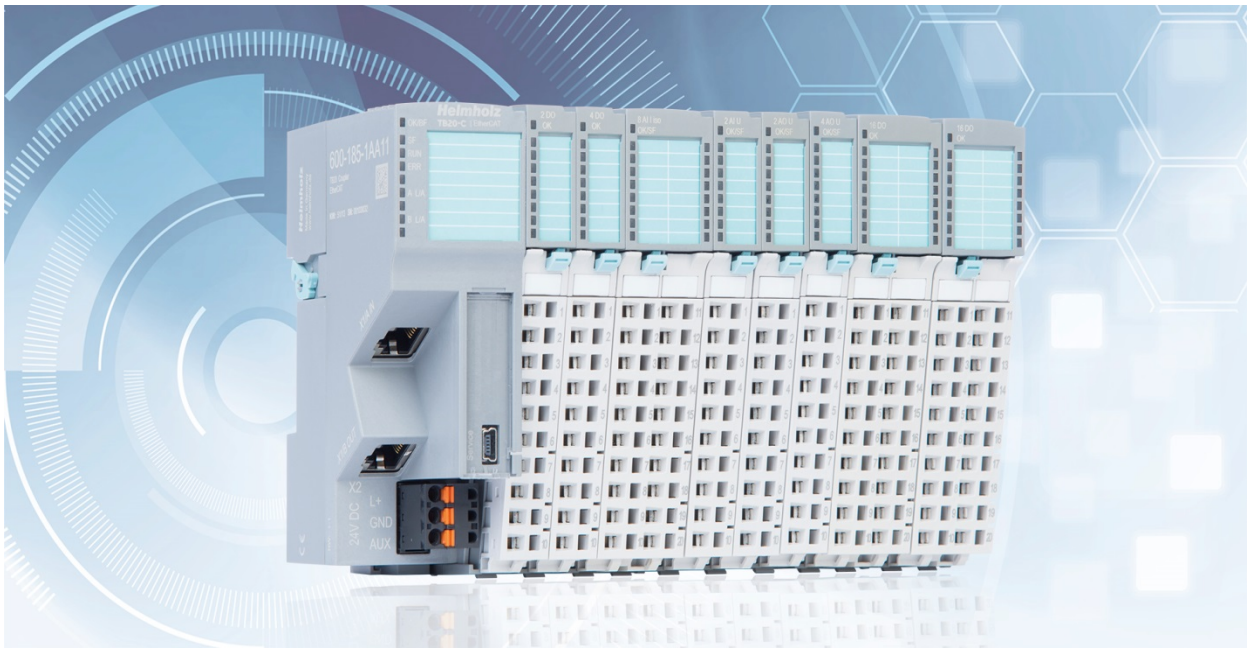
Helmholz
COMPATIBLE WITH YOU



TB20 – EtherCAT® bus coupler

Manual

Version 02 / 03.07.2019

Manual order number: 960-185-1AA11/en

Helmholz GmbH & Co. KG | Hannberger Weg 2 | 91091 Großenseebach | Germany
Phone +49 9135 7380-0 | Fax +49 9135 7380-110 | info@helmholz.de | www.helmholz.de

Notes

To download the latest version of this manual, please visit our website at www.helmholz.de.

We welcome all ideas and suggestions.

Contents

1 General

This operating manual applies only to devices, assemblies, software, and services of Helmholz GmbH & Co. KG.

1.1 Target audience for this manual

This description is only intended for trained personnel qualified in control and automation engineering who are familiar with the applicable national standards. For installation, commissioning, and operation of the components, compliance with the instructions and explanations in this operating manual is essential.

Configuration, execution, and operating errors can interfere with the proper operation of the TB20 devices and result in personal injury, as well as property or environmental damage. Only suitably qualified personnel may operate the TB20 devices!

Qualified personnel must ensure that the application and use of the products described meet all the safety requirements, including all relevant laws, regulations, provisions, and standards.

1.2 Safety instructions

The safety instructions must be observed in order to prevent harm or damage to persons, other living creatures, material goods, and the environment. The safety notes indicate possible hazards and provide information about how hazardous situations can be prevented.

1.3 Note symbols and signal words in the manual

HAZARD

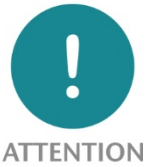
If the hazard warning is ignored, there is an imminent danger to life and health of people from electrical voltage.

WARNING

If the hazard warning is ignored, there is a probable danger to life and health of people from electrical voltage.

CAUTION

If the hazard warning is ignored, people can be injured or harmed.



ATTENTION

Draws attention to sources of error that can damage equipment or the environment.

NOTE

Gives an indication for better understanding or preventing errors.

1.4 Intended use

The TB20 I/O system is an open, modular, and distributed peripheral system designed to be mounted on a 35 mm DIN rail.

Communication with a higher-level control system is via a bus system / network and a TB20 bus coupler. Up to 64 modules from the TB20 range can be set up on a bus coupler. The bus couplers support hot plug for replacing modules during ongoing operation.

All components are supplied with a factory hardware and software configuration. The user must carry out the hardware and software configuration for the conditions of use. Modifications to hardware or software configurations that extend beyond the documented options are not permitted and nullify the liability of Helmholz GmbH & Co. KG.

The TB20 devices should not be used as the only means for preventing hazardous situations on machinery and equipment.

Successful and safe operation of the TB20 devices requires proper transport, storage, installation, assembly, installation, commissioning, operation, and maintenance.

The ambient conditions provided in the technical specifications must be adhered to.

The TB20 systems have a protection rating of IP20 and must have a control box/cabinet fitted to protect against environmental influences in an electrical operating room. To prevent unauthorized access, the doors of control boxes/cabinets must be closed and possibly locked during operation.

HAZARD

TB20 devices can be equipped with modules that can carry dangerously high voltages. The voltages connected to the TB20 devices can result in hazards during work on the TB20 devices.

1.5 Improper use

WARNING

The consequences of improper use may include personal injury to the user or third parties, as well as property damage to the control system, the product, or the environment. Use TB20 devices only as intended!

1.6 Installation

1.6.1 Access restriction

The modules are open operating equipment and must only be installed in electrical equipment rooms, cabinets, or housings. Access to the electrical equipment rooms, cabinets, or housings must only be possible using a tool or key, and access should only be granted to trained or authorized personnel.

1.6.2 Electrical installation

Regional safety regulations are to be observed.

TB20 devices can be equipped with modules that can carry dangerously high voltages. The voltages connected to the TB20 devices can result in hazards during work on the TB20 devices.

1.6.3 Protection against electrostatic discharges

To prevent damage through electrostatic discharges, the following safety measures are to be followed during assembly and service work:

- Never place components and modules directly on plastic items (such as polystyrene, PE film) or in their vicinity.
- Before starting work, suitable measures are to be taken to discharge static electricity.
- Only work with discharged tools.
- Do not touch components and assemblies on contacts.

1.6.4 Overcurrent protection

To protect the TB20 and the supply line, a *slow-blowing 8 A* line protection fuse is required.

1.6.5 EMC protection

To ensure electromagnetic compatibility (EMC) in your control cabinets in electrically harsh environments, the known rules of EMC-compliant configuration are to be observed in the design and construction.

1.6.6 Operation

Only operate the TB20 in flawless condition. The permissible operating conditions and performance limits must be adhered to. Retrofits, changes, or modifications to the device are strictly forbidden.

The TB20 is a piece of operating equipment intended for use in industrial plants. During operation, the TB20 can carry dangerous voltages. During operation, all covers on the unit and the installation must be closed in order to ensure protection against contact.

1.6.7 Liability

The contents of this manual are subject to changes resulting from the continuous development of products of Helmholz GmbH & Co. KG. In the event that this manual contains technical or clerical errors, we reserve the right to make changes at any time without notice. No claims for modification of delivered products can be asserted based on the information, illustrations, and descriptions in this documentation. Beyond the instructions contained in the operating manual, the applicable national and international standards and regulations must also be observed in any case.

1.6.8 Disclaimer of liability

Helmholz GmbH & Co. KG is not liable for damages if these were caused by use or application of products that was improper or not as intended.

Helmholz GmbH & Co. KG assumes no liability for any printing errors or other inaccuracies that may appear in the operating manual, unless these are serious errors of which Helmholz GmbH & Co. KG was already demonstrably aware.

Beyond the instructions contained in the operating manual, the applicable national and international standards and regulations must also be observed in any case.

Helmholz GmbH & Co. KG is not liable for damage caused by software that is running on the user's equipment which compromises, damages, or infects additional equipment or processes through the remote maintenance connection, and which triggers or permits unwanted data transfer.

1.6.9 Warranty

Report any defects to the manufacturer immediately after discovery of the defect.

The warranty is not valid in case of:

- Failure to observe these operating instructions
- Use of the device that is not as intended
- Improper work on and with the device
- Operating errors
- Unauthorized modifications to the device

The agreements met upon contract conclusion under "General Terms and Conditions of Helmholz GmbH & Co. KG" apply.

2 System overview

2.1 General

The TB20 I/O system is an open, modular, and distributed peripheral system designed to be mounted on a 35mm DIN rail.

It is made up of the following components:

- Bus couplers
- Peripheral modules
- Power and isolation modules
- Power modules

By using these components, you can build a custom automation system that is tailored to your specific needs and that can have up to 64 modules connected in series to a bus coupler. All components have a protection rating of IP20.

2.2 The components of the TB20 I/O system

2.2.1 Bus coupler

The system's bus coupler includes a bus interface and a power module. The bus interface is responsible for establishing a connection to the higher-level bus system and is used to exchange I/O signals with the automation system's CPU.

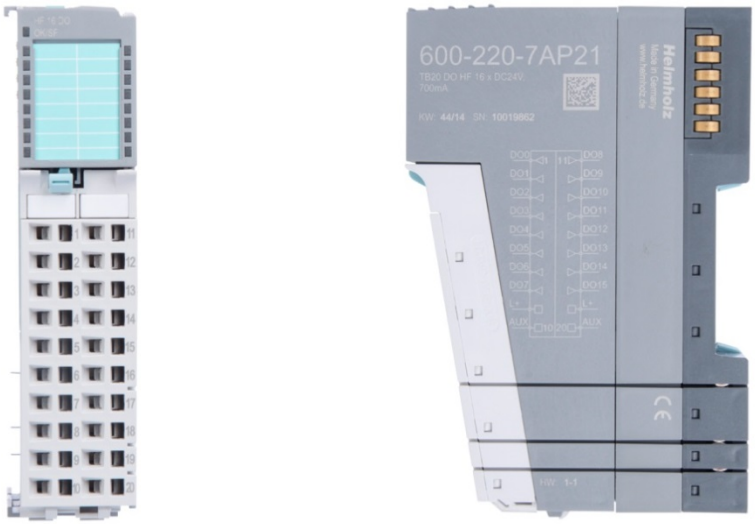
The power module is responsible for powering the coupler's electronics and all connected peripheral modules.

2.2.2 Peripheral modules

The system's peripheral modules are electronic components to which peripheral devices such as sensors and actuators can be connected. A variety of peripheral modules with different tasks and functions are available.

Example: Peripheral module with 10-terminal front connector

Example: Peripheral module with 20-terminal front connector



2.2.3 Power and isolation module

The system's bus coupler provides the supply voltage for the communications bus (5 V, top) and for external signals (24 V, bottom). These voltages are passed from module to module through the base modules.

Power and isolation modules make it possible to segment the power supply for external signals into individual power supply sections that are powered separately. The signals of the communications bus and the supply voltage for the communications bus are not affected by this and continue to be conducted.

NOTE

Power and insulation modules have a lighter housing color to help differentiate them better.

2.2.4 Power module

The system's bus coupler provides the supply voltage for the communications bus (5 V, top) and for external signals (24 V, bottom). These voltages are passed from module to module through the base modules.

Power modules make it possible to segment the power supply for both external signals and the communication bus into individual power supply sections that are powered separately.

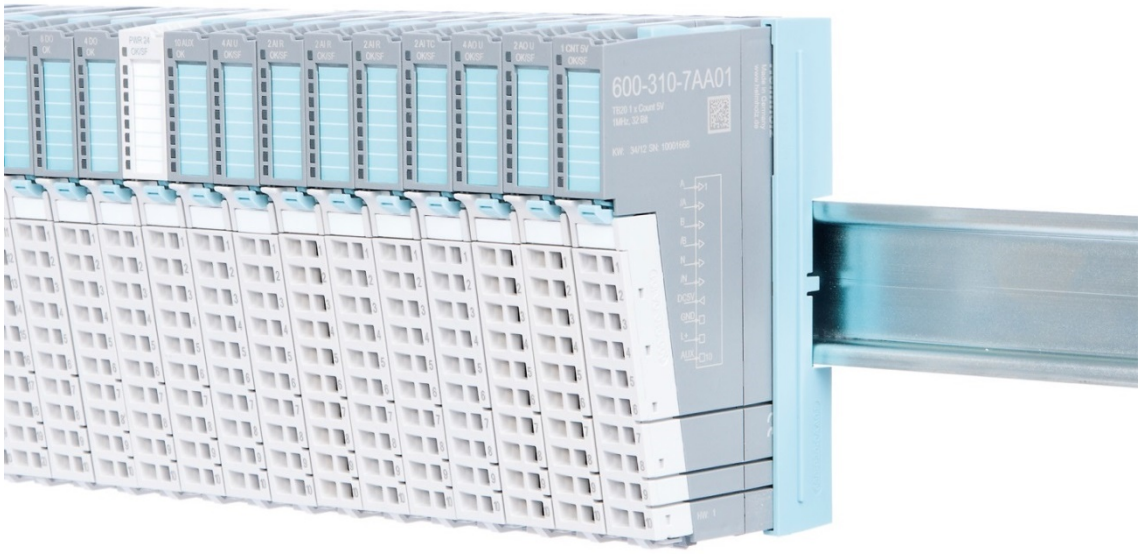
Power modules deliver all necessary power to the peripheral modules connected after them and, if applicable, all the way to the next power module or power and isolation module. A power module is required whenever the power supplied by the coupler alone is not sufficient, e.g., when there are a large number of modules with high power requirements. The "TB20 ToolBox" configuration program can be used to determine whether power modules are needed, as well as how many of them will be needed.

NOTE

Power modules have a lighter housing color to help differentiate them better.

2.2.5 Final cover

The final cover protects the contacts on the last base module from accidental contact by covering its outer right-hand side.



2.2.6 Components in a module

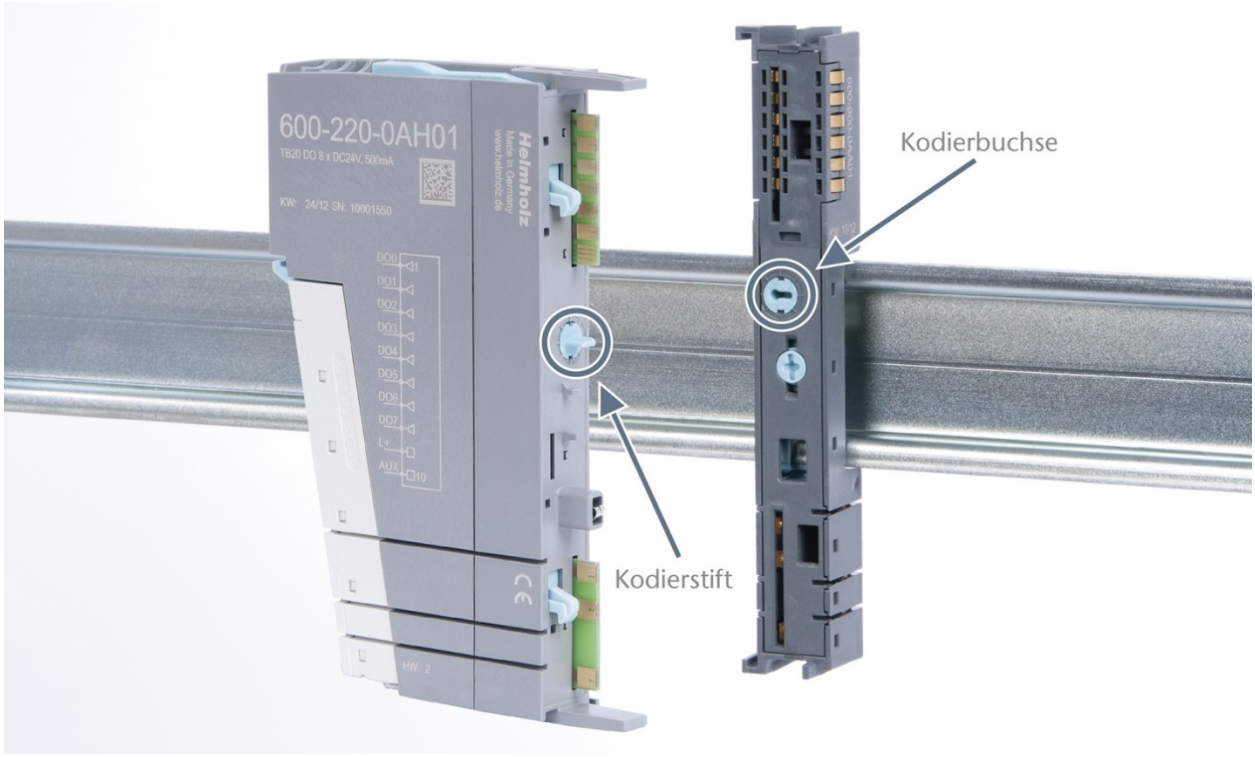
Each module consists of three parts:

- A base module
- An electronic module
- A front connector

2.2.7 Module Coding

Electronic modules and base modules feature coding elements meant to prevent the wrong spare electronic modules from being plugged in during maintenance and repairs.

These coding elements consist of a coding plug on the electronic module and a coding socket on the base module (see following figure).



The coding plug and coding socket can each be in one of eight different positions. Each of the eight positions can be assigned to a specific type of module (Digital In, Digital Out, Analog In, Analog Out, Power). It will only be possible to plug an electronic module into a base module if the position of the coding plug and the position of the coding socket match. If the positions differ, the electronic module is mechanically blocked.

3 Installation and removal

HAZARD

TB20 modules can carry lethal voltage.

Before starting any work on TB20 system components, make sure to deenergize all components and the cables supplying them with power! During work when the system is live, there is the risk of fatal electrocution!

ATTENTION

Installation must be carried out according to VDE 0100/IEC 364 or in accordance with applicable national standards. The TB20 IO system has protection rating IP20. If a higher protection rating is required, the system must be installed in a housing or control cabinet. In order to ensure safe operation, the ambient temperature must not exceed 60 °C.

3.1 Installation position

The TB20 I/O system can be installed in any position. Optimal ventilation and thus the maximum ambient temperature can only be achieved in the horizontal installation layout.

3.2 Minimum clearance

It is recommended to adhere to the minimum clearances specified when installing the coupler and modules. Adhering to these minimum clearances will ensure that:

- the modules can be installed and removed without having to remove any other system components
- there will be enough space to make connections to all existing terminals and contacts using standard accessories
- there will be enough space for cable management systems (if needed)

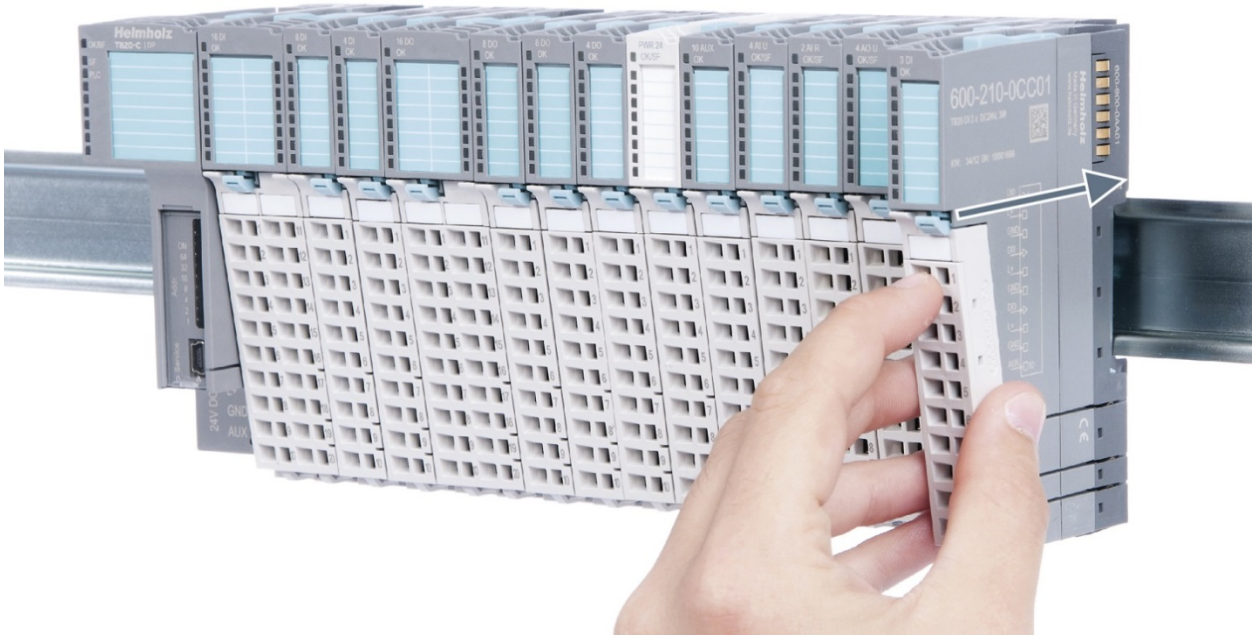
The minimum clearances for mounting TB20 components are: 30 mm on the top and on bottom and 10 mm on each side.

3.3 Installing and removing peripheral modules

3.3.1 Installation

Installing an assembled peripheral module:

1. Place the assembled module on the DIN rail by moving it straight towards the rail. Make sure that the module engages the upper and lower guide elements of the previous module.
2. Then push the upper part of the module towards the DIN rail until the rail fastener inside fastens into place with a soft click.



Installing the individual parts of a peripheral module one after the other:

1. Place the base module on the DIN rail from below in an inclined position.
2. Then push the upper part of the base module towards the rail until the module is parallel with the rail and the rail fastener on the inside snaps into place with a soft click.
3. Place an electronic module with matching coding (see section *2.2.7*) on the base module in a straight line from above and then gently push it onto the base module until both modules are fully resting on top of one another and the module fastener snaps into place with a soft click.
4. Finally, place the front connector on the electronic module from below in an inclined position and then gently push it onto the electronic module until the front connector fastener snaps into place with a soft click.

3.3.2 Removal

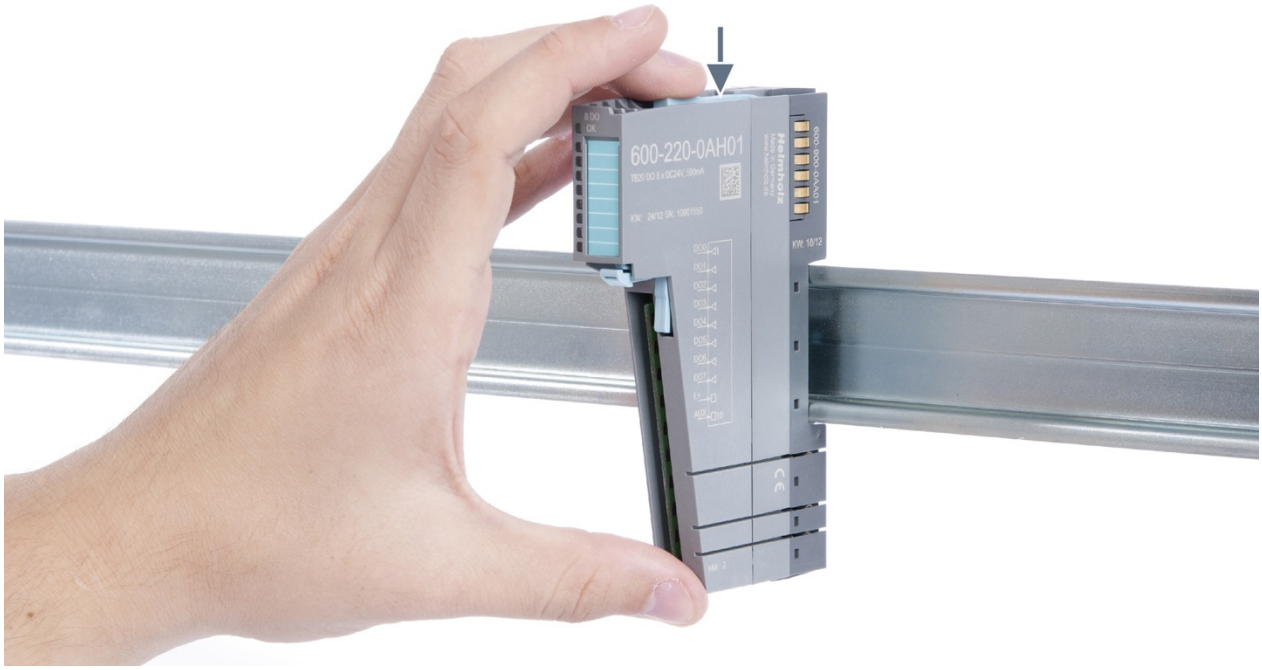
To remove a peripheral module, follow the four steps below:

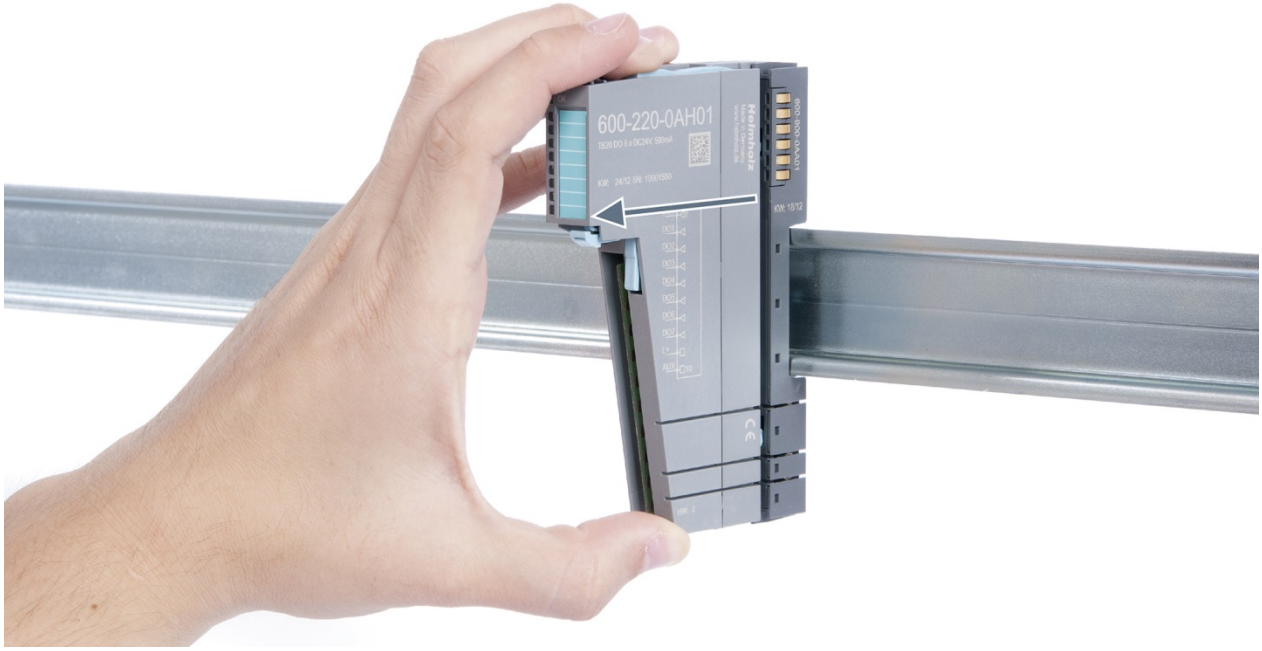
Step 1: Remove the front connector

To remove the front connector, push the tab above the front connector upwards from below. This will push out the front connector, after which you can pull it out.

Step 2: Remove the electronic module

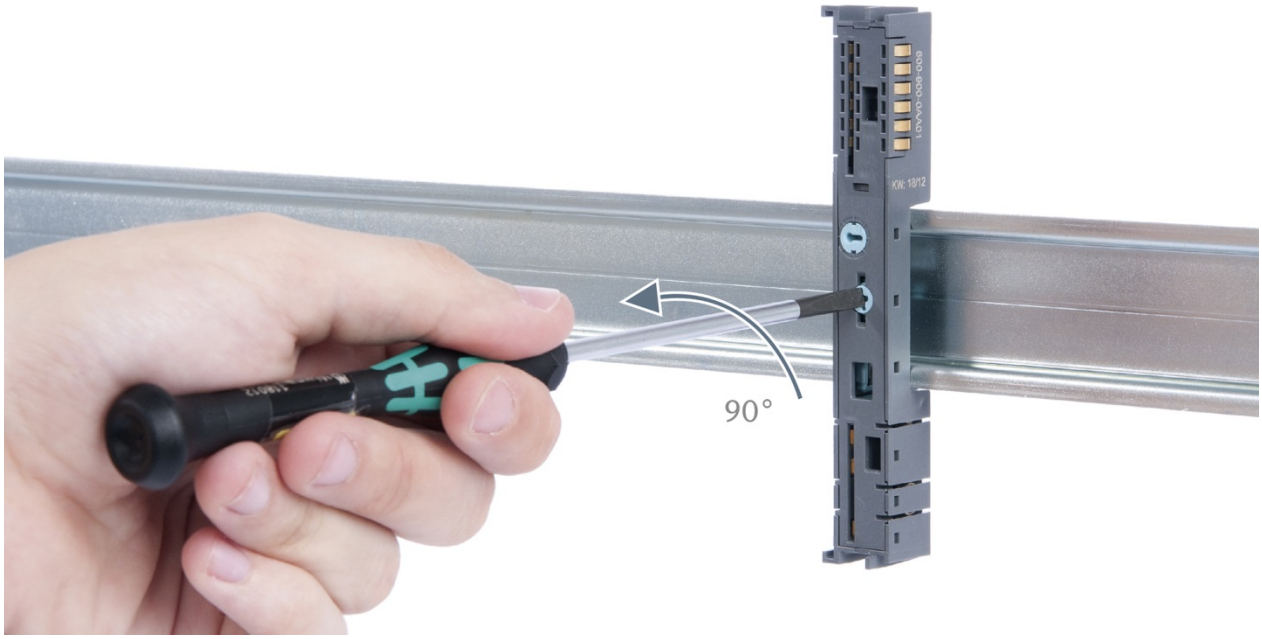
To do so, use your middle finger to push on the lever from above and then use your thumb and index finger to pull out the electronic module while holding the lever down.





Step 3: Release the base module

Use a screwdriver to release the base module. Turn the screwdriver 90° counterclockwise to release.



Step 4: Remove the base module

Remove the base module by pulling it towards you.

3.4 Replacing an electronic module

The procedure for replacing the electronic module on a peripheral module consists of four steps.

If you need to replace the electronic module while the system is running, make sure to take into account the general technical specifications for the bus coupler being used.

HAZARD

TB20 modules can carry lethal voltage.

Before starting any work on TB20 system components, make sure to deenergize all components and the cables supplying them with power! During work when the system is live, there is the risk of fatal electrocution!

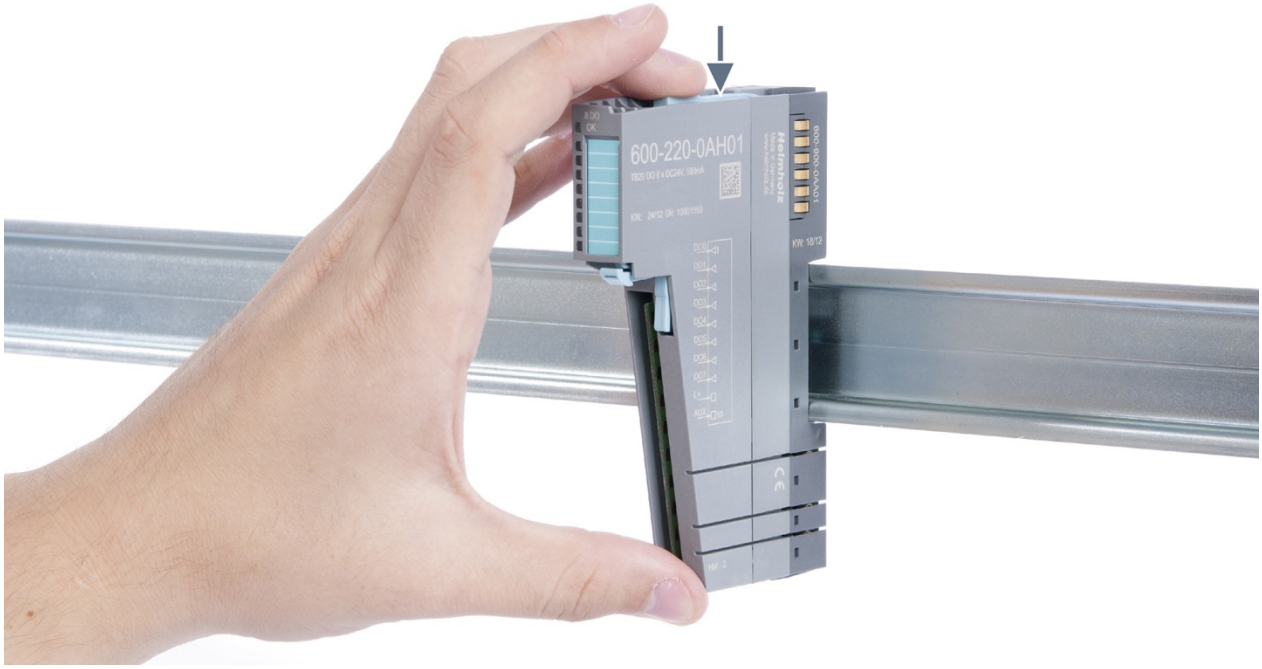
Note the wiring diagram of the system and switch off dangerous voltages before starting work!

