdoState	Status of the Tx-PDO		BOOLEAN	RO	0x00 (0_{dec})

Index F101 Network status

Index (hex)	Name	Meaning	Data type	Flags	Default
F101:0	Network status	Max. Subindex	UINT8	RO	0x11 (17_{dec})
F101:01	Device status	0: RUN MODE 1: IDLE MODE 2: Duplicate MacId Check failed, MAC ID used 3: Status: Selftest 4: Status: Standby 5: Status:Major Recoverable Fault 6: Status:Minor Recoverable Fault 7: DeviceNet® Voltage Error 8: DeviceNet® Access Error	UINT8	RO	0x00 (0_{dec})
F101:09	CAN BUS-OFF	CAN controller of the EL6752 is in state bus-off	BOOLEAN	RO	0x00 (0_{dec})
F101:0A	CAN warning limit	CAN controller of the EL6752 has exceeded the warning limit	BOOLEAN	RO	0x00 (0_{dec})
F101:0B	CAN Overrun	CAN controller of the EL6752 is in state bus-off	BOOLEAN	RO	0x00 (0_{dec})
F101:11	CAN BUS load	CAN bus load 0 - 100%	UINT16	RO	0x0000 (0_{dec})

Index F102 Status - MacState

Index (hex)	Name	Meaning	Data type	Flags	Default
F102:0	Status - MacState		UINT8	RO	0x01 (1_{dec})
F102:01					
...					
F102:FF					

Index F103 Diag data

Index (hex)	Name	Meaning	Data type	Flags	Default
F103:0	Diag data		UINT8	RO	0x01 (1_{dec})
F103:01					
...					
F103:FF					

Index F200 BitStrobe Bits

Index (hex)	Name	Meaning	Data type	Flags	Default
F200:0	BitStrobe Bits		UINT8	RO	0x40 (64_{dec})
F200:01					
...					
F200:FF					

Index F800 Bus Parameter set

Index (hex)	Name	Meaning		Data type	Flags	Default
F800:0	Bus Parameter set	Max. Subindex		UINT8	RW	0x0B (11_{dec})
F800:01	MAC ID	Device address of the DeviceNet device (see DeviceNet specification)		UINT16	RW	0x0000 (0_{dec})
F800:02	Baud rate	permitted values:		UINT16	RW	0x0100 (256_{dec})
		0	invalid			
		1	500 kbaud			
		2	250 kbaud			
		3	125 kbaud			
F800:03	Vendor ID	Vendor ID of the DeviceNet device (see DeviceNet specification)		UINT16	RW	0x0000 (0_{dec})
F800:04	DeviceType	Device type of the DeviceNet device (see DeviceNet specification)		UINT16	RW	0x0000 (0_{dec})

Index (hex)	Name	Meaning		Data type	Flags	Default
F800:05	Product Code	Product code of the DeviceNet device (see DeviceNet specification)		UINT16	RW	0x0000 (0_{dec})
F800:06	Revision Number	Version number of the DeviceNet device (see DeviceNet specification)		UINT16	RW	0x0000 (0_{dec})
F800:07	Serial Number	Serial number of the DeviceNet device (see DeviceNet specification)		UINT32	RW	0x00350000 (3473408_{dec})
F800:08	Product Name	Product name of the DeviceNet device (see DeviceNet specification)		OCTET-STRING[32]	RW	{0}
F800:09	Quick Connect	permitted values:		UINT16	RW	0x0000 (0_{dec})
		0	Quick Connect Disabled			
		1	Quick Connect DEnabled			
F800:0A	Reserved	reserved		UINT16	RW	0x0000 (0_{dec})
F800:0B	Reserved2	reserved		UINT16	RW	0x0000 (0_{dec})

Index F880 Vendor data

Index (hex)	Name	Meaning	Data type	Flags	Default
F880:0	Vendor data	Max. Subindex	UINT8	RO	0x01 (1_{dec})
F880:01	Serial Number	Serial Number	UINT32	RW	0x00000000 (0_{dec})

6.7.2.2 DeviceNet® Slave - EL6752-0010

EtherCAT XML Device Description

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

Parameterization via the CoE list (CAN over EtherCAT)

The EtherCAT device is parameterized via the CoE-Online tab (double-click on the respective object) or via the Process Data tab (allocation of PDOs). Please note the following general CoE notes [▶ 39] when using/manipulating the CoE parameters:

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

Introduction

The CoE overview contains objects for different intended applications:

- Objects required for parameterization during commissioning
- Objects for indicating internal settings (may be fixed)

The parameterization and the objects required for normal operation will be presented first of all below. All further objects that are not needed for the normal application case can be found in the lower section of the table.

6.7.2.2.1 Objects for the parameterization

Index 8000 configuration data

Index (hex)	Name	Meaning	Data type	Flags	Default
8000:0	Configuration data	Max. Subindex	UINT8	RW	0x33 (51_{dec})
8000:01	MAC ID	DeviceNet device address (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
8000:02	ProductName	Product name	OCTET-STRING[16]	RW	{0}
8000:03	Device Type	Device type (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
8000:04	Vendor ID	Vendor ID (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
8000:05	Product Code	Product code (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
8000:06	Revision Number	Revision number (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
8000:07	Serial Number	Serial number (see DeviceNet specification)	UINT32	RW	0x00000000 (0_{dec})
8000:08	Network Flags	Reserved for AMS via DeviceNet	UINT16	RW	0x0000 (0_{dec})
8000:09	Network Port	Reserved for AMS via DeviceNet	UINT16	RW	0x0000 (0_{dec})
8000:0A	Network Segment Address	Reserved for AMS via DeviceNet	OCTET-STRING[6]	RW	{0}
8000:0A	Reserved	reserved	UINT16	RW	0x0000 (0_{dec})
8000:0B	Electronic Key	Electronic key bit mask: Bit 0: Check Vendor Id Bit 1: Check DeviceType Bit 2: Check Product Code Bit 3: Check Revision Bit 4: reserved(0) Bit 5: reserved(0) Bit 6: reserved(0) Bit 7: reserved(0)	UINT16	RW	0x0100 (256_{dec})
8000:0D	Allocation Choice	DeviceNet mode selection (see DeviceNet specification) Bit 0: reserved (0) Bit 1: Polled Bit 2: Bit-Strobed Bit 3: reserved (0) Bit 4: Change of State Bit 5: Cyclic Bit 6: Acknowledge Suppression Bit 7: reserved(0)	UINT16	RW	0x0100 (256_{dec})
8000:0E	Expected Packet Rate - Poll	Timing parameter for the poll connection (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
8000:0F	Expected Packet Rate - Bit Strobe	Timing parameter for the bit strobe connection (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
8000:10	Expected Packet Rate - COS/Cyclic	Timing parameter for the COS/cyclic connection (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
8000:11	Produced Data Size - Poll	Data length in poll mode	UINT16	RW	0x0000 (0_{dec})
8000:12	Produced Data Size - Bit Strobe	Data length in bit strobe mode	UINT16	RW	0x0000 (0_{dec})
8000:13	Produced Data Size - COS/Cyclic	Data length in Change of State / Cyclic mode	UINT16	RW	0x0000 (0_{dec})
8000:14	Reserved1	reserved	UINT16	RW	0x0000 (0_{dec})
8000:15	Reserved2	reserved	UINT16	RW	0x0000 (0_{dec})
8000:16	Reserved3	reserved	UINT16	RW	0x0000 (0_{dec})
8000:17	Consumed Data Size - Poll	Data length in poll mode	UINT16	RW	0x0000 (0_{dec})
8000:18	Consumed Data Size - Bit Strobe	Data length in bit strobe mode	UINT16	RW	0x0000 (0_{dec})
8000:19	Consumed Data Size - COS/Cyclic	Data length in Change of State / Cyclic mode	UINT16	RW	0x0000 (0_{dec})
8000:1A	Reserved4	reserved	UINT16	RW	0x0000 (0_{dec})
8000:1B	Reserved5	reserved	UINT16	RW	0x0000 (0_{dec})

Index (hex)	Name	Meaning	Data type	Flags	Default
8000:1C	Reserved6	reserved	UINT16	RW	0x0000 (0_{dec})
8000:1D	Inhibit Time	Timing parameter for the COS/cyclic connection (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
8000:1E	Acknowledge Timer	Timing parameter for the COS/cyclic connection (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
8000:1F	Acknowledge Retry Limit	Timing parameter for the COS/cyclic connection (see DeviceNet specification)	UINT16	RW	0x0000 (0_{dec})
8000:20	Reserved7	reserved	UINT16	RW	0x0000 (0_{dec})
8000:21	Reserved8	reserved	UINT16	RW	0x0000 (0_{dec})

6.7.2.2.2 Objects for internal settings

6.7.2.2.2.1 Standard objects (0x1000-0x1FFF)

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

Index 1000 Device type

Index (hex)	Name	Meaning	Data type	Flags	Default
1000:0	Device type	Device type of the EtherCAT slave: The Lo-Word contains the CoE profile used (5001). The Hi-Word contains the module profile according to the modular device profile.	UINT32	RO	0x145A1389 (341447561_{dec})

Index 1008 Device name

Index (hex)	Name	Meaning	Data type	Flags	Default
1008:0	Device name	Device name of the EtherCAT slave	STRING	RO	EL6752-0010

Index 1009 Hardware version

Index (hex)	Name	Meaning	Data type	Flags	Default
1009:0	Hardware version	Hardware-Version des EtherCAT-Slaves	STRING	RO	00

Index 100A Software version

Index (hex)	Name	Meaning	Data type	Flags	Default
100A:0	Software version	Firmware version of the EtherCAT slave	STRING	RO	00

Index 1018 Identity

Index (hex)	Name	Meaning	Data type	Flags	Default
1018:0	Identity	Information for identifying the slave	UINT8	RO	0x04 (4_{dec})
1018:01	Vendor ID	Vendor ID of the EtherCAT slave	UINT32	RO	0x00000002 (2_{dec})
1018:02	Product code	Product code of the EtherCAT slave	UINT32	RO	0x1A603052 (442511442_{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	0x00000000 (0_{dec})
1018:04	Serial number	Serial number of the EtherCAT slave; the Low Byte (bit 0-7) of the Low Word contains the year of production, the High Byte (bit 8-15) of the Low Word contains the week of production, the High Word (bit 16-31) is 0	UINT32	RO	0x00000000 (0_{dec})

Index 10E2 Manufacturer-specific Identification Code

Index (hex)	Name	Meaning	Data type	Flags	Default
10E2:0	Manufacturer-specific Identification Code	Manufacturer-specific Identification Code	UINT8	RO	0x01 (1_{dec})
10E2:01	SubIndex 001	reserved	STRING	RO	

Index 1A10 DNM TxPDO-Map DeviceNet State

Index (hex)	Name	Meaning	Data type	Flags	Default
1A10:0	DNM TxPDO-Map DeviceNet State	PDO Mapping TxPDO 17	UINT8	RO	0x06 (6_{dec})
1A10:01	SubIndex 001	1. PDO Mapping entry (object 0xA000 (Status Data Slave), entry 0x01)	UINT32	RO	0xA000:01, 8
1A10:02	SubIndex 002	2. PDO Mapping entry (object 0xF100 (DeviceNet status), entry 0x01 (Communication status))	UINT32	RO	0xF100:01, 8
1A10:03	SubIndex 003	3. PDO Mapping entry (object 0xF100 (DeviceNet status), entry 0x10 (TxPdoState))	UINT32	RO	0xF100:10, 1

