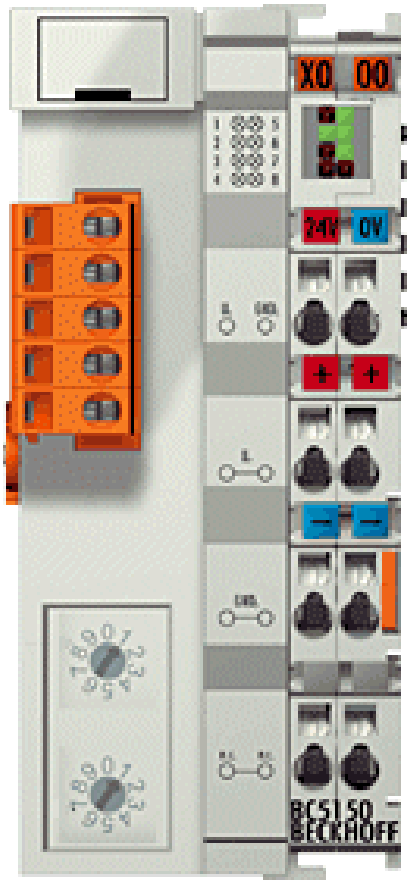



Documentation

# BC5150

## Bus Terminal Controller for CANopen

**Version:** 2.0.0
**Date:** 2017-07-17

BECKHOFF

# Table of contents

# 1 Foreword

## 1.1 Notes on the documentation

**Intended audience**

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning these components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

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

**Disclaimer**

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

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

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

**Trademarks**

Beckhoff®, TwinCAT®, EtherCAT®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

**Patent Pending**

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents: EP1590927, EP1789857, DE102004044764, DE102007017835 with corresponding applications or registrations in various other countries.

The TwinCAT Technology is covered, including but not limited to the following patent applications and patents: EP0851348, US6167425 with corresponding applications or registrations in various other countries.


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

**Copyright**

## 1.2 Safety instructions

### Safety regulations

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

### Exclusion of liability

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

### Personnel qualification

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

### Description of instructions

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

**⚠ DANGER**

**Serious risk of injury!**

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

**⚠ WARNING**

**Risk of injury!**

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

**⚠ CAUTION**

**Personal injuries!**

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

***NOTE***

**Damage to environment/equipment or data loss**

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

**Tip or pointer**

This symbol indicates information that contributes to better understanding.

## 1.3 Documentation revision levels

| Version | Comment |
|---|---|
| 2.0.0 | • Migration |
| 1.1.0 | • Notes to meet the UL requirements added. |
| 1.0.1 | • Chapter *Creating a boot project* corrected<br>• Minor routine corrections (typing errors, spelling, etc.)<br>• English version available |
| 1.0.0 | • First public issue (only in German) |

**Firmware BC5150**

For updating your firmware you need a serial cable, the KS2000 configuration software, or the firmware update program.

| Firmware | Comment |
|---|---|
| 0xB8 | Size of the Flash area from the tsm files increased |
| 0xB7 | Support 16 PDOs |
| 0xB6 | Internal version |
| 0xB5 | Support of persistent data |
| 0xB4 | Boot project creation optimized |
| 0xB3 | Fix for string to real conversion |
| 0xB2 | Optimization of terminal verification |
| 0xB1 | Internal version |
| 0xB0 | Firmware version 0xB0 |

The firmware and hardware versions (delivery state) can be found on the sticker on the underside of the Bus Terminal Controller.

# 2 Product overview

## 2.1 BCxx50 Overview

Bus Terminal Controllers are Bus Couplers with integrated PLC functionality. The BCxx50 Bus Terminal Controllers have a fieldbus interface, are intelligent slaves and can be used as decentralized intelligence within the system. They are located in a cost-optimized and compact housing. In contrast to the BCxx00 range, the BCxx50 range supports up to 255 Bus Terminals via the K-Bus extension.

The Bus Terminal Controller is programmed using the TwinCAT programming system according to IEC 61131-3. The BCxx50 configuration/programming interface is used for loading the PLC program. If the TwinCAT software PLC is in use, the PLC program can also be loaded via the fieldbus.

The inputs and outputs of the connected Bus Terminals are assigned in the default setting of the mini-PLC. Each individual Bus Terminal can be configured in such a way that it exchanges data directly through the fieldbus with the higher-level automation device. Similarly, pre-processed data can be exchanged between the Bus Terminal Controller and the higher-level controller via the fieldbus.

**Fieldbus interface**

The variants of the BCxx50 series Bus Terminal Controllers differ in terms of their fieldbus interfaces. Various versions cover the most important fieldbus systems:

- BC3150: PROFIBUS DP
- BC5150: CANopen
- BC5250: DeviceNet
- BC8050: RS485, various protocols
- BC8150: RS232, various protocols

**Programming**

The BCxx50 devices are programmed according to the powerful IEC 61131-3 standard. Like for all other BECKHOFF controllers, the TwinCAT automation software is the basis for parameterization and programming. Users therefore have the familiar TwinCAT tools available, e.g. PLC programming interface, System Manager and TwinCAT Scope. Data is exchanged optionally via the serial interface (COM1) or via the fieldbus through Beckhoff PC FCxxxx fieldbus cards.

**Configuration**

The configuration is also carried out using TwinCAT. The fieldbus interface can be configured and parameterized via the System Manager. The System Manager can read all connected devices and Bus Terminals. After the parameterization, the configuration is saved on the BCxx50 via the serial interface. The configuration thus created can be accessed again later.