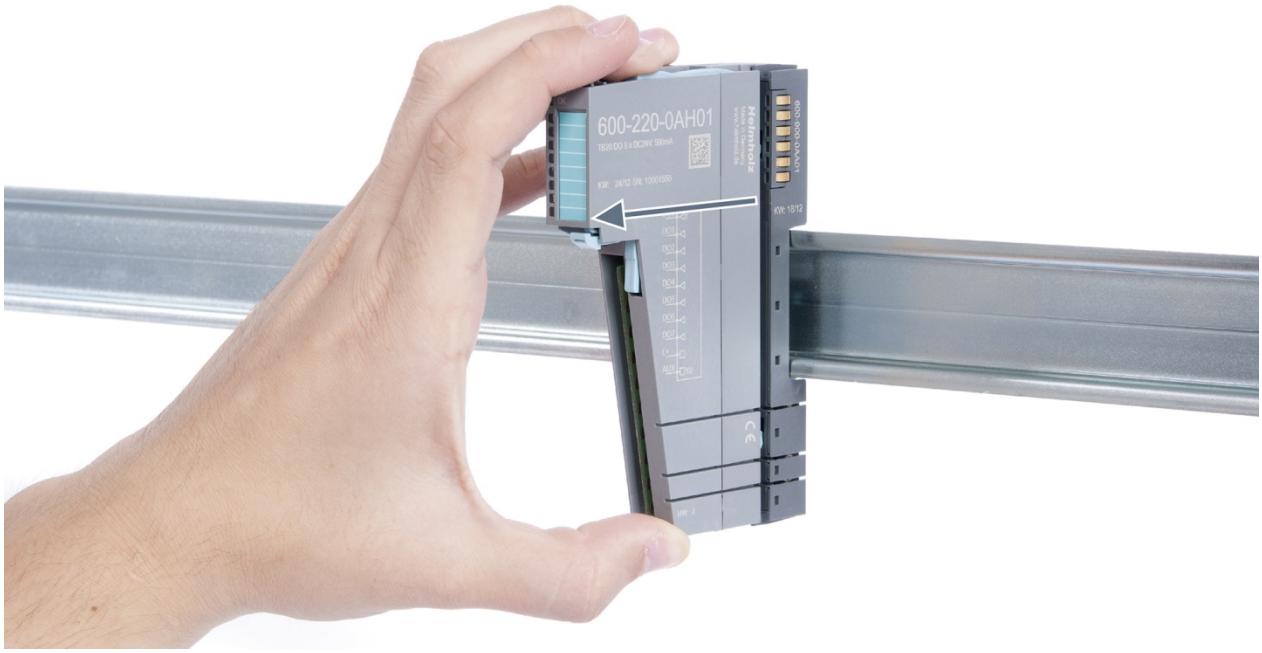
Step 1: Remove the front connector

To remove the front connector, push the tab above the front connector upwards (see the picture below). This will push out the front connector, after which you can pull it out.

Step 2: Remove the electronic module

To remove the electronic module, use your middle finger to push on the lever from above and then use your thumb and index finger to pull out the electronic module while holding the lever down (see the picture below).





Step 3: Plug in a new electronic module

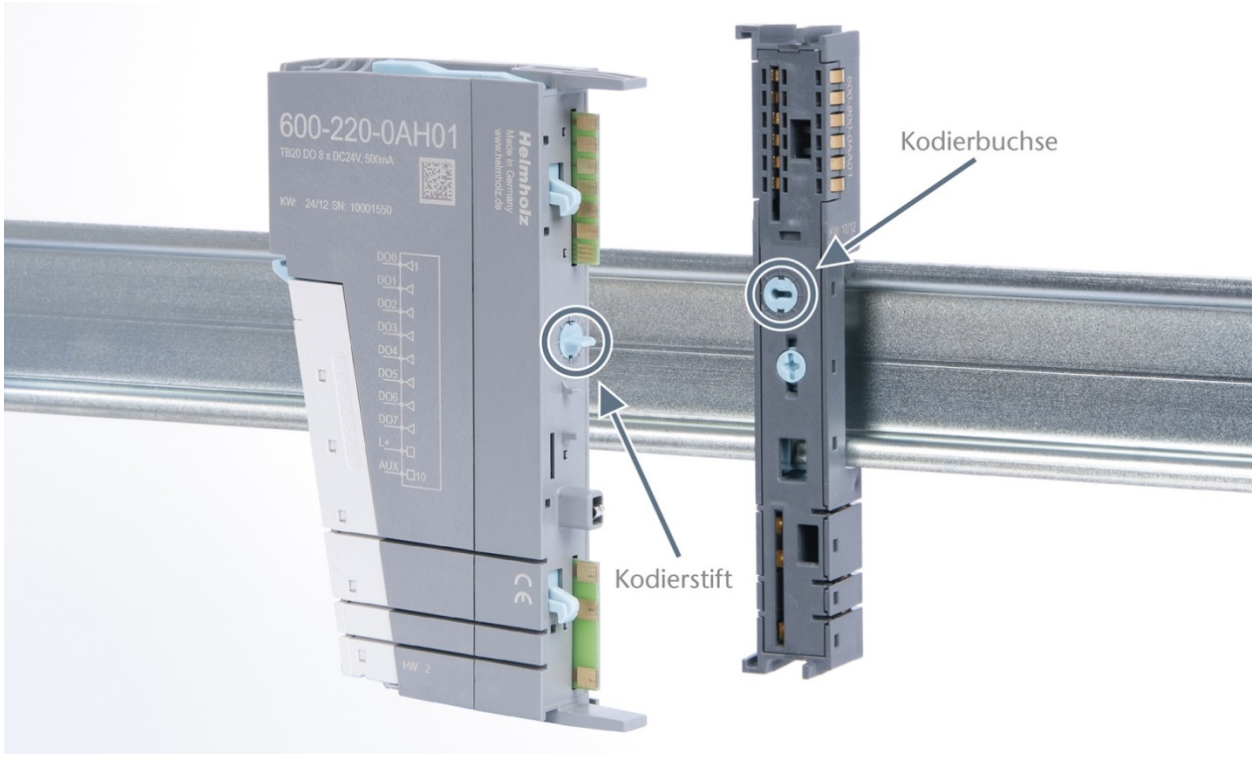
ATTENTION

The electronic module must be snapped into place on the base module with a single continuous movement. If the electronic module is not snapped into place firmly and straight on the base module, bus malfunctions may occur.

ATTENTION

If the electronic module cannot be plugged into the base module, check whether the coding elements on the electronic module and base module (see figure below) match. If the coding elements on the electronic module do not match those on the base module, you may be attempting to plug in the wrong electronic module.

For more information on coding elements, please consult section 2.2.7.

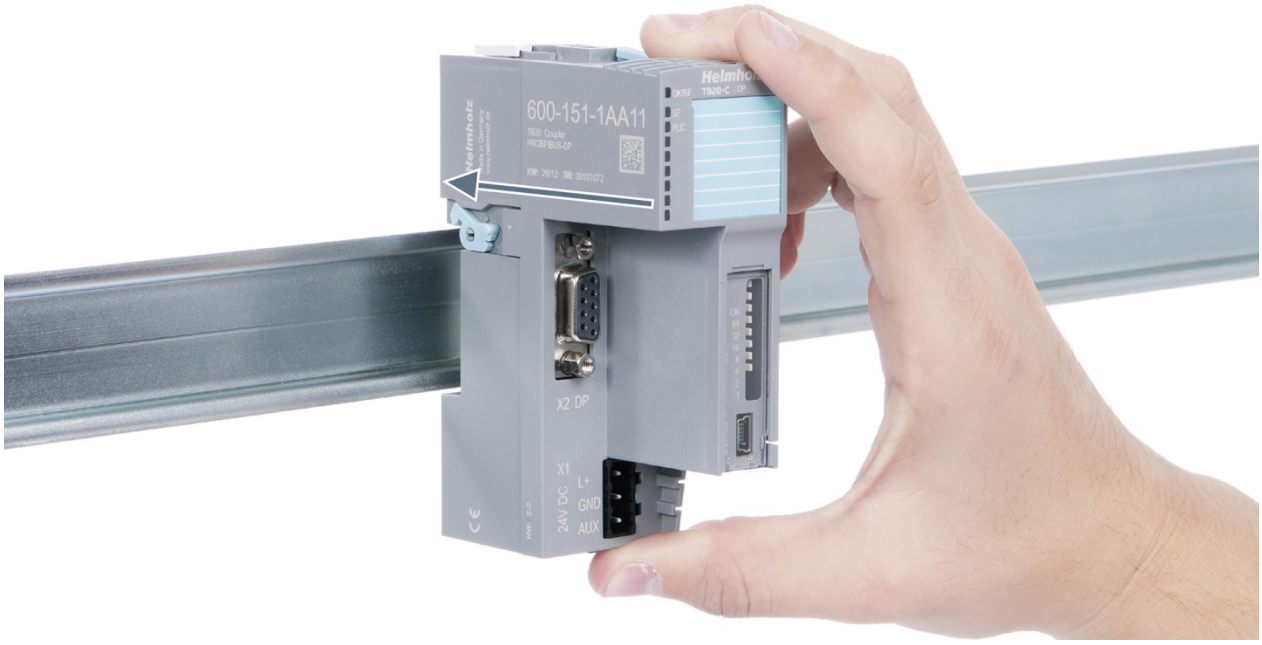


Step 4: Plug in the front connector

3.5 Installing and removing the coupler

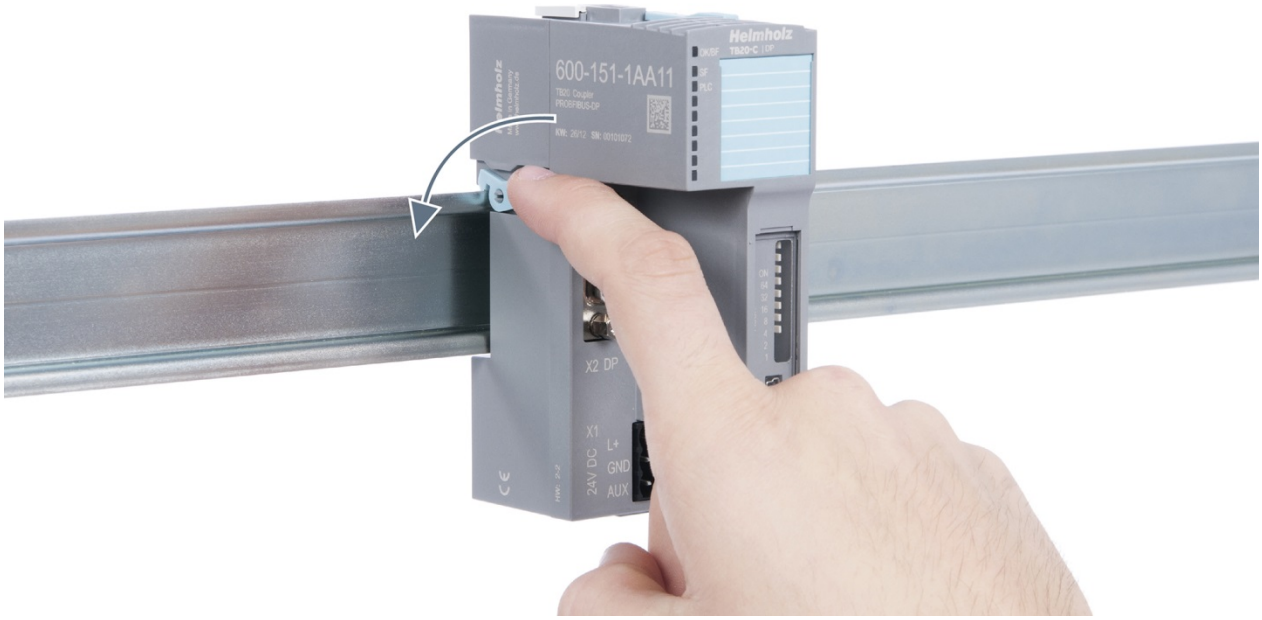
3.5.1 Installation

Place the coupler, together with the attached base module, on the DIN rail by moving it straight towards the rail. Then push the coupler towards the rail until the base module's rail fastener snaps into place with a soft click.



Step 2: Secure the coupler on the DIN rail

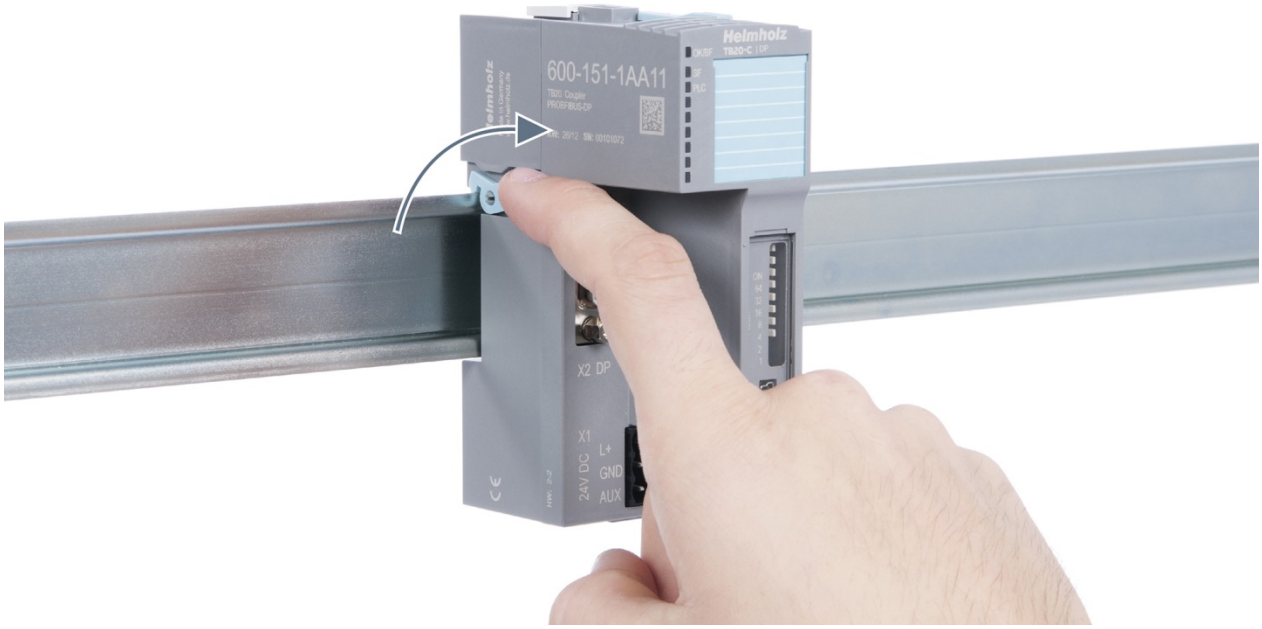
Use the locking lever on the left side of the coupler to lock the coupler into position on the DIN rail.



3.5.2 Removal

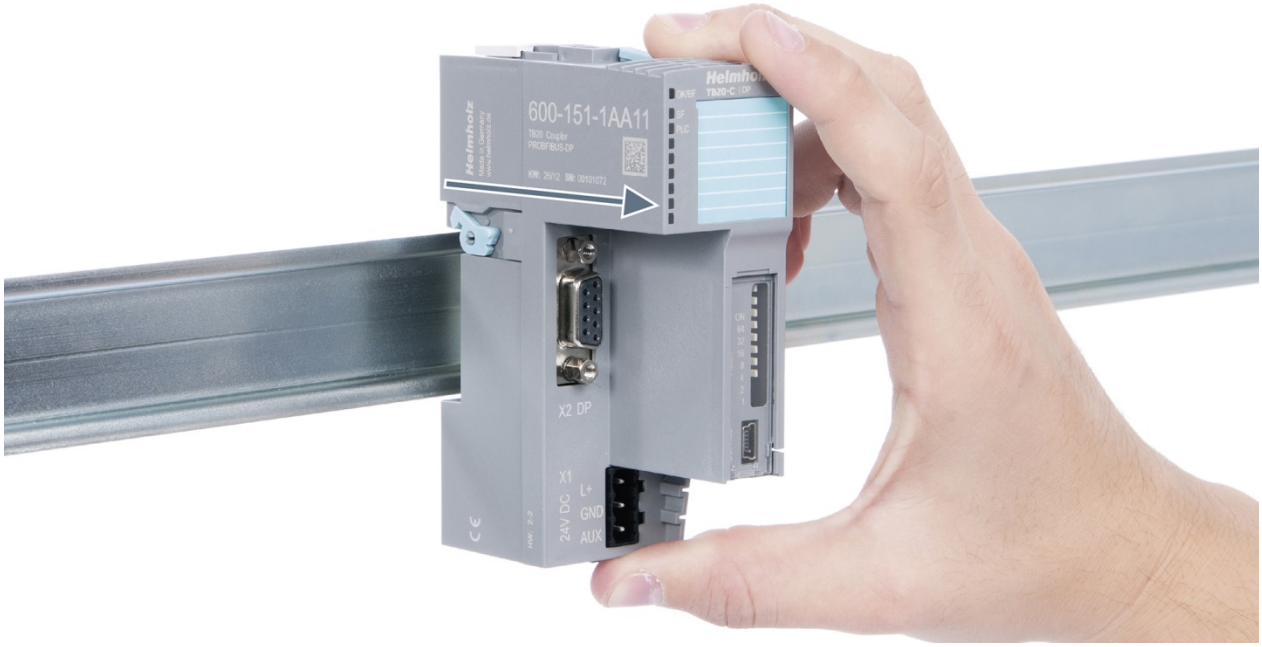
Step 1: Release the locking mechanism

Release the locking lever on the left side of the coupler in order to disengage it from the DIN rail.



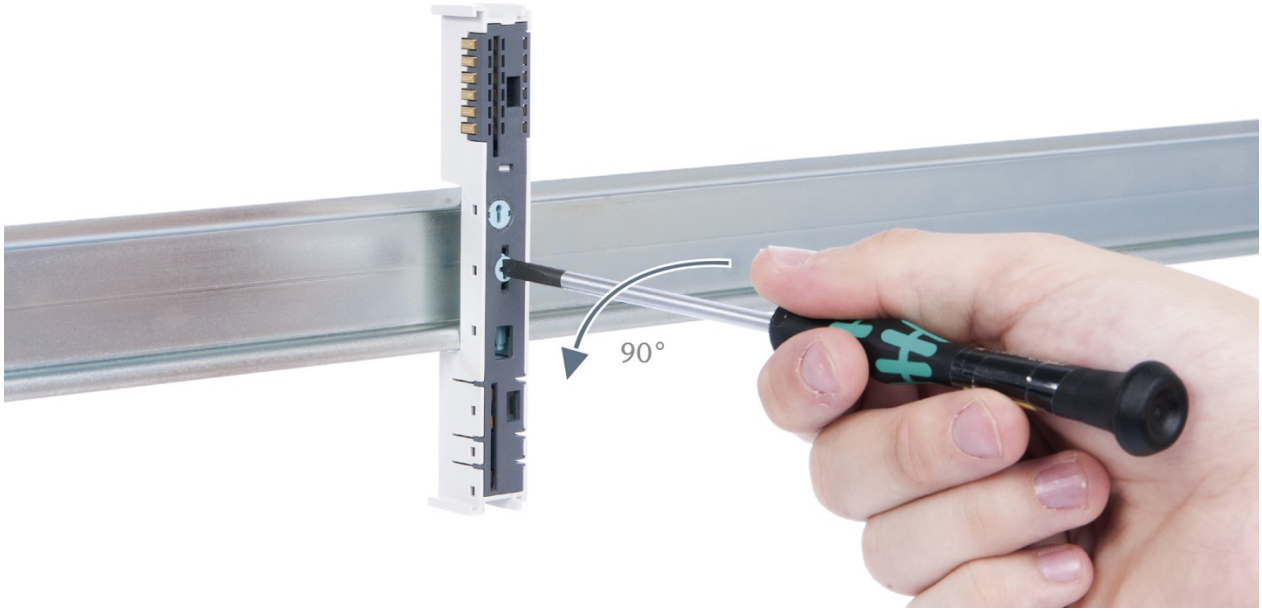
Step 2: Remove the coupler

Use your middle finger to push on the lever from above and use your thumb and index finger to pull out the coupler while holding the lever down.



Step 3: Release the base module

Use a screwdriver to release the base module.



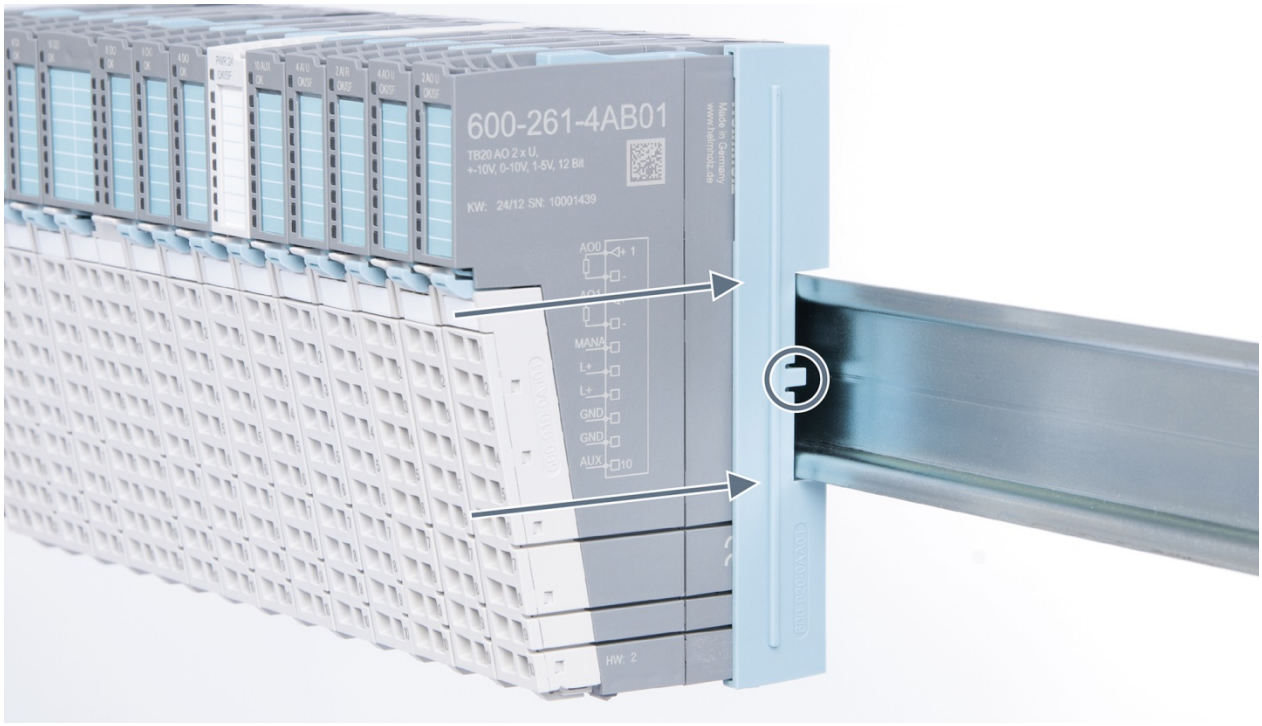
Step 4: Remove the base module

Remove the base module by pulling it towards you.

3.6 Installing and removing the final cover

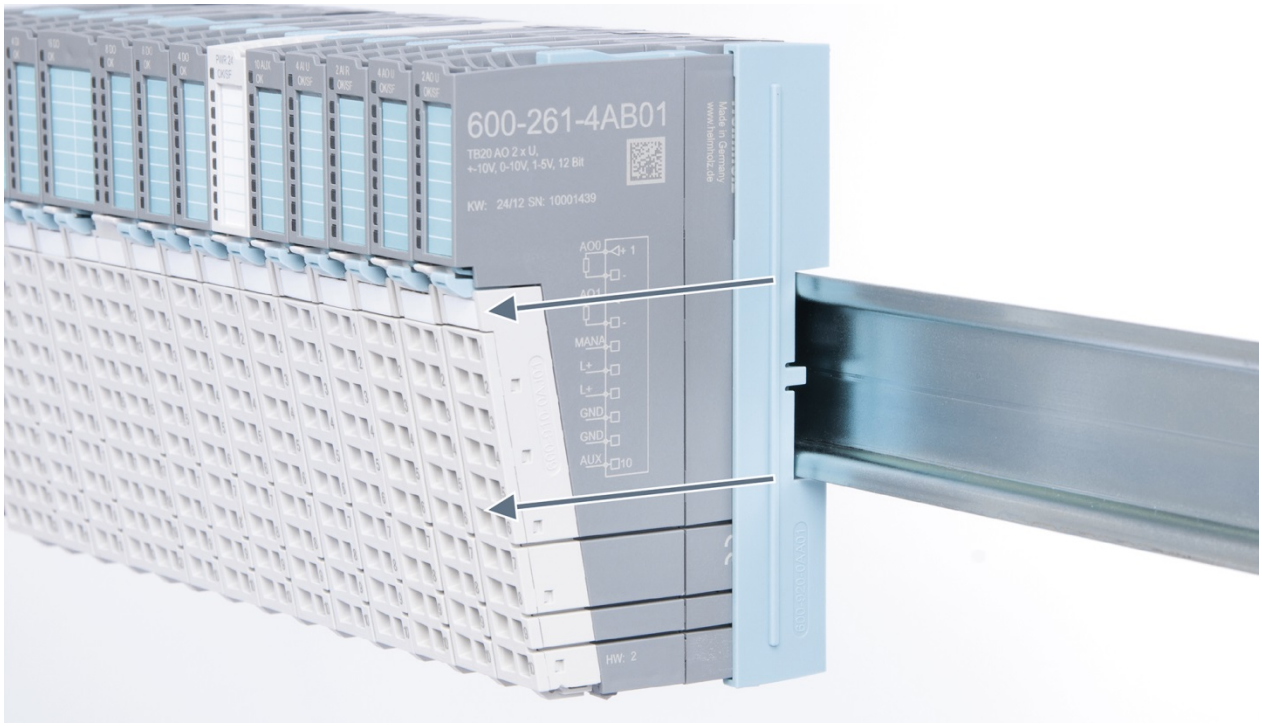
3.6.1 Installation

Slide the final cover onto the last module along the case, starting from the end with the front connector and moving towards the DIN rail, until the cover covers the base module's contacts and the tab snaps into place.



3.6.2 Removal

Pull the final bus cover upward along and off of the module.



4 Setup and wiring

4.1 EMC/safety/shielding

The TB20 IO system complies with EU Directive 2004/108/EC ("Electromagnetic Compatibility").

One effective way to protect against disturbances caused by electromagnetic interference is to shield electric cables, wires, and components.

ATTENTION

When setting up the system and routing the required cables, make sure to fully comply with all standards, regulations, and rules regarding shielding.

All work must be done professionally! Shielding faults can result in serious malfunctions, including the system's failure.

To ensure electromagnetic compatibility (EMC) in your control cabinets in electrically harsh environments, the following EMC rules are to be observed in the design:

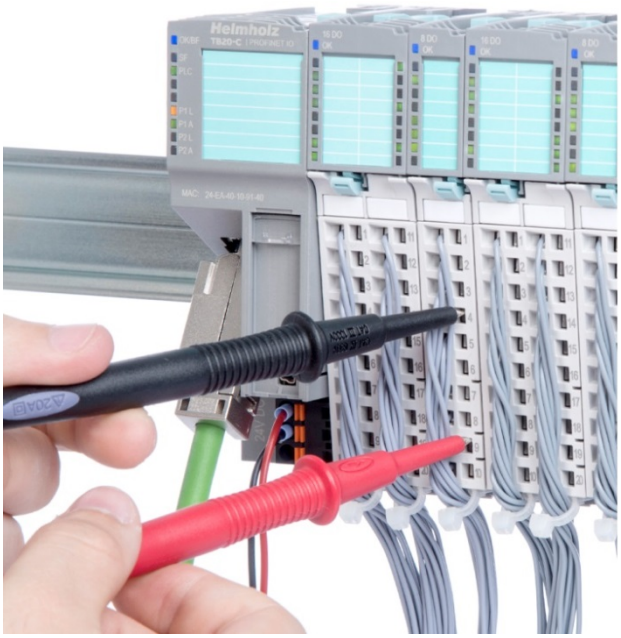
- All metal parts of the cabinet are to be connected with each other over a large area with good conductivity (no paint on paint). Where necessary, use contact washers or serrated washers.
- The cabinet door must be connected to the ground straps (top, middle, bottom) over as short a distance as possible.
- Signal cables and power cables are to be laid separated spatially by a minimum distance of 20 cm from each other in order to avoid coupling paths.
- Run signal lines only from one level into the cabinet if possible.
- Unshielded cables in the same circuit (outgoing and incoming conductors) must be twisted if possible.
- Contactors, relays, and solenoid valves in the cabinet, or in adjacent cabinets if applicable, must be provided with quenching combinations; e.g., with RC elements, varistors, and diodes.
- Do not lay wires freely in the closet; instead, run them as closely as possible to the cabinet housing or mounting panels. This also applies to reserve cables. These must be grounded on at least one end, and it is better if they are grounded at both ends (additional shielding effect).
- Unnecessary line lengths should be avoided. Coupling capacitances and inductances are kept low in this way.
- Analog signal lines and data lines must be shielded.

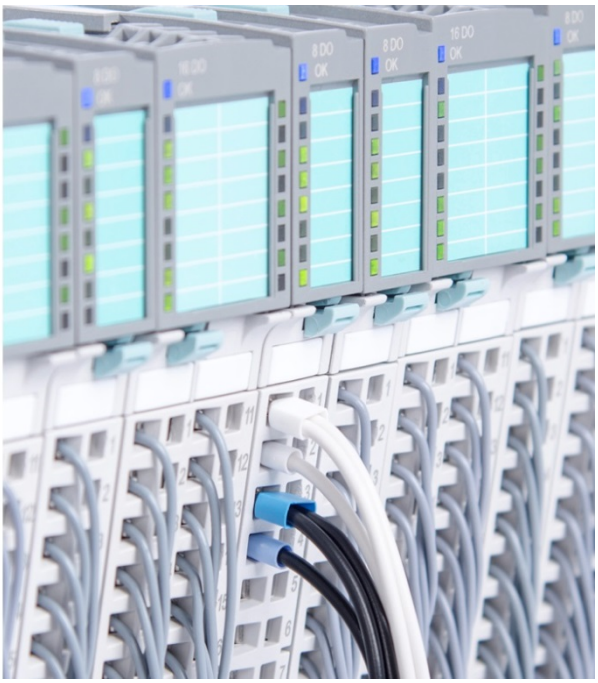
4.2 Front connectors

The front connector's spring-clamp terminals are designed for a cross-sectional cable area of up to 1.5 mm² (16–22 AWG) with or without ferrules.

It is also possible, for example, to connect two 0.75 mm² wires to a single spring-type terminal, provided the maximum cross-sectional cable area of 1.5 mm² per terminal is not exceeded.

The cables can be attached to the underside of the front connector with a cable tie.





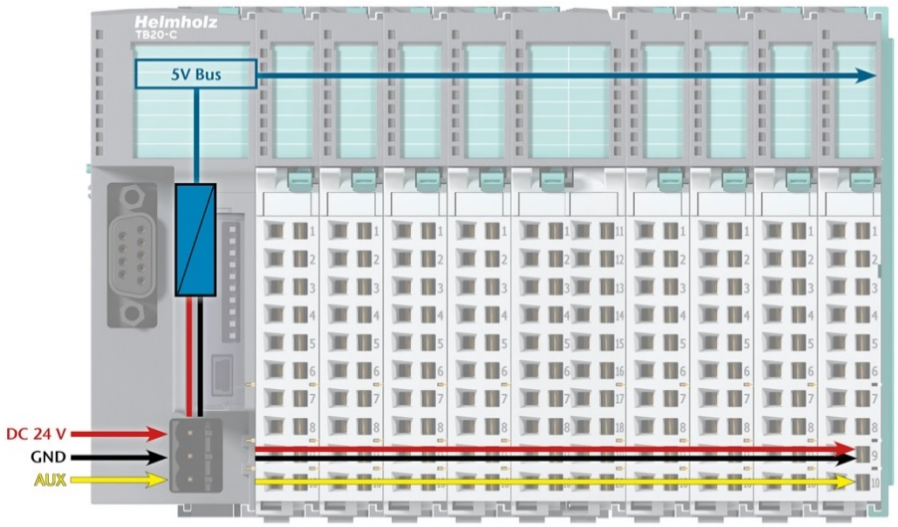
4.3 Wiring the coupler

A power supply unit is integrated into the bus coupler. The power supply unit is responsible for powering the peripheral modules connected to the coupler. In turn, it draws its own power from the three-pin connector on the front (24 VDC, GND, AUX).

The 24 V connector is used to power two buses:

- The power bus used to power the I/O components (24 VDC, GND, AUX)
- The communications bus used to power the electronics in the peripheral modules

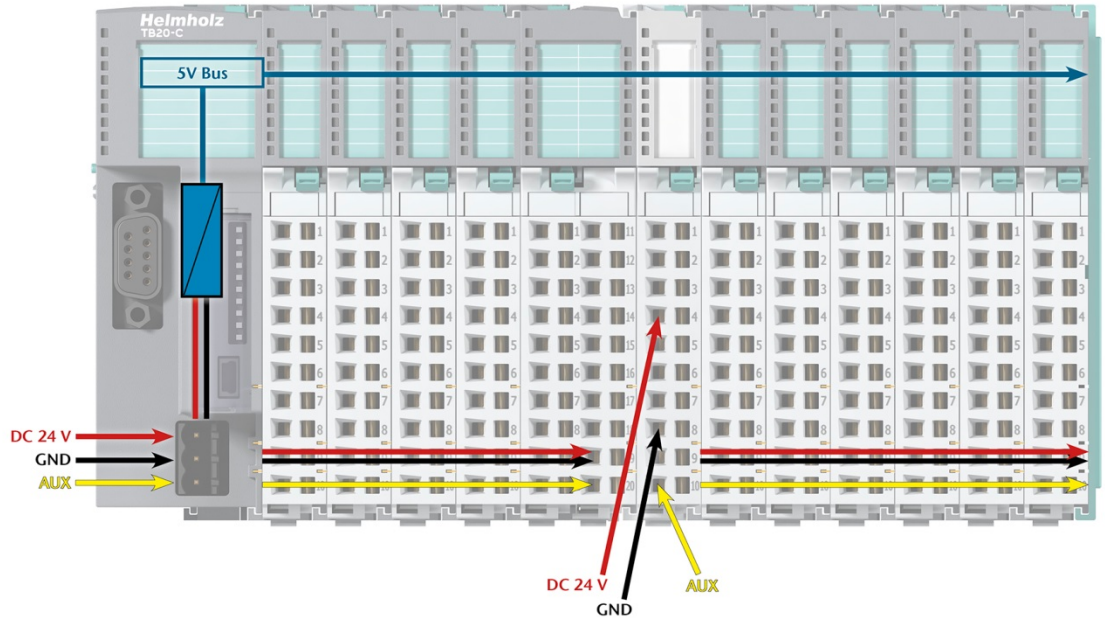
The AUX pin can be used to connect and use an additional voltage potential. Every peripheral module has an AUX terminal on its front connector (the bottommost terminal, i.e., terminals 10 and 20).



The coupler and the modules are grounded via the shield contact to the DIN rail. The DIN rail must be grounded. The surface of the DIN rail must be clean and conduct electricity well.

4.4 Using power and isolation modules

Power and isolation modules make it possible to segment the power supply for external signals (24 V, GND, AUX) into individual power supply sections that are powered separately.