Index (hex)	Name	Meaning	Data type	Flags	Default
1A10:04	SubIndex 004	4. PDO Mapping entry (object 0xF103 (Diag data), entry 0x01)	UINT32	RO	0xF103:01, 1
1A10:05	SubIndex 005	5. PDO Mapping entry (object 0xF600 (BitStrobe Bits), entry 0x01)	UINT32	RO	0xF600:01, 1
1A10:06	SubIndex 006	6. PDO Mapping entry (13 bits align)	UINT32	RO	0x0000:00, 13

Index 1C00 Sync manager type

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

Index 1C12 RxPDO assign

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

Index 1C13 TxPDO assign

Index (hex)	Name	Meaning	Data type	Flags	Default
1C13:0	TxPDO assign	PDO Assign Inputs	UINT8	RW	0x02 (2_{dec})
1C13:01	SubIndex 001	1. allocated TxPDO (contains the index of the associated TxPDO mapping object)	UINT16	RW	0x1A10 (6672_{dec})
1C13:02	SubIndex 002	2. allocated TxPDO (contains the index of the associated TxPDO mapping object)	UINT16	RW	0x0000 (0_{dec})

Index 1C32 SM output parameter

Index (hex)	Name	Meaning	Data type	Flags	Default
1C32:0	SM output parameter	Synchronization parameters for the outputs	UINT8	RO	0x20 (32_{dec})
1C32:01	Sync mode	Current sync mode: • 0: Free Run • 1: Synchron with SM 2 Event • 2: DC-Mode - Synchron with SYNC0 Event 3: DC-Mode - Synchron with SYNC1 Event	UINT16	RW	0x0000 (0_{dec})
1C32:02	Cycle time	Cycle time (in ns): • Free Run: cycle time of the local timer • Synchron with SM 2 Event: cycle time of the master 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	RO	0x00000000 (0_{dec})
1C32:04	Sync modes supported	Sync modes supported: • Bit 0 = 1: Free Run is supported • Bit 1 = 1: Synchron with SM 2 Event is supported • Bit 2-3 = 01: DC-Mode is supported • Bit 4-5 = 10: Output Shift with SYNC1 Event (DC Mode only) Bit 14 = 1: dynamic times (measurement through writing of 1C32:08)	UINT16	RO	0x4001 (16385_{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})
1C32:07	Minimum delay time		UINT32	RO	0x00000000 (0_{dec})

Index (hex)	Name	Meaning	Data type	Flags	Default
1C32:08	Get Cycle Time	• 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, 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})
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 intervals between SYNC0 and SYNC1 events that are 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 1C33 SM input parameter

Index (hex)	Name	Meaning	Data type	Flags	Default
1C33:0	SM input parameter	Synchronization parameters for the inputs	UINT8	RO	0x20 (32_{dec})
1C33:01	Sync mode	Current sync mode: • 0: Free Run • 1: Synchron with SM 3 Event (no outputs available) • 2: DC - Synchron with SYNC0 Event • 3: DC - Synchron with SYNC1 Event 34: Synchron 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, DC Mode only)	UINT32	RO	0x00000000 (0_{dec})
1C33:04	Sync modes supported	Sync modes supported: • Bit 0: Free Run is supported • Bit 1: Synchron with SM 2 Event is supported (outputs available) • Bit 1: Synchron 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	0x4001 (16385_{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 the inputs being available for the master (in ns, DC Mode only)	UINT32	RO	0x00000000 (0_{dec})
1C33:07	Minimum delay time		UINT32	RO	0x00000000 (0_{dec})
1C33:08	Get Cycle Time	as 1C32:08	UINT16	RW	0x0000 (0_{dec})
1C33:09	Maximum delay time	Time between SYNC1 event and reading of the inputs (in ns, DC Mode only)	UINT32	RO	0x00000000 (0_{dec})
1C33:0B	SM event missed counter	as 1C32:11	UINT16	RO	0x0000 (0_{dec})
1C33:0C	Cycle exceeded counter	as 1C32:12	UINT16	RO	0x0000 (0_{dec})
1C33:0D	Shift too short counter	as 1C32:13	UINT16	RO	0x0000 (0_{dec})
1C33:20	Sync error	as 1C32:32	BOOLEAN	RO	0x00 (0_{dec})

6.7.2.2.2.2 Profile-specific objects (0x6000-0xFFFF)

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

Index A000 Status Data Slave

Index (hex)	Name	Meaning	Data type	Flags	Default
A000	Status Data Slave	max. Subindex	UINT8	RO	0x02 (2_{dec})
A000:01					
...					
A000:FF					

Index F000 Modular device profile

Index (hex)	Name	Meaning	Data type	Flags	Default
F000:0	Modular device profile	General information for the modular device profile	UINT8	RO	0x02 (2_{dec})
F000:01	Module index distance	Index distance 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 F008 Code word

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

Index F100 DeviceNet status

Index (hex)	Name	Meaning	Data type	Flags	Default
F100:0	DeviceNet status	DeviceNet status of the EL6752-0010	UINT8	RO	0x10 (16_{dec})
F100:01	Communication status	Communication status of the EL6752-0010: 0 = No error 1 = Station deactivated 2 = Station not exists 18 = Station ready 31 = only for EtherCAT gateways: WC-State of cyclic EtherCAT frame is 1	UINT8	RO	0x00 (0_{dec})
F100:10	TxPdoState	Status of the Tx-PDO	BOOLEAN	RO	0x00 (0_{dec})

Index F101 Network status

Index (hex)	Name	Meaning	Data type	Flags	Default
F101:0	Network status	Max. Subindex	UINT8	RO	0x11 (17_{dec})
F101:01	Device status	0: RUN MODE 1: IDLE MODE 2: Duplicate MacId Check failed, MAC ID used 3: Status: Selftest 4: Status: Standby 5: Status:Major Recoverable Fault 6: Status:Minor Recoverable Fault 7: DeviceNet Voltage Error 8: DeviceNet Access Error	UINT8	RO	0x00 (0_{dec})
F101:09	CAN BUS-OFF	CAN controller of the EL6752-0010 is in state bus-off	BOOLEAN	RO	0x00 (0_{dec})
F101:0A	CAN warning limit	CAN controller of the EL6752-0010 has exceeded the warning limit	BOOLEAN	RO	0x00 (0_{dec})
F101:0B	CAN Overrun	CAN controller of the EL6752-0010 is in state bus-off	BOOLEAN	RO	0x00 (0_{dec})
F101:11	CAN BUS load	CAN bus load 0 - 100 %	UINT16	RO	0x0000 (0_{dez})

Index F102 Status - MacState

Index (hex)	Name	Meaning	Data type	Flags	Default
F102:0	Status - MacState		UINT8	RO	0x01 (1_{dec})
F102:01					
...					
F102:FF					

Index F103 Diag data

Index (hex)	Name	Meaning	Data type	Flags	Default
F103:0	Diag data		UINT8	RO	0x01 (1_{dec})
F103:01					
...					
F103:FF					

Index F600 BitStrobe Bits

Index (hex)	Name	Meaning	Data type	Flags	Default
F600:0	BitStrobe Bits		UINT8	RO	0x01 (01_{dec})
F600:01					
...					
F600:FF					

Index F800 Bus Parameter set

Index (hex)	Name	Meaning		Data type	Flags	Default
F800:0	Bus Parameter set	Max. Subindex		UINT8	RW	0x0B (11_{dec})
F800:01	MAC ID	Device address of the DeviceNet device (see DeviceNet specification)		UINT16	RW	0x0000 (0_{dec})
F800:02	Baud rate	permitted values:		UINT16	RW	0x0100 (256_{dec})
		0	invalid			
		1	500 kbaud			
		2	250 kbaud			
		3	125 kbaud			
F800:03	Vendor ID	Vendor ID of the DeviceNet device (see DeviceNet specification)		UINT16	RW	0x0000 (0_{dec})
F800:04	DeviceType	Device type of the DeviceNet device (see DeviceNet specification)		UINT16	RW	0x0000 (0_{dec})
F800:05	Product Code	Product code of the DeviceNet device (see DeviceNet specification)		UINT16	RW	0x0000 (0_{dec})
F800:06	Revision Number	Version number of the DeviceNet device (see DeviceNet specification)		UINT16	RW	0x0000 (0_{dec})
F800:07	Serial Number	Serial number of the DeviceNet device (see DeviceNet specification)		UINT32	RW	0x00350000 (3473408_{dec})
F800:08	Product Name	Product name of the DeviceNet device (see DeviceNet specification)		OCTET-STRING[32]	RW	{0}
F800:09	Quick Connect	permitted values:		UINT16	RW	0x0000 (0_{dec})
		0	Quick Connect Disabled			
		1	Quick Connect DEnabled			
F800:0A	Reserved	reserved		UINT16	RW	0x0000 (0_{dec})
F800:0B	Reserved2	reserved		UINT16	RW	0x0000 (0_{dec})

Index F880 Vendor data

Index (hex)	Name	Meaning	Data type	Flags	Default
F880:0	Vendor data	Max. Subindex	UINT8	RO	0x01 (1_{dec})
F880:01	Serial Number	Serial Number	UINT32	RW	0x00000000 (0_{dec})

7 Error handling and diagnostics

7.1 EL6752 - LED description

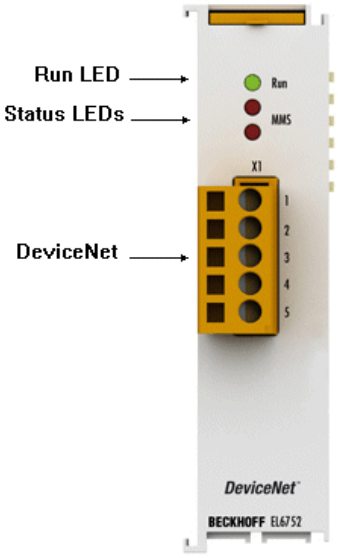


Fig. 68: LEDs

LED behaviour

The LEDs facilitate diagnosing of the main terminal states:

EL6752-0000 (DeviceNet® master terminal)

LED	Colours	Meaning	
RUN	green	This LED indicates the terminal's operating state:	
		off	State of the EtherCAT State Machine: **INIT** = initialization of the terminal; **BOOTSTRAP** = function for terminal firmware updates
		flashing	State of the EtherCAT State Machine: **PREOP** = function for mailbox communication and different standard-settings set
		Single flash	State of the EtherCAT State Machine: **SAFEOP** = verification of the sync manager channels and the distributed clocks. Outputs remain in safe state
		on	State of the EtherCAT State Machine: **OP** = normal operating state; mailbox and process data communication is possible
MNS green	green	off	Master is offline
		flashing	Master is online and is performing the Duplicate MAC-ID check
		on	Master is online and is communicating with the configured slaves
MNS red	red	flashing	Communication error of the master with one of the configured slaves
		on	DeviceNet Bus OFF, DeviceNet voltage error, Master failed Duplicate MAC-ID check

EL6752-0010 (DeviceNet® slave terminal)

LED	Color	Meaning	
RUN	green	This LED indicates the terminal's operating state:	
		off	State of the EtherCAT State Machine: **INIT** = initialization of the terminal; **BOOTSTRAP** = function for terminal firmware updates
		flashing	State of the EtherCAT State Machine: **PREOP** = function for mailbox communication and different standard-settings set
		Single flash	State of the EtherCAT State Machine: **SAFEOP** = verification of the sync manager channels and the distributed clocks. Outputs remain in safe state

LED	Color	Meaning	
		on	State of the EtherCAT State Machine: **OP** = normal operating state; mailbox and process data communication is possible
MNS green	green	off	Slave is offline
		flashing	Slave port has ended the Duplicate MAC-ID check (Network OK), communication error with the master.
		on	Slave port is online and is communicating with the master.
MNS red	red	flashing	Communication error of the slave port with the master, timeout of the slave port
		on	DeviceNet Bus OFF, DeviceNet voltage error, slave port error, error in Duplicate MAC-ID check

7.2 EL6752/-0010 diagnostics

The EL6752/-0010 feature various diagnostic variables that describe the state of the terminal and the DeviceNet® and can be linked in the PLC:

We recommend monitoring the following process data during each cycle:

- **WcState**: if ≠ 0, this EtherCAT device does not take part in the process data traffic
- **State**: if ≠ 8, the EtherCAT device is not in OP (operational) status

We recommend monitoring the following process data:

- **Error**: if ≠ 0, the indicated number of DeviceNet® devices has a BoxState not equal zero, i.e. check which DeviceNet® devices are not operating correctly in the bus
- **DiagFlag**: indicates pending diagnostic data

7.2.1 EL6752/-0010 - WC-State

For monitoring the EtherCAT communication the WC state (working counter) of the EL6752/-0010 must be checked. If the WC state is not equal "0" the EtherCAT communication is disturbed, i.e. data sent to the slave or the master are no longer transferred correctly and are not valid.