## 2.2 The principle of the Bus Terminal

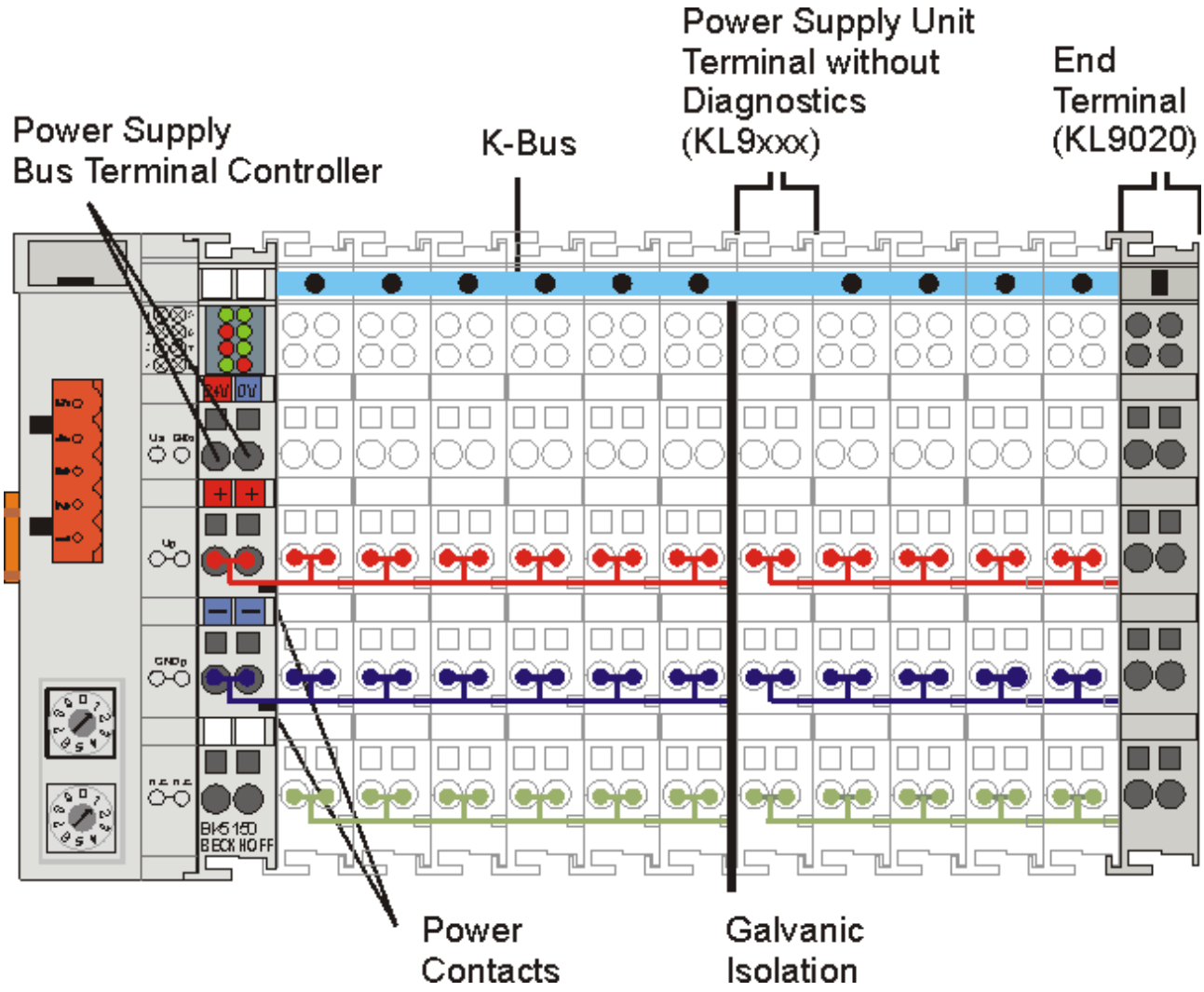



Fig. 1: The principle of the Bus Terminal

## 2.3 The Beckhoff Bus Terminal system

**Up to 256 Bus Terminals, with 1 to 16 I/O channels per signal form**

The Bus Terminal system is the universal interface between a fieldbus system and the sensor / actuator level. A unit consists of a Bus Coupler as the head station, and up to 64 electronic series terminals, the last one being an end terminal. Up to 255 Bus Terminals can be connected via the K-bus extension. For each technical signal form, terminals are available with one, two, four or eight I/O channels, which can be mixed as required. All the terminal types have the same mechanical construction, so that difficulties of planning and design are minimized. The height and depth match the dimensions of compact terminal boxes.

**Decentralised wiring of each I/O level**

Fieldbus technology allows more compact forms of controller to be used. The I/O level does not have to be brought to the controller. The sensors and actuators can be wired decentrally, using minimum cable lengths. The controller can be installed at any location within the plant.

**Industrial PCs as controllers**

The use of an Industrial PC as the controller means that the operating and observing element can be implemented in the controller's hardware. The controller can therefore be located at an operating panel, in a control room, or at some similar place. The Bus Terminals form the decentralised input/output level of the

controller in the control cabinet and the subsidiary terminal boxes. The power sector of the plant is also controlled over the bus system in addition to the sensor/actuator level. The Bus Terminal replaces the conventional series terminal as the wiring level in the control cabinet. The control cabinet can have smaller dimensions.

### Bus Couplers for all usual bus systems

The Beckhoff Bus Terminal system unites the advantages of a bus system with the possibilities of the compact series terminal. Bus Terminals can be driven within all the usual bus systems, thus reducing the controller parts count. The Bus Terminals then behave like conventional connections for that bus system. All the performance features of the particular bus system are supported.

### Mounting on standardized mounting rails

The installation is standardized thanks to the simple and space-saving mounting on a standardized mounting rail (EN 60715, 35 mm) and the direct wiring of actuators and sensors, without cross connections between the terminals. The consistent labelling scheme also contributes.

The small physical size and the great flexibility of the Bus Terminal system allow it to be used wherever a series terminal is also used. Every type of connection, such as analog, digital, serial or the direct connection of sensors can be implemented.

### Modularity

The modular assembly of the terminal strip with Bus Terminals of various functions limits the number of unused channels to a maximum of one per function. The presence of two channels in one terminal is the optimum compromise of unused channels and the cost of each channel. The possibility of electrical isolation through potential feed terminals also helps to keep the number of unused channels low.

### Display of the channel state

The integrated LEDs show the state of the channel at a location close to the sensors and actuators.

### K-bus

The K-bus is the data path within a terminal strip. The K-bus is led through from the Bus Coupler through all the terminals via six contacts on the terminals' side walls. The end terminal terminates the K-bus. The user does not have to learn anything about the function of the K-bus or about the internal workings of the terminals and the Bus Coupler. Many software tools that can be supplied make project planning, configuration and operation easy.

### Potential feed terminals for isolated groups

The operating voltage is passed on to following terminals via three power contacts. You can divide the terminal strip into arbitrary isolated groups by means of potential feed terminals. The potential feed terminals play no part in the control of the terminals, and can be inserted at any locations within the terminal strip.

Up to 64 Bus Terminals can be used in a terminal block, with optional K-bus extension for up to 256 Bus Terminals. This count does include potential feed terminals, but not the end terminal.

### Bus Couplers for various fieldbus systems

Various Bus Couplers can be used to couple the electronic terminal strip quickly and easily to different fieldbus systems. It is also possible to convert to another fieldbus system at a later time. The Bus Coupler performs all the monitoring and control tasks that are necessary for operation of the connected Bus Terminals. The operation and configuration of the Bus Terminals is carried out exclusively by the Bus Coupler. Nevertheless, the parameters that have been set are stored in each Bus Terminal, and are retained in the event of voltage drop-out. Fieldbus, K-bus and I/O level are electrically isolated.

If the exchange of data over the fieldbus is prone to errors or fails for a period of time, register contents (such as counter states) are retained, digital outputs are cleared, and analog outputs take a value that can be configured for each output when commissioning. The default setting for analog outputs is 0 V or 0 mA. Digital

outputs return in the inactive state. The timeout periods for the Bus Couplers correspond to the usual settings for the fieldbus system. When converting to a different bus system it is necessary to bear in mind the need to change the timeout periods if the bus cycle time is longer.

**The interfaces**

A Bus Coupler has six different methods of connection. These interfaces are designed as plug connectors and as spring-loaded terminals.