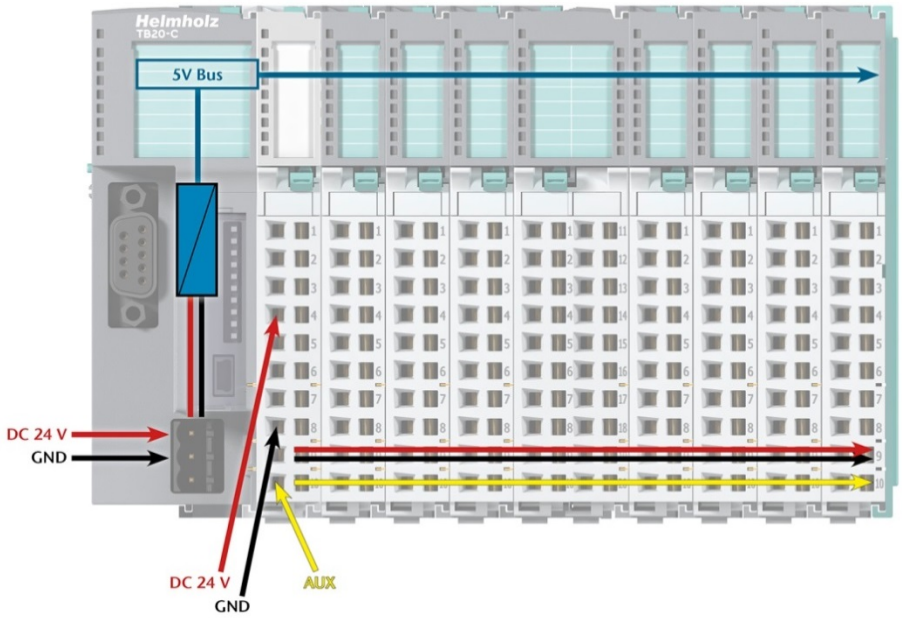


The order no. for the power and isolation module for 24 V signals is 600-710-0AA01.

Its electronic module and base module have the same light gray color as the front connector, ensuring that all power and isolation modules will stand out visually in the system and make it easy to clearly distinguish each individual power supply segment.

4.5 Separate power supply segments for the coupler and the I/O components

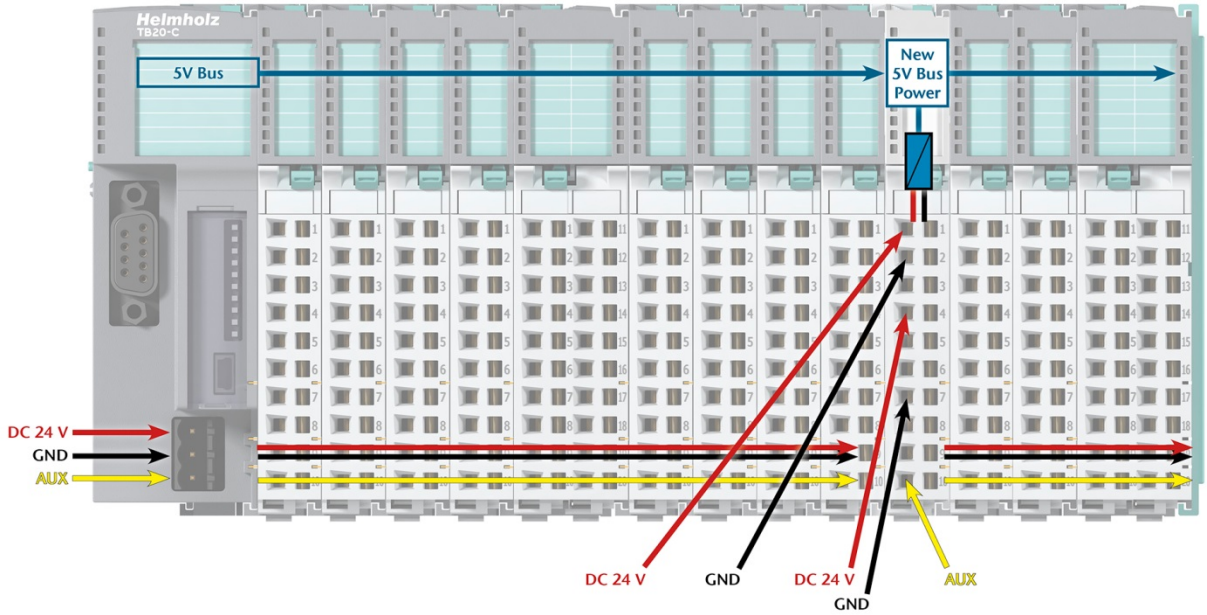
If the power supply for the coupler needs to be separate from the power supply for the I/O modules, a power and isolation module can be used right after the coupler.



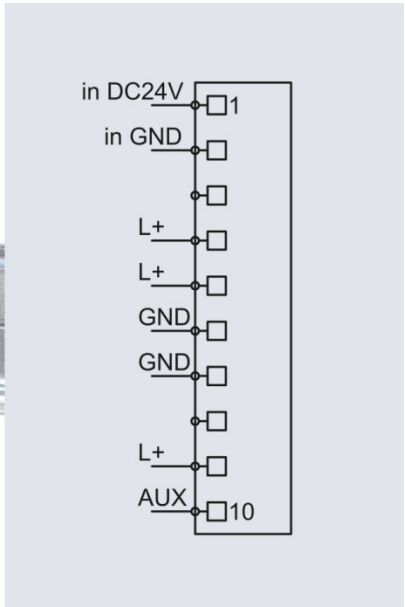
4.6 Using power modules

Power modules deliver all necessary power to the connected peripheral modules and, if applicable, all the way to the next power module or power and isolation module. Power modules must be used whenever the power supplied by the coupler alone is not sufficient, that is, when there are a large number of modules on the bus. The *TB20 ToolBox* configuration and diagnostics program can be used to calculate the current draw.

24 V DC, GND, and AUX are fed into the terminals, whereas the supply of the sequenced modules runs via the bus system of the base modules.



The order no. for the power module is 600-700-0AA01. The electronic module of the power module is light gray like the front connector. The base module of the power module is light gray with a dark top part.



4.7 Electronic nameplate

All of a TB20 module's important information can be found on its electronic nameplate. This information includes, for example, the corresponding module ID, module type, order number, unique serial number, hardware version, firmware version, and internal range of functionalities.

This information can be read out in a number of ways, one of which is using the *TB20 ToolBox* configuration and diagnostics software. The electronic nameplate make it possible to prevent configuration errors during commissioning and make maintenance easier.

4.8 Fusing

The TB20 coupler's and power modules' power supply must be externally fused with a slow-blowing fuse, maximum 8 A, appropriate for the required maximum current.

5 Properties of the TB20 EtherCAT® bus coupler

The TB20 EtherCAT® bus coupler has the following properties:

- EtherCAT slave with Modular Device Profile (MDP)
- 2 EtherCAT ports for connection to the EtherCAT network
- CoE object directory with support of SDO info and complete access
- Up to 64 TB20 modules
- Maximum 1024 byte process input and 1024 byte process output data
- Automatic process data configuration
- Module exchange possible while in operation (hot swap)
- Cycle-current module diagnostics
- Can be configured in any EtherCAT configuration tool with provided ESI file
- Comfortable configuration and diagnostics capabilities via the free PC software TB20 Toolbox (connection via USB)
- Module configuration can be stored
- (Re)configuration of the modules possible via SDO communication
- Status information/diagnostics via 6 LEDs
- 24 V DC power supply
- Integrated power supply unit for powering peripheral modules (2.5 A)
- Supplies the system's I/O voltage (24 VDC)

The Distributed Clocks functionality of EtherCAT® is currently not supported.

6 Introduction to EtherCAT®

EtherCAT® is the abbreviation for *Ethernet for Control Automation Technology* and stands for an Ethernet-based, real time-compatible fieldbus technology. The EtherCAT protocol is disclosed and standardized in IEC 61158.

The important foundations of EtherCAT® will be described in the following. These are helpful for an understanding of the functionality of the TB20 EtherCAT® bus coupler. For detailed information, we refer you to the Internet site www.ethercat.org of the *EtherCAT Technology Group*.

EtherCAT® is a registered trademark and a patented technology, licensed through the Beckhoff Automation GmbH, Germany.

6.1 Network and topology

An EtherCAT network consists of a master and up to 65,535 slaves. The master is generally a component of a programmable logic controller (PLC) and is equipped with one, or in the case of cable redundancy, with two EtherCAT ports for the connection of the slave. The slaves are field devices, meaning sensors and actors. The TB20 EtherCAT® bus coupler is also an EtherCAT slave.

Most slaves are equipped with two EtherCAT ports: One EtherCAT port designated with *IN* for connection from the direction of the master and one EtherCAT port designated with *OUT* for connecting subsequent slaves. This generally results in a line topology of slaves connected in series. Tree or star topologies can also be realized through the use of slaves with more than two EtherCAT ports or corresponding port expansions. Switches or other active infrastructure components are not required for this purpose.

EtherCAT also supports the connection and removal of individual slaves or entire network segments during ongoing operation, which is designated as 'Hot Connect'. To this purpose, the relevant slaves must be allocated to so-called Hot Connect groups with the help of the EtherCAT configuration tool.

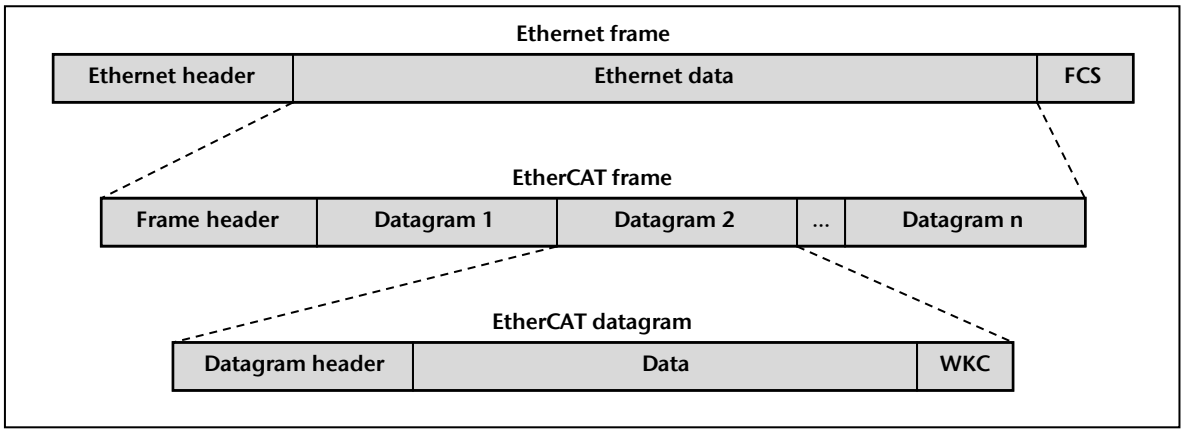
6.2 Communication principle

EtherCAT communication is based on Ethernet. For the purpose of data exchange, standard Ethernet frames are used, which, however, are only transmitted by the master and forwarded and further processed by the slaves in a special fashion:

An Ethernet frame sent by the master is forwarded in such a way that it passes through all slaves connected in a network and subsequently returns to the master. Each slave thereby already takes the data addressed to it during the frame pass and adds its data requested by the master. This makes it possible for the master, for example, to carry out an I/O data exchange with a large number of slaves by sending a single Ethernet frame. This special type of frame processing and forwarding is realized by the EtherCAT slave controller (ESC), which is contained in all slaves.

6.3 Structure of an Ethernet frame

The structure of an Ethernet frame used for EtherCAT communication is represented in the following figure.



The *Ethernet header* as a rule contains the MAC addresses of the sender and recipient, as well as the EtherType of the Ethernet frame. However, these MAC addresses are of no significance for addressing the slave (↗ 6.4). With the EtherType 0x88A4 reserved for EtherCAT, it is shown that the *Ethernet data* contains an EtherCAT frame.

An *EtherCAT frame* consists of a frame header with length and protocol information, as well as one or more *EtherCAT datagrams*. These datagrams are added to the EtherCAT frame by the master for the data exchange with the slaves. The *datagram header* defines which slave(s) are to be addressed by the datagram, and which slave data is to be accessed and how. Access can take place as writing, reading or writing & reading.

Depending upon the type of access, the data share of relevance for an addressed slave is adopted within the datagram and/or overwritten. In the event of the successful carrying out of access, the *Working Counter*(WKC) found at the end of the datagram is increased by the addressed slave. Following the return of the Ethernet frame to the master, this can monitor the processing status of the datagrams on the basis of the WKC.

The *Frame Check Sequence* (FCS) is found at the end of each Ethernet frame. It serves the purpose of detecting transmission errors and is evaluated both by the master and by the slaves. If the slave carries out changes to the EtherCAT datagrams contained in the Ethernet frame, then the FCS not only evaluates, but also makes adjustments accordingly.

6.4 Slave addressing

When addressing the slaves through the EtherCAT datagrams, a differentiation is made between various types of device addressing and logical addressing.

6.4.1 Device addressing

During *device addressing*, a slave can be addressed on the basis of its position in the network, its stations address or through a broadcast.

The *position-dependent addressing of devices* takes place with the help of the so-called auto-increment address. This contains the position of the slave in the form of a negative value and is automatically incremented by each slave during the frame run. The slave, with an address having the value 0, is addressed by the corresponding datagram. This type of addressing is only used when starting up and subsequently only sporadically for the detection of newly connected slaves.

The *addressing of devices via a stations address* takes place using the configured station address or the configured station alias. The *configured station address* is assigned by the master and configured when starting up in the slave. Alternatively, the *Configured Station Alias* can be used for addressing. This involves an address stored in the slave by the user that can also be used to clearly identify this slave. The device addressing via a station address is primarily used when the master wants to individually access the ESC directory of a slave.

Chapter 7.5.1 provides further information on the Configured Station Alias with regard to the TB20 EtherCAT® bus coupler.

In the case of a datagram sent as a *broadcast*, all slaves are addressed. This is used, for example, to carry out an initialization of all slaves or to check whether all slaves are found in the expected EtherCAT state.

6.4.2 Logical addressing

The logical addressing is used by the EtherCAT datagrams for the process data communication. To this purpose, the process data of all slaves is represented in common logical process data memory. The addressing of the process data within this logical memory takes place by way of 32 bit-wide logical addresses.

By using logical addressing, it is possible to address the process data of several slaves with only one datagram, to the extent that they lie in a common area of the logical process data memory. The EtherCAT configuration tool is responsible for the compilation of the slave process data within the logical process data memory. Under normal circumstances it ensures that efficient logical process data addressing is possible.

In the slaves, the EtherCAT slave controller (ESC) makes available a storage area in which the process data of the slave is filed for exchange with the master. Addressing within this memory takes place via local addresses.

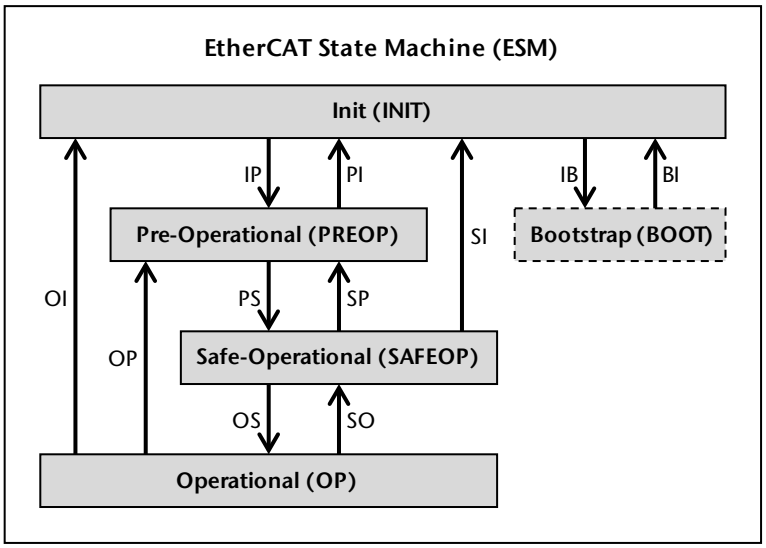
The *Fieldbus Memory Management Units* (FMMUs) are responsible for the assignment of logical to local addresses when accessing the process data . These are a component of the ESCs and are automatically configured by the master when starting up.

6.5 EtherCAT state machine (ESM)

The EtherCAT state machine (ESM) defines the EtherCAT-specific conduct of the slave on the basis of certain states (EtherCAT states) and state transitions. This serves the purpose of coordination of the interplay between master and slave.

6.5.1 The EtherCAT states and their transitions

The various EtherCAT states of the ESM are shown in the following figure. In addition to the complete designation, a short designation is also indicated in brackets, which is primarily used in this manual. The arrows identify the possible transitions between the EtherCAT states. They are provided with abbreviations, which are respectively composed of the first letter of the output and the target state. These abbreviations are used by several EtherCAT configurators for designating the state transitions.



INIT state

The INIT state is the initial state of the ESM. In this EtherCAT state, the master carries out a basic initialization of the slave, which includes, among other things, the configuration of the irregular mailbox communication.

PREOP state

As of the PREOP state, the object directory can be accessed via mailbox communication. Application-specific settings can be made and information read out concerning the objects contained within them. Before a change is made to the SAFEOP state, the slave is configured by the master for process data communication.

SAFEOP state

The cyclical exchange of process input and process output data with the master takes place as of the SAFEOP state. As of now, the slave provides current input data, but maintains its outputs in a secure state. This means that the output data transmitted by the master is not yet adopted.

OP state

With the OP state, the target state for the cyclical I/O data operation of the slave has been achieved. The slave provides current input data, and the output data transmitted by the master is now adopted by the slave and its outputs are activated accordingly.

BOOT state

The BOOT state is an optional EtherCAT state that is planned for the transmission of a new firmware to the slave. The BOOT state is not supported by the TB20 EtherCAT® bus coupler, as an updating of the firmware via USB takes place with the help of the TB20 ToolBox (↗ 7.3.10).

The EtherCAT state machine of a slave is normally controlled by the master, but the slave itself can also trigger a change to a lower EtherCAT state on the basis of a problem. If the latter is the case, or if the slave could not change to the EtherCAT state called for by the master for certain reasons, then the EtherCAT state machine is in a state of error. This is usually shown in that the designation of the current EtherCAT state has the English word 'ERROR' either as a prefix or a suffix (e.g. PREOP ERROR). Following the occurrence of an error, the master can only then once again request the change to a new EtherCAT state when it has confirmed the error. A change to the INIT state is on the other hand always possible.

This general introduction to the EtherCAT states and the state transitions of the EtherCAT state machine is supplemented by chapter 8, in which the conduct of the TB20 EtherCAT® bus coupler is described in detail.

6.5.2 The ESM directories in the EtherCAT slave controller

The following directories are used in the EtherCAT slave controller for controlling the EtherCAT state machine by the master, and for responses concerning its status.

ESM directory	ESC address	Short description
AL control directory (↗ 6.5.2.1)	0x0120/0x0121	Contains the EtherCAT state requested by the master
AL status directory (↗ 6.5.2.2)	0x0130/0x0131	Contains the current EtherCAT state of the ESM of the slave
AL status code directory (↗ 6.5.2.3)	0x0134/0x0135	Contains the current error code of the ESM of the slave

As an exception, these directories are described more precisely here, as they reflect central aspects of the functionality of the EtherCAT state machine.

6.5.2.1 The AL control directory

The AL control directory is described by the master in order to control the EtherCAT state machine of the slave. It also contains the requested EtherCAT state and offers the possibility to confirm an error state displayed in the AL status directory, and thus to switch it off. The requesting of a certain EtherCAT state via the AL control directory is designated as an *AL Control Request*.

The AL control directory can only be read by the slave. A transition to the requested EtherCAT state from the current EtherCAT state is triggered by an AL Control Request in the slave. Depending upon the marginal conditions, this transition is concluded either successfully or unsuccessfully, which is reported back via the AL status directory and the AL status code directory.

The 16-bit AL control directory is structured as follows:

Bit	Description	Meaning of the values
0 - 3	EtherCAT state requested by the master	1 = INIT 2 = PREOP 3 = BOOT 4 = SAFEOP 8 = OP All other values are invalid and result in an error flag being placed in the AL status directory (↗ 8.2.22).
4	Flag for the confirmation/switching off of an error displayed in the AL status directory	0 = Error is not confirmed 1 = Error is confirmed
5	Flag for requesting device identification via the AL status code directory (This device identification designated as 'Explicit Device Identification' is not supported by the TB20 EtherCAT® bus coupler)	0 = no request for device identification 1 = request for device identification
6 - 15	Reserved	Should always be described by the master with the value 0

The EtherCAT state machine of a slave can usually also be controlled from the EtherCAT configurator used for configuration/commissioning. This control possibility can usually be found under the designation 'Online' or 'State Machine' following selection of the slave. In the process, the AL control directory is (indirectly via the master) is described in the EtherCAT slave controller.

In the case of the TB20 EtherCAT® bus coupler, the EtherCAT state currently being requested via the AL control directory can also be displayed in the TB20 ToolBox, in that the bus coupler and then the 'Info' tab are selected during the real time diagnosis (↗ 7.3.7).

6.5.2.2 The AL status directory

The AL status directory provides information about the current state of the EtherCAT state machine of the slave. In addition to the current EtherCAT state, the directory also contains an error flag. This always indicates an error when a change to the EtherCAT state requested by the master is not possible or when the slave has itself initiated the change to a lower EtherCAT state due to a problem. The AL status code directory described in the following then contains an error code which provides an indication of the cause of the error.

The error flag is set upon the occurrence of an error in connection with the EtherCAT state machine, whereby the AL status code directory contains an error code as an indication of the cause of the error. The error flag can only be extinguished by an AL Control Request with a set error confirmation flag or by a request for INIT from the master.

The 16-bit AL status directory is structured as follows:

Bit	Description	Meaning of the values
0 - 3	Current EtherCAT state of the slave	1 = INIT 2 = PREOP 4 = SAFEOP 8 = OP
4	Error flag	0 = no error occurred (AL status code = 0x0000) 1 = error occurred (AL status code > 0x0000)
5	Device identification via the AL status code directory (This device identification designated as 'Explicit Device Identification' is not supported by the TB20 EtherCAT® bus coupler)	0 = no device identification 1 = AL status code directory contains information on device identification
6 - 15	Reserved	Always contains the value 0

The current EtherCAT state can usually be displayed in the EtherCAT configuration tool following selection of the slave under the designation 'Online' or 'State Machine'. The AL status directory is read out (indirectly via the master) to this purpose.

In the case of the TB20 EtherCAT® bus coupler, the current EtherCAT state can also be displayed in the TB20 ToolBox, in that the bus coupler and then the 'Info' tab are selected during the real time diagnosis (↗ 7.3.7). 'ERROR also shown when the error flag is set in the AL status directory.

6.5.2.3 AL status code directory

The AL status code directory contains the 16 bit *AL status code*, which should provide an indication of the cause of an error following its occurrence. The master therefore always reads this directory when the error flag is set in the previously described AL status directory. If there is currently no error, the directory will then contain the value 0x0000.

The current AL status code can usually be displayed in the EtherCAT configuration tool following selection of the slave under the designation 'Online' or 'State Machine'. The AL status directory is read out (indirectly via the master) to this purpose.

In the case of the TB20 EtherCAT® bus coupler, the current AL status code can also be displayed in the TB20 ToolBox, in that the bus coupler and then the 'Info' tab are selected during the real time diagnosis (↗ 7.3.7). In addition to the AL status code, a brief description of the error is also shown here. A list of all AL status codes used by the bus coupler in the event of an error can be found in chapter 8.3.

6.6 Configuration

The configuration of the master and the connected slaves takes place with the help of a software or software components, which are designated here as the EtherCAT configuration tool. A complete series of different EtherCAT configuration tools are available, which could come under consideration, depending upon the master or the programmable logic controller (PLC). For this reason, this manual attempts to provide generally valid information on the configuration of the TB20 EtherCAT® coupler in such a tool.

In the case of EtherCAT, slaves from different manufacturers can be connected to form a network and used on a master. The information necessary for configuration independent of the individual slaves is made available via device description files (↗ 6.6.1), which are designated EtherCAT-specific as ESI files. Depending upon the EtherCAT configuration tool used, configuration without a device description file may also be possible (↗ 6.6.2).

The configuration of the master *and* of the slaves carried out via the EtherCAT configuration tool is subsequently filed in the master. This takes place either implicitly through the configuration tool, or explicitly in the form of an XML-based ENI file (ENI = EtherCAT Network Information). When starting up, the master makes all of the settings required for the individual slaves on the basis of the filed configuration.

6.6.1 ESI files for the description of the device

In the case of the ESI files (ESI = EtherCAT Slave Information), these are XML-based device description files made available by the respective manufacturers for the offered slaves. They contain all relevant information necessary for a comfortable configuration of the slaves within an EtherCAT configuration tool. The connection between a slave and its device description takes place by way of the clear combination of vendor ID (manufacturer code) and product code, as well as product revision.

The TB20 EtherCAT® bus coupler is a modular EtherCAT slave with a CoE object directory (↗ Chapter 9). Its ESI file therefore also contains descriptions of all available modules, as well as an offline variant of the object directory.

In order to provide the EtherCAT configuration tool with the required ESI files, these must be installed or imported. Please find the procedure required for this in the documentation of the tool you are using.

NOTE

The current version of the ESI file for the TB20 EtherCAT® bus coupler can be downloaded via the download area of the Helmholz website www.helmholz.de.

6.6.2 Configuration without ESI file

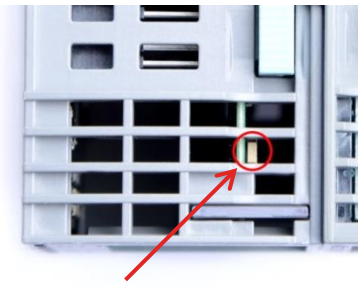
In the event that no ESI file is available, an online device description can be called up from the slave itself. However, this is reduced to the central information required for a foundational configuration. For example, no information on modules or on objects of the CoE object directory are provided by the online device description. While this doesn't make configuration with modules absolutely impossible, the process does become considerably more difficult and less comfortable.

The calling up and the alternative use of the online device description of a slave are not supported by all EtherCAT configuration tools. In this case, a slave configuration without an ESI file is not possible.

In the case of the TB20 EtherCAT® bus coupler, part of the limitations are compensated for again, in that the selection and configuration of the modules is carried out with the help of the TB20 ToolBox (↗ 7.5.6). Access to the object directory of the bus coupler is possible when the EtherCAT configuration tool used supports the calling up of the objects present in the bus coupler via SDO info. We nonetheless recommend carrying out the configuration of the TB20 EtherCAT® bus coupler in an EtherCAT configuration tool principally using the ESI file.

7 Setup and use

7.1 Reset to factory setting



The TB20 EtherCAT® bus coupler can be reset to factory settings with a concealed button. The button can be accessed from above through the ventilation openings of the bus coupler housing.

To reset to factory settings, the button must be held pressed when switching on the voltage supply until the top three LEDs light up within a few seconds. The button can then be released and the bus coupler restarts with the factory setting.

After resetting the bus coupler to the factory setting,

- the Configured Station Alias (↗ 7.5.1) has the value 0 and
- the parameters of the bus coupler (↗ 7.5.2) are set to standard values, and
- a *stored module configuration* (↗ 7.4.4.3) is no longer present.

Resetting to factory settings via the TB20 ToolBox is also possible (↗ 7.3.9).

7.2 Connection to an EtherCAT network

For the purposes of connecting with an EtherCAT network (↗ 6.1), the TB20 EtherCAT® bus coupler is equipped with two RJ45 ports, which are labeled on the left housing side with 'X1/A IN' and 'X1/B OUT'. The top port 'X1/A IN' is planned for connection from the direction of the master. The bottom port 'X1/B OUT' can be used for connecting the following additional slaves.

ATTENTION

Please make sure that you use the two ports exactly as described for the connection of the bus coupler to an EtherCAT network. An interchanged terminal results in the incorrect determination of the position of the bus coupler within the EtherCAT network. When port 'X1/A IN' is also open, this may lead to the complete failure of EtherCAT communication.

The physical communication by way of the two ports takes place according to the Ethernet standard 100BASE-TX. In this case, industrial Ethernet patch cables (at least CAT5, no crossover) with a maximum length of 100 meters are prescribed as connection lines.

The current connection status of the two EtherCAT ports can be read on the basis of the LEDs (↗ 7.7.1) labeled with 'A L/A' and 'B L/A'.

7.3 The TB20 ToolBox

The TB20 ToolBox is free PC software for the configuration, commissioning, and diagnostics of TB20 systems. Their use is not absolutely necessary for configuring and operating the EtherCAT® bus coupler as desired. However, the TB20 ToolBox enables, for example, the comfortable configuration of the bus coupler and of the modules and can provide helpful information concerning the current system status.

A USB connection is required for communication between the TB20 ToolBox and bus couplers. To this purpose, a standard USB 2.0 cable no longer than 5 meters is required, which is equipped with a Mini-B plug for the bus coupler end.

7.3.1 Download and installation of the TB20 ToolBox

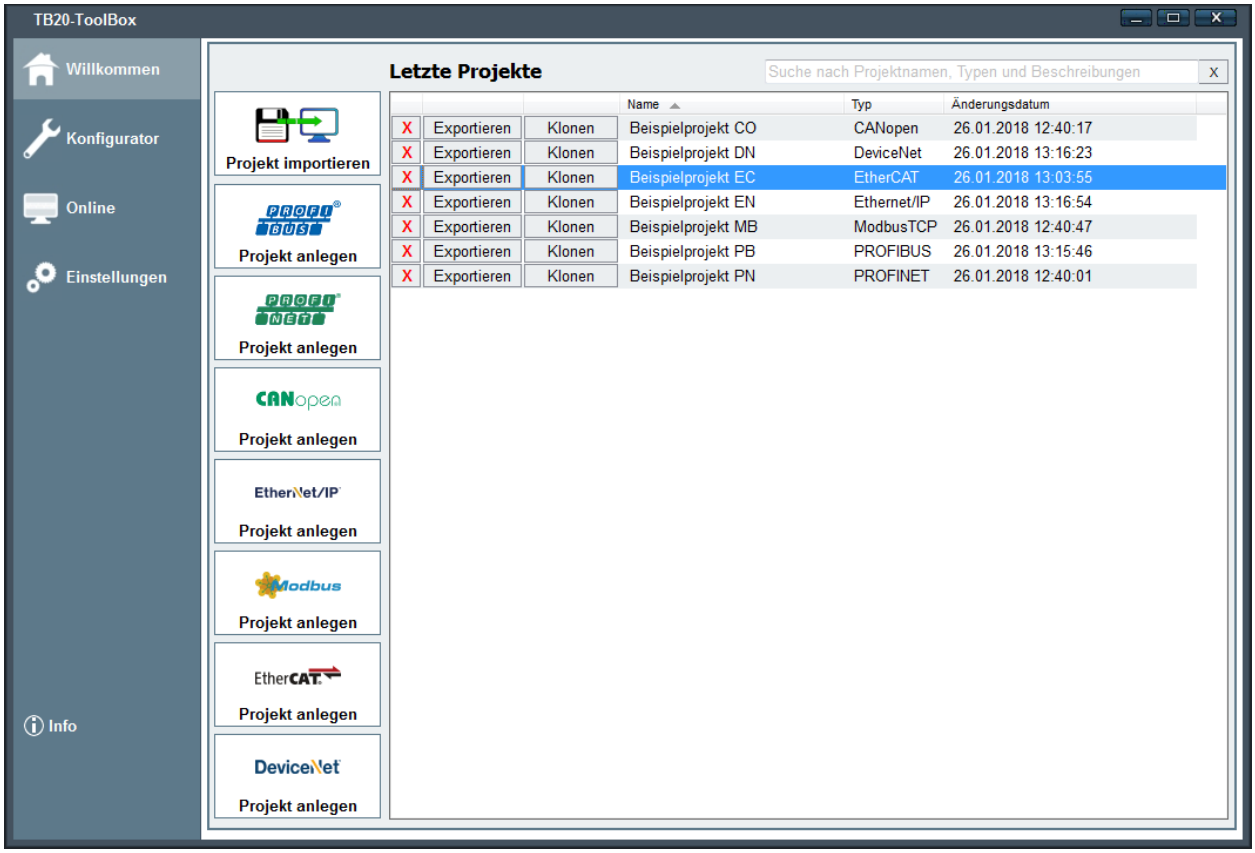
The software can be downloaded at no cost in the download area of the website www.helmholz.de. It is designed for use on PC systems with Windows 7 or Windows 10.

To install, start the downloaded setup file and follow the instructions. If an older version of the TB20 ToolBox is already installed, you will be automatically prompted to uninstall this version. All of the already present project files are maintained during the deinstallation.

The driver required for USB communication with the TB20 EtherCAT® bus coupler is also copied onto the hard drive during the installation of the TB20 ToolBox. You will find the USB driver and instructions for installation in the 'Settings' view of the started TB20 ToolBox.

7.3.2 Program start and project administration

After the start of the TB20 ToolBox, you find yourself in the 'Welcome' view. Here you can open and administer your most recently stored projects, as well as create and import new projects.

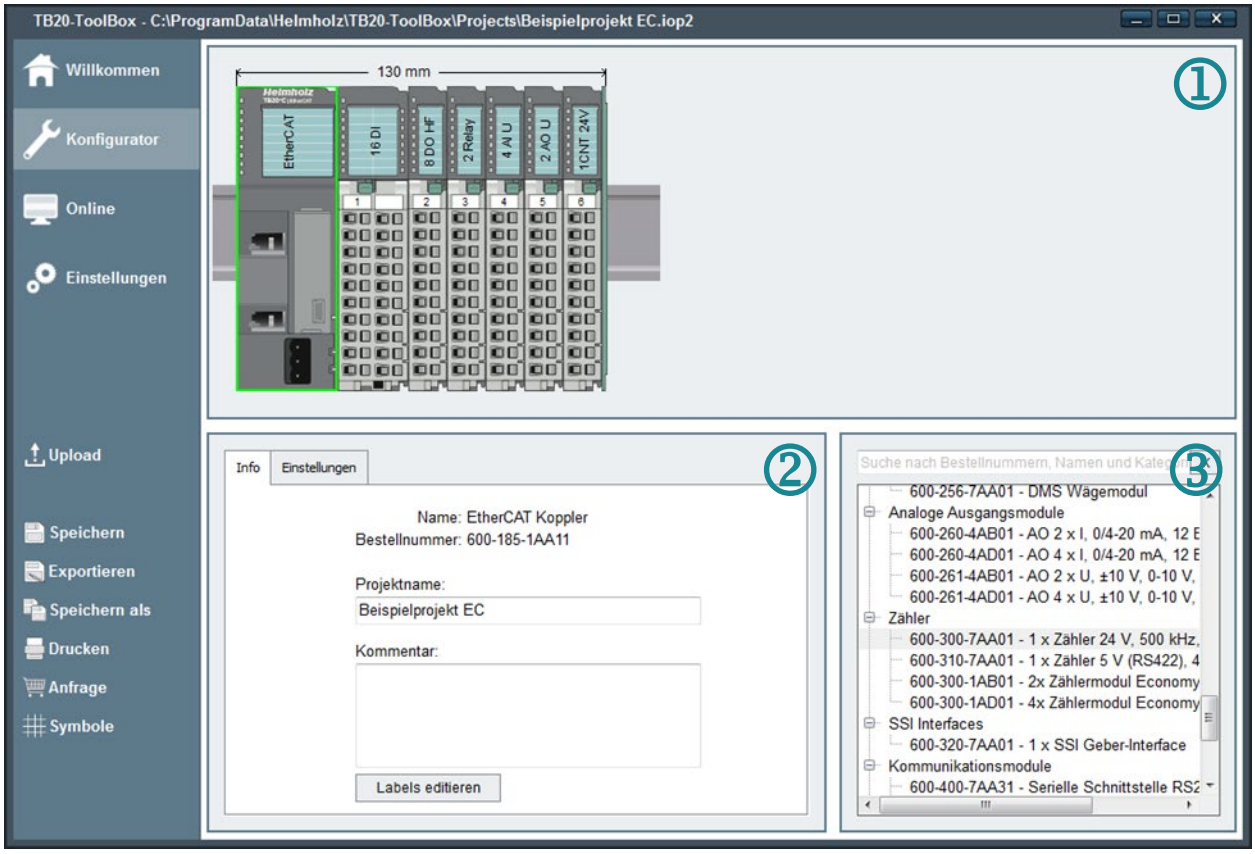


7.3.3 Creating a new EtherCAT project

If you have decided to configure the TB20 EtherCAT® bus coupler and its modules by way of the TB20 ToolBox (↗ 7.5.6), you must first create a new EtherCAT project. To this purpose, click the 'Create EtherCAT project' button in the 'Welcome' view. In the subsequently appearing dialog you will be prompted to enter a name and an optional description for the new project. You will subsequently automatically find yourself in the configurator view of the TB20 ToolBox described in the following.