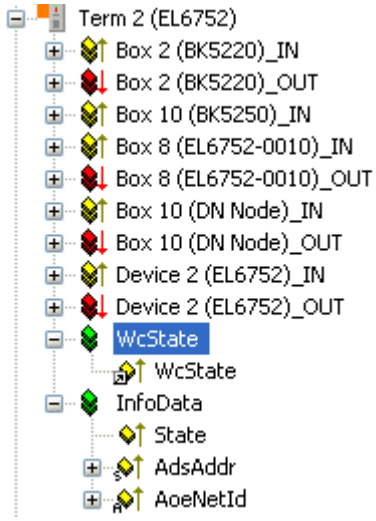


Fig. 69: WCState in the TwinCAT tree

WcState

0: data are valid
1: data are not valid, problem in the EtherCAT communication

7.2.2 EL6752/-0010 - State

The diagnostic state variable indicates the current EtherCAT state of the EL6752/-0010.

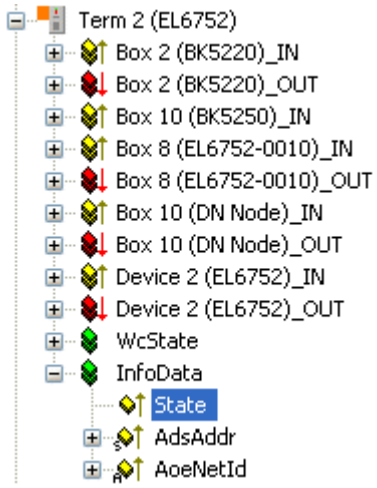


Fig. 70: State diagnostic variable in the TwinCAT tree

State

0x___1 = Slave in 'INIT' state
0x___2 = Slave in 'PREOP' state
0x___3 = Slave in 'BOOT' state
0x___4 = Slave in 'SAFEOP' state
0x___8 = Slave in 'OP' state
0x001_ = Slave signals error
0x002_ = Invalid vendorId, productCode... read
0x004_ = Initialization error occurred
0x010_ = Slave not present
0x020_ = Slave signals link error
0x040_ = Slave signals missing link
0x080_ = Slave signals unexpected link
0x100_ = Communication port A
0x200_ = Communication port B
0x400_ = Communication port C
0x800_ = Communication port D

7.2.3 EL6752/-0010 - Error / DiagFlag

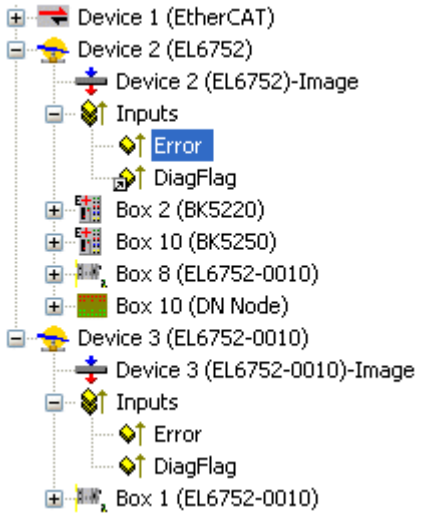


Fig. 71: Error and DiagFlag in the TwinCAT tree

Error

0: all DeviceNet® devices have BoxState zero
>0: number of DeviceNet® devices with BoxState not equal zero.

DiagFlag

0 = no diagnostic data are pending
1 = diagnostic data are pending and can be read via AdsRead services

7.3 DeviceNet® device diagnostics

DeviceNet® slave devices feature different diagnostic variables that describe the DeviceNet® communication state and can be linked in the PLC:

We recommend monitoring the following process data during each cycle:

- **MacState**: if ≠ 0, this DeviceNet® device is not participating correctly in the process data traffic
- **CouplerState**: for Beckhoff Bus Couplers, the terminal communication of the Bus Coupler may be disturbed or diagnostic data may be present if ≠ 0

7.3.1 DeviceNet® slave device / EL6752-0010 - MacState

For monitoring the DeviceNet communication the MacState of the DeviceNet device / EL6752-0010 must be checked. If the MacState is not equal zero, the DeviceNet slave is not participating correctly in the DeviceNet data exchange.

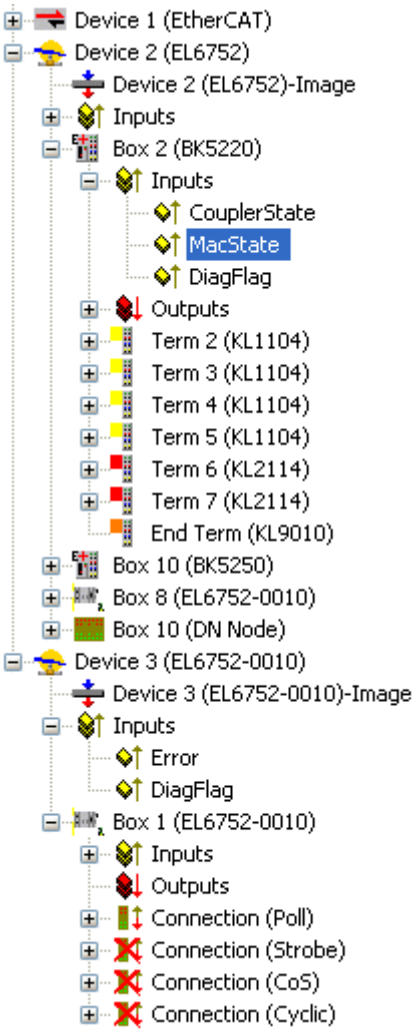


Fig. 72: MacState in the TwinCAT tree

MacState

0 = No error
1 = Station deactivated
2 = Station not exists
18 = Station ready
40 = Heartbeat Message not received
41 = Shutdown Message received
42 = Electronic Key Fault: Vendor Id
43 = Electronic Key Fault: Device Type

44 = Electronic Key Fault: Product Code
45 = Electronic Key Fault: Revision
46 = Fault while writing Start-Up Attributs
47 = wrong Produced IO-Data Size
48 = wrong Consumed IO-Data Size
49 = Idle Mode

7.3.2 DeviceNet® slave device / EL6752-0010 - DiagFlag

The DiagFlag indicates pending diagnostic data. Pending diagnostic data can be read via an AdsRead command.

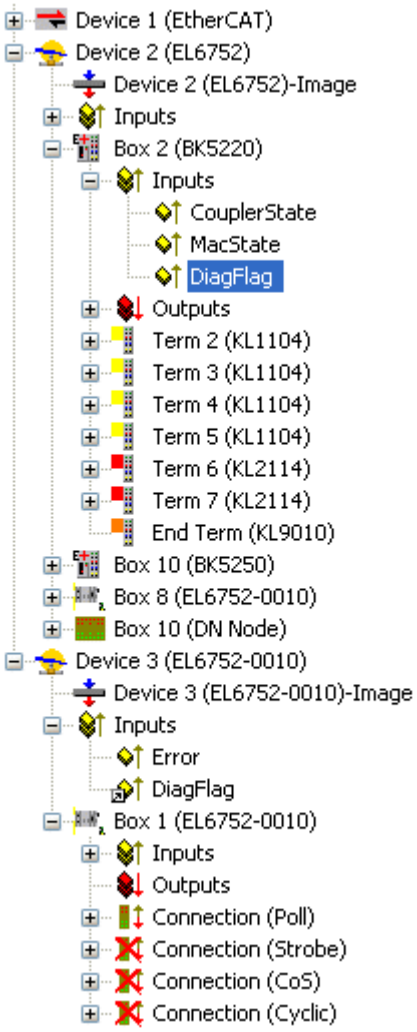


Fig. 73: DiagFlag in the TwinCAT tree

DiagFlag

0 = no diagnostic data are pending
1 = diagnostic data are pending and can be read via AdsRead services

7.3.3 Beckhoff DeviceNet® slave device - CouplerState

The CouplerState provides information on the terminal bus communication of the Beckhoff Bus Coupler. This information is available for Beckhoff BK52x0 Bus Couplers, devices from the IPxxxx-B520 IP Box-family and the IP Link family.

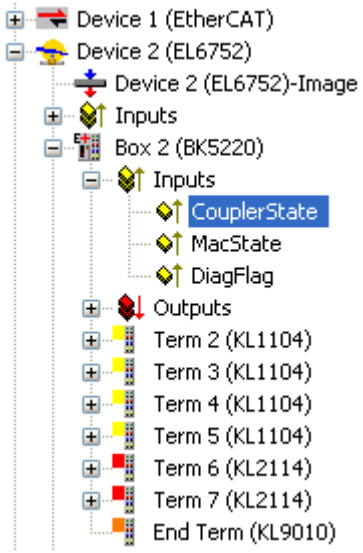


Fig. 74: CouplerState in the TwinCAT tree

CouplerState

0x00 = I/O Run
0x01 = I/O Error (KBus, IO or Terminal Error)
0x80 = I/O idle mode / fieldbus error, no output data are written
0x08= diagnostic information for an analog/special function terminal is pending. This function first hast to be activated at the couplers. The diagnostic data can then be read in the associated registers of the terminals, IP/IL modules

7.4 EL6752/-0010 - ADS Error Codes

The ADS error codes have the following meaning:

Error	Description
	Error during ADS/AMS data exchange
0x1001	Insufficient memory for AMS command
0x1101	Incorrect data length at StartFieldbus
0x1102	Incorrect DeviceState at StartFieldbus
0x1103	Device cannot change from INIT to RUN
0x1104	Incorrect AdsState in INIT state
0x1105	Incorrect DeviceState at StopFieldbus
0x1106	Device cannot change from STOP to RUN if a CDL is not defined
0x1107	Device cannot change from STOP to RUN if a box is not defined
0x1108	Incorrect data length at StartDataTransfer
0x1109	Incorrect DeviceState at StartDataTransfer
0x110A	Incorrect AdsState in STOP state
0x110B	Device cannot change from RUN to INIT
0x110C	Incorrect data length at StopDataTransfer
0x110D	Incorrect DeviceState at StopDataTransfer
0x1110	Incorrect AdsState in RUN state
0x1111	Loading the device parameters is only permitted in the INIT state
0x1112	Incorrect data length at SetDeviceState
0x1113	AddBox not allowed in INIT state
0x1114	Incorrect data length at AddBox
0x1115	DeleteBox not allowed in INIT state
0x1116	Incorrect IndexOffset at DeleteBox
0x1117	Incorrect data length at DeleteBox
0x1118	ReadBox only with AdsRead
0x1119	AddCdl not allowed in INIT state
0x111A	Incorrect data length at AddCdl
0x111B	DeleteCdl not allowed in INIT state
0x111C	Incorrect IndexOffset at DeleteCdl
0x111D	Incorrect data length at DeleteCdl
0x111E	Incorrect IndexGroup at AdsWrite
0x111F	Device parameters cannot be read