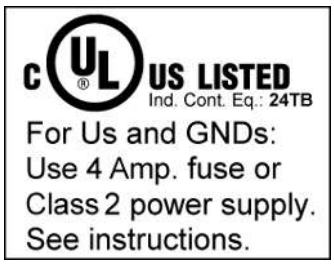
# 2.4 Technical data

## 2.4.1 Technical Data - BCxx50

| Technical data | BCxx5x |
|---|---|
| Processor | 16 bit micro-controller |
| Diagnostic LEDs | 2 x power supply, 2 x K-Bus |
| Configuration and programming software | TwinCAT PLC |

| Fieldbus interface | BC5150 |
|---|---|
| Fieldbus | CANopen |

| Interfaces | |
|---|---|
| Serial interface | COM1 (RS232 for configuration and programming, automatic baud rate detection 9600/19200/38400 baud) |
| Terminal Bus (K-Bus) | 64 (255 with K-bus extension) |

| Technical data | BC5150 |
|---|---|
| Digital peripheral signals | 2040 inputs/outputs |
| Analog peripheral signals | 1024 inputs/outputs |
| Configuration possibility | via TwinCAT or the controller |
| Maximum fieldbus byte number | 16 Tx/Rx PDOs |
| Maximum number of bytes - PLC | 2048 bytes of input data, 2048 bytes of output data |
| Bus connection | Open Style Connector, 5-pin |
| Power supply (Us)<br>  | 24 $V_{DC}$ (-15% /+20%) Use a 4 A fuse or an *NEC Class 2* power supply to meet the UL requirements! |
| Input current (Us) | 60 mA + (total K-bus current)/4 |
| Starting current | approx. 2.5 x continuous current |
| K-bus current (5 V) | maximum 1000 mA |
| Power contact voltage (Up) | maximum 24 $V_{DC}$ |
| Power contact current load (Up) | maximum 10 A |
| Recomm. back-up fuse (Up) | ≤10 A |
| Dielectric strength | 500 V (power contact/supply voltage/fieldbus) |
| Weight | approx. 100 g |
| Dimensions (W x H x D) | approx. 44 mm x 100 mm x 68 mm |
| Permissible ambient temperature range during operation | -25 °C ... +60 °C |
| Permissible ambient temperature range during storage | -40 °C ... +85 °C |
| Relative humidity | 95 % no condensation |
| Vibration / shock resistance | conforms to EN 60068-2-6 / EN 60068-2-27 |
| EMC immunity / emission | conforms to EN 61000-6-2 / EN 61000-6-4 |
| Installation position | variable |
| Protection class | IP20 |
| Approvals | CE, cULus, ATEX |

## 2.4.2 Technical Data - CANopen

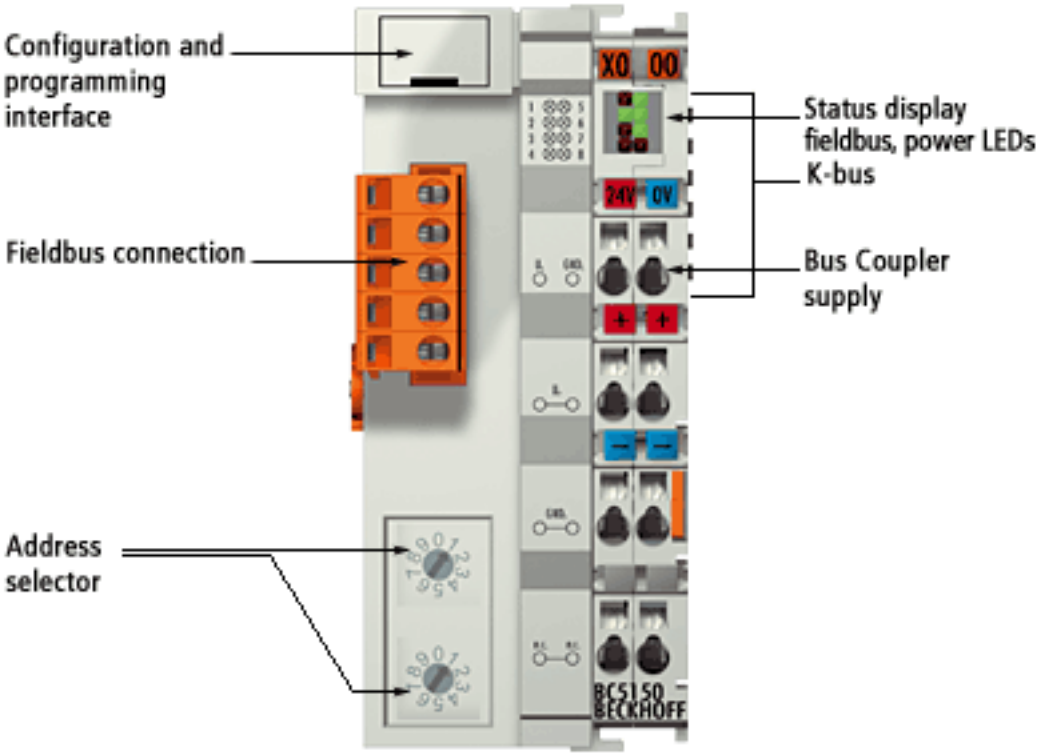



Fig. 2: BC5150

| System data | CANopen (BC5150) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Number of nodes | 63 (with repeater 99) | | | | | | | |
| Number of I/O points | depending on controller | | | | | | | |
| Data transfer medium | shielded, twisted copper cable, 2 x signal, 1 x CAN ground (recommended) | | | | | | | |
| Cable length | 5000 m | 2500 m | 1000 m | 500 m | 250 m | 100 m | 50 m | 25 m |
| Data transfer rate | 10 kbaud | 20 kbaud | 50 kbaud | 125 kbaud | 250 kbaud | 500 kbaud | 800 kbaud | 1 Mbaud |
| I/O communication types | event driven, cyclic, synchronous, polling | | | | | | | |
| Number of PDOs | 16 send process data objects and 16 receive process data objects | | | | | | | |
| Baud rate | Automatic baud rate detection (for possible baud rates see data transfer rate) | | | | | | | |

### 2.4.3 Technical Data - PLC

| PLC data | BCxx5x |
|---|---|
| Programmability | via serial programming interface or via the fieldbus |
| Program memory | 48 kByte |
| Source code memory | 128 kByte |
| Data memory | 32 kByte |
| Remanent flags | 2 kByte |
| PLC cycle time | Approx. 3.0 ms for 1000 IL commands (without I/O cycle) |
| Programming languages | IEC 6-1131-3 (IL, LD, FBD, ST, SFC) |
| Runtime | 1 SPS Task |
| Online Change | Yes |
| Up/Down Load Code | Yes/Yes |

# 3 Mounting and wiring

## 3.1 Mounting

### 3.1.1 Dimensions

The Beckhoff Bus Terminal system is characterized by low physical volume and high modularity. When planning a project it must be assumed that at least one Bus Coupler and a number of Bus Terminals will be used. The mechanical dimensions of the Bus Couplers are independent of the fieldbus system.