7.3.4 Configuration of the bus coupler and the modules

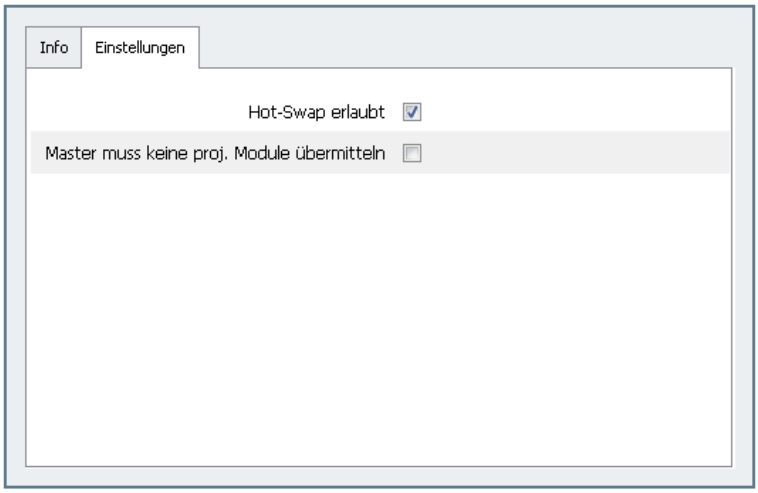
The configuration of the bus coupler and its modules takes place in the configurator view of the TB20 ToolBox. To this purpose, an EtherCAT project must be opened or a new project created in advance of this.



The configured structure of the TB20 system is graphically represented in the upper layout area. You can move, copy, insert and delete modules there with the help of the mouse. Using the module catalog (3) located to the bottom right, you can add further modules to the structure with a double-click or drag and drop.

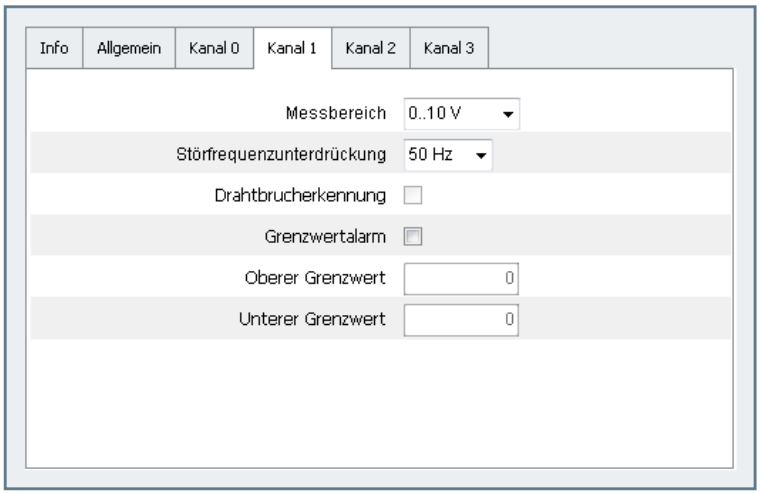
To the left next to the module catalog is an info & parameter area (2) with bus coupler or module-specific content, depending upon what is currently selected in the system structure shown above, and identified by a green outline. In addition to the display of information, the configuration of the bus coupler and of the modules takes place in this area.

The following figure shows the two parameter settings for the TB20 EtherCAT® bus coupler, the meanings of which are described in chapter 7.5.2.



For the modules, you can comfortably make the parameter settings using check boxes, drop down lists and input fields with value range testing. The meaning of the individual parameters can be looked up in the respective TB20 module manuals. You can download these in the download area of the Helmholz website www.helmholz.de.

The next figure shows the parameter settings of channel 1 for the analog module '4 AI U Iso' (600-252-7BD01) as an example.



After you have completed the configuration of the bus coupler and the modules, you should store the settings in the current project. You can use the export function for creating an external backup or for forwarding the project.

7.3.5 Upload of the configuration into the bus coupler

With the 'Upload' button you can upload the created configuration into a TB20 EtherCAT® bus coupler connected by USB, where it is stored safe from power failures. Following an automatic restart, the bus coupler is preconfigured with this stored configuration.

ATTENTION

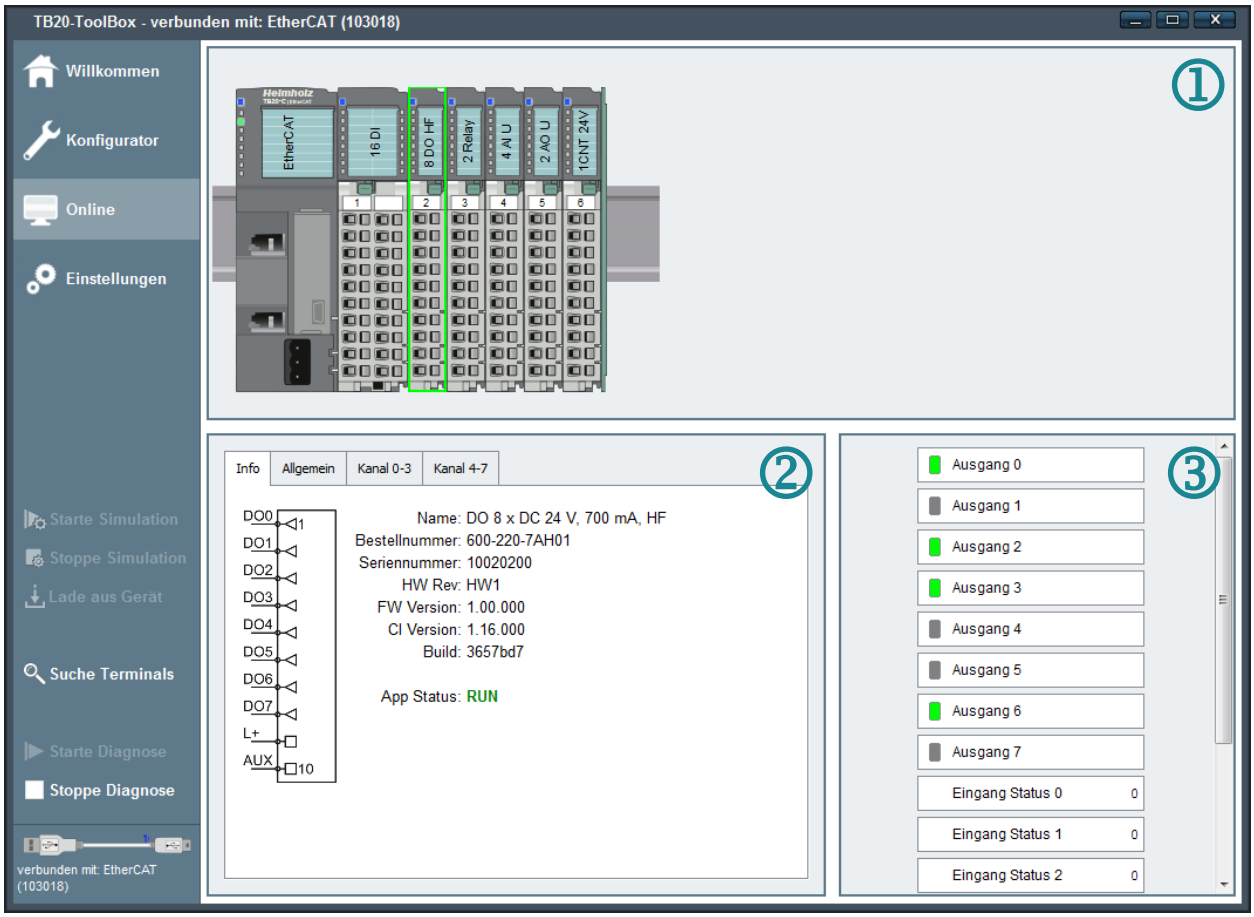
Please note that the bus coupler reverts to the INIT state when uploading the configuration. A cyclical I/O data exchange with the master is canceled and is only possible again following the automatic restart of the bus coupler.

The modules you have compiled and their parameter data are designated as a *stored module configuration* (↗ 7.4.4.3) following the uploading of the configuration. When starting up with a *stored module configuration*, the bus coupler expects that the *stored modules* are plugged in and automatically uses the *stored parameter data* for this.

You also have the possibility to upload a configuration to the bus coupler without added and configured modules. In this case, the set bus coupler parameters are adopted, but a *stored module configuration* is then no longer present in the bus coupler. The configuration of the modules must then be carried out with the help of the EtherCAT configuration tool.

7.3.6 The online view

Click the 'Online' button to the left to go to the online view of the TB20 ToolBox. Here you can carry out a real time diagnosis (↗ 7.3.7) of the TB20 system or change to simulation operation (↗ 7.3.8) for commissioning. The prerequisite for this is that the relevant bus coupler is connected by USB with the PC. In addition to this, you can reset the connected bus coupler to the factory setting from the online view (↗ 7.3.9) or carry out an update of the bus coupler firmware (↗ 7.3.10).



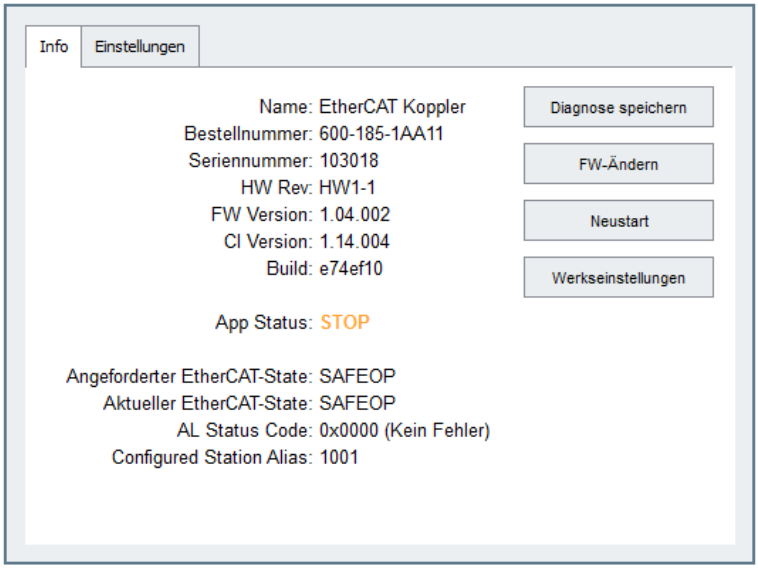
Similar to the configurator view, the online view is divided into three areas. In the upper layout area (1), the connected bus coupler is represented graphically with its currently plugged modules. By clicking with the mouse, you can select the bus coupler or a module there, which is then identified by a green outline. The information corresponding with this selection is shown in the info & parameter area (2) located at the bottom left. To the right if this is the I/O data area (3), in which, irrespective of the current status, the input and output data of a selected module is shown.

7.3.7 Real time diagnosis

For the real time diagnosis of a TB20 EtherCAT® coupler connected via a USB and its modules, you must change to the previously described online view. In the layout area you can see the current structure of the connected TB20 system. The current status of the status LEDs of bus couplers and modules is also shown in this view.

In the info & parameter area you can, depending upon the selection, have the current status of the bus coupler or of a module, as well as the currently used parameter settings displayed. A potentially present diagnostic message may also be shown here. You can recognize the diagnostic message at the bus coupler and the modules on the basis of a solid red or flashing red OK/BF or OK/SF LED.

As can be seen in the following figure, EtherCAT-specific information is also shown for the bus coupler in the info section. This includes the current and the requested EtherCAT state (↗ 6.5.1), the AL status code (↗ 6.5.2.3) with a brief description and the Configured Station Alias (↗ 7.5.1).



If the bus coupler is in the SAFEOP or OP state, meaning in the cyclical I/O data exchange with the master, then the current input and output data of the selected module is displayed in the I/O data range.

7.3.8 Simulation mode

With the simulation mode you can assume control of the output data of the modules from the TB20 ToolBox. As result, and independent of an EtherCAT master or a PLC, you have the possibility to commission the TB20 system and check it for correct function and wiring.

In order to start the simulation mode, you must be in the online view (↗ 7.3.6), have selected the bus coupler connected by USB and click 'Start simulation' to the left on the button. The prerequisite for a successful change to simulation mode is that at least one module is plugged in at the bus coupler and that no module gaps are present.

ATTENTION

Please note that the TB20 EtherCAT® bus coupler reverts to the INIT state on the EtherCAT side when changing to the simulation mode. A cyclical I/O data exchange with the EtherCAT master is canceled and is only possible again following the end of simulation mode.

If the simulation mode is started in the INIT, PREOP or SAFEOP state of the bus coupler, the currently plugged in modules are activated with their current parameter settings in the app state RUN. When starting from the OP state, the modules are already in the app state RUN. In this case, the output data of the modules most recently sent by the master are further used without gaps.

During simulation mode, the OK/BF LED of the bus coupler flashes blue rapidly, and 'SIMULATION' is clearly displayed as the 'App Status' in the info area of the bus coupler. In addition to the possibilities familiar from real time diagnostics (↗ 7.3.7), you now also have the possibility to control the output data of the modules and to change their parameter settings. In both cases, you must press the 'Activate' button to accept the changes made.

To end the simulation mode, click the 'Stop simulation' button. The simulation mode also automatically ends when the USB connection with the bus coupler is interrupted or when at least one or, in the case of an allowed hot swap (↗ 7.6), at least two modules have been pulled.

Following the end of simulation mode, the plugged modules are then switched to the app state IDLE or STOP, depending upon the then *active module configuration* (↗ 7.4.4.1). The bus coupler is still in the INIT state, but can now once again be controlled from the master.

7.3.9 Reset to factory setting

As an alternative to the reset to factory settings with a button described in chapter 7.1, you can also carry out this function with the help of the TB20 ToolBox. To this purpose, connect the TB20 EtherCAT® bus coupler via USB and change to the online view (↗ 7.3.6). Following the selection of the bus coupler in the layout area, the 'Factory setting' button is available in the layout area, with which you can initiate the reset to factory settings.

ATTENTION

Please note that the TB20 EtherCAT® bus coupler reverts to the INIT state when resetting to the factory setting. A cyclical I/O data exchange with the master is canceled and is only possible again following the automatic restart of the bus coupler.

7.3.10 Firmware update of the bus coupler

The firmware files required for an update of the bus coupler firmware are contained as a standard in the installation scope of the TB20 ToolBox. In order that the most recent available firmware version is available for a firmware update, you should download and install the latest version of the TB20 ToolBox prior to this (↗ 7.3.1).

The version of the firmware currently found on a bus coupler can be shown in that you connect the bus coupler via USB, change to the online view (↗ 7.3.6) of the TB20 ToolBox, and select the bus

coupler in the top layout area. You will then be shown the version information in the info area with 'FW Version'. If an orange symbol with an exclamation mark is next to this version indication, than a more recent firmware version for updating the bus coupler is available.

To carry out a firmware update, click in the previously named info area on the 'FW Update' button. If the TB20 ToolBox is not aware of any new firmware version, then this button will instead be called 'Change FW'. After clicking on the button, a dialog for the selection of the desired firmware version appears. The newest available firmware version is first in line and is preselected as a standard. In order to confirm the selection and start of the firmware update, click 'OK' to conclude.

During the firmware update, the OK/BF LED of the bus coupler flashes red and the progress of the update is displayed in the form of a progress bar in the TB20 ToolBox. The firmware update ends with a restart of the bus coupler. Shortly following this, the version of the updated bus coupler firmware is shown in the info area.

ATTENTION

Please note that the TB20 EtherCAT® bus coupler reverts to the INIT state when starting the firmware update. A cyclical I/O data exchange with the master is canceled and is only possible again following the firmware update and an automatic restart of the bus coupler.

7.4 The modules

This chapter contains important information concerning the use of the modules on the TB20 EtherCAT® bus coupler.

7.4.1 Module designations and their meaning

In connection with the TB20 EtherCAT® bus coupler, a differentiation is made between *detected, stored, projected*, and *configured modules*. These module designations are written in italics in the manual to explicitly draw attention to their special importance.

7.4.1.1 Detected or plugged modules

Those modules are designated as *detected modules* that are recognized (detected) by the bus coupler on the backplane bus. As a rule this should involve the modules plugged into the backplane bus, which is why the designation *plugged modules* is used equivalently.

The *detected modules* are provided by the bus coupler via the object 0xF050 (Detected Module Ident List, ↗ 9.6.14). If, however, a *stored module configuration* (↗ 7.4.4.3) is present, then this object is generally overlaid with the *stored modules*.

7.4.1.2 Stored modules

Those modules are designated as *stored modules* that are components of a *stored module configuration* (↗ 7.4.4.3). If a *stored module configuration* is present in the bus coupler, then the object 0xF050 (Detected Module Ident List, ↗ 9.6.9) is generally overlaid with the *stored modules*. This object otherwise contains the *detected modules*.

7.4.1.3 Projected modules

The modules selected in the EtherCAT configuration tool for the operation of the bus coupler are designated as *projected modules*. The term *'projected'* was selected to be able to more clearly differentiate these modules from the *modules configured* in the bus coupler. An automated comparison of the *configured* with the *projected modules* usually takes place prior to the change of the bus coupler following SAFEOP. To this purpose, the master describes object 0xF030 (Configured Module Ident List, ↗ 9.6.13) of the bus coupler with the *projected modules*.

7.4.1.4 Configured modules

The modules contained within the object 0xF030 (Configured Module Ident List, ↗ 9.6.13) are designated as *configured modules*. The *configured modules* define the plugged modules with which the bus coupler should be operated in SAFEOP and OP. You also already define in PREOP the modules for which corresponding module objects (↗ 9.7) are shown in the object directory of the bus coupler. A configuration of the currently *configured modules* is possible via the shown parameter data objects. The *configured modules* and the related parameter data are summarized as the *current module configuration* (↗ 7.4.4.2) of the bus coupler.

During the transition from INIT to PREOP, the object 0xF030 named above is initialized with the *stored modules* where a *stored module configuration* is present; otherwise with the *detected modules*. This means that either the *stored* or the *detected modules* are initially used as *configured modules* in the PREOP state. However, the object 0xF030 is still normally described by the master with the *projected modules* during the PREOP state.

Whether the *configured modules* are currently based on the *detected*, the *stored*, or the *projected modules* can be determined from the subindices 2 to 4 of the object 0x2100 (Module Configuration Status, ↗ 9.6.7).

7.4.2 Module IdentNumbers

For the purposes of differentiating the various types of modules that can be used on the TB20 EtherCAT® bus coupler, clear module identifiers are used for the EtherCAT, which are designated as Module Ident Numbers. The Module Ident Numbers are used, for example, in connection with the objects 0xF030 (Configured Module Ident List, ↗ 9.6.13) and 0xF050 (Detected Module Ident List, ↗ 9.6.14) of the bus coupler.

The Module Ident Numbers encompass 32 bit and are composed as follows.

Bit 31 = 16	Bit 15-8	Bit 7-0
TB20 module identification	Operating mode ID	reserved (always 0)

The 16 bit-wide *TB20 module identifier* is a clearly assigned identifier for TB20 modules. It is listed in the module manuals respectively in the 'Technical Data'.

In the case of modules with various operating modes, the 8 bit-wide *operating mode ID* ensures that every operating mode is assigned its own Module Ident Number. This is necessary because, depending upon the operating mode, such modules usually provide different parameters, as well as input and output data, and therefore represent different modules from an EtherCAT perspective.

For the purposes of selecting the operating mode, the first byte of the parameter data contains the operating mode parameters for the relevant modules. The value that needs to be set first for the selection of a particular operating mode corresponds to the value used as the operating mode ID in the respective module. In the case of modules that do not support different operating modes, the value 0 is contained in the Module Ident Number as the operating mode ID.

7.4.3 Parameter data and configuration

For a large part of the TB20 modules it is possible to adapt the behavior and the function to the respective application case by way of configuration. All settings relating to such a configuration are contained in the parameter data of the respective module.

Several configurable modules support various operating types. Different parameters are generally defined for each operating mode, so that they also differ in terms of structure and content of the parameter data. The only unifying factor is that the operating mode parameters are always contained in the first byte of the parameter data. The various operating modes of a module represent different modules with different Module Ident Numbers (↗ 7.4.2) from the perspective of the EtherCAT.

The parameter data of the *configured modules* are contained in the object directory of the bus coupler in the form of parameter data objects (↗ 9.7.3). When starting up the bus coupler, these objects can be described by the master with the module parameter data previously configured in the EtherCAT configuration tool (↗ 6.6). Reconfiguration of the modules during ongoing operation is also possible, in that the parameter data objects are accessed with writing rights from the PLC program.

In order to carry out the module configuration via the parameter data objects, it is generally necessary to inform oneself in the corresponding module manual in detail about the coding and the structure of the parameter data.

An alternative configuration of the modules without the detailed knowledge required for the parameter data objects can be carried out with the help of the TB20 ToolBox (↗ 7.5.6). The configurator (↗ 7.3.4) contained in the TB20 ToolBox is used to this purpose. Following the selection of the desired modules, the settings for the individual module parameters can be made comfortably by way of check boxes, drop down lists, and input fields with value range testing. The selected modules are then transferred to the bus coupler together with the resulting parameter data and stored there as a *stored module configuration* (↗ 7.4.4.3). Separate configuration of the modules in the EtherCAT configuration tool is then no longer necessary. However, the prerequisite for this is that the modules *projected* there and *configured* in the bus coupler match with the *stored modules*. If this is not the case, the bus coupler cannot change to the EtherCAT state SAFEOP.

7.4.4 Module configurations of the bus coupler

A certain compilation of modules and the parameter data of these modules are designated in a summary fashion as a *module configuration*. A differentiation is made in the bus coupler between an *active*, a *current*, and a *stored module configuration*.

7.4.4.1 Active module configuration

As a result of the *active module configuration*, the app state and the parameter data of the modules plugged in at the the backplane bus are defined:

- Plugged modules that correspond to the *active module configuration* are found, depending upon the current EtherCAT state of the bus coupler, in the app state STOP or RUN. Configurable modules are also supplied with the parameter data contained in the *active module configuration*.
- Plugged modules that *do not* correspond to the *active module configuration* are found, depending upon the current EtherCAT state of the bus coupler, in the app state IDLE. Configurable modules are reset to their default parameters.

In the case of an arbitrary state transition to INIT, an existing *stored module configuration* is adopted as an *active module configuration*. If no *stored module configuration* is present, then the *active module configuration* contains no modules. In this case, all plugged modules are then in the app state IDLE.

In the case of the transition from PREOP to SAFEOP, the *current module configuration*, to the extent valid, is adopted as an *active module configuration*. This adoption takes place irrespective of whether the transition to SAFEOP could ultimately be carried out successfully or not.

Other state transitions have no effect on the *active module configuration*. However, changes to the parameter data objects that have been carried out in SAFEOP or OP are not only adopted in the current, but also in the *active module configuration*, and are thus transmitted to the corresponding module. If access takes place without complete access, the changes must subsequently be confirmed for reasons of consistency. In order to confirm, the subindex 0 of the changed parameter data object must be described with the value already contained in subindex 0. If the parameter data object is changed via complete access, this confirmation is no longer necessary.

7.4.4.2 Current module configuration

The *current module configuration* is reflected through the content of the subsequently described objects in the object directory of the bus coupler:

Object 0xF030 (Configured Module Ident List, ↗ 9.6.13)

This object contains the list of the current *configured modules*. During the transition from INIT to PREOP, it is initialized with the *stored modules* where a *stored module configuration* is present; otherwise with the *detected modules*. In the latter case, changes to the *detected modules* updated in the object for as long as the bus coupler is in PREOP and the object has not been overwritten by the master.

The object 0xF030 is usually overwritten by the master in connection with the state transitions INIT→PREOP or PREOP→SAFEOP. In this way, the *projected modules* expected by the master are transmitted to the bus coupler and adopted as *configured modules*.

In SAFEOP and OP no more changes to the *configured* and now operating *modules* can carried out by way of this object. Writing access to the object is therefore no longer allowed in these EtherCAT states.

Parameter data objects of the configured, configurable modules (↗ 9.7.3)

Available online in the bus coupler are only the parameter data objects from such modules as are currently configured in the object 0xF030. In the PREOP state, the *configured module* must also be plugged or match with a *stored module*.

The initial values of the parameter data objects in the PREOP state depend upon whether the related *configured module* matches with the *stored* or with the *active module configuration*. When the module matches with the *stored module configuration*, then the related parameter data object is initialized with the *stored parameter data*. When the module matches with the *active module configuration*, but not with the *stored module configuration*, then the related parameter data object is initialized with the *active parameter data*. If there is no match of the module with the *stored* or with the *active module configuration*, then the related parameter data object is initialized with the default parameters read out from the module.

If a module with different operating modes is involved, a match with the operating mode with regard to the *stored* or the *active module configuration* must also prevail in order that the corresponding parameter data can be adopted as initial values. If this match does not prevail and the configured operating mode also doesn't match with the default operating mode of the module, then the related parameter data object is initialized with zeros with the exception of the contained operating mode parameters.

The parameter data objects available online can be described from the master or from the EtherCAT configuration tool into PREOP, SAFEOP and OP, in order to change the parameter data of the *configured modules*. Changes carried out in SAFEOP and OP are not adopted only into the *current*, but also into the *active module configuration*, and are thus transmitted directly to the corresponding module. If access takes place without complete access, the changes must subsequently be confirmed for reasons of consistency. In order to confirm, the subindex 0 of the changed parameter data object

must be described with the value already contained in subindex 0. If the parameter data object is changed via complete access, this confirmation is no longer necessary.

Information on the status of the *current module configuration* can be called up via the object 0x2100 (Module Configuration Status, ↗ 9.6.7). On the basis of the subindices 2 to 4, for example, it can be determined whether the list of the *configured modules contained in object 0xF030* is based on the *detected*, the *stored* or the *projected modules* transmitted by the master. And subindex 5 indicates in this connection whether this list is valid because it contains at least one module and no module gaps.

In PREOP, the object entry 0x2110:04 (Command: Store current configuration, ↗ 9.6.8) can be used to store the current module configuration in the bus coupler so that it is safe from power failures. It is then available as a *stored module configuration*.

When a transition from PREOP to SAFEOP is initiated and the list of the *configured modules* contained in the object 0xF030 is valid, then the *current module configuration* is adopted as an *active module configuration*. This means that all of the modules matching with the updated *active module configuration* are activated in the app state STOP. Configurable modules have previously also received the *stored module parameters*. All other modules are activated with default configuration in the app state IDLE.

7.4.4.3 Stored module configuration

A module configuration filed safe from power failures in the bus coupler is designated as a *stored module configuration*. Analog with this, the modules and parameter data contained within it are designated as *stored modules* (↗ 7.4.1.2) and *stored module parameters*.

If the configuration of the bus coupler and of the modules has been carried out via the TB20 ToolBox (↗ 7.5.6), then, in the case of an upload of the configuration (↗ 7.3.5), the modules contained within it and their parameter data are filed as a *stored module configuration* in the bus coupler. In the event that no modules have been configured, no *stored module configuration* is present in the bus coupler (any longer) following an upload of the configuration.

If an online configuration of the modules is carried out via the EtherCAT configuration tool, the settings made are initially contained only temporarily within the *current module configuration* of the bus coupler. The object entry 0x2110:04 (Command: Store current configuration, ↗ 9.6.8) can be used upon completion of the online configuration to file the *current module configuration* in the bus coupler as a *stored module configuration*.

It can be determined whether a *stored module configuration* is present in the bus coupler by way of the object entry 0x2100:01 (Stored configuration present). The object 0x2100 (Module Configuration Status, ↗ 9.6.7) still contains additional status information concerning a *stored module configuration* in the other subindices.

Behavior of the bus coupler

If a *stored module configuration* is present, then this has the following influence on the behavior of the bus coupler and the app state (↗ 7.4.9) of the modules:

- In the case of an arbitrary state transition to INIT, the *stored module configuration* is adopted as an *active module configuration*. This means that the plugged modules matching with the *stored module configuration* are activated in the app state STOP. Configurable modules have previously also received the *stored module parameters*. All other modules are activated with default configuration in the app state IDLE.
- In the case of the state transition from INIT to PREOP, the *stored module configuration* is adopted as a *current module configuration*. This means that the object 0xF030 (Configured Module Ident List, ↗ 9.6.13) is preconfigured with the *stored modules*. The respective parameter data objects of the modules (↗ 9.7.3) are also initialized with the *stored module parameters*.

- A transition from PREOP to SAFEOP is only possible when the *stored modules* match with the *detected modules* and, when transmitted by the master, also with the *projected modules*. This should prevent the *stored module parameters* from unintentionally not being used at all or only in part.

Deletion of a stored module configuration

A *stored module configuration* present in the bus coupler can be deleted in three different ways:

- If the bus coupler is in the PREOP state, deletion of the *stored module configuration* can then take place via the object entry 0x2110:05 (Command: Erase stored configuration, ↗ 9.6.8).
- From the TB20 ToolBox, a deletion of the *stored module configuration* can be achieved in that a configuration is uploaded into the bus coupler that contains no modules (↗ 7.3.5).
- A *stored module configuration* is also deleted by resetting the bus coupler to the factory setting (↗ 7.1). However, the bus coupler parameters and the Configured Station Alias are then also reset to their standard values.

7.4.5 Input and output data

In order to enable cycle-current access to the input and output data of the *configured modules*, these are always automatically contained in the process input or process output data of the bus coupler (↗ 7.5.4). They are also shown as input data objects (↗ 9.7.4) and output data objects (9.7.5) in the object directory of the bus coupler. These objects are irrelevant for access to the input and output data of the modules. However, they are required as reference objects for the PDO mapping objects (↗ 9.8.1), which, in connection with the PDO assignment objects (↗ 9.8.4), clearly define how the process data of the bus coupler is compiled.

Which input and output data a module possesses, and how this is structured is described in the respective module manual.

7.4.6 Submodule status

A module status is provided for each *configured module*, which contains significant information concerning the current status of the module. In order to enable cycle-current evaluation in the PLC, the module statuses of the *configured modules* are therefore also contained in the process input data (↗ 7.5.4) of the bus coupler. The module status of each *configured module* is also shown in the object directory in the form of module status objects (↗ 9.7.6).

The module status encompasses a byte and indicates in bits 0 and 1 whether the relevant module is *present* and *operational*.

Bit 7 = 2	Bit 1	Bit 0
Reserved always 0	**'Operational'** 0 = *not operational* 1 = *operational*	**'Present'** 0 = *not present* 1 = *present*

On bit 0 (Present):

A *configured module* is then shown as *present* when a compatible module is plugged into the corresponding plug point. If no or an incorrect module is plugged there, then the *configured module* is shown as *not present*.

When a module exchange during ongoing operation (hot swap, ↗ 7.5.2.1) is allowed, then the PLC program should carry out an evaluation of this bit for all modules in order to be able to react accordingly to the pulling of a module.

On bit 1 (Operational):

In order that a *configured module* is displayed as *operational*, the following conditions must be met:

- the bus coupler must be in the EtherCAT state SAFEOP or OP
- the module must be *present* according to bit 0
- in the case of diagnostics-capable modules, no module-wide diagnostic message may be present (↗ 7.4.8)

If one of these conditions is not met, the module will then be displayed as *not operational*.

Presuming that the bus coupler is in the SAFEOP or OP state and the module is *present*, then a not operational state means that there is a module-wide diagnostic message in the module. In this case, the cause for the diagnostic message can be read out of the diagnostic status (↗ 7.4.8) of the module.

7.4.7 Channel status

A channel status is provided by the bus coupler for each diagnostics-capable *configured module* with input and/or output channels. This contains important status information concerning these channels. In order to enable cycle-current evaluation in the PLC, the channel statuses of the modules are therefore also contained in the process input data (↗ 7.5.4) of the bus coupler. The channel status of each *configured module* is also shown in the object directory in the form of channel status objects (↗ 9.7.7).

The channel status of a module contains a bit for every channel which displays the status of the channel with 1 = *OK* and 0 = *not OK*. These status bits are ordered with an ascending channel number starting with bit 0 in the channel status. Depending upon the number of channels, the channel status encompasses 1 to a maximum of 4 bytes. Unused bits within the channel status are indicated with zeros. The following represents the structure of the channel status using a 4-channel module as an example.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Not used 0	Not used 0	Not used 0	Not used 0	'Channel 3 OK' 0 = not OK 1 = OK	'Channel 2 OK' 0 = not OK 1 = OK	'Channel 1 OK' 0 = not OK 1 = OK	'Channel 0 OK' 0 = not OK 1 = OK

The status of a channel is then shown as *OK* when the following conditions have been met:

- the bus coupler is in the EtherCAT states SAFEOP or OP
- the module is (↗ 7.4.6) *present* according to the module status and *operational*
- there is no diagnostic message present for the channel

If one of these conditions is not met, the channel status will then be displayed as *not OK*.

If the status of a channel is *not OK*, although the module is *present* and *operational* according to the module status, then a diagnostic message is present for this channel. In this case, the cause for the channel-related diagnostic message can be read out of the diagnostic status (↗ 7.4.8) of the module.

In the PLC, the channel status should also always be evaluated in connection with the input and output data of a diagnostics-capable module. It can thereby serve as an indicator for the validity of input data and the effectiveness of output data.

7.4.8 Diagnostic status

A diagnostic status is provided by the bus coupler for each diagnostics-capable *configured module*. It contains information on the diagnostic messages present in the module. Access to the diagnostic status must take place by way of the object directory of the bus coupler, where it is contained for the relevant modules in the form of diagnostic status objects (↗ 9.7.8).