Error	Description
	Error during ADS/AMS data exchange
0x1120	Box parameters cannot be read
0x1121	Cdl parameters cannot be read
0x1122	DeleteBox or DeleteCdl only with AdsWrite
0x1123	ReadBox only possible in STOP state
0x1124	Incorrect IndexOffset at ReadBox
0x1125	Incorrect data length at ReadBox
0x1126	Incorrect IndexGroup at AdsRead
0x1127	AddDeviceNotification not allowed in INIT state
0x1128	DelDeviceNotification not allowed in INIT state
0x1129	IndexOffset too large during reading of the device diagnostic data
0x112B	IndexOffset too large during reading of the box diagnostic data
0x112F	Insufficient memory for ReadBox response
0x1201	AddCdl: CDL no. is too large
0x1202	DeleteCdl only possible when CDL is stopped
0x1203	DeleteCdl not possible as no CDL defined
0x1204	Cycle could not be completed within the internal watchdog time
0x1301	AddCdl: I/O access multiplier is too large
0x1302	AddCdl: Start cycle must be smaller than I/O access multiplier
0x1303	AddCdl: Incorrect data length for output area
0x1304	AddCdl: Incorrect data offset for output area

Error	Description
	Error during ADS/AMS data exchange
0x1305	AddCdl: Output area is already defined
0x1306	AddCdl: Incorrect data length for input area
0x1307	AddCdl: Incorrect data offset for input area
0x1308	AddCdl: Input area is already defined
0x1309	AddCdl: Incorrect area type
0x130A	AddCdl: BoxNo has not been defined with AddBox
0x130B	AddCdl: Incorrect action type
0x130C	AddCdl: Insufficient memory for poll list
0x130D	AddCdl: Insufficient memory for poll list array
0x130E	AddCdl: Insufficient memory for actions
0x130F	AddCdl: CdlNo already exists

Error	Description
	Error during ADS/AMS data exchange
0x1310	DeleteCdl: CDL is not stopped
0x1311	AddCdl: Insufficient memory for asynchronous transmit list
0x1312	AddCdl: Insufficient memory for synchronous receive list
0x1313	AddCdl: Insufficient memory for asynchronous receive list
0x1316	AddCdl: Insufficient memory for synchronous receive list
0x1318	AddCdl: Only slave action allowed
0x1319	AddCdl: Insufficient memory for slave list
0x1601	AddBox: BoxNo is too large
0x1602	AddBox: Insufficient memory for ADS StartUp telegram
0x1604	DeleteBox: Box is not stopped
0x1605	AddBox: Insufficient memory for CDL telegram
0x1606	AddBox: Number of CDL telegrams is too large
0x1607	BoxRestart: Box is not stopped
0x1608	BoxRestart: AdsWriteControl syntax error
0x1609	BoxRestart: Incorrect AdsState
0x160A	Syntax error in AdsWrite to box port
0x160B	AMS CmdId is not supported by box port
0x160E	AdsReadState is not supported by box port
0x160F	AddBox: Insufficient memory for the ADS interface
0x1610	AddBox: AMS channel is invalid
0x1611	Error communicating with an AMS box
0x1613	Error communicating with an AMS box: Incorrect offset
0x1614	Error communicating with an AMS box: Data packet is too large
0x1615	Error communicating with an AMS box: AMS command is too large
0x1616	Error communicating with an AMS box: First data packet is too large
0x1617	Error communicating with an AMS box: First offset is incorrect

Error	Description
	Error during ADS/AMS data exchange
0x1701	AddDeviceNotification: Length of device diagnostic data to small
0x1702	AddDeviceNotification: Length of device diagnostic data to large
0x1703	AddDeviceNotification: Length of box diagnostic data to small
0x1704	AddDeviceNotification: Length of box diagnostic data to large
0x1705	AddDeviceNotification: Box is not defined
0x1706	AddDeviceNotification: Incorrect IndexGroup
0x1707	AddDeviceNotification: No more resources for client
0x1708	DelDeviceNotification: Incorrect handle
0x1801	StartFieldbus: In equidistant operation, shift time + safety time + 2*PLL sync. time must be greater than the cycle time
0x1802	StartFieldbus: Cycle time is too large
0x1803	StartFieldbus: Cycle time is too large
0x1804	StartFieldbus: Shift time is too large
0x1805	StartFieldbus: PLL sync time is too large

Error	Description
	Error during ADS/AMS data exchange
0x1806	StartFieldbus: Safety time is too large
0x1807	StartFieldbus: Cycle times shorter than 1 ms must be integral divisors of 1 ms
0x1A01	Memory could not be allocated from the huge heap, because it is larger than 0x8000 bytes
0x1A02	Memory could not be allocated from the near heap, because it is larger than 0x1000 bytes
0x1A03	Memory could not be allocated from the huge heap, because it is 0 bytes
0x1A04	Memory could not be allocated from the near heap, because it is 0 bytes
	Error during initialization of the DeviceNet® configuration
0x2001 .. 0x2xxx	
	Error during explicit DeviceNet® data exchange
0x2300	GENERR_RESUNAVAILABLE
0x2301	ADSERR_DEVICE_SRVNOTSUPP
0x2302	GENERR_INVALATTRVAL
0x2303	GENERR_ALRERADYINREQU
0x2304	GENERR_OBJECTSTATECONF
0x2305	GENERR_ATTRNOTSETABLE
0x2306	GENERR_PRIVVIOLATION
0x2307	GENERR_REPLDATTOOLARGE
0x2308	GENERR_NOTENOUGHDATA
0x2309	GENERR_ATTRNOTSUPP
0x230A	GENERR_TOOMUCHDATA
0x230B	GENERR_OBJECTNOTEXIST
0x230C	GENERR_NOSTOREATTRDATA
0x230D	GENERR_STOREOPFAIL
0x230E	GENERR_VENDORSPEC
0x230F	GENERR_INVALPARAM
0x2310	GENERR_INVALMEMBERID
0x2311	GENERR_MEMBERNOTSET
0x2312	ADSERR_DEVICE_SYMBOLNOTFOUND
0x2313	GENERR_OBJECTSTATECONF

7.5 DeviceNet® / CAN Trouble Shooting

Error Frames

One sign of errors in the CAN wiring, the address assignment or the setting of the baud rate is an increased number of error frames: the diagnostic LEDs then show *Warning Limit exceeded* or *Bus-off state entered*.

> **DeviceNet® / CAN network analysis**
>
> CAN warning limit exceeded, passive error or bus-off state are indicated first of all at the node that has detected the most errors. These nodes are not necessarily the cause for the occurrence of error frames! If, for instance, one node contributes unusually heavily to the bus traffic (e.g. because it is the only one with analog inputs, the data for which triggers event-driven messages at a high rate), then the probability of its telegrams being damaged increases. Its error counter will, correspondingly, be the first to reach a critical level.

MAC ID / baud rate setting

Duplicate allocation of node addresses / MAC IDs must be avoided.

Test 1

Check MAC ID. If the DeviceNet® communication works at least temporarily and all devices support the duplicate MAC ID check, the address assignment can also be checked by logging the duplicate MAC ID check messages when the devices are switched on, although this procedure does not detect incorrect allocation of node addresses.

Test 2

Check that the same baud rate has been set everywhere.

Testing the DeviceNet®/CAN cabling

These tests should not be carried out if the network is active: No communication should take place during the tests. The following tests should be carried out in the stated sequence, because some of the tests assume that the previous test was successful. Not all the tests are generally necessary.

Network terminator and signal leads

The nodes should be switched off or the CAN cable unplugged for this test, because the results of the measurements can otherwise be distorted by the active CAN transceiver.

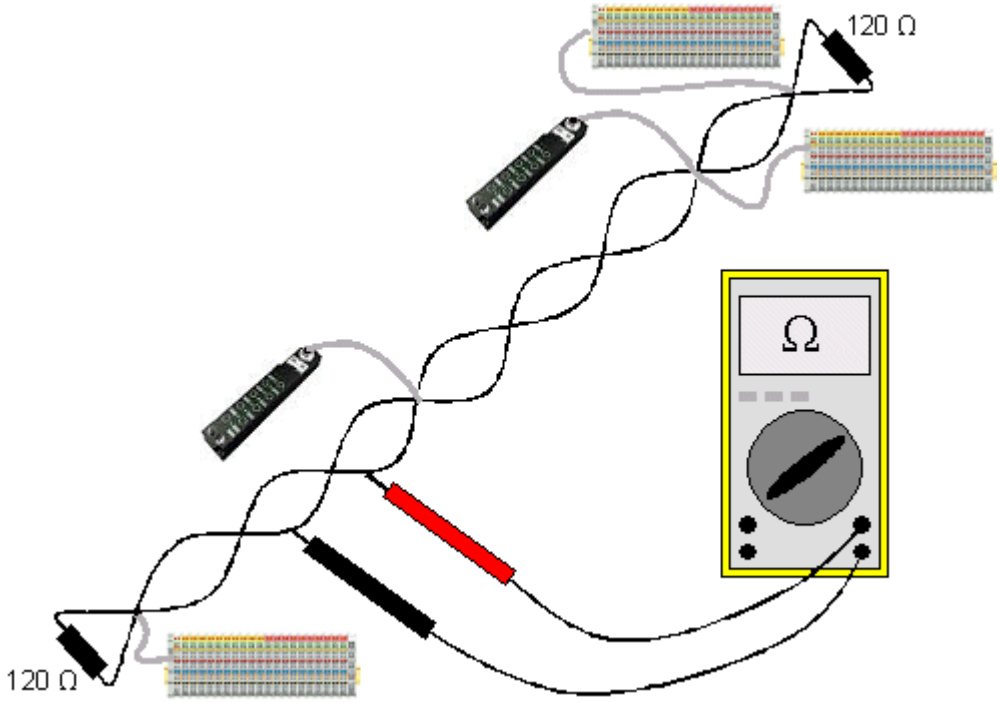


Fig. 75: Wiring diagram for test setup

Test 3

Determine the resistance between CAN high and CAN low - at each device, if necessary.

If the measured value is greater than 65 Ohms, it indicates the absence of a terminating resistor or a break in a signal lead. If the measured value is less than 50 Ohms, look for a short circuit between the CAN lines, more than the correct number of terminating resistors, or faulty transceivers.

Test 4

Check for a short circuit between the CAN ground and the signal leads, or between the screen and signal leads.

Test 5

Remove the earth connection from the CAN ground and screen. Check for a short circuit between the CAN ground and screen.

Topology

The possible cable length in CAN networks depends heavily on the selected baud rate. CAN will tolerate short drop lines - although this again depends on the baud rate. The maximum permitted drop line length should not be exceeded. The length of cable that has been installed is often underestimated - estimates can even be a factor of 10 less than the actual length. The following test is therefore recommended:

Test 6

Measure the drop line lengths and the total bus length (a rough estimate is not sufficient) and compare the values with the topology rules (depending on the baud rate).