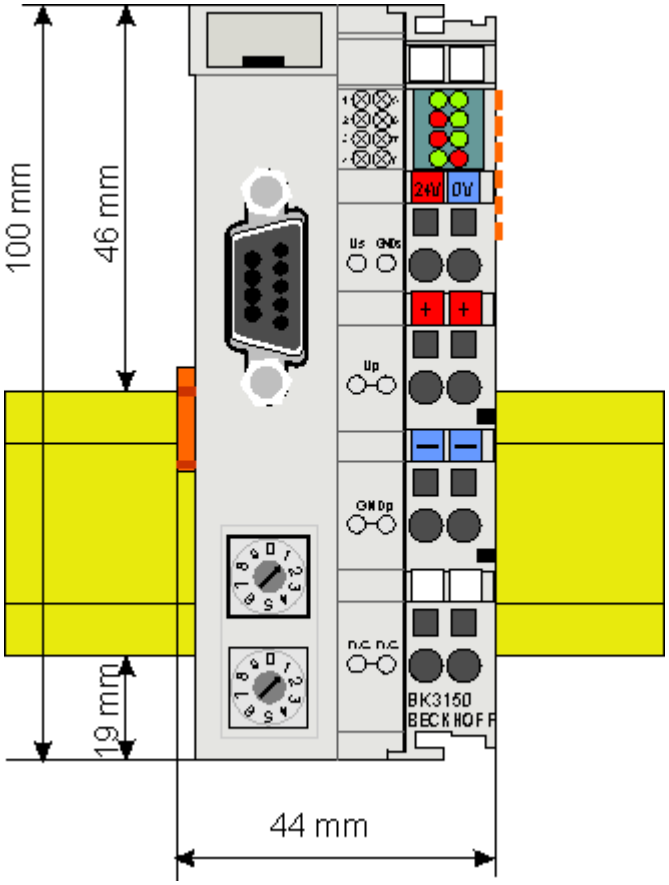



Fig. 3: BCxx50

The total width of the fieldbus station is the width of the Bus Coupler/Bus Terminal Controller plus the width of the Bus Terminals being used (incl. KL9010 bus end terminal). Depending on design, the Bus Terminals are 12 mm or 24 mm wide. The height is 100 mm.

The BCxx50 series Bus Terminal Controllers are 68 mm deep.

### 3.1.2 Mounting

The Bus Coupler and all the Bus Terminals can be clipped, with a light press, onto a 35 mm mounting rail. A locking mechanism prevents the individual housings from being pulled off again. For removal from the mounting rail the orange colored tension strap releases the latching mechanism, allowing the housing to be pulled off the rail without any force.

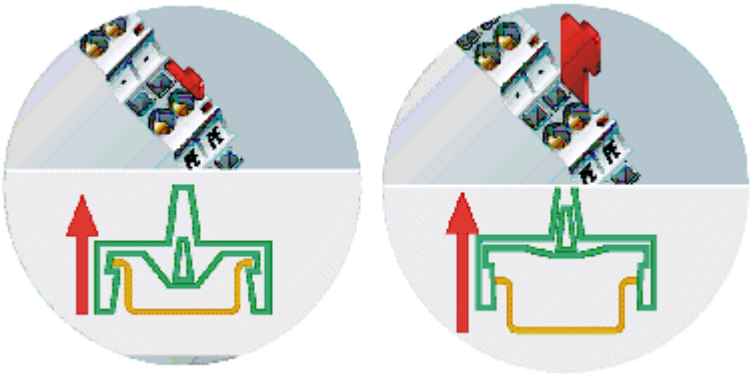


Fig. 4: Release the locking mechanism by pulling the orange tab

Up to 64 Bus Terminals can be attached to the Bus Coupler on the right hand side. When plugging the components together, be sure to assemble the housings with groove and tongue against each other. A properly working connection cannot be made by pushing the housings together on the mounting rail. When correctly assembled, no significant gap can be seen between the attached housings.

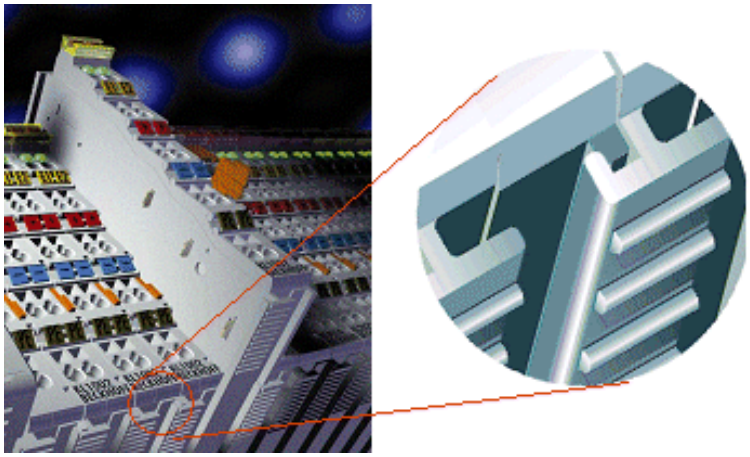


Fig. 5: Groove and tongue of the housings

**NOTE**

**Bus Terminals should only be pulled or plugged in switched-off state.**

Insertion and removal of Bus Terminals is only permitted when switched off. The electronics in the Bus Terminals and in the Bus Coupler are protected to a large measure against damage, but incorrect function and damage cannot be ruled out if they are plugged in under power.

# 3.2 Wiring

## 3.2.1 Potential groups, insulation testing and PE

### Potential groups

A Beckhoff Bus Terminal block usually has three different potential groups:

- The fieldbus interface is electrically isolated (except for individual Low Cost couplers) and forms the first potential group.
- Bus Coupler / Bus Terminal Controller logic, K-bus and terminal logic form a second electrically isolated potential group.
- The inputs and outputs are supplied via the power contacts and form further potential groups.

Groups of I/O terminals can be consolidated to further potential groups via potential supply terminals or separation terminals.