The diagnostic status of a module is composed of a module-related diagnosis ID, and for each channel of another channel-related diagnosis ID. The individual diagnosis IDs are precisely one byte each. The following represents the structure of the diagnostic status using a 4-channel diagnostic-compatible module as an example.

Byte	Content of the diagnostic status
0	Module-related diagnosis ID
1	Diagnostic ID of channel 0
2	Diagnostic ID of channel 1
3	Diagnostic ID of channel 2
4	Diagnostic ID of channel 3

The values for the diagnostic IDs contained in the diagnostic status are assigned to certain diagnostic messages. The following table displays this assignment for a selection of usual diagnostic messages.

Diagnostic ID	Diagnostic message
0	No error; there is no diagnostic message present
2	Undervoltage
3	Overvoltage
4	Overload
5	Excess temperature
6	Line break / Wire break
7	Upper measuring range limit exceeded (overflow)
8	Lower measuring range limit fallen below (underflow)
17	External reference voltage missing (L+)
18	Configuration error (general)
20	Configuration error (maximum parameter data length exceeded)

You can consult the respective module manual to determine which diagnostic messages are supported by a certain module in concrete terms.

The reading out and evaluation of the diagnostic status always makes sense when a more precise determination is required as to what a present diagnostic message refers to. Whether a diagnostic message is currently present for a module can be determined for the current cycle on the basis of the module status (↗ 7.4.6) and the channel status (↗ 7.4.7).

When a module-related diagnostic message is present, then the module is in its entirety, meaning including all channels, either no longer functional or only functional with restrictions. For this reason, the channel-related diagnostic ID then also contains the value of the module-related diagnostic ID.

7.4.9 Operating state app state

The operating state of the modules is designated as app state. A differentiation is made between the four app states INIT, IDLE, STOP, and RUN. In most cases, the OK(/SF) LED of a module can be used to read the app state in which the module is currently found (↗ 7.7.2). The current app state of the modules can otherwise also be displayed in the TB20 ToolBox by using the real time diagnostics (↗ 7.3.7).

App state INIT

The app state INIT is a brief, temporary state that occurs after plugging in the module or switching on the supply voltage. It can also be triggered by the bus coupler to reset the module in a targeted fashion.

After the module has completed an internal initialization, it changes automatically to the app state IDLE. In the case of configurable modules, the loading of default parameters is also part of the internal initialization.

App state IDLE

The app state IDLE is is the basic state of a module. There is no exchange of I/O data with the bus coupler, and the physical outputs of the module are found in a secure state.

App state STOP

An exchange of I/O data with the bus coupler can take place in the app state STOP. The module makes current input data available to the bus coupler in the process. However, output data transmitted from the bus coupler to the module is ignored. The physical outputs of the module are found in a secure or possibly specially configured state.

App state RUN

A cyclical exchange of I/O data with the bus coupler takes place in the app state RUN. The module makes current input data available to the bus coupler in the process. Output data transmitted from the bus coupler to the module is adopted and determines the state of the physical outputs of the module.

Dependency upon the active module configuration and the EtherCAT state

The app state of a module plugged into the backplane bus is initially dependent upon whether it matches with the *active module configuration* (↗ 7.4.4.1) of the bus coupler or not. Where there is no match, the module is in the app state IDLE. Where there is a match, the app state of the module is also then dependent upon the current EtherCAT state of the bus coupler: Such a module finds itself in the EtherCAT states INIT, PREOP and SAFEOP in the app state STOP. It is then in the app state RUN in the EtherCAT state OP.

7.5 Configuration of the bus coupler

In order to operate the TB20 EtherCAT® bus coupler on an EtherCAT master, a corresponding configuration of the bus coupler within an EtherCAT configuration tool is necessary. This configuration can be subdivided into the following areas:

- Configured Station Alias (↗ 7.5.1)
- Parameters of the bus coupler (↗ 7.5.2)
- Module configuration (↗ 7.5.3)
- Process data configuration (↗ 7.5.4)
- Process output data watchdog (↗ 7.5.5)

The TB20 ToolBox can be used to carry out a preconfiguration of the bus coupler (↗ 7.5.6) with regard to its parameters and the module configuration. The preconfigured settings are stored in the bus coupler (↗ 7.3.5) and need not then be carried out again in the EtherCAT configuration tool.

7.5.1 Configured Station Alias

The TB20 EtherCAT® bus coupler supports the use of the Configured Station Alias for the purpose of its clear identification within an EtherCAT network. In the process, this involves a 16 bit value that is stored in the bus coupler and can be read out by the master via the ESC directory to address 0x0012.

Unique identifiability is required by the master when the bus coupler is the first or only member of a hot connect group (↗ 6.1) that can be coupled with or decoupled from potentially different points of the EtherCAT network during ongoing operation. The identification can otherwise be used optionally to make an unintentional exchange of two slaves of the same type detectable when starting up.

The Configured Station Alias must be set to the value 0 in the bus coupler, and must be changed in advance for use for identification. The EtherCAT configuration tool is used to make this change. A new value for the Configured Station Alias can be set there and then stored in the bus coupler. In the process, the new value should be chosen such that it is only used by this bus coupler within the EtherCAT network. In order that saving the value from the configuration tool is possible, the bus coupler must be connected via EtherCAT and be accessible.

NOTE

Subsequent to saving, the bus coupler must be restarted, because the newly stored value for the Configured Station Alias is only then adopted into the corresponding ESC directory.

You can find out in detail how this configuration of the Configured Station Alias needs to be carried out with the EtherCAT configuration tool you are using from the corresponding documentation.

NOTE

When exchanging a bus coupler that is clearly identifiable by the master, it must be ensured that the same value has been stored for the Configured Station Alias in advance in the replacement device.

The current value of the Configured Station Alias in the bus coupler can be displayed with the help of the TB20 ToolBox. Using the real time diagnosis (↗ 7.3.7), it is displayed in the info area following the selection of the bus coupler.

7.5.2 Parameters of the bus coupler

The bus coupler has two parameters with which the behavior of the bus coupler can be influenced when starting up and during ongoing operation. Both parameters can be configured by way of the object directory of the bus coupler or with the help of the TB20 ToolBox.

When configuring by way of the object directory, the object 0x3000 (Coupler Parameters, ↗ 9.6.11) must be accessed. Changes to the parameters contained within it made online are adopted and stored safe from power failures. A change to the parameters during ongoing operation, meaning in the states SAFEOP and OP, is also possible in this way.

When using the TB20 ToolBox, the configuration of the bus coupler takes place by way of an EtherCAT project (↗ 7.3.4) created there. With the uploading of the project configuration into the bus

coupler, the set parameters are also stored in the bus coupler. Changing parameters during ongoing operation is not possible in this way.

7.5.2.1 Module exchange during ongoing operation (hot swap)

With the parameters described here it is possible to set whether a module exchange during ongoing operation (hot swap, ↗ 7.6) is allowed or not. In the standard setting, such a module exchange is not allowed and in SAFEOP and OP results in the bus coupler changing to PREOP-ERROR.

In the object directory, the parameter can be set by way of the object entry 0x3000:01 (Hot-swapping allowed, ↗ 9.6.11). In the TB20 ToolBox, the parameter has the designation 'Hot swap allowed' and can be found in the configurator view following selection of the bus coupler under 'Settings'.

7.5.2.2 Transfer of projected modules during startup

With the parameters described here, it can be set whether a successful startup of the bus coupler is also possible when the master does not transmit a list with *projected modules*. In the object directory, this parameter can be set using the object entry 0x3000:02 (Master need not provide projected modules, ↗ 9.6.11). In the TB20 ToolBox, the parameter has the designation 'Master need not transfer proj. modules' and can be found in the configurator view following selection of the bus coupler under 'Settings'.

For the following explanations of these parameters, a differentiation will be made between *projected*, *detected*, and *stored modules*. Chapter ↗ 7.4.1 describes which modules are involved.

A sensible process data exchange between master and bus coupler can only take place when the *projected modules* expected by the master match with the *detected modules* plugged in at the backplane bus. The bus coupler therefore carries out standard testing during the state transition PREOP→SAFEOP (↗ 8.2.8) to determine whether this match applies, and only then allows a change to the SAFEOP state including start of the cyclical process data communication.

If a *stored module configuration* is present, then there must also be a match between *projected* and *stored modules*. This should prevent the module parameter data contained in the *stored module configuration* from unintentionally not being used at all or only in part.

Carrying out these module tests presumes that the master has previously transmitted the *projected modules* to the bus coupler. In the standard setting of the parameters described here, the bus coupler cannot change to the SAFEOP state without this transmission and the subsequent successful testing of the modules. This means, when the standard setting is used, the bus coupler itself ensures that a cyclical process data exchange with the master only takes place when *projected* and *detected*, and possibly *stored modules* match with one another.

The ESI device description file of the bus coupler contains the specification that the *projected modules* should be transmitted in connection with the state transition INIT→PREOP. On the basis of this specification, the EtherCAT configuration tool automatically adds an entry to the CoE initialization list. Depending upon the tool used, this entry can be designated as 'Download Slot Configuration' or something similar. In concrete terms, the entry defines a writing access to be carried out by the master on the state INIT→PREOP involving the object 0xF030 (Configured Module Ident List, ↗ 9.6.13) of the bus coupler. With the writing access to this object, the master transmits the required *projected modules* to the bus coupler.

Several EtherCAT configuration tools configure the master in a way deviating from this, so that the writing access to object 0xF030 doesn't take place for INIT→PREOP, but instead first with PREOP→SAFEOP. This is unproblematic, because the *projected modules* are nonetheless available for a check on time.

It may be the case that the EtherCAT configuration tool or the master do not support the transmission of the *projected modules* in the previously described automatic fashion. Or transmission of this kind

may not be possible because you are carrying out the configuration of the bus coupler without an ESI file. Only in these cases is it then necessary to change the setting of the parameters in such a way that a change of the bus coupler following SAFEOP and a process data exchange with the master are also possible without transmission of the *projected modules*.

If, in the case of a changed parameter setting, no *projected modules* are in fact transmitted by the master to the bus coupler, then the checks with the *projected modules* for the state transition PREOP→SAFEOP are dispensed with without this being canceled. Whether a match between the *projected* and the *detected modules* exists can only still be determined indirectly and with considerable restrictions for the configuration check of the Sync Managers 2 and 3. If the size of the process input and process output data configured by the master do not match with the sizes expected by the bus coupler, this is generally because both sides have presumed different module constellations. In this case, the bus coupler changes with the AL status code 0x001D (Invalid process output data configuration for Sync Manager 2) or 0x001E (Invalid process input data configuration for Sync Manager 3) after PREOP-ERROR.

Conversely, it cannot be reliably concluded on the basis of matching process data sizes that the modules expected by the master and bus coupler match, because this can also be the case with differing module constellations. Especially differences in the order of the modules cannot be detected in this way, as the arrangement of the process data is influenced by this, but not their size. For this parameter setting, it is therefore the responsibility of the PLC programmer to ensure in a suitable fashion that the modules from the master and bus coupler match. This can take place, for example, from the PLC program in which the *detected modules* contained in the object 0xF050 (Detected Module Ident List, ↗ 9.6.14) are read out and compared with the expected modules.

CAUTION

If the master does not transmit a list with *projected modules* to the bus coupler, then the bus coupler cannot itself ensure for the state transition PREOP→SAFEOP that the modules expected by the master match with the modules *configured* or *detected* in the bus coupler. This is why it must be otherwise ascertained that a change of the bus coupler after SAFEOP and OP only takes place with a match of all modules!
If the change following SAFEOP and OP takes place without this match, then this must result in an incorrect assignment of input and output data, which then generally also affects those matching modules. The incorrect assignment of input and output data results in unintentional and for the most part unpredictable system behavior, which can cause damage to devices!

7.5.3 Module configuration

The configuration of the bus coupler modules encompasses a selection of the desired modules and their configuration. For EtherCAT®, this configuration usually takes place with an EtherCAT configuration tool (↗ 6.6). Alternatively, the TB20 ToolBox can also be used for the configuration of the modules (↗ 7.5.6). Especially the configuration of the modules is comfortably possible via the TB20 ToolBox and also requires no background knowledge concerning the structure of the parameter data. Reconfiguration of the modules during ongoing operation is also possible from the PLC program.

7.5.4 Process data configuration

7.5.4.1 General information on the process data of the bus coupler

The process data is data that is available for direct access to the PLC program and is to this purpose cyclically exchanged between master and bus coupler in the EtherCAT states SAFEOP and OP.

Depending upon the transmission direction, a differentiation is made between the process output data and the process input data of the bus coupler.

The *process output data* is cyclically transmitted from the master to the bus coupler. It contains the output data of the *configured modules* and can be described by the PLC program.

The process input data is cyclically transmitted from the bus coupler to the master. In addition to the input data of the *configured modules*, it also contains a module status (↗ 7.4.6) and, in the case of diagnostics-capable modules, also a channel status (↗ 7.4.7). The process input data can be accessed from the PLC program with reading authorization.

7.5.4.2 Composition and content of the process data

The composition and the content of the process data are directly dependent upon the *configured modules* and are determined automatically in accordance with the following fixed system:

- The process output data with an ascending plug point position is composed of the strung together output data of the *configured modules*. Modules without output data do not occupy space in the process output data. In the event that only modules without output data have been configured, the bus coupler has no process output data.
- The process input data with an ascending plug point position is composed of the strung together input data, as well as the module and channel status of the *configured modules*. With reference to a module, the potentially existing input data, then the module status, and finally the potentially present channel status are strung together.
- Integers contained in the input or output data of the modules that occupy 2 or 4 points are contained in the process data in the Little Endian format. This means that the individual bytes of these numerical values are structured starting with the lowest value and ascending to the byte with the highest value.

From an EtherCAT perspective, the process data of the slaves is composed of so-called process data objects (PDOs). The content and the arrangement of these PDOs are defined by way of PDO mapping and PDO assignment objects. You can find further details concerning this in chapter 9.8. The compilation of the process data for an exemplary module selection is represented there (↗ 9.8.4.3).

7.5.5 Process output data watchdog

With the process output data watchdog, the cyclical process data communication between master and bus coupler can be monitored with regard to the process output data in the EtherCAT state OP. In the watchdog, a timer runs to this purpose, which is restarted with each reception of a valid process output data datagram. Should this timer exceed a monitoring time configured for the watchdog, the bus coupler then carries out a state transition from OP to SAFEOP-ERROR (↗ 8.2.20). In this way it can be ensured that the bus coupler automatically changes to the secure state in the event of a failure of the master or communication and wiring problems.

In the event that the bus coupler does not possess any process output data, because all *configured modules* only provide input data, the bus coupler is unable to carry out such an automatic change to the SAFEOP state. This is, however, unproblematic, because the bus coupler is also in a secure state in the OP state due to an absence of module outputs.

The process output data watchdog of the TB20 EtherCAT® bus coupler is activated with a monitoring time of 100 ms as a standard. Should this standard setting not be appropriate for your application

case, you can use the EtherCAT configuration tool to also configure shorter or longer monitoring times. Deactivation of the watchdog is not recommended, but it is usually possible that the value 0 is configured for the monitoring time.

If the process output data watchdog has been deactivated, no automatic change to the secure state takes place upon interruption of communication with the master. This can result in undesired behavior on the part of the system and damage to devices.

In the various EtherCAT configuration tools, the process output data watchdog is often designated in a generalizing way as the process data watchdog, or also as Sync Manager watchdog or SM watchdog. You can find out how the configuration of the watchdog is to be carried out in detail in the documentation of the EtherCAT configuration tool used.

7.5.6 Preconfiguration by way of the TB20 ToolBox

For the preconfiguration of the bus coupler by way of the TB20 ToolBox, certain settings are made for the subsequent operation of the bus coupler and of the modules in the TB20 ToolBox and then stored in the bus coupler. These include the parameters of the bus coupler and the selection and configuration of the desired modules. For the subsequent configuration of the bus coupler in the EtherCAT configuration tool, these settings then no longer need to be made again. The preconfigured modules can usually be automatically added to the bus coupler in the configuration tool when a device or module scan is carried out there.

The main advantage of preconfiguration by way of the TB20 ToolBox is found in the configuration of the modules. This can be carried out significantly more comfortably in the TB20 ToolBox and also requires no special background knowledge concerning the structure of the respective parameter data.

Preconfiguration is disadvantageous in comparison with offline configuration in the EtherCAT configuration tool when an exchange of the bus coupler is required. This is because the replacement device must then also be preconfigured accordingly, which requires the TB20 ToolBox and the appropriate EtherCAT project.

How the TB20 ToolBox can be used for the preconfiguration of the bus coupler and its modules has already been described in chapter 7.3.4.

7.6 Module exchange during ongoing operation (hot swap)

The TB20 EtherCAT® bus coupler enables the exchange of a module during ongoing operations referred to as *hot swap*. This means that you have the possibility to exchange a potentially defective module for a module of the same type, without having to interrupt ongoing operations. The bus coupler and the remaining modules continue running normally during the exchange.

However, a module exchange during ongoing operations must also also be supported by the PLC side. Additional program technology precautions are generally necessary to be able to correctly deal with the intermittent absence of a module and the related effects. For this reason, the TB20 EtherCAT® bus coupler is configured as a standard in such a way that the exchange of a module is not permitted during ongoing operations. However, if a module is nonetheless pulled during ongoing operations, the bus coupler reverts to the EtherCAT state PREOP-ERROR.

Through activation of the bus coupler parameter 'Hot-swap allowed', the module exchange during ongoing operations can be allowed. This parameter setting can be carried out either by way of the TB20 ToolBox (↗ 7.3.4) or by way of the object directory of the bus coupler (↗ 9.6.11).

ATTENTION

Carry out the pulling and plugging of a module quickly in order to keep the time periods of occurring malfunctions on the backplane bus as short as possible. Otherwise, the possibility can also not be excluded for a permitted module exchange that the bus coupler leaves the cyclical I/O data exchange and reverts to PREOP-ERROR.
Also be sure to never pull more than one one module simultaneously, as the bus coupler will in this case also leave the cyclical I/O data exchange and revert to PREOP-ERROR.

7.6.1 Behavior of the bus coupler during the module exchange

The behavior of the bus coupler described in the following presumes that the module exchange during ongoing operations has been allowed by the corresponding bus coupler parameters (↗ 7.5.2.1). 'During ongoing operations' means that the bus coupler is in the EtherCAT states SAFEOP or OP and a cyclical I/O data exchange with the master is taking place.

When a module is pulled, the bus coupler sets the 'Present' flag and the 'Operational' flag to 0 in the related module status (↗ 7.4.6) in order to indicate that the module is *not present* and *not operational*. Because the module status of each module is contained in the process input data of the bus coupler, this information is made available cycle-current to the PLC program. In addition to this, the input data, the channel status, and the diagnostic status are reset to zero, to the extent present, in the case of a pulled module. The absence of a module is indicated on the bus coupler by a yellow, flashing SF LED.

When the pulled module is replaced by plugging in a module of the same type, the bus coupler resets the 'Present' flag to the value 1 in the related module status. Prior to this, the bus coupler has switched the newly plugged in module to the app state STOP or RUN, depending upon whether it is found in the EtherCAT state SAFEOP or OP. If a configurable module is involved, then the bus coupler has already transmitted the the currently set parameter data to the module. To the extent that no module-wide diagnostic message is present in the newly plugged in module, the 'Operational' flag is also once again set to the value 1. In addition to this, current values for the input data, the channel status and the diagnostic status of the module are once again provided, to the extent present. When no incorrect module is plugged in at another point, the SF LED of the bus coupler will now extinguish again.

If an incorrect module is plugged in place of the pulled module, the module will continue to be reported to the PLC as *not present*. The bus coupler leaves the incorrectly plugged module in the app state IDLE and switches its yellow SF LED on permanently.

7.7 Diagnosis via the bus coupler and module LEDs

7.7.1 The LEDs of the TB20 EtherCAT® bus coupler

The *blue/red OK/BF LED* indicates the basic status of the bus coupler:

State of the OK/BF LED	Explanation
is off	The bus coupler does not have an adequate voltage supply or there may be a hardware defect. → Check the voltage supply of the bus coupler (↗ 4.3). Should the problem continue to exist, please contact support.
slowly flashes blue	The bus coupler is in the startup phase (INIT or PREOP) without the presence of a diagnostic message.
lights steady blue	The bus coupler is in cyclical operation (SAFEOP or OP) without the presence of a diagnostic message.
quickly flashes blue	The bus coupler is in simulation mode, triggered via the TB20 ToolBox (↗ 7.3.8).
slowly flashes red	There is no connection with a master or the master no longer/doesn't communicate(s) with the bus coupler. → Check the connection with the master. Also check the current state and the configuration of the master.
lights steady red	There is a diagnostic message for at least one module. → In the TB20 ToolBox you have the possibility via real time diagnosis (↗ 7.3.7) to have the diagnostic message displayed.
quickly flashes red	The bus coupler is in the firmware update mode. → Complete the firmware update by way of the TB20 ToolBox or restart a firmware update (↗ 7.3.10).

The *yellow SF LED* indicates an error on the backplane bus.

State of the SF LED	Explanation
is off	No error was detected on the backplane bus.
slowly flashes	*In INIT/PREOP:* No modules are plugged or at least one *configured module* is present. → Check your module structure for a match with the *active module configuration* (↗ 7.4.4.1). *In SAFEOP/OP:* A module has been pulled during ongoing operation (Hot Swap, ↗ 7.6). → Replace the pulled module with a module of the same type.
lights steady	At least one module does not match with the *active module configuration*. → Incorrectly plugged modules remain in the app state IDLE, so that you can easily recognize it by the quickly flashing blue OK or OK/SF LED.

The *green RUN LED* shows the current EtherCAT state of the bus coupler:

State of the RUN LED	Current EtherCAT state
is off	Init (INIT, ↗ 8.1.1)
blinking	Pre-Operational (PREOP, ↗ 8.1.3)
flashes periodically	Safe-Operational (SAFEOP, ↗ 8.1.4)
lights steady	Operational (OP, ↗ 8.1.5)

In combination with an error state displayed by way of the ERR LED, the EtherCAT states are designated with INIT-ERROR, PREOP-ERROR, and SAFEOP-ERROR (↗ 8.1.6).

The *red ERR LED* shows an error state of the EtherCAT state machine of the bus coupler:

State of the ERR LED	Explanations
is off	No error state is present.
blinking	A change to the EtherCAT state requested by the master was not possible because, for example, the Sync Manager settings are invalid or the *configured modules* do not match with the *plugged modules*. → Evaluate the AL status code (↗ 8.3) to receive an indication of the cause of the error and then check the configuration and the module structure to that effect.
flashes periodically	The bus coupler itself has triggered a change to a lower EtherCAT state. Possible causes for this might included the starting of the simulation mode via the TB20 ToolBox or the pulling of modules. → Evaluate the AL status code (↗ 8.3) to receive an indication of the cause for the autonomously carried out state transition.
flashes periodically 2x	The process output data watchdog (↗ 7.5.5) has expired and the bus coupler has changed independently from OP to SAFEOP-ERROR. The AL status code in this case contains the value 0x001B. → Check the wiring and the state of the master. Also check your cycle time setting for the process data communication with regard to the configured watchdog timeout.

The *green 'A L/A' LED* and the *green 'B L/A' LED* show the connection and activity status of the two EtherCAT ports 'X1/A IN' and 'X1/B OUT' (↗ 7.2):

State of the L/A LED	Explanations for the relevant EtherCAT port
is off	There is no Ethernet connection established.
lights steady	No communication is currently running over the established Ethernet connection.
blinking	Communication is currently taking place over the established Ethernet connection.

7.7.2 Module LEDs

The topmost LED (OK-LED) on every module indicates the current module status. In the case of modules without diagnostics capability, it is labeled with 'OK' and lights up blue. In the case of diagnostics-capable modules, it is called 'OK/SF' and can also display the presence of a diagnostic message with a red light.

State of the OK(/SF) LED	Explanations
is off	The module does not have an adequate voltage supply or there may be a hardware defect. →Check for correct installation (↗ Chapter 3) and voltage supply (↗ 4.3, 4.6) of the TB20 system. If the problem continues, replace the module when possible with a replacement module and contact Support when necessary.
quickly flashes blue	The module is in the app state IDLE (↗ 7.4.9). In the case of diagnostics-capable modules, there is no configuration error. If the bus coupler has an *active module configuration* (↗ 7.4.4.1), then only incorrectly plugged modules remain in the app state IDLE. → In this case ensure a comparison between *configured* and *plugged modules*.
slowly flashes blue	The module is in the app state STOP (↗ 7.4.9). In the case of diagnostics-capable modules, no configuration errors and no other diagnostic message are present.
lights steady blue	The module is in the app state RUN (↗ 7.4.9). In the case of diagnostics-capable modules, there is no configuration error and no other diagnostic message.
slowly flashes red	A diagnostic message due to a configuration error is present in the module. The module can thereby currently be in the app state IDLE, STOP, or RUN[1]. The diagnosis status (↗ 7.4.8) of the module provides information on which type of configuration error is involved and whether this involved the entire module or individual channels. → Check and correct the parameter settings for the module. To this purpose consult with the corresponding manual.

State of the OK(/SF) LED	Explanations
lights steady red	There is a diagnostic message in the module, as at least one module or channel-related problem has been diagnosed. The module can thereby currently be in the app state STOP or RUN[1]. The diagnostic status (↗ 7.4.8) of the module provides information about which module or channel-related problems are involved. → Evaluate the diagnostic status and possibly attempt to correct the problem with the help of the corresponding module manual.

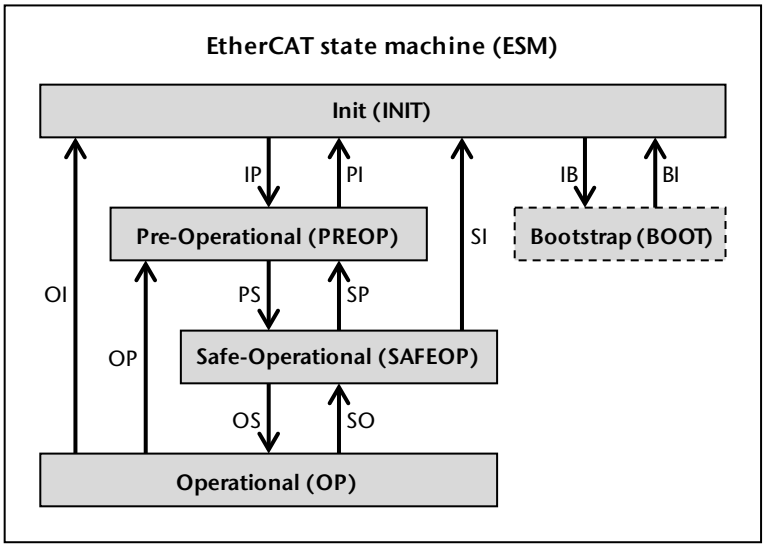
[1])If the OK/SF LED lights up red on the basis of a diagnostics message, reading the current app state on the basis of the LED is not possible at the moment. In this case, you can use the real time diagnosis (↗ 7.3.7) of the TB20 ToolBox to display the app state in the event of uncertainty.

All other present LEDs have module and application-specific meanings, about which you can inform yourself in the respective module manual.

8 Detailed information on the EtherCAT state machine

A general introduction to the EtherCAT state machine has already been provided in chapter 6.5. In this chapter, the behavior of the TB20 EtherCAT® bus coupler in connection with the various EtherCAT states and the possible transitions is now described in concrete terms. This description presumes that the differences between *detected, stored, projected*, and *configured modules* (↗ 7.4.1), as well as between *active, current*, and *stored module configurations* (↗ 7.4.4) are familiar.

For purposes of a better overview, here is the figure for the EtherCAT state machine familiar from the introductory chapter.



8.1 The EtherCAT states

8.1.1 Init (INIT)

The *INIT state* is the initial state after switching on the power. In this EtherCAT state, no mailbox and no process data communication is possible. The INIT state can be recognized through a switched off RUN LED.

The status of the modules depends upon whether a *stored module configuration* has been filed in the bus coupler or not:

Status of the modules without stored module configuration

Without *stored module configuration*, the *active module configuration* contains no modules in the INIT state. All plugged modules are therefore in the app state IDLE with a quickly flashing blue OK(/SF) LED.

Status of the modules with stored module configuration

If a *stored module configuration* is present, this is adopted as an *active module configuration*. Depending upon whether a plugged module matches with the *active module configuration* or not, this involves a correctly plugged or an incorrectly plugged module.

Correctly plugged modules are found in the app state STOP. Configurable modules have also received the module parameters from the *stored module configuration*. The OK(/SF) LED of the correctly plugged modules slowly flashes blue. Where a diagnostic message is present, the OK/SF LED flashes or lights up red in contrast (↗ 7.7.1).

In contrast, *incorrectly plugged modules* remain in the app state IDLE with quickly flashing blue OK(/SF) LED.

It is thus already possibly to optically recognize which modules have been correctly plugged and which haven't in the IMIT state on the basis of the OK(/SF) LEDs of the module, even when no master is present.

8.1.2 Bootstrap (BOOT)

The BOOT state is an optional EtherCAT state for updating the firmware of a slave. It is not supported by the TB20 EtherCAT® bus coupler, as firmware updates are carried out with the help of the TB20 ToolBox (↗ 7.3.10).

8.1.3 Pre-Operational (PREOP)

Mailbox communication is possible in the PREOP state, but not process data communication. The object directory of the bus coupler can now be accessed via the mailbox communication. This access is necessary, among other reasons, in order to be able to transfer the list of *projected modules* to the bus coupler and to be able to make and adjust settings on the bus coupler and module parameters. The PREOP state is indicated by a flashing green RUN LED.

The status of the modules depends upon the *active module configuration* adopted from the previous EtherCAT state.

All of the modules plugged correctly according to the *active module configuration* are in the app state STOP. Configurable modules have received the module parameters filed in the *active module configuration*. The OK(/SF) LED of the correctly plugged modules slowly flashes blue. Where a diagnostic message is present, the OK/SF LED lights up red in contrast (↗ 7.7.1).

Incorrectly plugged modules are found in the app state IDLE. They can be recognized by a quickly flashing blue OK(/SF) LED.

8.1.4 Safe-Operational (SAFEOP)

As of the SAFEOP state, process data communication is now also possible with the master in addition to mailbox communication. The green RUN LED of the bus coupler indicates this EtherCAT state with periodic flashing.

In the case of process data communication, a cyclical exchange of the process output and process input data of the bus coupler with the master takes place.

The *process output data* sent by the master to the bus coupler contains the output data for the *configured modules*. Because the correctly plugged modules are in the app state STOP for SAFEOP, the provided output data is not accepted at all. The physical outputs of the modules instead remain in a secure state that can also be configurable depending upon the module.

The *process input data* sent by the bus coupler to the master contains the current input data of the *configured modules*. The process input data also contains a module status (↗ 7.4.6) for each module and possibly also a channel status (↗ 7.4.7).

The object directory of the bus coupler can still be accessed via mailbox communication. This makes it possible, for example, to read out the diagnostic status of a module via the object directory as needed or to change the parameter settings of a module.

In the SAFEOP state, all plugged modules normally correspond to the *active module configuration*. They have received their parameter data and are in the app state STOP. The OK(/SF) LED of these modules slowly flashes blue. Where a diagnostic message is present, the OK/SF LED flashes or lights up red in contrast (↗ 7.7.1).

Only modules that have been incorrectly plugged in the context of a module exchange (↗ 7.6) are in the app state IDLE. The OK(/SF) LED of these incorrectly plugged modules quickly flashes blue.

8.1.5 Operational (OP)

In the OP state, the cyclical process data communication with the master is continued in the SAFEOP state. The decisive difference, however, is that the correctly plugged modules are now in the app state RUN and accept the output data transmitted by the master. Access to the object directory is also still possible in the OP state via mailbox communication. The RUN LED of the bus coupler lights up steady green.

All plugged modules normally correspond to the *active module configuration* in the OP state and are in the app state RUN. The OK(/SF) LED of these modules flashes steady blue. Where a diagnostic message is present, the OK/SF LED flashes or lights up red in contrast (↗ 7.7.1).

Only modules that have been incorrectly plugged in the context of a module exchange (↗ 7.6) are in the app state IDLE. The OK(/SF) LED of these incorrectly plugged modules quickly flashes blue.

8.1.6 INIT-ERROR, PREOP-ERROR, and SAFEOP-ERROR

The addition 'ERROR' with the EtherCAT states INIT, PREOP, and SAFEOP should make clear that the EtherCAT state machine of the bus coupler is currently in an error state. In optical terms, this error state can be recognized on the basis of the ERR-LED (↗ 7.7.1), which presents a particular lighting behavior depending upon the cause of the error. In the direction of the master, the error state is indicated with a set error flag in the AL status directory (↗ 6.5.2.2). The error code contained in the AL status code directory (↗ 6.5.2.3) provides information on the cause of the error. A list of these AL status codes is found in chapter 8.3.

An error state of the EtherCAT state machine is on the one hand always evoked when the change to a higher EtherCAT state requested by the master was not possible. Causes of this may include, for example, invalid Sync Manager settings or deviations between c*onfigured* and *plugged modules*. The bus coupler in this case remains in the current EtherCAT state and the ERR LED flashes red.

On the other hand, it can happen that a problem may make it necessary for the bus coupler itself to make a change toward a lower EtherCAT state. For example, the bus coupler will automatically change from OP to SAFEOP-ERROR in the secure operating state when the process data watchdog has expired. In this error case, the red ERR-LED of the bus coupler periodically flashes twice in brief succession.

Other causes for an automatic change to a lower EtherCAT state might be the pulling of modules of the start of simulation mode via the TB20 ToolBox. In such cases, the ERR-LED of the bus coupler flashes red periodically.

In order to be able to once again leave the error state and potentially change to another EtherCAT state, a corresponding AL control request from the master is required. A particular flag for the confirmation and switching off of the error must be set in it, as the AL control request will otherwise be ignored. Deviating from this, a change to the INIT state can always be successfully requested.

8.2 The EtherCAT state transitions

All possible state transitions of the EtherCAT state machine of the bus coupler are described in the following. The meaning of the 'ERROR' addition with the state designations has been previously explained in chapter 8.1.6. AL status and AL status code respectively refer in abbreviated form to the content of the AL status directory (↗ 6.5.2.2) or to the AL status code directory (↗ 6.5.2.3).

8.2.1 Reset→INIT

After switching on the voltage supply, a change is automatically made from a reset state to the INIT state. If a *stored module configuration* is present in the bus coupler, this is adopted as an *active module configuration*. The *active module configuration* otherwise contains no modules and no parameter data.

Depending upon the *active module configuration*, plugged modules that match are switched to the app state STOP. If these are configurable modules, these are also supplied with the parameter data originating from the *stored module configuration*. All other modules remain in the app state IDLE.

The state transition is concluded with AL status = INIT and AL status code = 0x0000 (no error).

8.2.2 INIT-ERROR→INIT

In the case of this state transition, only the current error state is switched off by a corresponding AL control request from the master. The state transition ends with *AL status = INIT* and *AL status code = 0x0000* (no error).