Screening and earthing

The power supply and the screen should be carefully earthed at the power supply unit, once only and with low resistance. At all connecting points, branches and so forth the screen of the CAN cable (and possibly the CAN GND) must also be connected, as well as the signal leads. In the Beckhoff IP20 Bus Couplers, the screen is grounded for high frequencies via an R/C element.

Test 7

Use a DC ammeter (16 A max.) to measure the current between the power supply ground and the screen at the end of the network most remote from the power supply unit. An equalization current should be present. If there is no current, then either the screen is not connected all the way through, or the power supply unit is not properly earthed. If the power supply unit is somewhere in the middle of the network, the measurement should be performed at both ends. When appropriate, this test can also be carried out at the ends of the drop line.

Test 8

Interrupt the screen at a number of locations and measure the connection current. If current is flowing, the screen is earthed at more than one place, creating a ground loop.

Potential differences

The screen must be connected all the way through for this test, and must not be carrying any current - this has previously been tested.

Test 9

Measure and record the voltage between the screen and the power supply ground at each node. The maximum potential difference between any two devices should be less than 5 volts.

Detect and localize faults

The "low-tech approach" rule is the best localisation method: disconnect parts of the network, and observe when the fault disappears.

But: the approach based on this method is inadequate in the event of problems such as excessive potential differences, ground loops, EMC and signal corruption, since in many cases making the network smaller solves the problem notwithstanding the fact that the "missing" component may not have caused it. The bus load also changes as the network is reduced in size, which can mean that external interference "hits" CAN telegrams less often.

Diagnosis with an oscilloscope is not usually successful: even when they are in good condition, CAN signals can look really chaotic. It may be possible to trigger on error frames using a storage oscilloscope - this type of diagnosis, however, is only possible for expert technicians.

Protocol problems

In rare cases, protocol problems (e.g. faulty or incomplete DeviceNet® implementation, unfavorable timing at boot up, etc.) can be the cause of faults. In this case it is necessary to trace the bus traffic for evaluation by DeviceNet® experts - the Beckhoff support team [▶ 133] can help here.
A free channel on a Beckhoff FC5102 CANopen PCI card is appropriate for such a trace - Beckhoff make the necessary trace software available on the internet. Alternatively, it is of course possible to use a normal commercial CAN analysis tool.

8 Appendix

8.1 EtherCAT AL Status Codes

For detailed information please refer to the EtherCAT system description.

8.2 Firmware compatibility

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

Note

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

NOTICE

Risk of damage to the device!

Pay attention to the instructions for firmware updates on the separate page [▶ 120]. 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!

EL6752			
Hardware (HW)	**Firmware (FW)**	**Revision no.**	**Release date**
06 - 24*	07	EL6752-0000-0016	2008/06
	08		2008/11
	09		2010/05
		EL6752-0000-0017	2011/10
	10		2012/01
		EL6752-0000-0018	2012/10
	11	EL6752-0000-0019	2014/07
	12	EL6752-0000-0020	2014/06
	13		2015/02
	14		2019/04
	15		2020/06
	16*		2025/04

EL6752-0010			
Hardware (HW)	**Firmware (FW)**	**Revision no.**	**Release date**
06 - 24*	06	EL6752-0010-0016	2008/04
	07		2008/06
	08		2008/11
		EL6752-0010-0017	2011/10
	09		2012/01
	10		2012/05
		EL6752-0010-0018	2012/10
	11	EL6752-0010-0019	2014/07
	12	EL6752-0010-0020	2014/06
	13		2015/02
	14		2019/04
	15		2020/03
	16*		2024/02

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

8.3 Firmware Update EL/ES/EM/ELM/EP/EPP/ERPxxxx

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

NOTICE

Only use TwinCAT 3 software!

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

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

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

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

Storage locations

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

- Each EtherCAT slave has a device description, consisting of identity (name, product code), timing specifications, communication settings, etc.
  This device description (ESI; EtherCAT Slave Information) can be downloaded from the Beckhoff website in the download area as a zip file and used in EtherCAT masters for offline configuration, e.g. in TwinCAT.
  Above all, each EtherCAT slave carries its device description (ESI) electronically readable in a local memory chip, the so-called **ESI EEPROM**. When the slave is switched on, this description is loaded locally in the slave and informs it of its communication configuration; on the other hand, the EtherCAT master can identify the slave in this way and, among other things, set up the EtherCAT communication accordingly.

NOTICE

Application-specific writing of the ESI-EEPROM

The ESI is developed by the device manufacturer according to ETG standard and released for the corresponding product.
- Meaning for the ESI file: Modification on the application side (i.e. by the user) is not permitted.
- Meaning for the ESI EEPROM: Even if a writeability is technically given, the ESI parts in the EEPROM and possibly still existing free memory areas must not be changed beyond the normal update process. Especially for cyclic memory processes (operating hours counter etc.), dedicated memory products such as EL6080 or IPC's own NOVRAM must be used.

- Depending on functionality and performance EtherCAT slaves have one or several local controllers for processing I/O data. The corresponding program is the so-called **firmware** in *.efw format.
- In some EtherCAT slaves the EtherCAT communication may also be integrated in these controllers. In this case the controller is usually a so-called **FPGA** chip with *.rbf firmware.

Customers can access the data via the EtherCAT fieldbus and its communication mechanisms. Acyclic mailbox communication or register access to the ESC is used for updating or reading of these data.

The TwinCAT System Manager offers mechanisms for programming all three parts with new data, if the slave is set up for this purpose. Generally the slave does not check whether the new data are suitable, i.e. it may no longer be able to operate if the data are unsuitable.

Simplified update by bundle firmware

The update using so-called **bundle firmware** is more convenient: in this case the controller firmware and the ESI description are combined in a *.efw file; during the update both the firmware and the ESI are changed in the terminal. For this to happen it is necessary

- for the firmware to be in a packed format: recognizable by the file name, which also contains the revision number, e.g. 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

NOTICE

Risk of damage to the device!

✓ Note the following when downloading new device files

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

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

c) The power supply must adequately dimensioned. The signal level must meet the specification.

⇨ In the event of malfunctions during the update process the EtherCAT device may become unusable and require re-commissioning by the manufacturer.

8.3.1 Device description ESI file/XML

NOTICE

Attention regarding update of the ESI description/EEPROM

Some slaves have stored calibration and configuration data from the production in the EEPROM. These are irretrievably overwritten during an update.

The ESI device description is stored locally on the slave and loaded on start-up. Each device description has a unique identifier consisting of slave name (9 characters/digits) and a revision number (4 digits). Each slave configured in the System Manager shows its identifier in the EtherCAT tab:

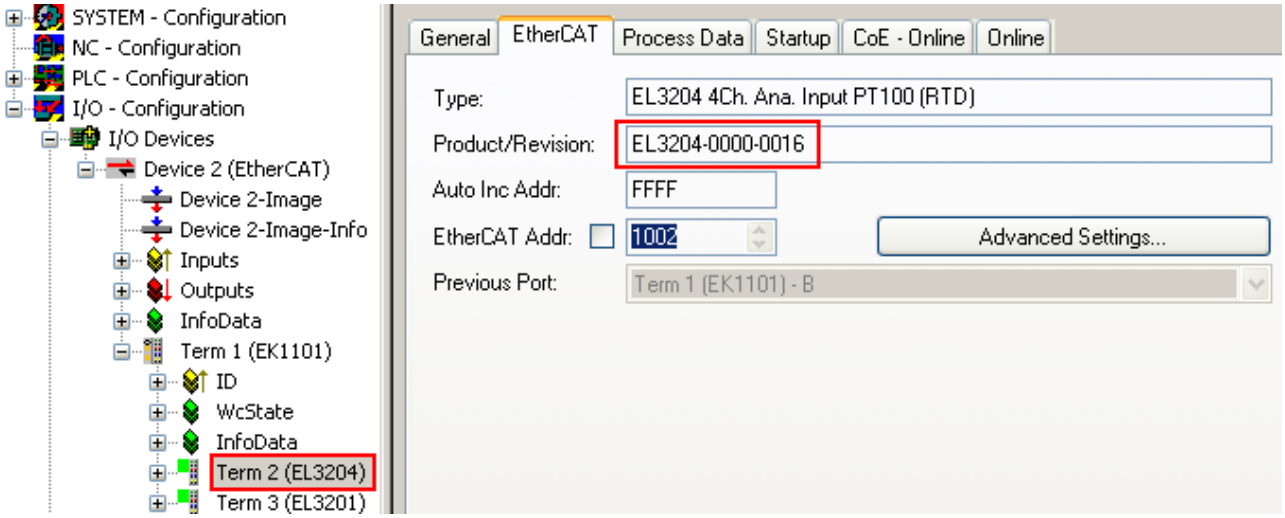


Fig. 76: Device identifier consisting of name EL3204-0000 and revision -0016

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:

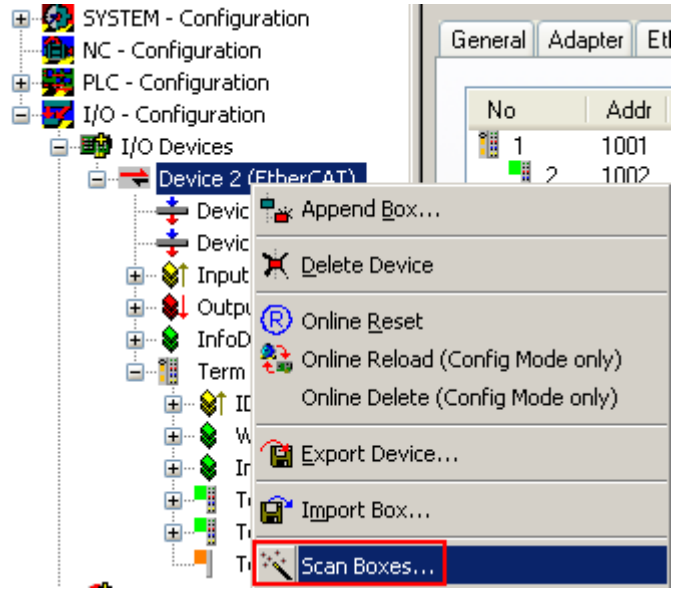


Fig. 77: Scan the subordinate field by right-clicking on the EtherCAT device

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

Fig. 78: Configuration is identical

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

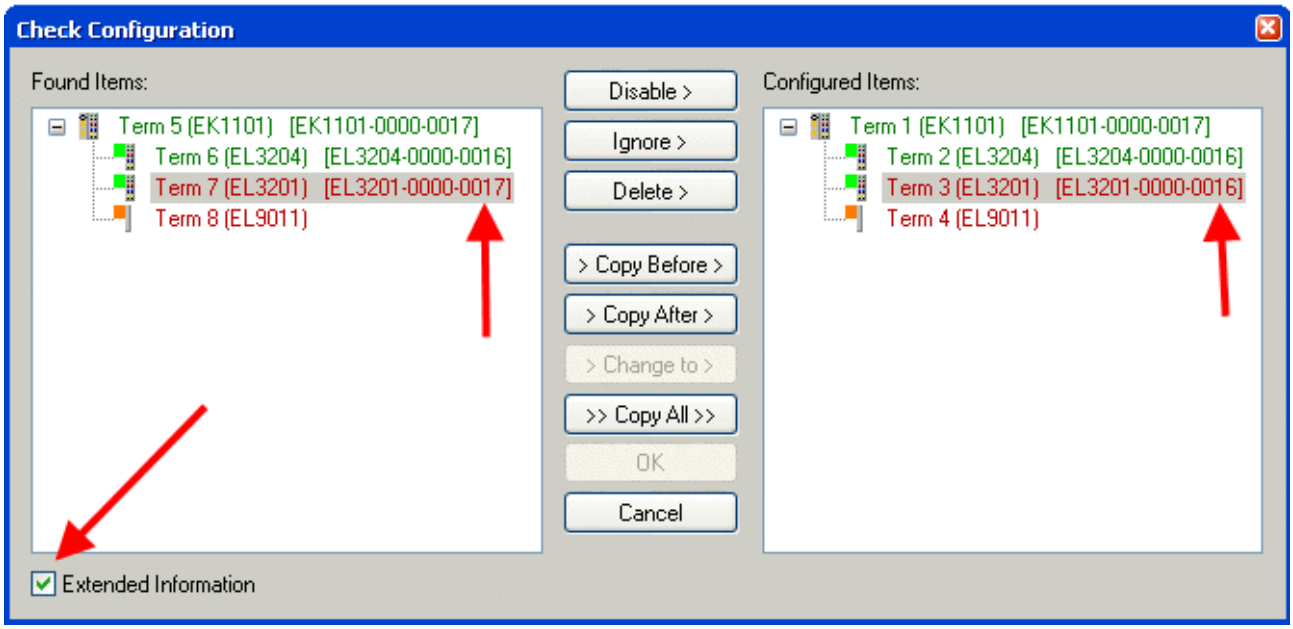


Fig. 79: Change dialog

In this example in Fig. *Change dialog*, an EL3201-0000-**0017** was found, while an EL3201-0000-**0016** was configured. In this case the configuration can be adapted with the *Copy Before* button. The *Extended Information* checkbox must be set in order to display the revision.