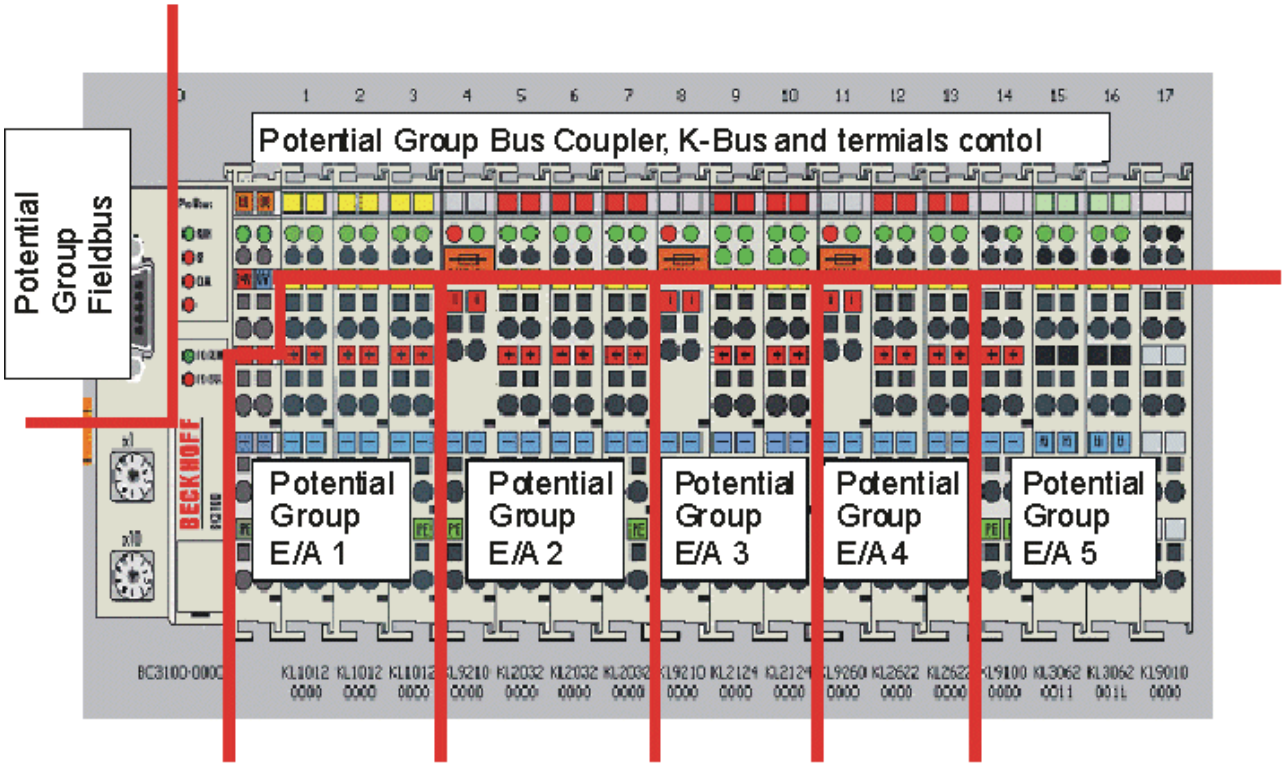



Fig. 6: Potential groups of a Bus Terminal block

### Insulation testing

The connection between Bus Coupler / Bus Terminal Controller and Bus Terminals is realized automatically by latching the components. The transfer of the data and the supply voltage for the intelligent electronics in the Bus Terminals is performed by the K-bus. The supply of the field electronics is performed through the power contacts. Plugging together the power contacts creates a supply rail. Since some Bus Terminals (e.g. analog Bus Terminals or 4-channel digital Bus Terminals) are not looped through these power contacts or not completely the Bus Terminal contact assignments must be considered.

The potential feed terminals interrupt the power contacts, and represent the start of a new supply rail. The Bus Coupler / Bus Terminal Controller can also be used for supplying the power contacts.

### PE power contacts

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

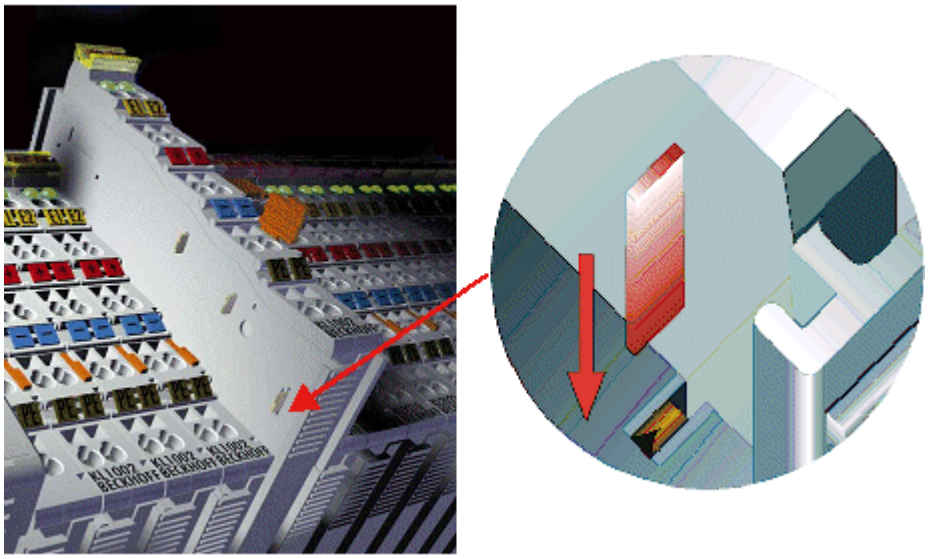


Fig. 7: Power contact on the left

It should be noted that, for reasons of electromagnetic compatibility, the PE contacts are capacitively coupled to the mounting rail. This can both lead to misleading results and to damaging the terminal during insulation testing (e.g. breakdown of the insulation from a 230 V power consuming device to the PE conductor). The PE supply line at the Bus Coupler / Bus Terminal Controller must be disconnected for an insulation test. In order to uncouple further feed locations for the purposes of testing, the feed terminals can be pulled at least 10 mm out from the connected group of other terminals. In that case, the PE conductors do not have to be disconnected.

The power contact with the label PE must not be used for other potentials.

## 3.2.2 Power supply

**⚠ DANGER**

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

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

### Supply of Bus Terminal Controller and Bus Terminals (Us)

The Bus Terminal Controller requires a supply voltage of 24 $V_{DC}$.

The connection is made by means of the upper spring-loaded terminals labelled *24 V* and *0 V*. This supply voltage is used for the electronic components of the Bus Coupler / Bus Terminal Controllers and (via the K-bus) the electronic components of the Bus Terminals. It is galvanically separated from the field level voltage.