8.2.3 INIT(-ERROR)→BOOT

This state transition is initiated by a corresponding AL control request from the master. Because the bus coupler does not support the optional BOOT state, the state transition is canceled with *AL status = INIT-ERROR* and *AL status code = 0x0013* (EtherCAT state bootstrap is not supported).

8.2.4 INIT(-ERROR)→PREOP

This state transition is initiated by a corresponding AL control request from the master. In the process, the bus coupler may neither be in simulation mode (↗ 7.3.8) nor be currently receiving a new configuration (↗ 7.3.5) by way of the TB20 ToolBox. The state transition is otherwise canceled with *AL status = INIT-ERROR* and *AL status code = 0x8000* (no EtherCAT operation possible).

When the precondition has been fulfilled, the configuration of the Sync Managers 0 and 1 is then checked, which are required for the mailbox communication. The master previously carried out the configuration of the two Sync Managers via the corresponding ESC directory.

In the event of a valid Sync Manager configuration, the mailbox communication is activated. Access to the object directory (↗ Chapter 9) of the bus coupler is then possible through this as of PREOP. The state transition is successfully concluded with *AL status = PREOP* and *AL status code = 0x0000* (no error).

If the Sync Manager configuration is invalid, then the state transition is canceled with *AL status = INIT-ERROR* and *AL status code = 0x0016* (invalid mailbox configuration of the Sync Manager 0 & 1).

8.2.5 PREOP(-ERROR)→INIT

This state transition is initiated by a corresponding AL control request from the master. For one, the mailbox communication is once again deactivated in the process. For another, the *active module configuration* is reset or, when present, overwritten with the *stored module configuration*. A reset *active module configuration* contains no modules and no parameter data.

Depending upon the now *active module configuration*, plugged modules that match are switched to the app state STOP. If these are configurable modules, these are also supplied with the parameter data originating from the *stored module configuration*. All other modules are switched to the app state IDLE.

The state transition is concluded with *AL status = INIT* and *AL status code = 0x0000* (no error).

8.2.6 PREOP(-ERROR)→INIT-ERROR

This state transition is initiated by the bus coupler when either the upload of a configuration (↗ 7.3.5) or the simulation mode (↗ 7.3.8) is started via the TB20 ToolBox. The mailbox communication is deactivated. The state transition is subsequently ended with *AL status = INIT-ERROR* and *AL status code = 0x8000* (no EtherCAT operation possible).

8.2.7 PREOP-ERROR→PREOP

In the case, of this state transition, only the current error state is switched off by a corresponding AL control request from the master. The state transition ends with *AL status = PREOP* and *AL status code = 0x0000* (no error).

8.2.8 PREOP(-ERROR)→SAFEOP

This state transition is initiated by a corresponding AL control request from the master.

If the *current module configuration* is valid because it contains at least one *configured module* and no module gaps, then it will be accepted as an *active module configuration*. As a consequence of this, the plugged modules now matching are switched to the app state STOP and receive, when configurable, the currently configured parameter data. All other modules are switched to the app state IDLE.

If the *current module configuration* is invalid, it has no influence on the *active module configuration* and the current app state of the modules.

As of SAFEOP, the bus coupler is in cyclical process data communication with the master. In order that this communication functions as expected, an entire series of checks take place in advance, which are described more precisely in the following. A change after SAFEOP is only possible when each of the checks described in the following could be carried out successfully.

Check of the plugged/detected modules:

All of the modules plugged into the backplane bus are normally detected by the bus coupler and administered in the object 0xF050 (Detected Module Ident List, ↗ 9.6.9). On the basis of this list, a check is made to determine whether at least one module is plugged into the backplane bus and that no module gaps exist. If this check is unsuccessful, the state transition will then be canceled with *AL status = PREOP-ERROR* and *AL status code = 0x8100* (No modules or module gaps detected on the backplane bus).

Checks for the presence of a stored configuration:

If a *stored module configuration* is stored in the bus coupler, then it is checked to determine whether the modules contained within it match with the *projected modules*. If no *projected modules* have been transmitted by the master, this check instead takes place with the *detected modules*.

If there is no match between the *stored* and the *projected* or *detected modules*, then the state transition is canceled with *AL status = PREOP-ERROR* and *AL status code = 0x8122* (No match between *projected* and *stored modules*) or *AL status code = 0x8110* (No match between *stored* and *detected modules*).

Checks for the projected modules:

When not explicitly deactivated by way of a particular bus coupler parameter (↗ 7.5.2.2), the master must transmit the list of the *projected modules* to the bus coupler master through description of the object 0xF030 (Configured Module Ident List, ↗ 9.6.13). If this does not occur, the state transition will then be canceled with *AL status = PREOP-ERROR* and *AL status code = 0x8123* (Master hasn't transmitted list of projected modules).

If the master has transmitted a list with *projected modules* to the bus coupler, a check then takes place, irrespective of the previously mentioned bus coupler parameter, to determine whether this list is valid and whether it matches with the *detected, meaning plugged modules*.

If the transmitted list of of the *projected modules* is invalid, either because it is empty or because it contains module gaps, then the state transition is canceled with *AL status = PREOP-ERROR* and *AL status code = 0x8120* (List of *projected modules* transmitted by the master is invalid).

Should there be no match between the transmitted *projected* and the *detected modules*, then the state transition is canceled with *AL status = PREOP-ERROR* and *AL status Code = 0x8121* (No match between *projected* and *detected modules*).

Checking the configuration of Sync Managers 2 and 3:

Sync Managers 2 and 3 are necessary for the cyclical process data communication between bus coupler and master. In concrete terms, they are responsible for the consistent exchange of the process output data (Sync Manager 2) and the process input data (Sync Manager 3) by way of the EtherCAT slave controller.

The master has previously undertaken the configuration of the two Sync Managers by way of corresponding directories of the EtherCAT slave controller. Of special relevance for checking are the sizes of the process output and process input data set by the master, which must match with the process data sizes expected by the bus coupler. If this match does not prevail, this is generally because master and bus coupler presume different module configurations. In this case, the state transition is then canceled with *AL status = PREOP-ERROR* and *AL status code = 0x001D* (Invalid process output data configuration for Sync Manager 2) or *AL status code = 0x001E* (Invalid process input data configuration for Sync Manager 3).

The master should have made all other required settings on the Sync Managers 2 and 3 independently of the project on the basis of the specifications from the ESI file of the bus coupler. It is therefore somewhat improbable that they are the cause for an invalid Sync Manager configuration and the cancellation of the state transition.

Successful completion of the checks:

After all checks have been successfully completed, the bus coupler prepares itself for the start of cyclical process data communication. This also includes the fact that the bus coupler starts a local cyclical exchange of module input and module output data with the modules found on the backplane bus

The state transition is then successfully concluded with *AL status = SAFEOP* and *AL status code = 0x0000* (no error).

8.2.9 SAFEOP(-ERROR)→INIT

This state transition is initiated by a corresponding AL control request from the master. In the process, for one thing, the process data and the mailbox communication are deactivated. For another, the *active module configuration* is reset or, when present, overwritten with the *stored module configuration*. A reset *active module configuration* contains no modules and no parameter data.

Depending upon the now *active module configuration*, plugged modules that match are switched to the app state STOP. If these are configurable modules, these are also supplied with the parameter data originating from the *stored module configuration*. All other modules are switched to the app state IDLE.

The state transition is concluded with *AL status = INIT* and *AL status code = 0x0000* (no error).

8.2.10 SAFEOP(-ERROR)→INIT-ERROR

This state transition is initiated by the bus coupler when either the upload of a configuration (↗ 7.3.5) or the simulation mode (↗ 7.3.8) is started via the TB20 ToolBox. The process data and the mailbox communication are deactivated. The state transition is subsequently ended with *AL status = INIT-ERROR* and *AL status code = 0x8000* (no EtherCAT operation possible).

8.2.11 SAFEOP(-ERROR)→PREOP

This state transition is initiated by a corresponding AL control request from the master. The process data communication with the master is deactivated and the local cyclical exchange of input and output data between bus coupler and modules is ended. The state transition is subsequently ended with *AL status = PREOP* and *AL status code = 0x0000* (no error).

8.2.12 SAFEOP(-ERROR)→PREOP-ERROR

This state transition is initiated by the bus coupler due to a communication problem on the backplane bus or due to a module pulled without permission. The process data communication with the master is deactivated and the local cyclical exchange of input and output data between bus coupler and modules is ended. The state transition is subsequently ended with *AL status = PREOP-ERROR* and *AL status code = 0x8200* (Communication problem on the backplane bus) or *AL status code = 0x8201* (Unauthorized pulling of a module).

8.2.13 SAFEOP-ERROR→SAFEOP

In the case of this state transition, only the current error state is switched off by a corresponding AL control request from the master. The state transition ends with *AL status = SAFEOP* and *AL status code = 0x0000* (no error).

8.2.14 SAFEOP(-ERROR)→OP

This state transition is initiated by a corresponding AL control request from the master. All correctly plugged modules are switched to the app state RUN, so that the output data provided by the master is now used in the modules. The state transition is ended with *AL status = OP* and *AL status code = 0x0000* (no error).

8.2.15 OP→INIT

This state transition is initiated by a corresponding AL control request from the master. The process data communication with the master is deactivated and the local cyclical exchange of input and output data between bus coupler and modules is ended. The mailbox communication also ends, so that the access to the object directory of the bus coupler is no longer possible.

The *active module configuration* is still reset or, when present, overwritten with the *stored module configuration*. A reset *active module configuration* contains no modules and no parameter data.

Depending upon the now *active module configuration*, plugged modules that match are switched to the app state STOP. If these are configurable modules, these are also supplied with the parameter data originating from the *stored module configuration*. All other modules are switched to the app state IDLE.

The state transition is concluded with *AL status = INIT* and *AL status code = 0x0000* (no error).

8.2.16 OP→INIT-ERROR

This state transition is initiated by the bus coupler when either the upload of a configuration (↗ 7.3.5) or the simulation mode (↗ 7.3.8) is started via the TB20 ToolBox. The process data and the mailbox communication are deactivated. The state transition is subsequently ended with *AL status = INIT-ERROR* and *AL status code = 0x8000* (no EtherCAT operation possible).

8.2.17 OP→PREOP

This state transition is initiated by a corresponding AL control request from the master. The process data communication with the master is deactivated and the local cyclical exchange of input and

output data between bus coupler and modules is ended. All correctly plugged modules are also switched back to the app state STOP. The state transition is subsequently ended with *AL status = PREOP* and *AL status code = 0x0000* (no error).

8.2.18 OP→PREOP-ERROR

This state transition is initiated by the bus coupler due to a communication problem on the backplane bus or due to a module pulled without permission. The process data communication with the master is deactivated and the local cyclical exchange of input and output data between bus coupler and modules is ended. All correctly plugged modules are also switched back to the app state STOP. The state transition is subsequently ended with *AL status = PREOP-ERROR* and *AL status code = 0x8200* (Communication problem on the backplane bus) or *AL status code = 0x8201* (Non-authorized pulling of a module).

8.2.19 OP→SAFEOP

This state transition is initiated by a corresponding AL control request from the master. All correctly plugged modules are switched back to the app state STOP. The state transition is subsequently ended with *AL status = SAFEOP* and *AL status code = 0x0000* (no error).

8.2.20 OP→SAFEOP-ERROR

This state transition is initiated by the bus coupler because the process data watchdog has expired. All correctly plugged modules are switched back to the app state STOP. The state transition is subsequently ended with *AL status = SAFEOP-ERROR* and *AL status code = 0x001B* (Process data watchdog expired).

8.2.21 Invalid EtherCAT state transition

An invalid state transition is initiated when an EtherCAT state is requested by the master in the AL control request (↗ 6.5.2.1) into which it is not possible to change from the current EtherCAT state. Examples of this are INIT→SAFEOP, INIT→OP, and PREOP→OP.

If the state transition was requested from OP, then the bus coupler changes to SAFEOP-ERROR. The correctly plugged modules are thereby switched back to the app state STOP. The changed state transition is ended with *AL status = SAFEOP-ERROR* and *AL status code = 0x0011* (Invalid EtherCAT state transition).

If the state transition took place from a different EtherCAT state, then the current EtherCAT state will be retained as usual and the error flag set in the AL status. The state transition is canceled with *AL status code = 0x0011* (Invalid EtherCAT state transition).

8.2.22 Transition to unknown EtherCAT state

A transition to an unknown EtherCAT state is initiated when the AL control request from the master (↗ 6.5.2.1) contains a value that could not be assigned to any known EtherCAT state.

If the state transition was requested from OP, then the bus coupler changes to SAFEOP-ERROR. The correctly plugged modules are thereby switched back to the app state STOP. The changed state transition is ended with *AL status = SAFEOP-ERROR* and *AL status code = 0x0012* (Unknown EtherCAT state).

If the state transition took place from a different EtherCAT state, then the current EtherCAT state will be retained as usual and the error flag set in the AL status. The state transition is canceled with *AL status code = 0x0012* (Unknown EtherCAT state).

8.3 List of AL status codes

All AL status codes that can occur for the EtherCAT state machine of the TB20 EtherCAT® bus coupler are listed in the following and described briefly. As of 0x8000, these are manufacturer or product-specific AL status codes.

AL status code	Description
0x0000	No error
0x0001	Non-specific error An error has occurred that cannot be precisely defined.
0x0011	Invalid EtherCAT state transition A change to the EtherCAT state requested by the master is not possible from the current EtherCAT state (↗ 8.2.21).
0x0012	Unknown EtherCAT state The EtherCAT state requested by the master is unknown (↗ 8.2.22).
0x0013	EtherCAT state bootstrap is not supported The optional EtherCAT state bootstrap (BOOT) is not supported (↗ 8.1.2).
0x0016	Invalid mailbox configuration (Sync Managers 0 & 1) A transition from INIT to PREOP was not possible because the mailbox communication undertaken by the master from Sync Managers 0 and 1 is invalid (↗ 8.2.4).
0x001B	Process data watchdog expired The bus coupler has automatically carried out a change from OP to SAFEOP-ERROR, because the process data watchdog has expired (↗ 8.2.20).
0x001D	Invalid process output data configuration (Sync Manager 2) A transition from PREOP to SAFEOP was not possible because the process output data configuration undertaken by the master from Sync Manager 2 is invalid (↗ 8.2.8).
0x001E	Invalid process input data configuration (Sync Manager 3) A transition from PREOP to SAFEOP was not possible because the process input data configuration undertaken by the master from Sync Manager 3 is invalid (↗ 8.2.8).
0x001F	Invalid configuration of the process data watchdog A transition from PREOP to SAFEOP was not possible because the configuration of the process data watchdog undertaken by the master is invalid.
0x0028	Requested sync mode is not supported A transition from PREOP to SAFEOP was not possible because the configuration of the sync mode undertaken by the master is invalid. The TB20 EtherCAT® bus coupler only supports a process data exchange in the free run mode.
0x0029	Invalid buffer configuration for Sync Managers 2 and 3 A transition from PREOP to SAFEOP was not possible because the buffer configuration for sync managers 2 and/or 3 is invalid. In the free run mode supported by the bus coupler, the two sync managers must be configured in the 3 buffer mode.
0x8000	EtherCAT operation not possible The bus coupler has changed automatically to the INIT-ERROR state or doesn't allow for leaving the INIT state, as EtherCAT operation is not currently possible. This is always the case when the bus coupler is in simulation mode or when a new configuration is uploaded via the TB20 ToolBox.
0x8100	No modules or module gaps detected in the backplane bus A transition from PREOP to SAFEOP was not possible, because no modules or a module gap were detected on the backplane bus (↗ 8.2.8).
0x8110	No match between stored and detected modules A transition from PREOP to SAFEOP was not possible because the modules of an existing *stored module configuration* do not match with the *modules detected/plugged in* at the backplane bus (↗ 8.2.8).
0x8120	List of projected modules transmitted by the master is invalid A transition from PREOP to SAFEOP was not possible because the list of the *projected modules* transmitted by the master is either empty or contains module gaps (↗ 8.2.8).
0x8121	No match between projected and detected modules A transition from PREOP to SAFEOP was not possible because the list of *projected modules* transmitted by the master does not match with the *modules detected/plugged in* at the backplane bus (↗ 8.2.8).

0x8122	No match between projected and stored modules A transition from PREOP to SAFEOP was not possible because the list of *projected modules* transmitted by the master does not match with the modules of an existing *stored module configuration* (↗ 8.2.8).
0x8123	Master has not transmitted the list of projected modules A transition from PREOP to SAFEOP was not possible because the master has not transmitted a list with *projected modules*, but this is required according to bus coupler parameters (↗ 7.5.2.1)(↗ 8.2.8).
0x8200	Communication problem at the backplane bus The bus coupler has reverted from SAFEOP/OP to PREOP-ERROR because a communication problem with the modules has occurred at the backplane bus.
0x8201	Unauthorized pulling of a module The bus coupler has reverted from SAFEOP/OP to PREOP-ERROR because a module has been pulled at the backplane bus without authorization. When hot swap is permitted by the bus coupler configuration (↗ 7.5.2.1), then a maximum of one and otherwise no modules may be pulled.

9 Detailed information on the object directory

9.1 General

Objects for the identification, configuration and diagnosis of the bus coupler and its modules are provided in the object directory of the TB20 EtherCAT® bus coupler. The object directory and access to it are based on the CANopen protocol, which was originally developed for CAN devices (CAN = Controller Area Network). The adaption for EtherCAT is therefore referred to as "CANopen over EtherCAT" (CoE).

Each object contained in the object directory is assigned a clear index for addressing. The index is a 16 bit value that is usually indicated in a four-digit, hexadecimal notation (e.g. 0xF030). Each object has a designation, a data type and particular flags. With the flags, the possibilities for accessing the object are defined, among other things, meaning whether the object can be read and/or described, and in which EtherCAT states (PREOP, SAFEOP, OP) this is permitted.

In the following, a differentiation is made between bus coupler and module objects, depending upon whether the information contained within them refers to the bus coupler itself or to the modules.

9.2 Modular Device Profile (MDP)

With the TB20 EtherCAT® bus coupler, this is a modular EtherCAT slave, which is why the Modular Device Profile (MDP) is used with regard to the object directory. For the MDP, the profile ID 5001 is specified, which is in this case indicated in the object 0x1000 (Device Type, ↗ 9.6.1).

The Modular Device Profile defines the following, among other things:

- how the profile-dependent range of the object directory is structured
- which information certain objects contain, and whether these must be made available or are optional
- in which way the module objects (↗ 9.7) of the *configured modules* are shown in the object directory

9.3 Ranges of the object directory

The following table provides an overview of the various ranges of the object directory and how these are used with the TB20 EtherCAT bus coupler.

Index range	Description	Use with the bus coupler
0x0000 – 0x0FFF	**Range for data type definitions**	Not in use
0x1000 – 0x1FFF	**Profile-independent range**	Contains profile-independent bus coupler objects
0x1XXX	Various information	incl. identification and characteristics of the bus coupler
0x1400 – 0x15FF	RxPDO configuration	Not in use
0x1600 – 0x17FF	RxPDO configuration	Illustration of the module output data on RxPDOs
0x1800 – 0x19FF	TxPDO configuration	Not in use
0x1A00 – 0x1BFF	TxPDO configuration	Illustration of the module input data and of the module and channel status on TxPDOs
0x1C10 – 0x1C2F	PDO allocation	Allocation of the Rx and TxPDOs to the Sync Managers
0x2000 – 0x5FFF	**Manufacturer-specific range**	Contains proprietary bus coupler and module objects
0x2100 – 0x2130	Module configuration	Status and control objects for module configuration
0x3000 – 0x3400	Parameter data	Configuration of the bus coupler and the modules

Index range	Description	Use with the bus coupler
0x6000 – 0xEFFF	**MDP-specific module range**	Contains proprietary bus coupler and module objects
0x6000 – 0x6FFF	Input data range	Definition of the input data of the modules
0x7000 – 0x7FFF	Output data range	Definition of the output data of the modules
0x8000 – 0x8FFF	Configuration range	Not in use
0x9000 – 0x9FFF	Information range	Not in use
0xA000 – 0xAFFF	Diagnostics range	Diagnostics information on the modules
0xB000 – 0xBFFF	Service transfer range	Not in use
0xF000 – 0xFFFF	MDP-specific bus coupler range	Contains MDP-specific bus coupler objects - MDP characteristics of the bus coupler - List of the *modules detected by the bus coupler* - List of the *modules configured in the bus coupler*

9.4 Data types of the objects

Each object represents its data in a defined data type. This is either a so-called base data type or the structured data types ARRAY and RECORD.

9.4.1 Base data types

The following table lists the base data types used by the bus coupler.

Base data type	Bit size	Value range	Description
BOOL	1	0 (FALSE) and 1 (TRUE)	A Boolean data type encompassing one bit
BYTE	8	0 to 255	Unsigned integer data type (byte) used for bit fields
SINT	8	-128 to 127	Signed integer data type (short integer)
USINT	8	0 to 255	Unsigned integer data type (unsigned short integer)
INT	16	-32768 to 32767	Signed integer data type (integer)
UINT	16	0 to 65535	Unsigned integer data type (unsigned integer)
DINT	32	-2^{31} to 2^{31} -1	Signed integer data type (double integer)
UDINT	32	0 to 2^{32} -1	Unsigned integer data type (unsigned double integer)
STRING(n)	8·n	8 bits per character	Character string with n characters à 8 bits (without zero termination)

9.4.2 Structured data types ARRAY and RECORD

In the case of objects of data type ARRAY or RECORD, the data contained is distributed to several object entries. The addressing of the individual object entries takes place with the indication of a subindex. The subindex is an 8-bit value in the range from 0 to 255, so that a maximum of 256 different object entries can be addressed. The term subindex is often also used as a synonym for the object entry addressed with it (e.g. 'Subindex 3 contains the value X' instead of 'the object entry with subindex 3 contains the value X').

The following notation from the object index and subindex are used as a reference to a particular object entry: <hexadecimal object index>:<decimal subindex>

e.g. 0xF030:12 (Subindex 12 of object 0xF030)

Subindex 0

Subindex 0 is a special subindex that, in the case of ARRAY and RECORD objects, contains the number of the following available subindices (0 to maximum 255). The subindex 0 always has the base data type USINT (8 bits) and bears the designation "SubIndex 000". Through corresponding access flags, an indication is made for subindex 0 of whether this can only be read or can also be written in, and in which EtherCAT state this is respectively possible.

Subindex 1 to 255 of an ARRAY object

In the case of an ARRAY object, the same base data type and the same access settings are defined for all subindices from 1 to a maximum of 255. In addition to this, a uniform designation following the system of "SubIndex *nnn*" is used for all these subindices, whereby *nnn* stands for the three-decimal place subindex.

The objects 0xF030 (Configured Module Ident List, ↗ 9.6.13) and 0xF050 (Detected Module Ident List, ↗ 9.6.14) are listed as examples of ARRAY objects here.

Subindices 1 to 255 of a RECORD object

In contrast with the ARRAY objects, the base data type and the access settings for each of the subindices 1 to a maximum of 255 are individually defined for the RECORD objects. Individual designations can also be used for each of the subindices.

The object 0x2100 (Module Configuration Status, ↗ 9.6.7) and the parameter data objects of the modules (↗ 9.7.3) are named here as examples of RECORD objects.

9.5 Access to the object directory

Access to the object directory takes place from the master by way of the following CoE services:

- SDO upload for the reading of object data
- SDO download for the writing of object data
- SDO info for reading out the index lists of the objects available online, as well as further information on the individual objects (incl. designation, data type, flags)

SDO is the abbreviation for "Service Data Object" and designates the data that is exchanged with access to the object directory by way of the CoE services. The CoE service uses the mailbox service, which is available as of the EtherCAT state PREOP, for carrying out SDO communication. In the EtherCAT state INIT it is therefore not yet possible to access the object directory.

The data of the object directory that should be accessible for reading or writing is indicated by the index and, in the case of the ARRAY and RECORD objects, also by the subindex. The transmission of the SDO data takes place byte-oriented and in the Little Endian format:

- An entire byte is also transmitted in the case of individual access to subindices of the data type BOOL with data encompassing only one bit. The relevant information is in this case contained in the bit (bit 0) with the lowest value of the transmitted byte.
- When accessing subindices of the data type INT, DINT, UINT and UDINT, the relevant numerical values are transmitted in the Little Endian Format, thus beginning with the byte of the lowest value.

Access to ARRAY and RECORD objects can take place either subindex-wise or via complete access.

9.5.1 Access via complete access

In the event of access via complete access, all subindices of an ARRAY or RECORD object can be accessed with reading or writing rights with only one SDO order. In addition to a speed advantage, this primarily represents easing of corresponding access to objects from a PLC program, where otherwise all subindices would have to be successively accessed individually as needed.

The access via complete access is not supported by all, but by most ARRAY and RECORD objects of the bus coupler. To this purpose, a corresponding entry is present for the description of the individual objects respectively.

Only the following values can be used as a subindex with complete access:

- 0 = access begins at subindex 0 (subindex 0 is thus also included)
- 1 = access begins at subindex 1 (subindex 0 is thus not included)

If a subindex other than 0 or 1 is used, then the SDO order is canceled with the SDO abort code 0x06010000 (access method not supported).

In the event of access via complete access, the object data to be read or written is composed of the individual data of the subindices. The bit offset indicated with the description of the ARRAY and RECORD objects for each subindex shows the bit position of the respective subindex data within the object data. The indication of the bit offset thereby refers to the object data with complete access including subindex 0. The data from subindex 1 in this case always starts at bit offset 16, as an additional full byte is inserted behind the data of subindex 0 as a standard. In the case of complete access excluding subindex 0, the object data begins with the data from subindex 1. The bit offset entries of the subindices must then be reduced by the value 16 in order to receive the respective bit position within the object data.

If, for example, a parameter data object is accessed via complete access, then the object data contained as of subindex 1 corresponds precisely to the byte-wise structure of the parameter data described in the respective module manual.

9.5.2 Access from an EtherCAT configuration tool

It is usually possible in the EtherCAT configuration tools to access the object directory of the bus coupler. In the process, it is for the most part possible to differentiate between an offline and an online variant of the object directory.

9.5.2.1 Offline object directory

The offline object directory is generated from the EtherCAT configuration tool from the information contained in the ESI file (↗ 6.6.1). That means that the ESI file of the bus coupler must be installed to display the offline object directory. Insofar as no online values are to be displayed in the offline object directory, the bus coupler need not be accessible via EtherCAT.

All module objects (↗ 9.7), for example, the configuration data objects of the modules, are dynamically shown in the offline object directory by the EtherCAT configuration tool, depending upon the currently *projected modules*.

In the case of the offline object directory, it is generally possible to switch back and forth between the display of offline and online values for the contained objects:

For the display of *offline values*, values can actually only be displayed for such objects and object entries for which the corresponding default values have been stored in the ESI file. This is currently only the case for the parameter data objects of the modules, because it is primarily advantageous for an offline configuration of the bus coupler. No offline values are therefore shown for all other objects.

The display of *online values* can only be switched to when the corresponding bus coupler is accessible via EtherCAT and is at least in the EtherCAT state PREOP. For module objects, the display of online

values also only functions reliably when the *projected modules* in the EtherCAT configuration tool match with the modules *configured* and *detected* in the bus coupler, because the module objects shown in the offline object directory are only then actually provided online by the bus coupler.

For the display of online values in the offline object directory, it is generally also possible to describe objects or object entries, insofar as these allow for the respective access settings of the object in the current EtherCAT state of the bus coupler.

9.5.2.2 Online object directory

For the display of the online object directory, it is necessary that the corresponding bus coupler be accessible via EtherCAT and be at least in the EtherCAT state PREOP. All of the information required for the representation of the online object directory is read out of the bus coupler by way of the CoE service 'SDO Info.' The respective online values for the objects are normally also automatically called up by the bus coupler and displayed.

The module objects contained in the online object directory are based on the modules *configured* in the bus coupler. In order that the bus coupler can also actually provide the module objects of a particular *configured module* online, the corresponding module must be plugged into the backplane bus. This is necessary because the bus coupler itself does not possess any detailed knowledge about the individual modules and must read all information required from the modules. Deviating from this, parameter data objects can also then be provided when a *stored module configuration* is present in the bus coupler and the *configured module* matches with the corresponding *stored module*.

9.5.3 Access from a PLC program

Access from a PLC program takes place with the help of certain functional components, through which the object directory of the bus coupler can be accessed with reading and writing rights. The prerequisite for this is that the bus coupler be connected to the master via EtherCAT and be at least in the EtherCAT state PREOP. The required functional components are usually provided by the manufacturer of the programmable logic controller (PLC) or of the EtherCAT master in the form of a function library.

The terms 'CoE' (CANopen over EtherCAT) and/or 'SDO' (Service Data Object) are generally found in the names of the functional components in connection with 'Read' and 'Write'. In the case of ARRAY or RECORD objects it can normally be indicated whether the access should refer only to a subindex or to the entire object via complete access.

When the functional components are called up, a corresponding SDO communication then takes place in the background between the master and the bus coupler, for the processing of which several task cycles are generally required. In the case of failed access, a corresponding SDO abort code (↗ 9.5.4) is returned as an error code, which provides an indication of the cause of the error.

Reference is made to the respective documentation of the provider of the functional components for concrete details on the functional components and their use.

Relevant application cases for this access path to the object directory are:

- the reading out of the detailed diagnostic status object (↗ 9.7.8) of a module after a problem has been displayed by way of the module or channel status provided for the current cycle
- the reconfiguration of modules during ongoing operation by way of the corresponding parameter data objects (↗ 9.7.3)

9.5.4 SDO abort codes

The following listed SDO abort codes are used by the TB20 EtherCAT® bus coupler for failed SDO orders.

SDO abort code	Description
0x05030000	new SDO inquiry without changed toggle bit
0x05040001	SDO inquiry with unknown 'Command Specifier'
0x06010000	not supported access method
0x06010002	not permitted writing access to an only readable object/an only readable object entry
0x06010003	change to the object entry not possible, as subindex 0 wasn't previously set to the value 0
0x06010004	access via complete access is not supported for this object
0x06020000	relevant object does not exist
0x06040043	object entry cannot be described with transmitted value
0x06070010	length of the data not compatible with the object/object entry
0x06090011	relevant object entry does not exist
0x06090030	value to be written is too large for the object entry
0x08000000	general, unspecific error
0x08000022	access not possible in the current EtherCAT state

9.6 The bus coupler objects

The following describes the bus coupler objects statically present in the object directory. They serve the purposes of identification and configuration of the bus coupler. The bus coupler objects for PDO configuration are described separately in chapter 9.8.

9.6.1 Object 0x1000 - Device Type

Index 0x1000	Device Type
Short description	Type ID of the device
Data type (bit size)	UDINT (32)
Access	only reading
Online value	0x00001389 (5001) = device uses the Modular Device Profile (MDP, ↗ 9.2)

9.6.2 Object 0x1008 - Device Name

Index 0x1008	Device Name
Short description	Name of the device
Data type (bit size)	STRING (variable)
Access	only reading
Online value	"TB20 Coupler EtherCAT"

9.6.3 Object 0x1009 - Hardware Version

Index 0x1009	Hardware Version
Short description	Hardware version of the device
Data type (bit size)	STRING (variable)
Access	only reading
Online value	Example: "HW1-1"

9.6.4 Object 0x100A - Software Version

Index 0x100A	Software Version
Short description	Software version of the device
Data type (bit size)	STRING (variable)
Access	only reading
Online value	Example: "1.04.002"

9.6.5 Object 0x1018 - Identity

Index 0x1018	Identity
Short description	Characteristics of the device
Data type (bit size)	RECORD (144)
Access via complete access	possible
Subindex 0	**SubIndex 000**
Short description	Number of subsequent subindices
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	4
Subindex 1	**Vendor ID**
Short description	EtherCAT-specific identifier of the device manufacturer
Data type (bit size/bit offset)	UDINT (32 / 16)
Access	only reading
Online value	0x00000223 (547) = identifier of Helmholz GmbH & Co. KG
Subindex 2	**Product code**
Short description	Product identifier of the device
Data type (bit size/bit offset)	UDINT (32 / 48)
Access	only reading
Online value	0x00000009 (9) = identifier of the TB20 EtherCAT® bus coupler
Subindex 3	**Revision**
Short description	Revision identifier of the device
Data type (bit size/bit offset)	UDINT (32 / 80)
Access	only reading
Online value	e.g. 0x00000002 (2)
Subindex 4	**Serial number**
Short description	Serial number of the device
Data type (bit size/bit offset)	UDINT (32 / 112)
Access	only reading
Online value	e.g. 0x0001E240 (123456)

9.6.6 Object 0x1C00 - Sync Manager Communication Types

Index 0x1C00	Sync Manager Communication Types
Short description	Communication types of the available Sync Managers
Data type (bit size)	ARRAY OF USINT (48)
Access via complete access	possible

Index 0x1C00	Sync Manager Communication Types
Subindex 0	**SubIndex 000**
Short description	Number of subsequent subindices = Number of available Sync Managers
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	4
Subindex 1	**SubIndex 001**
Short description	Communication type of Sync Manager 0
Data type (bit size/bit offset)	USINT (8 / 16)
Access	only reading
Online value	1 = Mailbox communication (Master → Slave)
Subindex 2	**SubIndex 002**
Short description	Communication type of Sync Manager 1
Data type (bit size/bit offset)	USINT (8 / 24)
Access	only reading
Online value	2 = Mailbox communication (Slave → Master)
Subindex 3	**SubIndex 003**
Short description	Communication type of Sync Manager 2
Data type (bit size/bit offset)	USINT (8 / 32)
Access	only reading
Online value	3 = Process output data communication (Master → Slave)
Subindex 4	**SubIndex 004**
Short description	Communication type of Sync Manager 3
Data type (bit size/bit offset)	USINT (8 / 40)
Access	only reading
Online value	4 = Process input data communication (Slave → Master)

9.6.7 Object 0x2100 - Module Configuration Status

Index 0x2100	Module Configuration Status
Short description	Current status of the module configuration
Data type (bit size)	RECORD (24)
Access via complete access	possible
Additional details	The subindices 2 to 7 are in a direct relationship with the subindices 1 to 6 of object 0x2130 (Configured Module List Status).
Subindex 0	**SubIndex 000**
Short description	Number of subsequent subindices
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	8
Subindex 1	**Stored configuration present**
Short description	indicates whether a *stored module configuration* is present
Data type (bit size/bit offset)	BOOL (1 / 16)
Access	only reading
Online value	0 (FALSE) = not present; 1 (TRUE) = present

Index 0x2100	Module Configuration Status
Subindex 2	**Module list: based on detected modules**
Short description	indicates whether the list of *configured modules* is based on the *detected modules*
Data type (bit size/bit offset)	BOOL (1 / 17)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	The *configured* are always then based on the *detected modules*when no *stored module configuration* is present and the master has not (yet) transmitted a list with *projected modules*.
Subindex 3	**Module list: based on stored modules**
Short description	indicates whether the list of *configured modules* is based on the *stored modules from an existing stored module configuration*
Data type (bit size/bit offset)	BOOL (1 / 18)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	The *configured* modules are always then based on the *stored modules* when a *stored module configuration* is present and the master has not (yet) transmitted a list with *projected modules*.
Subindex 4	**Module list: based on projected modules**
Short description	indicates whether the list of *configured modules* is based on the *projected modules transmitted by the master*
Data type (bit size/bit offset)	BOOL (1 / 19)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	The *configured* are always then based on the *projected modules* when the master has transmitted the list with the *projected modules*.
Subindex 5	**Module list: valid (not empty, no gaps)**
Short description	Indicates whether the list of *configured modules* is valid (not empty and without gaps)
Data type (bit size/bit offset)	BOOL (1 / 20)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	The state transition PREOP→SAFEOP is only possible with a valid module list.
Subindex 6	**Module list: matches detected modules**
Short description	indicates whether the *configured* match with the *detected modules*
Data type (bit size/bit offset)	BOOL (1 / 21)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	The state transition PREOP→SAFEOP is only possible when the *configured* modules match with the *detected modules*
Subindex 7	**Module list: matches stored modules**
Short description	Indicates whether the *configured* modules match with the modules from an existing *stored module configuration*
Data type (bit size/bit offset)	BOOL (1 / 22)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	When a *stored module configuration* is present, then a state transition PREOP→SAFEOP is only possible when the *configured* modules match with the *stored modules*.