Changing the ESI slave identifier

The ESI/EEPROM identifier can be updated as follows under TwinCAT:

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

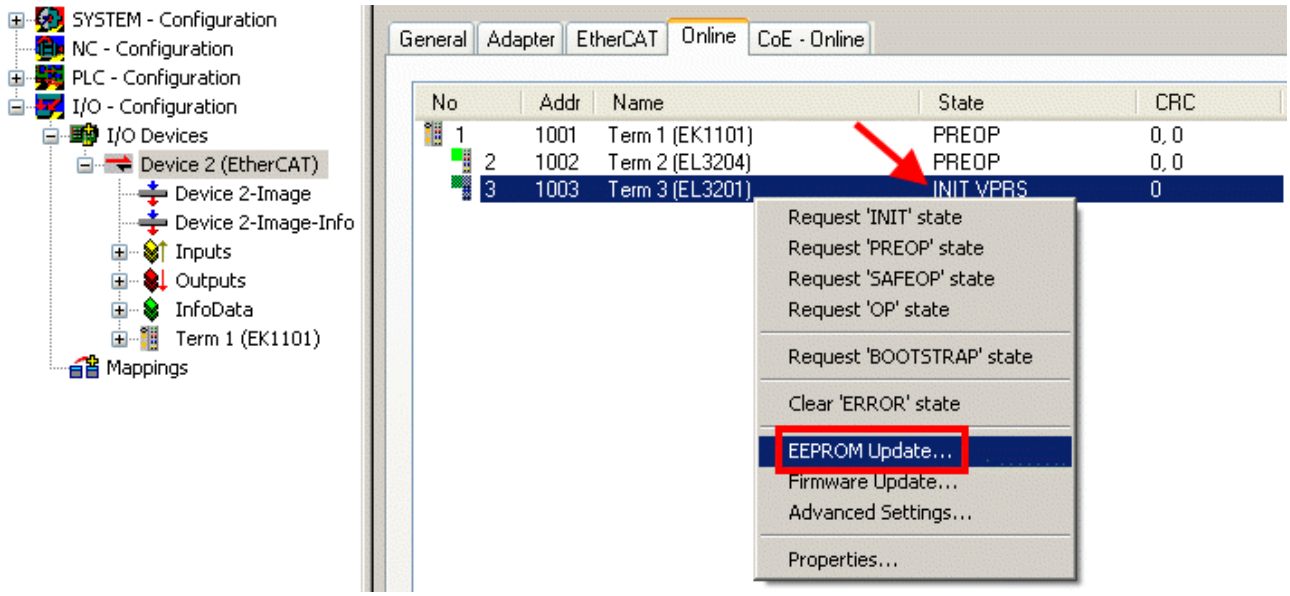


Fig. 80: EEPROM Update

The new ESI description is selected in the following dialog, see Fig. *Selecting the new ESI.* The checkbox *Show Hidden Devices* also displays older, normally hidden versions of a slave.

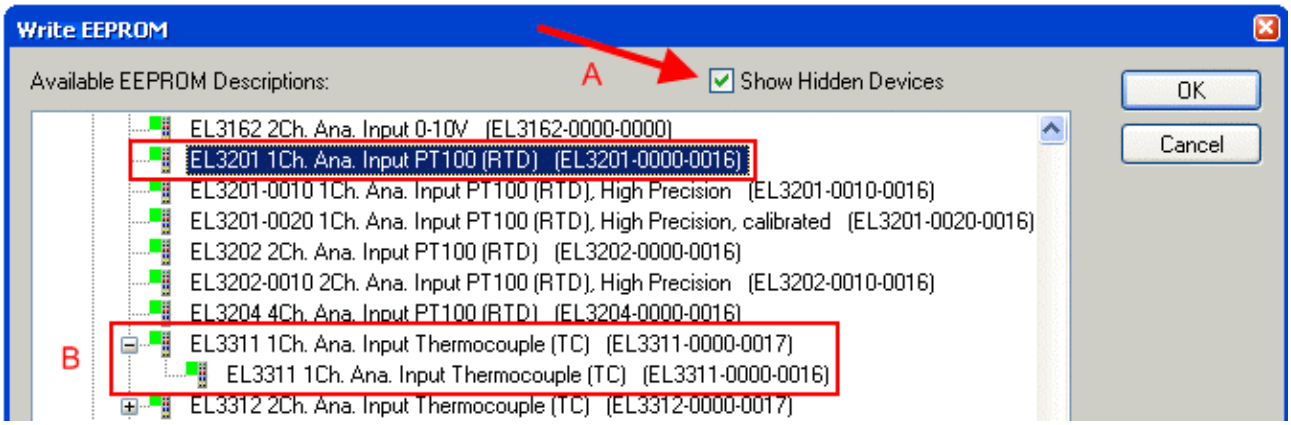


Fig. 81: Selecting the new ESI

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

The change only takes effect after a restart.

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

8.3.2 Firmware explanation

Determining the firmware version

Determining the version via the TwinCAT System Manager

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

CoE Online and Offline CoE

Two CoE directories are available:
• **online**: This is offered in the EtherCAT slave by the controller, if the EtherCAT slave supports this. This CoE directory can only be displayed if a slave is connected and operational.
• **offline**: The EtherCAT Slave Information ESI/XML may contain the default content of the CoE. This CoE directory can only be displayed if it is included in the ESI (e.g. “Beckhoff EL5xxx.xml”).

The Advanced button must be used for switching between the two views.

In Fig. *Display of EL3204 firmware version* the firmware version of the selected EL3204 is shown as 03 in CoE entry 0x100A.

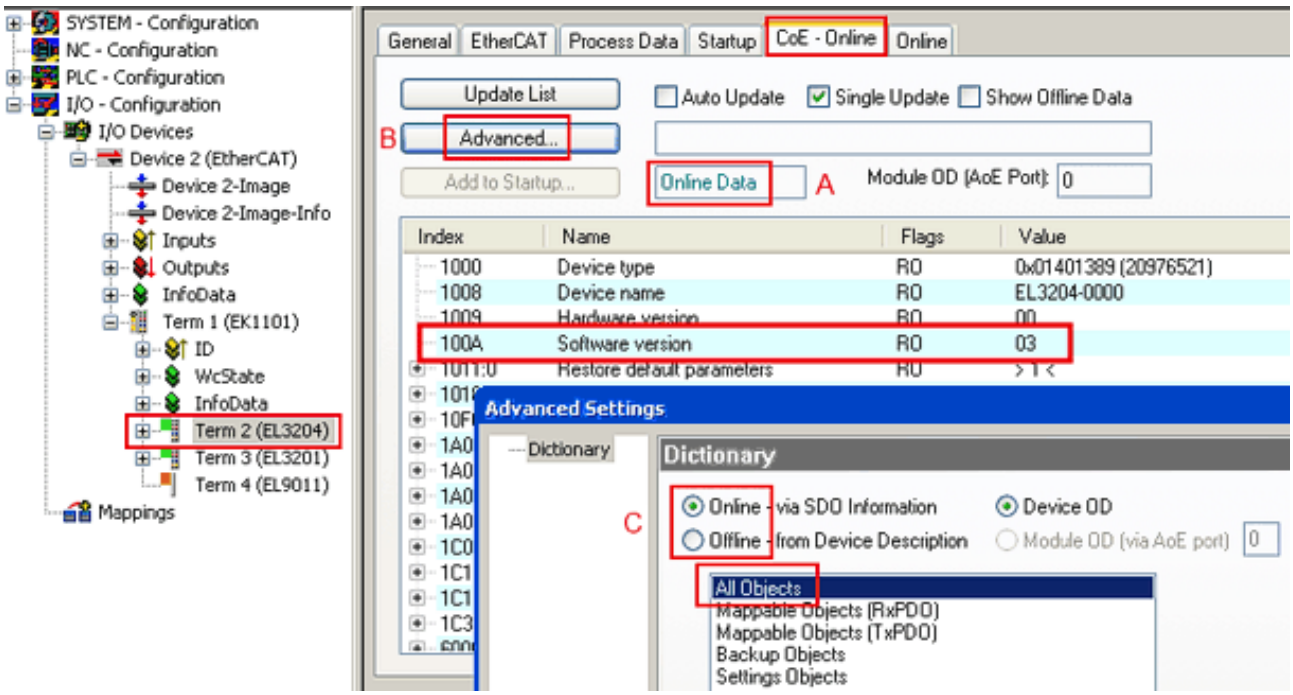


Fig. 82: Display of EL3204 firmware version

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

8.3.3 Updating controller firmware *.efw

CoE directory

The Online CoE directory is managed by the controller and stored in a dedicated EEPROM, which is generally not changed during a firmware update.