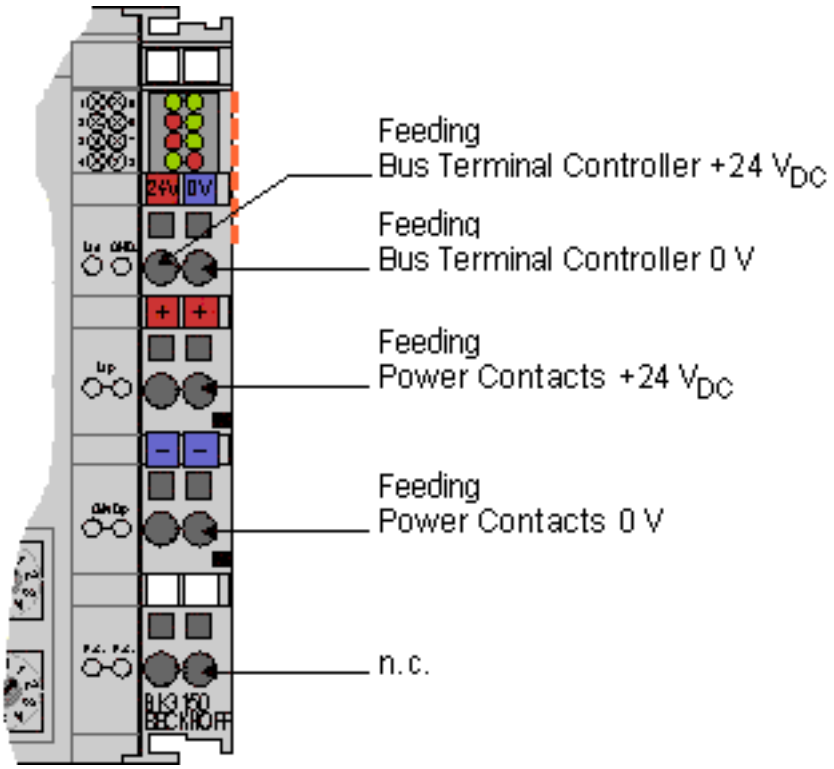



Fig. 8: Terminal points for the Bus Terminal Controller supply

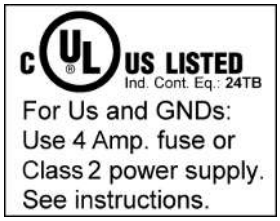



Fig. 9: UL identification

**⚠ DANGER**

**Note the UL requirements for the power supply.**

To comply with the UL requirements, the 24 $V_{DC}$ supply voltage for Us must originate

- from an isolated source protected by a fuse of max. 4 A (according to UL248) or
- from a voltage supply complying with *NEC class 2*.
  An *NEC class 2* voltage source must not be connected in series or parallel with another *NEC class 2* corresponding voltage source!

**⚠ DANGER**

**No unlimited voltage sources!**

To comply with the UL requirements, Us must not be connected with unlimited voltage sources.

### Power contacts supply (Up)

The bottom six connections with spring-loaded terminals can be used to feed the supply for the peripherals. The spring-loaded terminals are joined in pairs to a power contact. The feed for the power contacts has no connection to the voltage supply for the BC electronics.

The spring-loaded terminals are designed for wires with cross-sections between 0.08 mm$^2$ and 2,5 mm$^2$.

The assignment in pairs and the electrical connection between feed terminal contacts allows the connection wires to be looped through to various terminal points. The current load from the power contact must not exceed 10 A for long periods. The current carrying capacity between two spring-loaded terminals is identical to that of the connecting wires.

**Power contacts**

On the right hand face of the Bus Terminal Controller there are three spring contacts for the power contact connections. The spring contacts are hidden in slots so that they cannot be accidentally touched. By attaching a Bus Terminal the blade contacts on the left hand side of the Bus Terminal are connected to the spring contacts. The tongue and groove guides on the top and bottom of the Bus Terminal Controllers and of the Bus Terminals guarantees that the power contacts mate securely.

## 3.2.3 Programming cable

Use the KS2000-Z2 programming cable for serial programming of the Bus Terminal Controller. This cable is included in the scope of supply of the KS2000 software, or it can be ordered separately (order identifier KS2000-Z2).

Fig. 10: Programming cable KS2000-Z2

**KS2000-Z2**

The programming cable offers the option of programming the BCxx50 via the serial interface.

**NOTE**

**Do not interrupt the ground connection for the supply voltage**

When the programming cable (between BCxx50 and PC) is connected, the ground connection of the Bus Terminal controller must not be interrupted or disconnected, since this may destroy the programming cable.

## 3.2.4 CANopen cabling

Notes related to checking the CAN wiring can be found in the Trouble Shooting section.

### 3.2.4.1 CAN topology

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

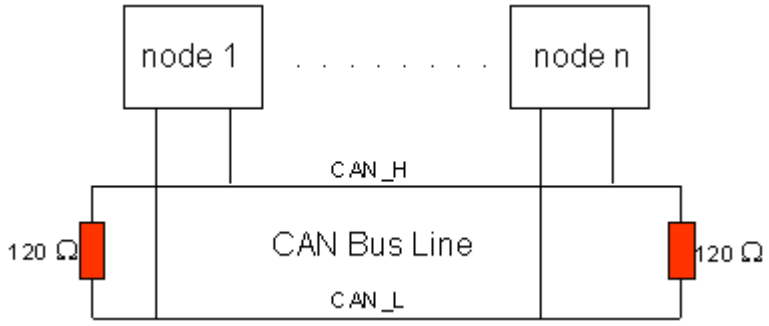



Fig. 11: Termination of the bus with a 120 Ohm termination resistor

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

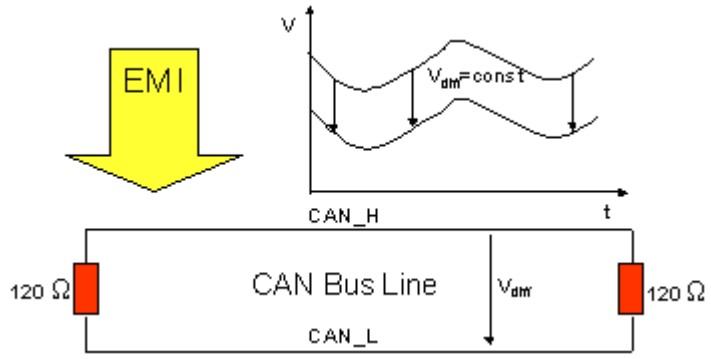



Fig. 12: Insensitivity to incoming interference

### 3.2.4.2 Bus length

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

| Baud rate | Bus length |
|---|---|
| 1 Mbit/s | < 20 m* |
| 500 kbit/s | < 100 m |
| 250 kbit/s | < 250 m |
| 125 kbit/s | < 500 m |
| 50 kbit/s | < 1000 m |
| 20 kbit/s | < 2500 m |
| 10 kbit/s | < 5000 m |

*) A figure of 40 m at 1 Mbit/s is often found in the CAN literature. This does not, however, apply to networks with optically isolated CAN controllers. The worst case calculation for opto-couplers yields a figure 5 m at 1 Mbit/s - in practice, however, 20 m can be reached without difficulty.

It may be necessary to use repeaters for bus lengths greater than 1000 m.

### 3.2.4.3 Drop lines

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

| Baud rate | Drop line length | Total length of all drop lines |
|---|---|---|
| 1 Mbit/s | < 1 m | < 5 m |
| 500 kbit/s | < 5 m | < 25 m |
| 250 kbit/s | < 10 m | < 50 m |
| 125 kbit/s | < 20 m | < 100 m |
| 50 kbit/s | < 50 m | < 250 m |

Drop lines must not have terminating resistors.