Index 0x2100	Module Configuration Status
Subindex 8	**Module parameters: match stored parameters**
Short description	Indicates whether the current parameters of the *configured modules* match with those in the *stored module configuration*
Data type (bit size/bit offset)	BOOL (1 / 23)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	When no *stored module configuration* is present or the *configured* modules do not match with the *stored modules*, then the value is always 0 (FALSE). With the object entry 0x2110:03 (Command: Reload stored parameters), the stored module parameters can be restored in PREOP.

9.6.8 Object 0x2110 - Module Configuration Control

Index 0x2110	Module Configuration Control
Short description	Control object of the module configuration
Data type (bit size)	RECORD (24)
Access via complete access	not possible
Additional details	In the case of the subindices 1 to 5, an action only takes place when it is described with the value 1 (TRUE). A description with the value 0 (FALSE) thus has no effect. Writing access to the subindices 3 to 5 is only permitted in PREOP; the SDO abort code 0x08000022 (Access in the current state not possible) is otherwise the response. Reading access to subindices 3 to 5 always delivers the value 0 (FALSE), which is of no further relevance.
Subindex 0	**SubIndex 000**
Short description	Number of subsequent subindices
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	5
Subindex 1	**Setting: Module scan provides detected modules**
Short description	Setting that provides the *detected modules* for a module or device scan carried out by the EtherCAT configuration tool
Data type (bit size/bit offset)	BOOL (1 / 16)
Access	reading and writing
Online value	0 (FALSE) = deactivated (standard value when a *stored module configuration* is present) 1 (TRUE) = activated (standard value when no *stored module configuration* is present)
Additional details	In concrete terms, this setting determines that object 0xF050 (Detected Module Ident List) contains the *detected modules*. The setting can be activated via a description with 1 (TRUE), whereby the setting of subindex 2 is then automatically deactivated.

Index 0x2110	Module Configuration Control
Subindex 2	**Setting: Module scan provides stored modules**
Short description	Setting that provides the *stored modules* for a module or device scan carried out by the EtherCAT configuration tool
Data type (bit size/bit offset)	BOOL (1 / 17)
Access	reading and writing
Online value	0 (FALSE) = deactivated (standard value when no *stored module configuration* is present) 1 (TRUE) = activated (standard value when a *stored module configuration* is present)
Additional details	In concrete terms, this setting determines that object 0xF050 (Detected Module Ident List) contains the *stored modules*. The setting can be activated via a description with 1 (TRUE) given the presence of a *stored module configuration*, whereby the setting of subindex 1 is then automatically deactivated. A description 1 (TRUE) without the presence of a *stored module configuration* results in cancellation with SDO abort code 0x08000000 (General Error).
Subindex 3	**Command: Reload stored parameters**
Short description	Command for the restoration of the module parameters in PREOP
Data type (bit size/bit offset)	BOOL (1 / 18)
Access	reading (always) and writing (only in PREOP)
Online value	0 (FALSE) = no importance
Additional details	The command is triggered through description with the value 1 (TRUE) with the presence of a *stored module configuration*. The command restores the *stored parameter data* for those *configured modules* that match with the *stored modules*. No parameter changes are made for modules that do not match with the *stored modules*. A description 1 (TRUE) without the presence of a *stored module configuration* results in cancellation with SDO abort code 0x08000000 (General Error).
Subindex 4	**Command: Store current configuration**
Short description	Command for saving the *current module configuration* in PREOP
Data type (bit size/bit offset)	BOOL (1 / 19)
Access	reading (always) and writing (only in PREOP)
Online value	0 (FALSE) = no importance
Additional details	The command is triggered through description with the value 1 (TRUE). As a precondition, the list of *configured modules* must be valid and each *configured module* must match with either the *plugged* or the *stored module* of the same plug point. The response is the SDO abort code 0x08000000 (General Error) when these preconditions are not fulfilled. The *current module configuration* is stored as a *stored module configuration* protected against power failures in the bus coupler with the command. A previously existing *stored module configuration* is overwritten as a result.
Subindex 5	**Command: Erase stored configuration**
Short description	Command for the deletion of the *stored module configuration*
Data type (bit size/bit offset)	BOOL (1 / 20)
Access	reading (always) and writing (only in PREOP)
Online value	0 (FALSE) = no importance
Additional details	The command is triggered through description with the value 1 (TRUE). No *stored module configuration* is then present any longer.

9.6.9 Object 0x2120 - Detected Module List (0xF050) Status

Index 0x2120	Detected Module List (0xF050) Status
Short description	Status for the list of the *detected modules* in the object 0xF050 (↗ 9.6.14)
Data type (bit size)	RECORD (24)
Access via complete access	possible
Subindex 0	**SubIndex 000**
Short description	Number of subsequent subindices
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	4
Subindex 1	**Valid (not empty, no gaps)**
Short description	indicates whether the list of *detected modules* is valid, and thus not empty and without gaps
Data type (bit size/bit offset)	BOOL (1 / 16)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	A state transition from PREOP to SAFEOP is only then possible when the list of *detected modules* is valid.
Subindex 2	**Matches configured modules**
Short description	indicates whether the *detected modules* match with the *detected modules*
Data type (bit size/bit offset)	BOOL (1 / 17)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	A state transition from PREOP to SAFEOP is only then possible when the list of *configured modules* is valid.
Subindex 3	**Matches stored modules**
Short description	Indicates whether the *detected* modules match with the modules from an existing *stored module configuration*.
Data type (bit size/bit offset)	BOOL (1 / 18)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	When a *stored module configuration* is present, then a state transition from PREOP to SAFEOP is only possible when the *detected* modules match with the *stored modules*.
Subindex 4	**Overlaid with stored modules**
Short description	Indicates whether the object 0xF050 (Detected Module Ident List) lists the modules from an existing *stored module configuration* in place of the *detec*ted modules
Data type (bit size/bit offset)	BOOL (1 / 19)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	Whether the *detected* or the *stored modules* are listed in the object 0xF050 depends on whether the setting is currently activated in 0x2110:01 or in 0x2110:02 (↗ 9.6.8). In the event of the presence of a *stored module configuration*, the *stored modules* are generally listed, otherwise the *detected*. Irrespective of this, the information in the subindices 1 to 3 always refer to the *detected modules*!

9.6.10 Object 0x2130 - Configured Module List (0xF030) Status

Index 0x2130	Configured Module List (0xF050) Status
Short description	Status for the list of the *configured modules* in the object 0xF030 (↗ 9.6.13)
Data type (bit size)	RECORD (24)
Access via complete access	possible
Additional details	The subindices 1 to 6 are in a direct relationship with the subindices 2 to 7 of object 0x2100 (Module Configuration Status, ↗ 9.6.7).
Subindex 0	**SubIndex 000**
Short description	Number of subsequent subindices
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	6
Subindex 1	**Preconfigured with detected modules**
Short description	indicates whether the list of *configured modules* in object 0xF030 is preconfigured with the *detected modules*
Data type (bit size/bit offset)	BOOL (1 / 16)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	Object 0xF030 is always preconfigured with the *detected modules* when no *stored module configuration* is present and the master has not (yet) transmitted a list with *projected modules since the last state transition INIT→PREOP*. As long as this is the case, changes occurring in PREOP are automatically fed to the *detected modules*.
Subindex 2	**Preconfigured with stored modules**
Short description	indicates whether the list of *configured modules* has been preconfigured with the modules *from an existing stored module configuration*
Data type (bit size/bit offset)	BOOL (1 / 17)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	Object 0xF030 is always preconfigured with the *stored modules* when a *stored module configuration* is present and the master has not (yet) transmitted a list with *projected modules since the last state transition INIT→PREOP*.
Subindex 3	**Written by master with projected modules**
Short description	indicates that the list of *configured modules* has been described by the master with the *projected modules*
Data type (bit size/bit offset)	BOOL (1 / 18)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	The bus coupler is configured as a standard in such a way (↗ 7.5.2.2) that a state transition from PREOP to SAFEOP is only possible when the master has previously described the object 0xF030 with the *projected modules*.
Subindex 4	**Valid (not empty, no gaps)**
Short description	indicates whether the list of *configured modules* is valid (not empty and without gaps)
Data type (bit size/bit offset)	BOOL (1 / 19)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	A state transition from PREOP to SAFEOP is only possible with a valid module list.

Index 0x2130	Configured Module List (0xF050) Status
Subindex 5	**Matches detected modules**
Short description	indicates whether the *configured* match with the *detected modules*
Data type (bit size/bit offset)	BOOL (1 / 20)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	A state transition from PREOP to SAFEOP is only possible when the *configured* modules match with the *detected modules*.
Subindex 6	**Matches stored modules**
Short description	indicates whether the *configured* modules match with the modules from an existing *stored module configuration.*
Data type (bit size/bit offset)	BOOL (1 / 21)
Access	only reading
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	When a *stored module configuration* is present, then a state transition from PREOP to SAFEOP is only possible when the *configured* modules match with the *stored modules*.

9.6.11 Object 0x3000 - Coupler Parameters

Index 0x3000	Coupler Parameters
Short description	Parameters of the bus coupler
Data type (bit size)	RECORD (24)
Access via complete access	possible
Additional details	The two parameters of the bus coupler are described in chapter 7.5.2. Settings for the parameters can be carried out in PREOP, SAFEOP, and OP and are automatically stored protected against power failures.
Subindex 0	**SubIndex 000**
Short description	Number of subsequent subindices
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	2
Subindex 1	**Hot-swapping allowed**
Short description	defines whether the exchanging of a module (hot swap, ↗7.6) during ongoing operation (SAFEOP/OP) is allowed
Data type (bit size/bit offset)	BOOL (1 / 16)
Access	reading and writing
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	See chapter 7.5.2.1
Subindex 2	**Master need not provide projected modules**
Short description	defines whether a state transition from PREOP to SAFEOP should also be possible without the transmission of projected modules by the master
Data type (bit size/bit offset)	BOOL (2 / 16)
Access	reading and writing
Online value	0 (FALSE) = no, 1 (TRUE) = yes
Additional details	See chapter 7.5.2.2

9.6.12 Object 0xF000 - Modular Device Profile

Index 0xF000	Modular Device Profile
Short description	Characteristics for the Modular Device Profile (MDP) used
Data type (bit size)	RECORD (48)
Access via complete access	possible
Subindex 0	**SubIndex 000**
Short description	Number of subsequent subindices
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	2
Subindex 1	**Module index distance**
Short description	Index distance between the module objects of two successive slots
Data type (bit size/bit offset)	UINT (16 / 16)
Access	only reading
Online value	0x0010 (16)
Additional details	The index distance is used for the following module objects, whereby *MN* is a filler for the two-digit, hexadecimal module number: • Objects for the input data (0x6*MN*0) and output data (0x7*MN*0) • Objects for the module status (0xA*MN*0), channel status (0xA*MN*1), and diagnostic status (0xA*MN*2)
Subindex 2	**Maximum number of modules**
Short description	Maximum number of modules
Data type (bit size/bit offset)	UINT (16 / 32)
Access	only reading
Online value	64 = up to 64 modules can be plugged into the bus coupler

9.6.13 Object 0xF030 - Configured Module Ident List

Index 0xF030	Configured Module Ident List
Short description	List of *configured modules*
Data type (bit size)	ARRAY OF UDINT (2064)
Access via complete access	possible
Additional details	The object contains the module ident values (↗ 7.4.2) of the *configured modules* in the subindices 1 to a maximum of 64. The subindices 1 to 64 thereby correspond to the plug points or module numbers 1 to 64. The Module Ident Number 0x00000000 stands for 'no module configured' and thus corresponds to a module gap. The list of the *configured modules* is then only valid when at least one module is configured and there are no module gaps. In the event of the presence of a *stored module configuration*, the object is preconfigured with the *stored*, otherwise with the *detected modules*. It can be described by the master with the *projected modules*. Object 0x2130 (Configured Module Ident List Status, ↗ 9.6.10) contains important status information on this object.
Subindex 0	**SubIndex 000**
Short description	Number of subsequent subindices = number of configured plug points/ modules
Data type (bit size/bit offset)	USINT (8 / 0)
Access	reading (always) and writing (only PREOP)
Online value	e.g. 10 = list encompasses 10 *configured modules* for the plug points 1 to 10 in the subindices 1 to 10
Additional details	a maximum of 64 modules can be configured.
Subindex 1	**SubIndex 001**

Index 0xF030	Configured Module Ident List
Short description	Module Ident Number of the module for the first plug point
Data type (bit size/bit offset)	UDINT (32 / 16)
Access	reading (always) and writing (only PREOP)
Online value	e.g. 0x07080000 (8 DI module)
Subindex 2	SubIndex 002
Short description	Module Ident Number of the module for the second plug point
Data type (bit size/bit offset)	UDINT (32 / 48)
Access	reading (always) and writing (only PREOP)
Online value	e.g. 0x0AF00000 (8 DO module)
...	...

9.6.14 Object 0xF050 - Detected Module Ident List

Index 0xF050	Detected Module Ident List
Short description	List of *detected*, meaning *plugged modules*
Data type (bit size)	ARRAY OF UDINT (2064)
Access via complete access	possible
Additional details	The object contains the module ident values (↗ 7.4.2) of the *detected modules* in the subindices 1 to a maximum of 64. If, however, a *stored module configuration* is present in the bus coupler, then the list is overlaid with the *stored modules* as a standard, which is indicated by the object entry 0x2120:04 (Overlaid with stored modules,↗ 9.6.9). The modules contained in object 0xF050 can be read out by the EtherCAT configuration tool in the context of a module or device scan and be adopted as *projected modules*. With the help of subindices 1 and 2 of the object 0x2110 (Module Configuration Control, ↗ 9.6.8), it can be manually defined whether the object 0xF050 should contain the Module Ident Values of the *detected* or the *saved modules*. The list of the *detected modules* is only valid when it contains at least one module and there are no module gaps. A module gap is represented by the Module Ident Number 0x00000000. The subindices 1 to 64 correspond to the plug points or module numbers 1 to 64. This connection may no longer apply for plug points located behind a module gap, as the actual size of a module gap can't always be detected. Object 0x2130 (Configured Module Ident List Status, ↗ 9.6.9) contains important status information on this object.
Subindex 0	SubIndex 000
Short description	Number of subsequent subindices = (number of *detected modules* + number of detected module gaps) or number of *stored modules*
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	e.g. 10 = 10 modules detected
Additional details	The list can encompass a maximum of 64 modules.
Subindex 1	SubIndex 001
Short description	Module Ident Number of the *detected* or *stored module* in the first plug point
Data type (bit size/bit offset)	UDINT (32 / 16)
Access	only reading
Online value	e.g. 0x07080000 (8 DI module)

Index 0xF050	Detected Module Ident List
Subindex 2	**SubIndex 002**
Short description	Module Ident Number of the *detected* or *stored module* in the second plug point
Data type (bit size/bit offset)	UDINT (32 / 48)
Access	only reading
Online value	e.g. 0x0AF00000 (8 DO module)
...	...

9.7 The module objects

The module objects are shown dynamically in the object directory, depending upon which modules *are projected* or *configured*. Which module objects are shown for a certain module depends upon its functional scope. The various module objects are listed in the following table with a brief description.

Module object	Description
Parameter data object (↗ 9.7.3)	This object contains the parameter data of a module and is therefore only present for configurable modules. For the configuration or reconfiguration of a module, this object can be described in PREOP, SAFEOP, and OP.
Input data object (↗ 9.7.4)	This object contains the input data of a module and is therefore only present for modules with input data. It is not required for access to the input data, as this is provided current with the cycle via the process input data of the bus coupler.
Output data object (↗ 9.7.5)	This object contains the output data of a module and is therefore only present for modules with output data. Writing access to the output data via this object is not permitted and also not necessary, as this is provided via the process output data of the bus coupler.
Module status object (↗ 9.7.6)	This object contains the module status and is present for all modules. It is not required for access to the module status, as this is also provided current with the cycle via the process input data of the bus coupler.
Channel status object (↗ 9.7.7)	This object contains the channel status of a module and is therefore only present for diagnostics-capable modules with I/O data. It is not required for access to the channel status, as this is also provided current with the cycle via the process input data of the bus coupler.
Diagnostic status object (↗ 9.7.8)	This object contains the diagnostic status of a module and is therefore only present for diagnostic-capable modules. Access to the diagnostic status of a module must take place via this object.

9.7.1 Index of module objects

The following table lists the ranges of the object directory in which the various module objects are shown and how the index of such an object can be calculated for a certain module.

Module object	Range	Index calculation
Parameter data object	0x3010 – 0x3400	0x3000 + 0x10 · <Module numbers>
Input data object	0x6010 – 0x6400	0x6000 + 0x10 · <Module numbers>
Output data object	0x7010 – 0x7400	0x7000 + 0x10 · <Module numbers>
Module status object	0xA010 – 0xA400	0xA000 + 0x10 · <Module numbers>
Channel status object	0xA011 – 0xA401	0xA001 + 0x10 · <Module numbers>
Diagnostic status object	0xA012 – 0xA402	0xA002 + 0x10 · <Module numbers>

The module number used for the index calculation corresponds to the plug point number and lies in the range from 1 to 64. The distance between module objects of the same kind of directly successive modules is designated as 'Module Index Distance' and is indicated in the object 0xF000 (Modular Device Profile, ↗ 9.6.12). In the case of the TB20 EtherCAT® bus coupler, a fixed index distance of 0x10 (=16) used.

9.7.2 Module objects and their indices for an exemplary module selection

The following table shows which module objects are available for an exemplary module selection and via which indices they can be addressed.

Module numbers	1	2	3	4	5	6	7	8
Module (Order number)	**PS 24V** (600-700-0AA01)	**8 DI** (600-210-0AH01)	**4 DO** (600-220-0AD01)	**8 DI/8 DO** (600-230-0AP21)	**4 DO HF** (600-220-7AD01)	**PWR 24V** (600-710-0AA01)	**2 AI U** (600-252-4AB01)	**4 AO U** (600-261-4AD01)
Parameter data	0x3010	-	-	-	0x3050	-	0x3070	0x3080
Input data	-	0x6020	-	0x6040	0x6050	-	0x6070	-
Output data	-	-	0x7030	0x7040	0x7050	-	-	0x7080
Submodule status	0xA010	0xA020	0xA030	0xA040	0xA050	0xA060	0xA070	0xA080
Channel status	-	-	-	-	0xA051	-	0xA071	0xA081
Diagnostic status	0xA012	-	-	-	0xA052	-	0xA072	0xA082

9.7.3 The parameter data objects

A corresponding parameter data object will be shown in the object directory of the bus coupler for each *configured module* that is configurable. The prerequisite for this is that the respective module is plugged into the backplane bus or matches with the *stored module* of an existing *stored module configuration*. Only in these two cases does the bus coupler have the required information on the parameter data at its disposal.

The parameter data objects have the data type RECORD. The individual object entries within such an object are dynamically generated in the basis of the structure of the parameter data. The structure of the parameter data can be looked up in the corresponding module manual.

Subindex 0 of a parameter data object contains the number *n* of the subsequent subindices. The individual parameter data elements are shown as follows on subindices 1 to *n*:

- The representation takes place within the parameter data ordered according to ascending position.
- Parameter data elements that encompass 1, 2, or 4 bytes are respectively represented on their own subindex. The data type (↗ 9.4) of the subindex is thereby oriented to the bit size (8, 16, or 32) and type (with or without prefixes) of the contained parameter data element.
- Parameter data elements that encompass only individual bits and are contained in a shared parameter data byte are shown summarized on a subindex of the data type BYTE.
- In the case of access via Complete Access, the read or written object data as of subindex 1 corresponds precisely to the parameter data described in the respective module manual.

Reading and writing access to the parameter data objects is possible in the EtherCAT states PREOP, SAFEOP, and OP. Changes made to the parameter data objects made in the EtherCAT state PREOP are initially only adopted into the *current module configuration*. The transmission of the changed parameter data to the respective modules first takes place during the state transition from PREOP to SAFEOP, where the *current* is adopted as the *active module configuration*.

Changes to parameter data objects carried out in the EtherCAT states SAFEO and OP with Complete Access are immediately adopted in the active module configuration and directly transmitted to the respective modules. When the changes are not made via Complete Access, but instead subindex-wise, then subindex 0 must subsequently be described with the already contained value in order for the changes to take effect.

9.7.3.1 General structure of the parameter data object

Parameter data object	
Index range	0x3010 - 0x3400 (manufacturer-specific)
Index calculation	0x3000 + 0x10 · <Module numbers>
Offline designation	"Parameters (<Short identifier> - <Order number>)"
Online designation	"Modules <Module numbers> - Parameters"
Data type	RECORD
Bit size	dependent upon the number of available subindices and their data types
Access via complete access	possible
Subindex 0	
Short description	Number of subsequent subindices
Offline/online designation	“SubIndex 000”
Data type (bit size/bit offset)	USINT (8 / 0)
Access	reading and writing
Offline/online value	dependent upon the structure of the parameter data of the module
Additional details	Through the description of subindex 0 with the already contained value, the consistent transmission of previously changed parameter data to the relevant module can be initiated in SAFEOP and OP. If an attempt is made to describe subindex 0 with a different value, access is denied with the SDO abort code 0x06040043.
Subindex <*s*>	
Short description	Date of parameter data element <*e*> with *e* = *s* - 1 (enumeration begins with 0 in subindex 1)
Offline designation	A parameter-specific designation is used that originates from the module description in the ESI file. If this involves a summarized parameter data element of the data type BYTE, then the generic designation "Parameter byte [<Byte Offset>] is used as an alternative. <Byte Offset> is a filler for the byte offset, meaning the position within the parameter data.
Online designation	"Parameter <*e*>"
Data type	dependent upon bit size and type of the parameter data element
Bit offset	begins at subindex 1 with 16 and increases with each additional subindex by the bit size of the previous data type
Access	reading and writing
Offline value	Default value according to module description in the ESI file
Online value	current value of the parameter
…	

9.7.3.2 Concrete structure on the basis of an example

The module 'AO 2x I, 0/4-20 mA, 12 Bit' (600-260-4AB01) is used as an example. According to the "Manual TB20 modules", the parameter data is structured as follows:

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Diagnosis alarm	-	-	1	-	-	Representation values	
1	Wire break detection Channel 0	Channel 0 behavior at CPU-STOP			Channel 0 output range			
2	Channel 0 substitute value							
3								
4	Wire break detection Channel 1	Channel 1 behavior at CPU-STOP			Channel 1 output range			
5	Channel 1 substitute value							
6								

Derived from this, the following structure of the parameter data object for the module results:

Index 0x3050	
Short description	Parameter data object from module 5
Offline designation	"Parameters (2 AO I - 600-260-4AB01)"
Online designation	"Module 5 - Parameters"
Data type (bit size)	RECORD (72)
Subindex 0	
Short description	Number of subsequent subindices
Offline/online designation	"SubIndex 000"
Data type (bit size/bit offset)	USINT (8 / 0)
Subindex 1	
Short description	corresponds to the module parameters summarized in byte 0
Offline designation	"Parameter byte [0]"
Online designation	"Parameter 0"
Data type (bit size/bit offset)	BYTE (8 / 16)
Subindex 2	
Short description	corresponds to the module parameters summarized in byte 1
Offline designation	"Parameter byte [1]"
Online designation	"Parameter 1"
Data type (bit size/bit offset)	BYTE (8 / 24)
Subindex 3	
Short description	corresponds to the "Substitute value channel 0" contained in bytes 2 and 3
Offline designation	"Channel 0: Substitute value"
Online designation	"Parameter 2"
Data type (bit size/bit offset)	INT (16 / 32)
Subindex 4	
Short description	corresponds to the module parameters summarized in byte 4
Offline designation	"Parameter byte [4]"
Online designation	"Parameter 3"
Data type (bit size/bit offset)	BYTE (8 / 48)

Index 0x3050	
Subindex 5	
Short description	corresponds to the "Substitute value channel 1" contained in bytes 5 and 6
Offline designation	"Channel 1: Substitute value"
Online designation	"Parameter 4"
Data type (bit size/bit offset)	INT (16 / 56)

9.7.4 The input data objects of the modules

A corresponding input data object will be shown in the object directory of the bus coupler for each *configured module* having input data. The prerequisite for this is that the respective module is plugged into the backplane bus, as the bus coupler can only then call up the required information on the input data from the module.

The input data objects have the data type RECORD. The individual object entries within such an object are dynamically generated on the basis of the structure of the input data. The structure of the input data can be looked up in the corresponding module manual.

Subindex 0 of an input data object contains the number *n* of the subsequent subindices. The individual input data elements are shown as follows on subindices 1 to *n*:

- The representation takes place within the input data ordered according to ascending position.
- Input data elements that encompass 1, 2, or 4 bytes are respectively represented on their own subindex. The data type (↗ 9.4) of the subindex is thereby oriented to the bit size (8, 16, or 32) and type (with or without prefixes) of the contained input data element.
- Input data elements that encompass only individual bits and are contained in a shared input data byte are shown summarized on a subindex of the data type BYTE.
- In the case of digital modules, the input data elements that contain the states of the digital input channels are resolved bit-precise and respectively represented on an own subindex of the data type BOOL. The representation thereby takes place within the input data ordered according to ascending byte and bit position.

Reading access to input data objects is possible in PREOP, SAFEOP, and OP. However, current input data is only made available in SAFEOP and OP. Zero values are returned in PREOP for the contained input data.

There is normally no reason to access the input data of the modules via these objects, as the input data of the modules is also made available via the cycle-current process input data of the bus coupler in SAFEOP and OP (↗ 7.4.5). The input data objects are instead required in the context of the PDO configuration as reference objects for the representation of the input data on TxPDOs (↗ 9.8.3).

9.7.4.1 General structure of the object

Input data object	
Index range	0x6010 - 0x6400 (MDP-specific)
Index calculation	0x6000 + 0x10 · <Module numbers>
Offline designation	"Inputs (<Short identifier> - <Order numbers>)"
Online designation	"Modules <Module numbers> - Inputs"
Data type	RECORD
Bit size	• is dependent upon the number and the data type of the available subindices • is rounded up to a multiple of 8
Access via complete access	not possible

Subindex 0	
Short description	Number of subsequent subindices
Offline/online designation	“SubIndex 000”
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	dependent upon the structure of the input data of the module
…	
Subindex <*s*>	
Short description	Data of input data element <*e*> with *e* = *s* - 1 (enumeration begins with 0 in subindex 1)
Offline designation	A data element-specific designation in accordance with the module description is used in the ESI file. If this involves a summarized input data element of the data type BYTE, then one of the following generic designations are used as an alternative. • "Input byte [<Byte Offset>]" (contains general input data) • "Status byte [<Byte Offset>]" (contains status information) • "Reserved byte [<Byte Offset>" (reserved, without function) <Byte Offset> is a filler for the byte offset, meaning the position within the input data.
Online designation	"Input <*e*>"
Data type	dependent upon bit site and type of the input data element
Bit offset	if subindex 1 begins with 16 and increases with each additional subindex by the bit size of the previous data type
Access	only reading
Online value	in PREOP: always 0 in SAFEOP and OP: current value from the module
…	

9.7.4.2 Concrete structure of the object on the basis of two examples

Example 1 based on the digital input module 'DI 4 x DC 24 V' (600-210-0AD01)

The input data of the module is structured as follows:

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	-	-	-	-	DI 3 (Input status of channel 3)	DI 2 (Input status of channel 2)	DI 1 (Input status of channel 1)	DI 0 (Input status of channel 0)

The structure of the corresponding input data object for the module makes clear the bit-precise resolution of digital input data:

Index 0x60B0	
Short description	Input data object from module 11
Offline designation	"Inputs (4 DI - 600-210-0AD01)"
Online designation	"Module 11 - Inputs"
Data type (bit size)	RECORD (24)
Subindex 0	
Short description	Number of subsequent subindices
Offline/online designation	“SubIndex 000”
Data type (bit size/bit offset)	USINT (8 / 0)
Subindex 1	
Short description	corresponds to the input data element 'DI 0' in bit 0 of input data byte 0
Offline designation	"Input 0"
Online designation	"Input 0"
Data type (bit size/bit offset)	BOOL (1 / 16)
Subindex 2	
Short description	corresponds to the input data element 'DI 1' in bit 1 of input data byte 0
Offline designation	"Input 1"
Online designation	"Input 1"
Data type (bit size/bit offset)	BOOL (1 / 17)
Subindex 3	
Short description	corresponds to the input data element 'DI 2' in bit 2 of input data byte 0
Offline designation	"Input 2"
Online designation	"Input 2"
Data type (bit size/bit offset)	BOOL (1 / 18)
Subindex 4	
Short description	corresponds to the input data element 'DI 3' in bit 3 of input data byte 0
Offline designation	"Input 3"
Online designation	"Input 3"
Data type (bit size/bit offset)	BOOL (1 / 19)

Example 2 based on the 5V counter module (600-310-7AA01) in the operating mode "Counter Mode"

Structure of the input data for the Counter Mode according to the manual of the counter module:

Byte	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0 - 3	Count							
4	Sensor supply short circuit	-	Parameter error	-	-	Status bits being reset	Error with loading function	Loading function running
5	“Down” counting direction	“Up” counting direction	-	-	-	-	-	Status SW gate
6	Zero crossing	Lower counter limit	Upper counter limit	Status Comparator 2	Status Comparator 1	-	-	Synchroniza tion status
7	-	-	-	-	-	-	-	-

The following represents the structure of the corresponding input data object for the module. Subindices 2 to 4 respectively are summarized input data elements of the data type BYTE. The input data bits contained in bytes 4 to 6 are summarized in it and represented in an identical layout .

Subindex 5 bears the offline designation "Reserved byte [5]", as the input data byte represented within it currently has no function.

Index 0x6140	
Short description	Input data object from module 20
Offline designation	"Inputs (1CNT 5V - 600-310-7AA01)"
Online designation	"Module 20 - Inputs"
Data type (bit size)	RECORD (80)
Subindex 0	
Short description	Number of subsequent subindices
Offline/online designation	"SubIndex 000"
Data type (bit size/bit offset)	USINT (8 / 0)
Subindex 1	
Short description	corresponds to the input data element 'Count' in the input data bytes 0 to 3
Offline designation	"Counter value"
Online designation	"Input 0"
Data type (bit size/bit offset)	BOOL (32 / 16)
Subindex 2	
Short description	corresponds to the summarized input data elements in input data byte 4
Offline designation	"Input byte [4]"
Online designation	"Input 1"
Data type (bit size/bit offset)	BOOL (8 / 48)
Subindex 3	
Short description	corresponds to the summarized input data elements in input data byte 5
Offline designation	"Input byte [5]"
Online designation	"Input 2"
Data type (bit size/bit offset)	BOOL (8 / 56)
Subindex 4	
Short description	corresponds to the summarized input data elements in input data byte 6
Offline designation	"Input byte [6]"
Online designation	"Input 3"
Data type (bit size/bit offset)	BOOL (8 / 64)
Subindex 5	
Short description	corresponds to the reserved input data byte 7
Offline designation	"Reserved byte [7]"
Online designation	"Input 4"
Data type (bit size/bit offset)	BOOL (8 / 72)

9.7.5 The output data objects

A corresponding output data object will be shown in the object directory of the bus coupler for each *configured module* having output data. The prerequisite for this is that the respective module is plugged into the backplane bus, as the bus coupler can only then call up the required information on the output data from the module.

The output data objects have the data type RECORD. The individual object entries within such an object are dynamically generated on the basis of the structure of the output data. The structure of the output data can be looked up in the corresponding module manual.

Subindex 0 of an output data object contains the number *n* of the subsequent subindices. The individual output data elements are shown as follows on subindices 1 to *n*:

- The representation takes place within the output data ordered according to ascending position.
- Output data elements that encompass 1, 2, or 4 bytes are respectively represented on their own subindex. The data type (↗ 9.4) of the subindex is thereby oriented to the bit size (8, 16, or 32) and type (with or without prefixes) of the contained output data element.
- Output data elements that encompass only individual bits and are contained in a shared output data byte are shown summarized on a subindex of the data type BYTE.
- In the case of digital modules, the output data elements that contain the states of the digital output channels are resolved bit-precise and respectively represented on an own subindex of the data type BOOL. The representation thereby takes place within the output data ordered according to ascending byte and bit position.

Only reading access to output data objects is possible in PREOP, SAFEOP, and OP. Writing access is not possible and also not necessary, as the actual setting of the output data of the modules in SAFEOP and OP takes place via the cycle-current process output data of the bus coupler. The output data objects are instead required in the context of the PDO configuration as reference objects for the representation of the output data on RxPDOs (↗ 9.8.2).

9.7.5.1 General structure of the object

Output data object	
Index range	0x7010 - 0x7400 (MDP-specific)
Index calculation	0x7000 + 0x10 · <Module numbers>
Offline designation	"Outputs (<Short identifier> - <Order numbers>)"
Online designation	"Modules <Module numbers> - Outputs"
Data type	RECORD
Bit size	• is dependent upon the number and the data type of the available subindices • is always rounded up to a multiple of 8
Access via complete access	not possible
Subindex 0	
Short description	Number of subsequent subindices
Offline/online designation	“SubIndex 000”
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	dependent upon the structure of the output data of the module
…	
Subindex <*s*>	
Short description	Data of output data element <*e*> with *e* = *s* - 1 (enumeration begins with 0 in subindex 1)
Offline designation	A data element-specific designation in accordance with the module description is used in the ESI file. If this involves a summarized output data element of the data type BYTE, then one of the following generic designations is used as an alternative. • "Output byte [<Byte Offset>]" (contains general output data) • "Reserved byte [<Byte Offset>" (reserved, without function) <Byte Offset> is a filler for the byte offset, meaning the position within the output data.
Online designation	"Output <*e*>"
Data type	dependent upon bit site and type of the output data element
Bit offset	begins at subindex 1 with 16 and increases with each additional subindex by the bit size of the previous data type
Access	only reading

Online value	in PREOP: always 0 in SAFEOP / OP: values set via the process output data of the bus coupler The output data contained here is only transmitted cycle-current to the respective module in OP. This transmission does not take place in SAFEOP, and the outputs of the module are in the secure state. This means that a match between the output data shown here and the secure state of the outputs at the module is not necessary in SAFEOP.
…	

The representation of the module output data on the output data objects takes place analog with the representation of the module input data on the input data objects. We therefore refer here to the two concrete examples of the structure of the input data objects in chapter 9.7.4.2.