Switch to the *Online* tab to update the controller firmware of a slave, see Fig. *Firmware Update.*

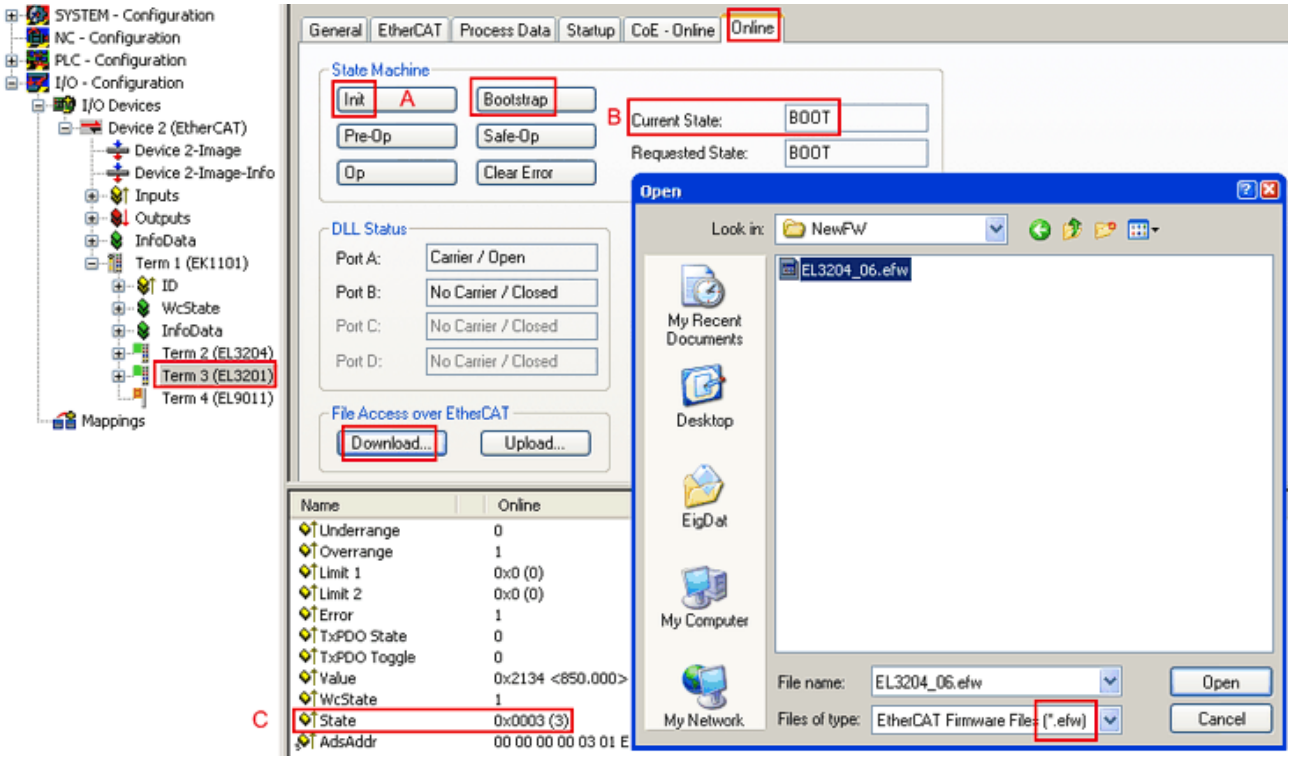
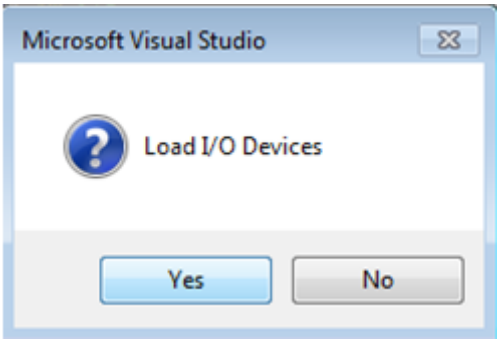
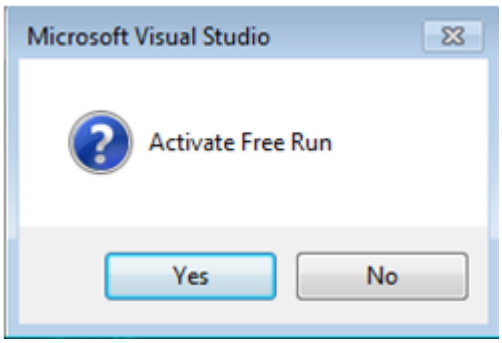


Fig. 83: Firmware Update

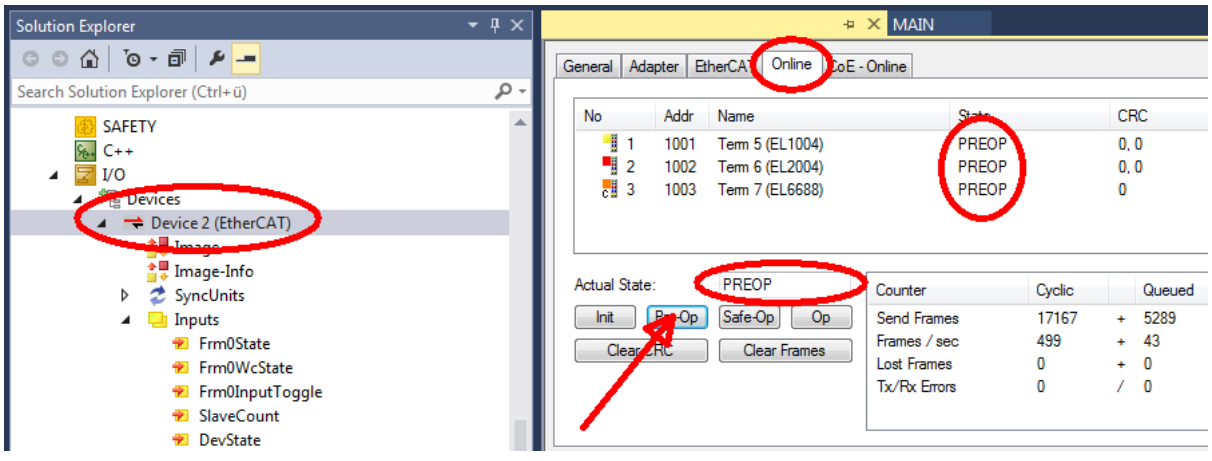
Proceed as follows, unless instructed otherwise by Beckhoff support. Valid for TwinCAT 2 and 3 as EtherCAT master.

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



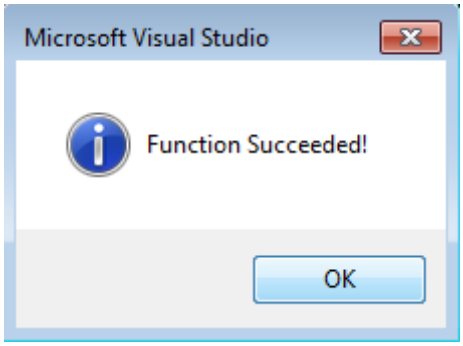


- Switch EtherCAT Master to PreOP



- Switch slave to INIT (A)
- Switch slave to BOOTSTRAP

- Check the current status (B, C)
- Download the new *efw file (wait until it ends). A password will not be necessary usually.



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

8.3.4 FPGA firmware *.rbf

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

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

The firmware version number included in the terminal serial number contains both firmware components. If one of these firmware components is modified this version number is updated.

Determining the version via the TwinCAT System Manager

The TwinCAT System Manager indicates the FPGA firmware version. Click on the Ethernet card of your EtherCAT strand (Device 2 in the example) and select the *Online* tab.

The *Reg:0002* column indicates the firmware version of the individual EtherCAT devices in hexadecimal and decimal representation.

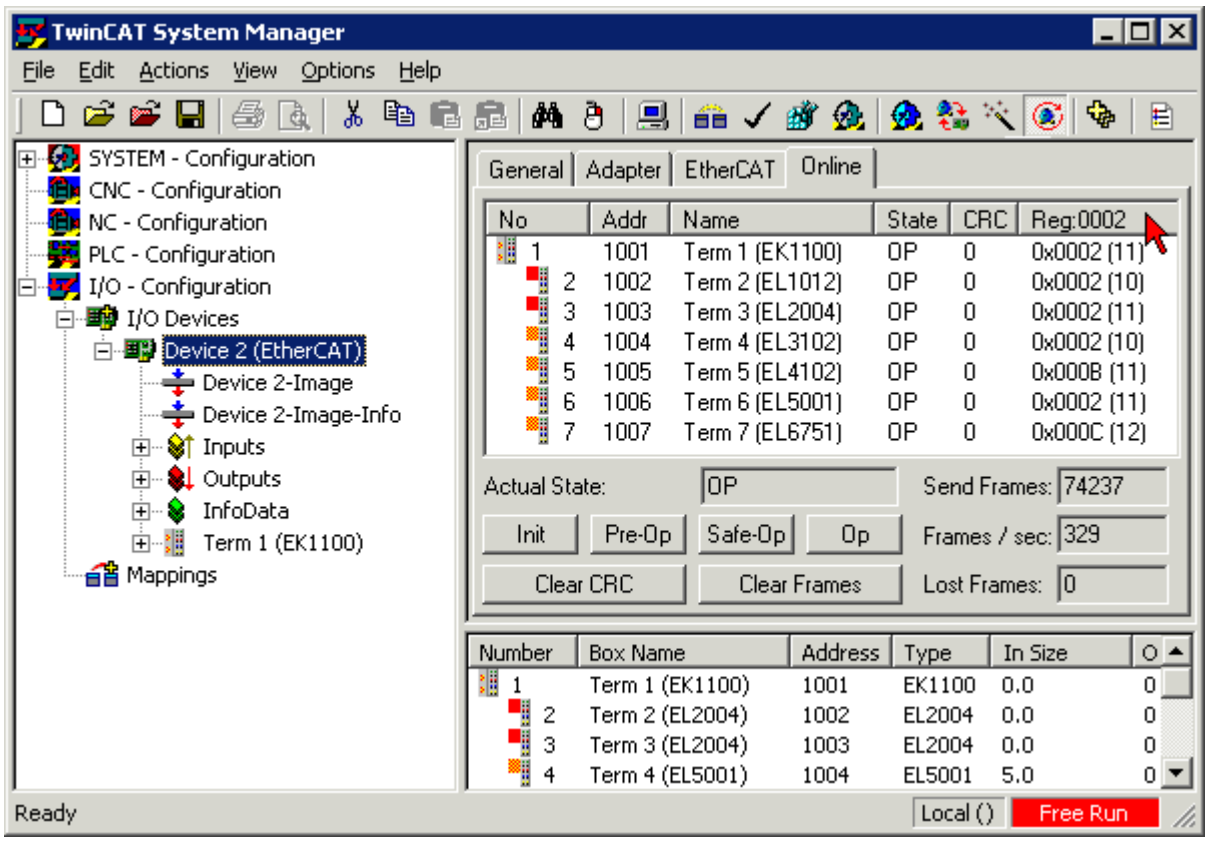


Fig. 84: FPGA firmware version definition

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

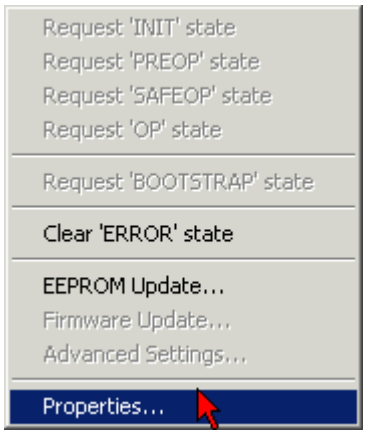


Fig. 85: Context menu *Properties*

The *Advanced Settings* dialog appears where the columns to be displayed can be selected. Under *Diagnosis/***Online View** select the *'0002 ETxxxx Build'* check box in order to activate the FPGA firmware version display.