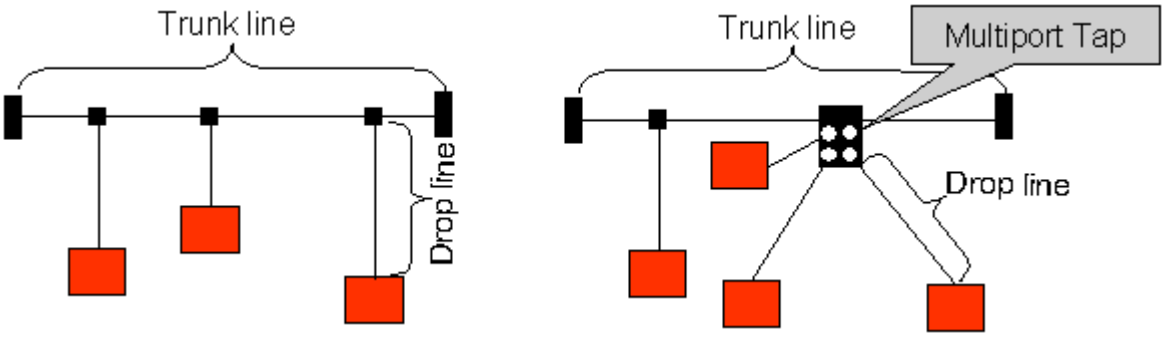



Fig. 13: Sample topology of drop lines

### 3.2.4.4 Star Hub (Multiport Tap)

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

| Baud rate | Drop line length with multiport topology | Trunk line length (without drop lines) |
|---|---|---|
| 1 Mbit/s | < 0,3 m | < 25 m |
| 500 kbit/s | < 1,2 m | < 66 m |
| 250 kbit/s | < 2,4 m | < 120 m |
| 125 kbit/s | < 4,8 m | < 310 m |

### 3.2.4.5 CAN cable

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

#### ZB5100 CAN Cable

A high quality CAN cable with the following properties is included in Beckhoff's range:

- 2 x 2 x 0.25 mm² (AWG 24) twisted pairs, cable colors: red/black + white/black
- double screened
- braided screen with filler strand (can be attached directly to pin 3 of the 5-pin connection terminal)
- flexible (minimum bending radius 35 mm when bent once, 70 mm for repeated bending)

- characteristic impedance (60 kHz): 120 ohm
- conductor resistance < 80 Ohm/km
- sheath: grey PVC, outside diameter 7.3 +/- 0.4 mm
- Weight: 64 kg/km.
- printed with "Beckhoff ZB5100 CAN-BUS 2x2x0.25" and meter marking (length data every 20cm)

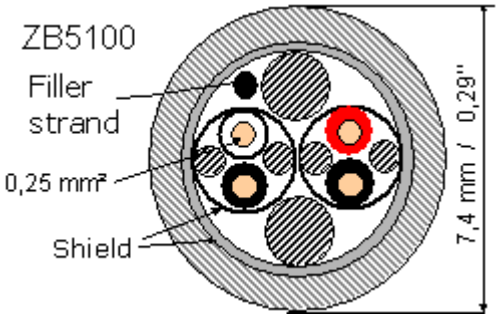



Fig. 14: Structure of CAN cable ZB5100

**ZB5200 CAN/DeviceNet Cable**

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

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

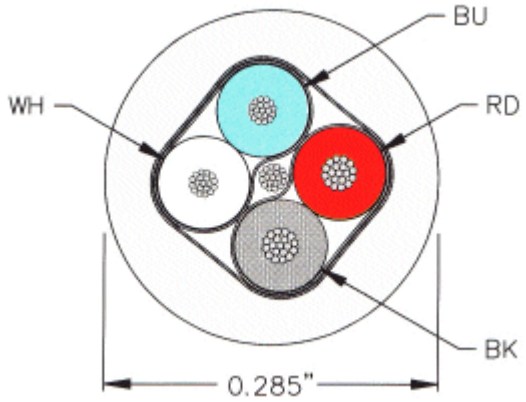



Fig. 15: Structure of CAN/DeviceNet cable ZB5200

### 3.2.4.6 Shielding

The screen is to be connected over the entire length of the bus cable, and only galvanically grounded at one point, in order to avoid ground loops.
The design of the screening, in which HF interference is diverted through R/C elements to the mounting rail

assumes that the rail is appropriately earthed and free from interference. If this is not the case, it is possible that HF interference will be transmitted from the mounting rail to the screen of the bus cable. In that case the screen should not be attached to the couplers - it should nevertheless still be fully connected through.

Notes related to checking the CAN wiring can be found in the Trouble Shooting section.

### 3.2.4.7 Cable colors

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

| BK51x0 pin PIN BX5100 (X510) | Pin BK5151 CX8050, CX8051, CXxxxx-B510/M510 | Fieldbus Box pin | Pin FC51xx | Function | ZB5100 cable color | ZB5200 cable color |
|---|---|---|---|---|---|---|
| 1 | 3 | 3 | 3 | CAN Ground | **black/** (red) | **black** |
| 2 | 2 | 5 | 2 | CAN Low | **black** | **blue** |
| 3 | 5 | 1 | 5 | Shield | Filler strand | Filler strand |
| 4 | 7 | 4 | 7 | CAN high | **white** | **white** |
| 5 | 9 | 2 | 9 | not used | **(red)** | **(red)** |

### 3.2.4.8 BK5151, FC51xx, CX with CAN interface and EL6751: D-sub, 9 pin

The CANbus cable is connected to the FC51x1, FC51x2 CANopen cards and in the case of the EL6751 CANopen master/slave terminal via 9-pin Sub-D sockets with the following pin assignment.

| Pin | Assignment |
|---|---|
| 2 | CAN low (CAN-) |
| 3 | CAN ground (internally connected to pin 6) |
| 6 | CAN ground (internally connected to pin 3) |
| 7 | CAN high (CAN+) |

The unlisted pins are not connected.
The mounting rail contact spring and the plug shield are connected together.

Note: an auxiliary voltage of up to 30 $V_{DC}$ may be connected to pin 9. Some CAN devices use this to supply the transceiver.