9.7.6 The module status objects

A module status object is shown in the object directory of the bus coupler for each *configured module*, which contains the module status described in chapter 7.4.6.

The module status object has the data type RECORD. The structure is the same for all modules and contains in subindices 1 and 2 the information of whether the module is *present* and *operational*. Subindex 0 with the number of subsequent subindices contains the value 2 as a result.

Reading access to module status objects is possible in PREOP, SAFEOP, and OP. However, such access doesn't make sense, as the module status is also contained in the process input data of the bus coupler transmitted cyclically to the master. The module status objects are instead required in the context of the PDO configuration as reference objects for the representation of the module status on TxPDOs (↗ 9.8.3).

9.7.6.1 Structure of the module status object

Module status object	
Index range	0xA010 - 0xA400 (MDP-specific)
Index calculation	0xA000 + 0x10 · <module number>
Offline designation	"Module Status"
Online designation	"Modules <Module numbers> - Module Status"
Data type	RECORD
Bit size	24
Access via complete access	possible
Subindex 0	
Short description	Number of subsequent subindices
Offline/online designation	"SubIndex 000"
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	2
Subindex 1	
Short description	indicates whether the module is *present*
Offline/online designation	"Present"
Data type (bit size/bit offset)	BOOL (1 / 16)
Access	only reading
Online value	0 (FALSE): Module is not *present* or the wrong module is plugged 1 (TRUE): Module is *present*

Module status object	
Subindex 2	
Short description	indicates whether the module is *operational*
Offline/online designation	"Operational"
Data type (bit size/bit offset)	BOOL (1 / 16)
Access	only reading
Online value	0 (FALSE): Module is *not operational* 1 (TRUE): Module is *operational*

9.7.7 The channel status objects

A channel status object that contains the channel status described in chapter 7.4.7 is shown in the object directory of the bus coupler for each *configured module* that is diagnosis-capable and possesses I/O data. The prerequisite for this is that the respective module is plugged into the backplane bus, as the bus coupler can only then call up the required information from the module.

The channel status objects have the data type RECORD. A corresponding object entry of the data type BOOL is contained in this object for each present channel of the diagnosis-capable module. The object entries show the status of the respective channels, which can be either "not OK" (0) or "OK" (1). The number of subsequent subindices contained in subindex 0 corresponds as a result to the number of channels of the respective module.

Reading access to channel status objects is possible in PREOP, SAFEOP, and OP. However, such access doesn't make sense, as the channel status of the diagnosis-capable *configured modules* is also contained in the process input data of the bus coupler transmitted cyclically to the master. The channel status objects are instead required in the context of the PDO configuration as reference objects for the representation of the channel status on TxPDOs (↗ 9.8.3).

9.7.7.1 General structure of the object

Channel status object	
Index range	0x7011 - 0x7401 (MDP-specific)
Index calculation	0x7001 + 0x10 · <Module numbers>
Offline designation	"Channel Status"
Online designation	"Modules <Module numbers> - Channel Status"
Data type	RECORD
Bit size	• is dependent upon the number of channels of the module • is rounded up to a multiple of 8
Access via complete access	possible
Subindex 0	
Short description	Number *n* of subsequent subindices
Offline/online designation	"SubIndex 000"
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	corresponds to the number of channels of the module
...	

Channel status object	
Subindex <*s*>	
Short description	Status of the channel <c> with c = *s* - 1 (enumeration begins with 0 in subindex 1)
Offline/online designation	"Channel <*c*> OK"
Data type (bit size/bit offset)	BOOL (1 / 16 + *c*)
Access	only reading
Online value	0 (FALSE): Channel is not 'OK' 1 (TRUE): Channel is 'OK' A channel is then shown as OK when the following conditions have been met: • the bus coupler is in SAFEOP or OP • the module is *present* and *operational* according to the module status • there is no diagnostic message present for the channel in the module
…	

9.7.8 The diagnostic status objects

A diagnostic status object is shown in the object directory of the bus coupler for each *configured module* that is diagnosis-capable, which contains the diagnostic status described in chapter 7.4.8. The prerequisite for this is that the respective module is plugged into the backplane bus, as the bus coupler can only then call up the required information from the module.

The diagnostic status objects have the data type RECORD. Each of the diagnosis IDs contained in the diagnostic status is represented in this object on a corresponding object entry of the data type USINT. This representation starts with ascending position within the diagnostic status with subindex 1. The number of subsequent subindices contained in subindex 0 corresponds to the number of diagnosis IDs contained in the diagnostic status.

Reading access to diagnostic status objects is possible in PREOP, SAFEOP, and OP. In contrast with the module and channel status, the diagnostic status is not contained in the process input data of the bus coupler transmitted cyclically to the master. This means that the diagnostic status of a module can only be determined through access to the respective diagnostic status object. This access only makes sense when the presence of a diagnostic message is displayed through the module or channel status of the respective module, and should then be defined in more detail to determine which diagnostic message(s) is/are involved.

9.7.8.1 General structure of the object

Diagnostic status object	
Index range	0x7012 - 0x7402 (MDP-specific)
Index calculation	0x7002 + 0x10 · <Module numbers>
Offline designation	"Diagnostic Status"
Online designation	"Modules <Module numbers> - Diagnostic Status"
Data type	RECORD
Bit size	16 + 8 · *n*
Access via complete access	possible
Subindex 0	
Short description	Number *n* of subsequent subindices
Offline/online designation	“SubIndex 000”
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	corresponds to the number of channels of the module + 1

Diagnostic status object	
Subindex 1	
Short description	contains the module-related diagnosis ID
Offline/online designation	"Modules"
Data type (bit size/bit offset)	USINT (8 / 16)
Access	only reading
Online value	0: no module-related diagnostic message is present 1-255: no defined module-related diagnostic message is present (↗ 7.4.8)
…	
Subindex <*s*>	
Short description	Diagnostic ID for channel <*c*> with *c* = *s* - 2 (enumeration begins with 0 in subindex 2)
Offline/online designation	"Channel <c>"
Data type (bit size/bit offset)	USINT (8 / 24 + 8 · c)
Access	only reading
Online value	0: no channel-related module diagnosis is present 1-255: a defined channel-related diagnostic message is present (↗ 7.4.8)
…	

9.8 The objects for PDO configuration of the bus coupler

The cyclical exchange of process data between the master and the bus coupler is organized in the form of process data objects (PDOs), whereby a differentiation is made depending upon the transmission direction between RxPDOs and TxPDOs:

RxPDOs are used for the transmission of process output data from the master to the bus coupler. From the perspective of the bus coupler, RxPDOs are thus received and therefore have the prefix Rx = Reception. The content of the output data objects of all *configured modules* is represented as process output data of the bus coupler in RxPDOs.

TxPDOs are used for the transmission of process input data from the bus coupler to the master. From the perspective of the bus coupler, TxPDOs are thus sent and therefore have the prefix Tx = Transmission. The content of the input data objects of all *configured modules* is represented as process input data of the bus coupler in TxPDOs. The content of the module status and channel status objects of these modules is also represented in TxPDOs.

The following described PDO mapping objects of the bus coupler are responsible for the representation of the content of certain module objects in Rx and TxPDOs. The two PDO assignment objects 0x1C12 and 0x1C13 of the bus coupler are then described (↗ 9.8.4). They ensure that the Rx and TxPDOs are assigned to the Sync Managers 2 and 3, which are responsible for the cyclical exchange of the process data with the master.

9.8.1 The PDO mapping objects

The PDO mapping objects are used to determine which data from the objects present in the object directory are to be represented in PDOs. To this purpose, the PDO mapping objects defined as RECORD in the subindices 1 to n contain reference data that refers to the corresponding objects or object entries. The structure of a PDO results from the gap-free stringing together of the referenced object data containing the ascending subindex.

The reference data contained in the subindices of the data type UDINT encompass 32 bits and are structured as follows:

Bits	Meaning
31 – 24	Index of the referenced object
23 – 16	
15 – 8	Subindex for reference to a RECORD or ARRAY object, otherwise 0
7 – 0	Bit size of the referenced data

With regard to the reference data, the following notation is used in this manual:

<hexadecimal index>:<decimal subindex>:<decimal bit size>

Example: 0x6040:03: 16

A 16-bit data element in subindex 3 of the object 0x6040 is referenced. This is the input data object of module 4.

So-called padding bits can also be added to a PDO by way of a special reference data entry. These are then used to fill up a PDO to a byte limit as needed. A padding entry always refers to the subindex 0 of object 0x0000 and contains the number of padding bits to be added as the bit size. Padding bits are assigned with zeros in the PDO with respect to values.

Example: 0x0000:00: 4

This reference entry encompasses 4 padding bits.

9.8.2 The RxPDO mapping objects

An RxPDO mapping object exists for each output data object present in the object directory. The RxPDO mapping object ensures that the data contained in the output data object is represented in an output data RxPDO.

The index of an RxPDO mapping object depends upon the number of the module to which its output data object is being referred to and calculates it as follows:

Index of the RxPDO mapping object = 0x1600 + <Module numbers>

The index of the output data object that is being referred to depends upon this module number:

Index of the output data object = 0x7000 + 0x10 · <Module numbers>

This results in the following context:

For a module with the module number 0x*MN*, the data contained in the output data object 0x7*MN*0 is represented by the RxPDO mapping object 0x16*MN* in an output data RxPDO.

All of the data elements contained in the output data object are always represented in the output data RxPDOs in the same arrangement. That means that a corresponding reference data entry referring to this subindex exists in the RxPDO mapping object for each subindex present in the output data object.

When necessary, an additional padding entry is present, with which it can be ensured through the attachment of a certain number of padding bits that the size of the RxPDO corresponds to a multiple of a byte. This ensures that the RxPDOs strung together free of gaps in the process output image of the bus coupler are always oriented to byte limits.

The RxPDO mapping objects and their content are automatically generated by the bus coupler on the basis of the output data objects available for the *configured modules*. They are only visible in the online object directory of the bus coupler and allow reading, but not writing access. This means that the RxPDO configuration of the bus coupler cannot be changed manually.

9.8.2.1 General structure of an RxPDO mapping object

RxPDO mapping object for output data	
Index range	0x1601 - 0x1640
Index calculation	0x1600 + <Module numbers> = 0x16*MN* (*MN* stands for the two-digit, hexadecimal module number)
Online designation	"RxPDO (Modules <Module numbers> - Outputs)"
Short description	Representation of the data of the output data object 0x7*MN*0 (= Reference object) in an output data RxPDO
Data type	RECORD
Bit size	16 + 32 · *n*
Access via complete access	possible
Subindex 0	
Short description	Number *n* of subsequent subindices
Online designation	“SubIndex 000”
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	• corresponds to the value of subindex 0 of the reference object, and thus to the number of the individual data elements contained in this object. • Increases by 1 in the event that a padding entry exists
…	
Subindex <*s*>	
Short description	Reference data entry <*e*> with *e* = *s* - 1, which refers to the subindex <*s*> in the reference object (enumeration starts with 0 in subindex 1)
Online designation	"Mapping entry <*e*>"
Data type (bit size/bit offset)	UDINT (32 / 16 + 32 · *e*)
Access	only reading
Online value	0x7*MN*0:<*s*>:<Bit size of the subindex in the reference object>
…	

9.8.2.2 Example of the content of an RxPDO object

In this example, the analog input module 'AI 4 x TC, Iso., 16 Bit' (600-254-4AD02) is used as module 3. The module possesses output data containing the values for the external temperature compensation of the 4 channels.

The representation of the output data contained in object 0x7030 in an RxPDO takes place in the RxPDO mapping object 0x1603 as shown in the following:

Output data object 0x7030				RxPDO mapping object 0x1603		
SI	Data type (bit size)	Description		SI	Online value	Description
0	USINT (8)	Subindex 0 (= 4)	→	0	4	Subindex 0
1	INT (16)	Ext. temp. compensation channel 0	→	1	0x7030:01:16	Reference to 0x7030:01 with 16 bits
2	INT (16)	Ext. temp. compensation channel 1	→	2	0x7030:2:16 AM	Reference to 0x7030:02 with 16 bits
3	INT (16)	Ext. temp. compensation channel 2	→	3	0x7030:3:16 AM	Reference to 0x7030:03 with 16 bits
4	INT (16)	Ext. temp. compensation channel 3	→	4	0x7030:4:16 AM	Reference to 0x7030:04 with 16 bits

In chapter 9.8.3.2, a similar representation for the content of the TxPDO mapping objects with regard to the input data and the module and channel status is contained for the module chosen as an example.

9.8.3 The TxPDO mapping objects

For each of the following module objects present in the object directory there exists a TxPDO mapping object that represents the contained data on a corresponding TxPDO:

Input data object → Input data TxPDO

Module status object → Module status TxPDO

Channel status object → Channel status TxPDO

Both the index of the TxPDO mapping object and the index of the referenced module object depend upon the respective number (1 - 64) of the relevant module:

TxPDO	Index of the TxPDO mapping object	Index of the reference object
Input data TxPDO	0x1A00 + 0x04 · <Module numbers>	0x6000 + 0x10 · <Module numbers>
Module status TxPDO	0x1A01 + 0x04 · <Module numbers>	0xA000 + 0x10 · <Module numbers>
Channel status TxPDO	0x1A02 + 0x04 · <Module numbers>	0xA001 + 0x10 · <Module numbers>

All of the data elements contained in the respective reference object are always represented in the various TxPDOs in the same arrangement. That means that a corresponding reference data entry referring to this subindex exists in the TxPDO mapping objects for each subindex present in the respective reference object.

When necessary, an additional padding entry is present, with which it can be ensured through the attachment of a certain number of padding bits that the size of the TxPDO corresponds to a multiple of a byte. This ensures that the TxPDOs strung together free of gaps in the process output image of the bus coupler are always oriented to byte limits.

The TxPDO mapping objects and their content are automatically generated by the bus coupler on the basis of the input data, module status, and channel status objects available for the *configured modules*. They are only visible in the online object directory of the bus coupler and allow reading, but not writing access. This means that the TxPDO configuration of the bus coupler cannot be changed manually.

9.8.3.1 General structure of the TxPDO

TxPDO mapping object for input data, module status, or channel status	
Index range	with reference to input data: 0x1A04 - 0x1B00 with reference to module status: 0x1A05 - 0x1B01 with reference to channel status: 0x1A06 - 0x1B02
Index calculation	with reference to input data: 0x1A00 + 0x04 · <Module numbers> with reference to module status: 0x1A01 + 0x04 · <Module numbers> with reference to channel status: 0x1A02 + 0x04 · <Module numbers>
Online designation	with reference to input data: "TxPDO (Modules <Module numbers> - Inputs)" with reference to module status: "TxPDO (Modules <Module numbers> - Module Status)" with reference to channel status: "TxPDO (Modules <Module numbers> - Channel Status)"
Short description	Representation of the data of the respective reference object on a TxPDO
Data type	RECORD
Bit size	16 + 32 · *n*
Access via complete access	possible
Subindex 0	
Short description	Number *n* of subsequent subindices
Online designation	“SubIndex 000”
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	• corresponds to the value of subindex 0 of the reference object, and thus to the number of the individual data elements contained in this object. • Increases by 1 in the event that a padding entry exists
…	
Subindex <*s*>	
Short description	Reference data entry <*e*> with *e* = *s* - 1, which refers to the subindex <*s*> in the reference object (enumeration starts with 0 in subindex 1)
Online designation	"Mapping entry <*e*>"
Data type (bit size/bit offset)	UDINT (32 / 16 + 32 · *e*)
Access	only reading
Online value	with reference to input data: 0x6*MN*0:<s>:<Bit size of the subindex in the reference object> with reference to the module status: 0xA*MN*0:<s>:<Bit size of the subindex in the reference object> with reference to channel status: 0xA*MN*1:<s>:<Bit size of the subindex in the reference object> (*MN* stands for the two-digit, hexadecimal module number)
…	

9.8.3.2 Example of the content of the TxPDO mapping objects

In this example, the analog input module 'AI 4 x TC, Iso., 16 Bit' (600-254-4AD02) is used as module 3. The module possesses input data containing the analog values for the 4 channels.

The bus coupler makes available the input data object 0x6030, the module status object 0xA030 and, due to the diagnosis-capability, the channel status object 0xA031 for this module. The following demonstrates how the content of these module objects are represented on TxPDOs via corresponding TxPDO mapping objects.

Input data object 0x6030				TxPDO mapping object 0x1A0C		
SI	Data type (bit size)	Description		SI	Online value	Description
0	USINT (8)	Subindex 0 (= 4)	→	0	4	Subindex 0
1	INT (16)	Analog value for channel 0	→	1	0x6030:01:16	Reference to 0x6030:01 with 16 bits
2	INT (16)	Analog value for channel 1	→	2	0x6030:02:16	Reference to 0x6030:02 with 16 bits
3	INT (16)	Analog value for channel 2	→	3	0x6030:3:16 AM	Reference to 0x6030:03 with 16 bits
4	INT (16)	Analog value for channel 3	→	4	0x6030:4:16 AM	Reference to 0x6030:04 with 16 bits

Module status object 0xA030				TxPDO mapping object 0x1A0D		
SI	Data type (bit size)	Description		SI	Online value	Description
0	USINT (8)	Subindex 0 (= 4)	→	0	2 + 1	Subindex 0
1	BOOL (1)	Present	→	1	0xA030:01:01	Reference to 0xA030:01 with 1 bit
2	BOOL (1)	Operational	→	2	0xA030:02:01	Reference to 0xA030:02 with 1 bit
				3	0x0000:00:06	Padding entry with 6 bits

Channel status object 0xA031				TxPDO mapping object 0x1A0E		
SI	Data type (bit size)	Description		SI	Online value	Description
0	USINT (8)	Subindex 0 (= 4)	→	0	4 + 1	Subindex 0
1	BOOL (1)	Channel 0 OK	→	1	0xA031:01:01	Reference to 0xA031:01 with 1 bit
2	BOOL (1)	Channel 1 OK	→	2	0xA031:2:01 AM	Reference to 0xA031:02 with 1 bit
3	BOOL (1)	Channel 2 OK	→	3	0xA031:3:01 AM	Reference to 0xA031:03 with 1 bit
4	BOOL (1)	Channel 3 OK	→	4	0xA031:4:01 AM	Reference to 0xA031:04 with 1 bit
				5	0x0000:12:04 AM	Padding entry with 4 bits

One additional padding entry each is present in the TxPDO mapping objects with reference to the module and the channel status. This ensures that the size of the TxPDOs increases to 8 bits, meaning 1 byte.

In chapter 9.8.2.2, a similar representation for the content of the RxPDO mapping object with regard to the output data is contained for the module chosen as an example.

9.8.4 The PDO assignment objects

The PDO assignment objects are responsible for the assignment of the available PDOs for Sync Managers 2 and 3, and thus for the compilation of the process data of the bus coupler.

The RECORD object 0x1C12 (Sync Manager 2 RxPDO Assignment) contains the information in its subindices as to which RxPDOs are assigned to the Sync Manager 2 responsible for the process output data communication, and are thus contained in the process output data of the bus coupler.

The RECORD object 0x1C13 (Sync Manager 3 TxPDO Assignment) contains the information in its subindices as to which TxPDOs are assigned to the Sync Manager 3 responsible for the process input data communication, and are thus contained in the process output data of the bus coupler.

The respective index of the related PDO mapping object is used in the subindices of the PDO assignment objects as a reference to a certain PDO.

The content of the two PDO assignment objects is generated automatically and in such a way that all available Rx and TxPDOs are contained in the process output or process input data of the bus coupler.

The sequence of the PDOs within the process data is prescribed by the sequence of the modules. With regard to the various TxPDOs contained in the process input data of the bus coupler for a single module, the following sequence applies:

- first the input data TxPDO of the module, where present
- then module status TxPDO of the module
- finally the channel status TxPDO of the module, where present

The automatically executed PDO assignment cannot be retroactively changed by the user. Only reading access to the PDO assignment objects is therefore possible.

9.8.4.1 General structure of the object 0x1C12

Index 0x1C12	**Sync Manager 2 RxPDO Assignment**
Short description	Assignment of the RxPDOs to Sync Manager 2
Data type (bit size)	ARRAY OF UINT (1040)
Additional details	All available RxPDOs with the output data of the *configured modules* are assigned to Sync Manager 2. This assignment takes place automatically and with ascending module numbers.
Access via complete access	possible
Subindex 0	Subindex 000
Short description	Number of subsequent subindices = Number of assigned RxPDOs
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	e.g. 8
Subindex 1	Subindex 001
Short description	Index of the 1st assigned RxPDOs
Data type (bit size/bit offset)	UINT (16 / 16)
Access	only reading
Online value	e.g. 0x1601 = RxPDO with output data of module 1
Subindex 2	SubIndex 002
Short description	Index of the 2nd assigned RxPDOs
Data type (bit size/bit offset)	UINT (16 / 32)
Access	only reading
Online value	e.g. 0x1604 = RxPDO with output data of module 4
…	…
Subindex 8	SubIndex 008
Short description	Index of the 8th assigned RxPDOs
Data type (bit size/bit offset)	UINT (16 / 128)
Access	only reading
Online value	e.g. 0x160C = RxPDO with output data of module 12

9.8.4.2 General structure of the object 0x1C13

Index 0x1C13	Sync Manager 3 TxPDO Assignment
Short description	Assignment of the TxPDOs to Sync Manager 3
Data type (bit size)	ARRAY OF UINT (3088)
Access via complete access	possible
Additional details	All available TxPDOs of the *configured modules* are automatically assigned to Sync Manager 3. This assignment takes place with ascending module numbers and referring to the individual module as follows: • TxPDO with the input data (where present) • TxPDO with the module status (always present) • TxPDO with the channel status (where present)
Subindex 0	**SubIndex 000**
Short description	Number of subsequent subindices = Number of assigned TxPDOs
Data type (bit size/bit offset)	USINT (8 / 0)
Access	only reading
Online value	e.g. 22
Subindex 1	**SubIndex 001**
Short description	Index of the 1st assigned TxPDO
Data type (bit size/bit offset)	UINT (16 / 16)
Access	only reading
Online value	e.g. 0x1A5 = TxPDO with module status of module 1
Subindex 2	**SubIndex 002**
Short description	Index of the 2nd assigned TxPDO
Data type (bit size/bit offset)	UINT (16 / 32)
Access	only reading
Online value	e.g. 0x1A08 = TxPDO with input data of module 2
Subindex 3	**SubIndex 003**
Short description	Index of the 3rd assigned TxPDO
Data type (bit size/bit offset)	UINT (16 / 48)
Access	only reading
Online value	e.g. 0x1A09 = TxPDO with module status of module 2
Subindex 4	**SubIndex 004**
Short description	Index of the 4th assigned TxPDO
Data type (bit size/bit offset)	UINT (16 / 64)
Access	only reading
Online value	e.g. 0x1A0C = TxPDO with input data of module 3
Subindex 5	**SubIndex 005**
Short description	Index of the 5th assigned TxPDO
Data type (bit size/bit offset)	UINT (16 / 80)
Access	only reading
Online value	e.g. 0x1A0D = TxPDO with module status of module 3
Subindex 6	**SubIndex 006**
Short description	Index of the 6th assigned TxPDO
Data type (bit size/bit offset)	UINT (16 / 96)
Access	only reading
Online value	e.g. 0x1A0E = TxPDO with channel status of module 3
…	…

9.8.4.3 Example of a PDO assignment and resulting process data

#	Module	Output data	Input data	Submodule status	Channel status
1	**16 DI** 600-210-0AP21	-	0x6010 → 0x1A04 (2 bytes)	0xA010 → 0x1A05 (1 bytes)	-
2	**8 DO** 600-220-0AH01	0x7020 → 0x1602 (1 bytes)	-	0xA020 → 0x1A09 (1 bytes)	-
3	**1 CNT 24V** 600-300-7AA01	0x7030 → 0x1603 (8 bytes)	0x6040 → 0x1A0C (8 bytes)	0xA030 → 0x1A0D (1 bytes)	0xA031 → 0x1A0E (1 bytes)
4	**PWR 24V** 600-710-0AA01	-	-	0xA040 → 0x1A11 (1 bytes)	-
5	**2 AO U** 600-261-4AB01	0x7050 → 0x1605 (4 bytes)	-	0xA050 → 0x1A15 (1 bytes)	0xA051 → 0x1A16 (1 bytes)
6	**8 AI U** 600-252-4AD01	-	0x6060 → 0x1A18 (16 bytes)	0xA060 → 0x1A19 (1 bytes)	0xA061 → 0x1A1A (1 bytes)

Object 0x1C12 (Sync Manager 2 RxPDO Assignment)				**Output process data of the bus coupler**		
SI	**Value**	**Description**		**Byte**	**Description**	**Size**
0	3	Number of subsequent subindices				
1	0x1602	Output data RxPDO from module 2	→	**0**	Output data from module 2	1 bytes
2	0x1603	Output data RxPDO from module 3	→	**1 - 8**	Output data from module 3	8 bytes
3	0x1605	Output data RxPDO from module 5	→	**9 - 12**	Output data from module 5	4 bytes

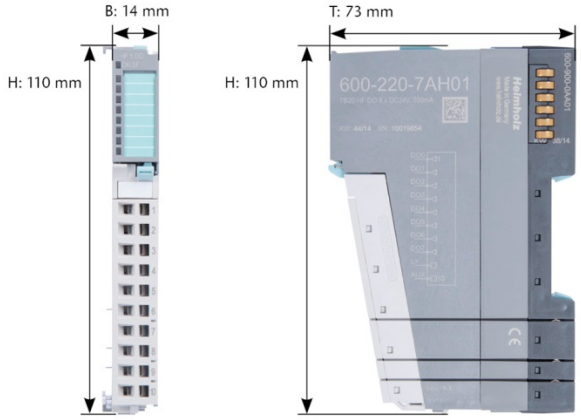
Object 0x1C13 (Sync Manager 3 TxPDO Assignment)				**Input process data of the bus coupler**		
SI	**Value**	**Description**		**Byte**	**Description**	**Size**
0	10	Number of subsequent subindices				
1	0x1A04	Input data TxPDO from module 1	→	**0 - 1**	Input data from module 1	2 bytes
2	0x1A05	Module status TxPDO from module 1	→	**2**	Module status from module 1	1 bytes
3	0x1A09	Module status TxPDO from module 2	→	**3**	Module status from module 2	1 bytes
4	0x1A0C	Input data TxPDO from module 3	→	**4 - 11**	Input data from module 3	8 bytes
5	0x1A0D	Module status TxPDO from module 3	→	**12**	Module status from module 3	1 bytes
6	0x1A0E	Channel status TxPDO from module 3	→	**13**	Channel status from module 3	1 bytes
7	0x1A11	Module status TxPDO from module 4	→	**14**	Module status from module 4	1 bytes
8	0x1A15	Module status TxPDO from module 5	→	**15**	Module status from module 5	1 bytes
9	0x1A16	Channel status TxPDO from module 5	→	**16**	Channel status from module 5	1 bytes
10	0x1A18	Input data TxPDO from module 6	→	**17 - 32**	Input data from module 6	16 bytes
11	0x1A19	Module status TxPDO from module 6	→	**33**	Module status from module 6	1 bytes
12	0x1A1A	Channel status TxPDO from module 6	→	**34**	Channel status from module 6	1 bytes

10 Technical data

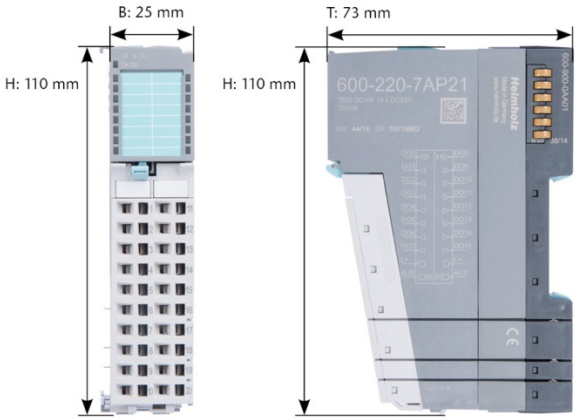
Order number	**600-185-1AA11**
Article designation	TB20 EtherCAT® bus coupler
EtherCAT interface	
Connection	Two RJ45
Protocol	EtherCAT according to IEC standard 61158
Physical layer	Ethernet, 100BASE-TX
Transmission rate	100 Mbps full duplex
I/O image size	max. 1024 bytes
Features	Automatic process data configuration CANopen over EtherCAT (CoE) Object directory according to Modular Device Profile (MDP) SDO Info, Complete Access
Diagnosis functions	Cycle-current module and channel status Diagnostic status in the object directory
USB port	
Protocol	Full-speed USB 1.1 device
Connection	Mini-USB
Isolation voltage	1.5 kV
Electrical isolation	Yes
Number of modules that can be connected in series	max. 64, all variants
Voltage supply	24 V DC; 18 … 28 V DC
Current consumption without modules	75 mA
Power dissipation	Max. 8 W
Power supply for modules	5 V DC; max. 2.5 A
Electrically separated from the backplane bus	No
Dimensions (H x W x D)	110 mm x 35 mm x 73 mm
Weight	115 g
Installation position	Any
Certification	CE, UL
Noise immunity	DIN EN 61000-6-2 “EMC Immunity”
Interference emission	DIN EN 61000-6-4 “EMC Emission”
Vibration and shock resistance	DIN EN 60068-2-8:2008 “Vibration” DIN EN 60068-27:2010 “Shock”
Protection rating	IP20
Relative air humidity	95% without condensation
Permissible ambient temperature	0 °C ... 60 °C for UL applications: 0 – 50 °C
Transport and storage temperature	-20 – 80 °C
Pollution degree	2

11 Dimensions

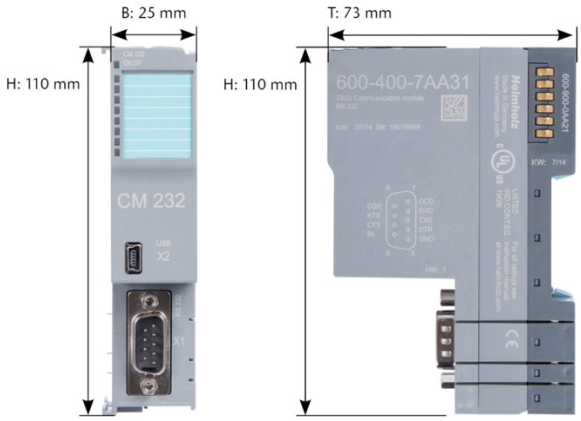
Einfachbreites Modul



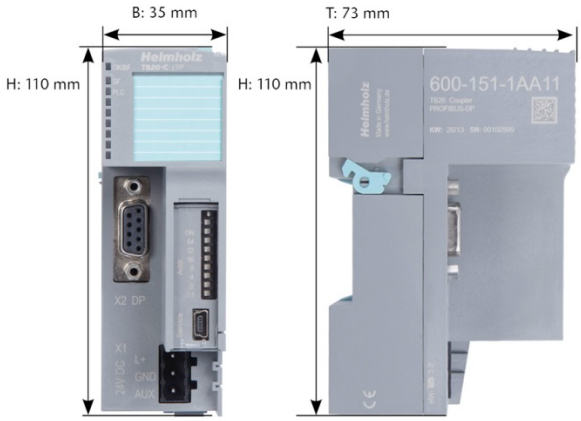
Doppelbreites Modul

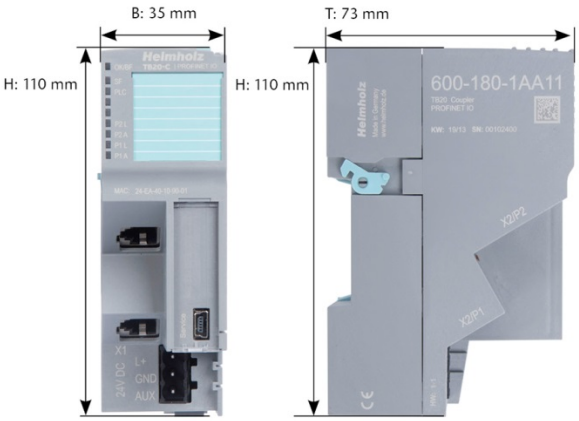


Kommunikations-Modul



Buskoppler





12 Spare parts

12.1 Base modules

12.1.1 14 mm width standard base module

The 14 mm standard base module is available in sets of five with order no. 600-900-9AA01.

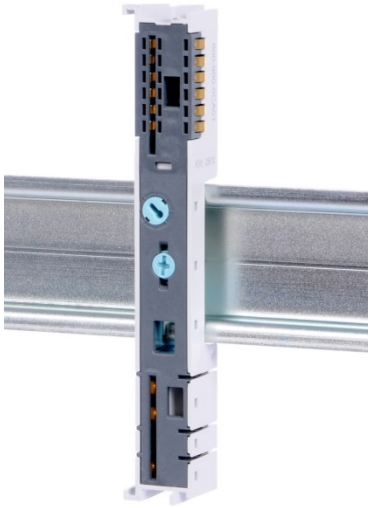
12.1.2 25 mm width base module

The 25 mm standard base module is available in sets of five with order no. 600-900-9AA21.

12.1.3 Power and isolation base module

The power and isolation base module is available in sets of five with order no. 600-900-9BA01.

12.1.4 Power base module

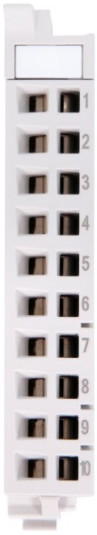


The power base module is available in sets of five with order no. 600-900-9CA01.

It can be used with the power module (600-700-0AA01) and with all bus couplers.

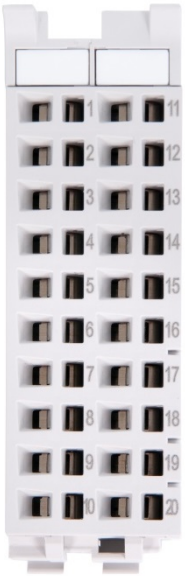
12.2 Front connectors

12.2.1 Front connector, 10-pin



The 10-terminal front connector is available in sets of five with order no. 600-910-9AJ01.

12.2.2 Front connector, 20-pin



The 20-pin front connector is available in sets of five with order no. 600-910-9AT21.

12.3 Electronic modules

Electronic modules can be ordered as spare parts with the order number of the original product. Electronic modules are always sent as a complete assembly, including the corresponding base module and front connector.

12.4 Final cover

The final cover is available in sets of five with order no. 600-920-9AA01.