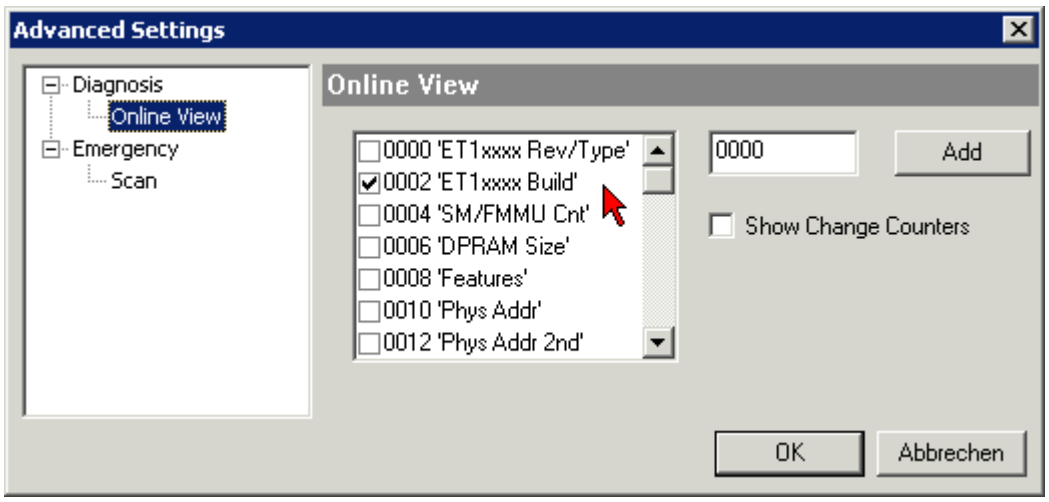


Fig. 86: Dialog *Advanced Settings*

Update

For updating the FPGA firmware

- of an EtherCAT coupler the coupler must have FPGA firmware version 11 or higher;
- of an E-Bus Terminal the terminal must have FPGA firmware version 10 or higher.

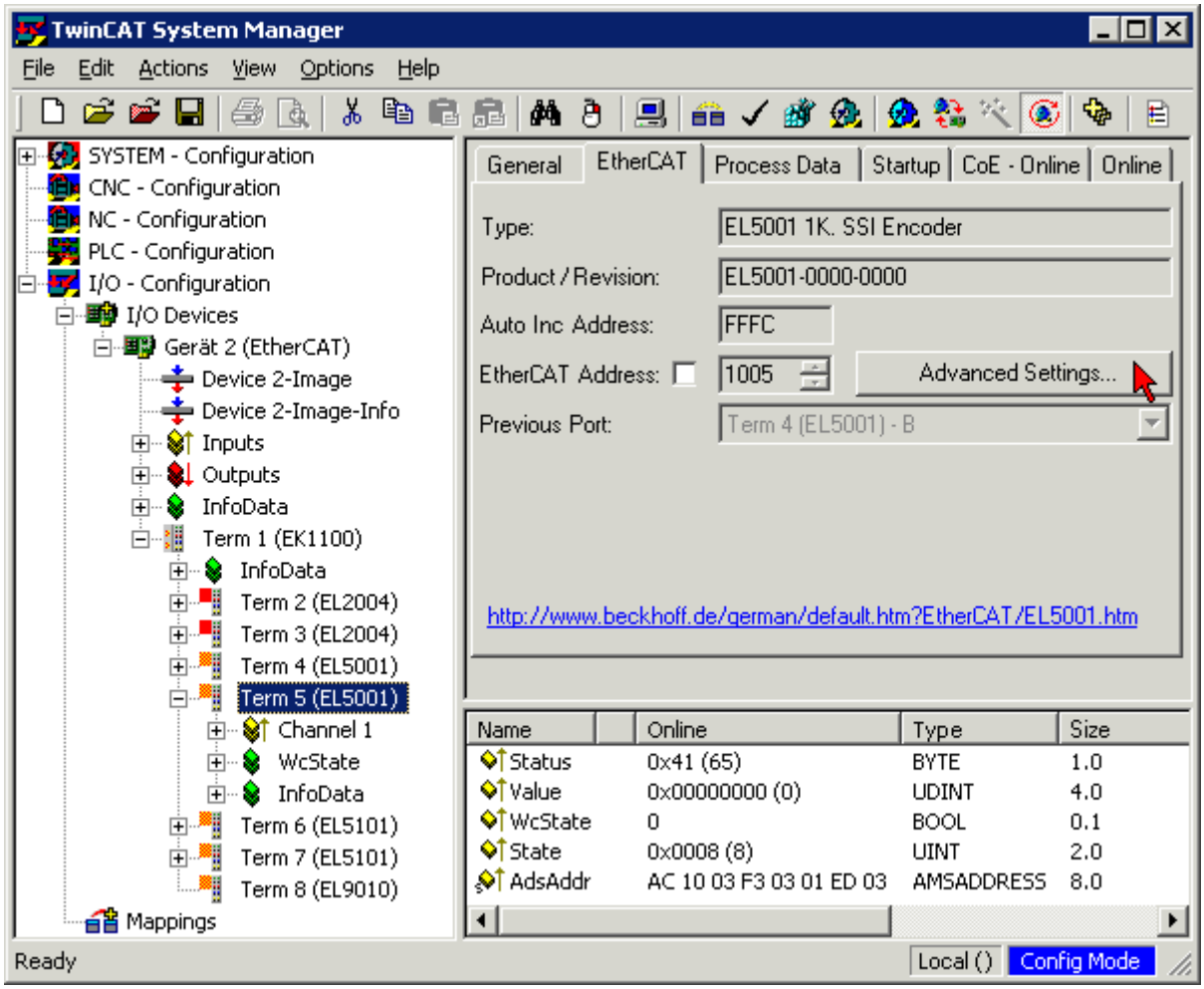
Older firmware versions can only be updated by the manufacturer!

Updating an EtherCAT device

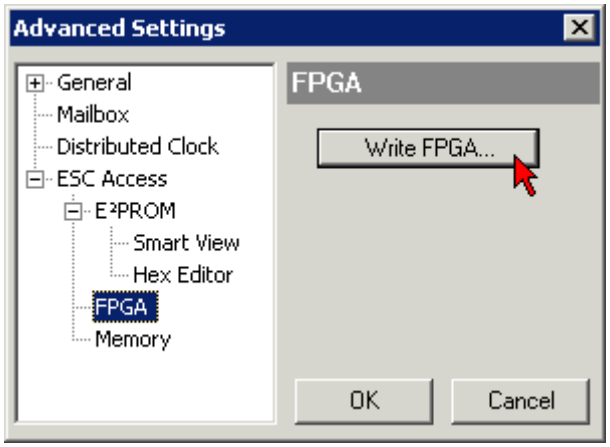
The following sequence order have to be met if no other specifications are given (e.g. by the Beckhoff support):

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

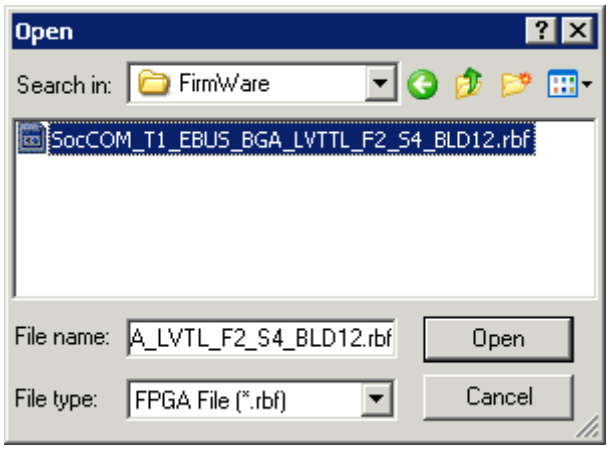
- In the TwinCAT System Manager select the terminal for which the FPGA firmware is to be updated (in the example: Terminal 5: EL5001) and
  click the *Advanced Settings* button in the *EtherCAT* tab:



- The *Advanced Settings* dialog appears. Under *ESC Access/E²PROM*/FPGA click on *Write FPGA* button:



- Select the file (*.rbf) with the new FPGA firmware, and transfer it to the EtherCAT device:



- Wait until download ends
- Switch slave current less for a short time (don't pull under voltage!). In order to activate the new FPGA firmware a restart (switching the power supply off and on again) of the EtherCAT device is required.
- Check the new FPGA status

NOTICE

Risk of damage to the device!

A download of firmware to an EtherCAT device must not be interrupted in any case! If you interrupt this process by switching off power supply or disconnecting the Ethernet link, the EtherCAT device can only be recommissioned by the manufacturer!

8.3.5 Simultaneous updating of several EtherCAT devices

The firmware and ESI descriptions of several devices can be updated simultaneously, provided the devices have the same firmware file/ESI.

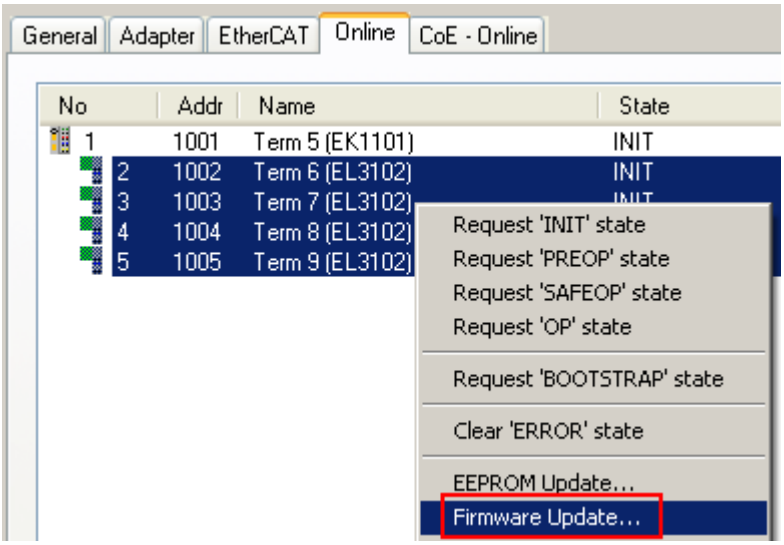


Fig. 87: Multiple selection and firmware update

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

8.4 Abbreviations

Abbreviation	Description
CAN	Controller Area Network. Serial bus system standardised in ISO 11898 that is used as the basic technology for CANopen
CiA	CAN in Automation e.V.. An international association of manufacturers and users based in Erlangen, Germany.
CoB	Communication Object. A CAN telegram with up to 8 data bytes.
CoB-ID	Communication Object Identifier. Telegram address (not to be confused with the node address). CANopen uses the 11-bit identifier according to CAN 2.0A.
CoE	CANopen over EtherCAT
ESC	EtherCAT Slave Controller
FBM	Fieldbus master
MC	Motion Control
NMT	Network Management. One of the service primitives of the CANopen specification. Network management is used to initialize the network and to monitor nodes.
OP	OPERATIONAL
PDO	Process Data Object. A CAN telegram for the transfer of process data (e.g. I/O data).
PREOP	PRE OPERATIONAL
RxPDO	Receive PDO. PDOs are always identified from the point of view of the device under consideration. Thus a TxPDO with input data from an I/O module becomes an RxPDO from the controller's point of view.
SAFEOP	SAFE OPERATIONAL
SDO	Service Data Object. A CAN telegram with a protocol for communication with data in the object directory (typically parameter data).
SI	Subindex
SM	SyncManager
SoE	Servo Profile over EtherCAT
TxPDO	Transmit PDO (named from the point of view of the CAN node).

8.5 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on her internet pages: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Support

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

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

Hotline:	+49 5246 963 157
e-mail:	support@beckhoff.com
web:	www.beckhoff.com/support

Service

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

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

Hotline:	+49 5246 963 460
e-mail:	service@beckhoff.com
web:	www.beckhoff.com/service

Headquarters Germany

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Germany

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

Trademark statements

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

Third-party trademark statements

CANopen and CANopen FD are registered trademarks of CAN in AUTOMATION - International Users and Manufacturers Group e.V.

DeviceNet and EtherNet/IP are trademarks of ODVA, Inc.

PCI Express®, PCIe®, PCI™ and PCI HOT PLUG™ are trademarks or registered trademarks and/or service marks of PCI-SIG.

More Information:
www.beckhoff.com/EL6752

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