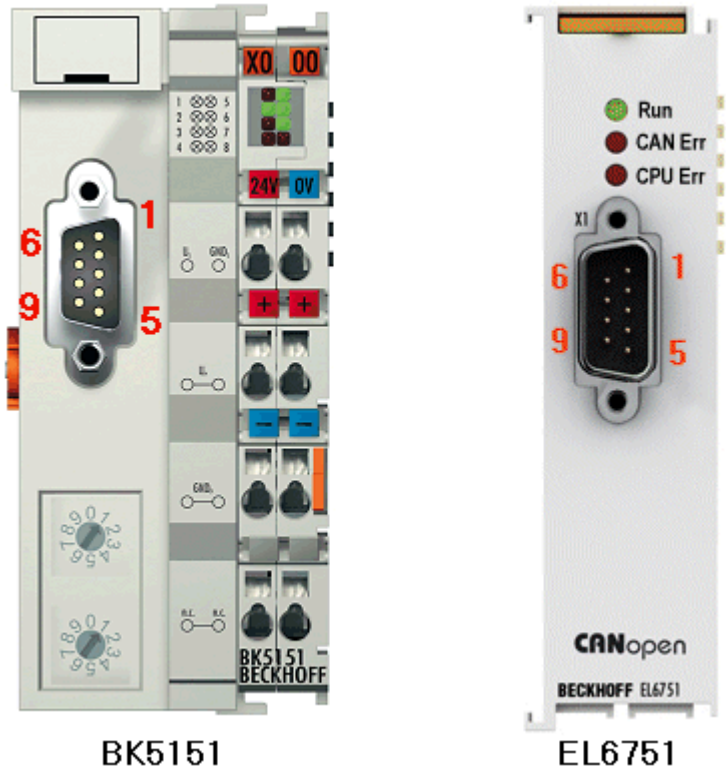



Fig. 16: BK5151, EL6751 pin assignment

**FC51x2:**

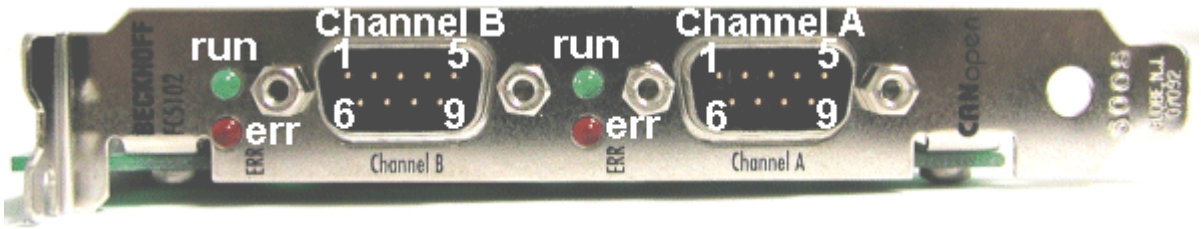



Fig. 17: FC51x2

### 3.2.4.9 BK51x0/BX5100: 5-pin open style connector

The BK51x0/BX5100 (X510) Bus Couplers have a recessed front surface on the left hand side with a five pin connector.
The supplied CANopen socket can be inserted here.

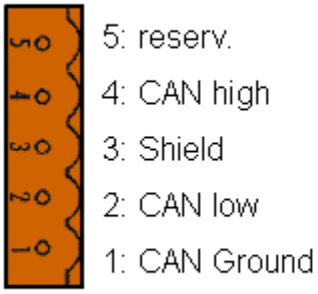





Fig. 18: BK51x0/BX5100 socket assignment

The left figure shows the socket in the BK51x0/BX5100 Bus Coupler. Pin 5 is the connection strip's top most pin. Pin 5 is not used. Pin 4 is the CAN high connection, pin 2 is the CAN low connection, and the screen is connected to pin 3 (which is connected to the mounting rail via an R/C network). CAN-GND can optionally be connected to pin 1. If all the CAN ground pins are connected, this provides a common reference potential for the CAN transceivers in the network. It is recommended that the CAN GND be connected to earth at one location, so that the common CAN reference potential is close to the supply potential. Since the CANopen BK51X0/BX5100 Bus Couplers provide full electrical isolation of the bus connection, it may in appropriate cases be possible to omit wiring up the CAN ground.

#### ZS1052-3000 Bus Interface Connector

The ZS1052-3000 CAN Interface Connector can be used as an alternative to the supplied connector. This makes the wiring significantly easier. There are separate terminals for incoming and outgoing leads and a large area of the screen is connected via the strain relief. The integrated terminating resistor can be switched externally. When it is switched on, the outgoing bus lead is electrically isolated - this allows rapid wiring fault location and guarantees that no more than two resistors are active in the network.

### 3.2.4.10 LC5100: Bus connection via spring-loaded terminals

In the low cost LC5100 Coupler, the CAN wires are connected directly to the contact points 1 (CAN-H, marked with C+) and 5 (CAN-L, marked with C-). The screen can optionally be connected to contact points 4 or 8, which are connected to the mounting rail via an R/C network.

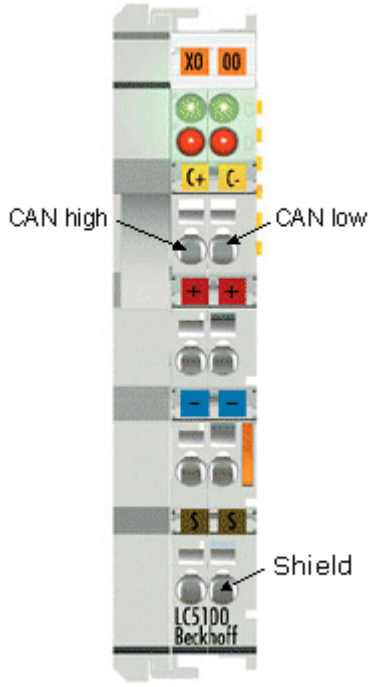



Fig. 19: LC5100

**NOTE**

**Risk of device damage!**

On account of the lack of electrical isolation, the CAN driver can be destroyed or damaged due to incorrect cabling. Always carry out the cabling in the switched-off condition.
First connect the power supply and then the CAN. Check the cabling and only then switch on the voltage.

### 3.2.4.11 Fieldbus Box: M12 CAN socket

The IPxxxx-B510, IL230x-B510 and IL230x-C510 Fieldbus Boxes are connected to the bus using 5-pin M12 plug-in connectors.

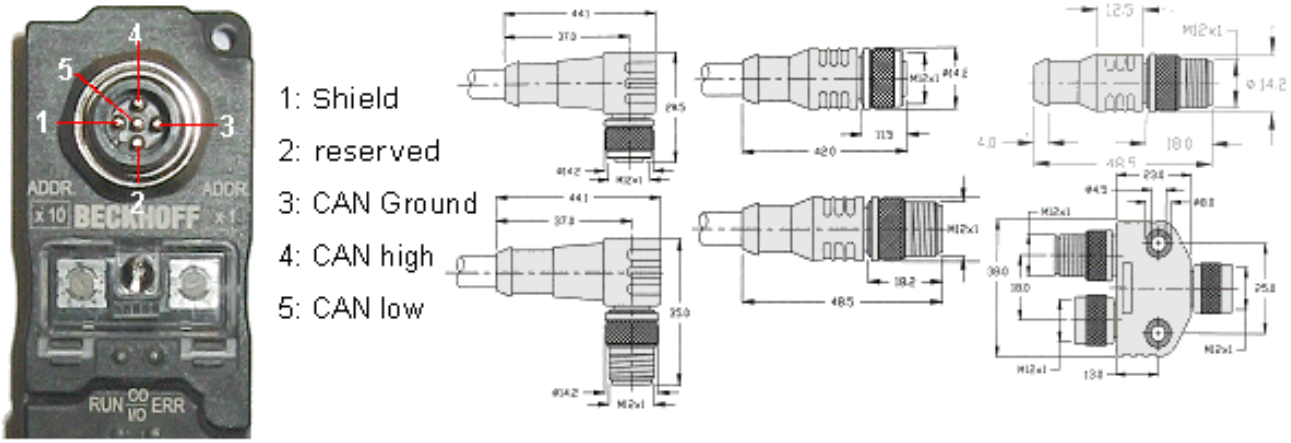



Fig. 20: Pin assignment: M12 plug, fieldbus box

Beckhoff offer plugs for field assembly, passive distributor's, terminating resistors and a wide range of pre-assembled cables for the Fieldbus Box system. Details be found in the catalogue, or under www.beckhoff.de.

## 3.2.5 ATEX - Special conditions (extended temperature range)

**⚠ WARNING**

**Observe the special conditions for the intended use of Beckhoff fieldbus components with extended temperature range (ET) in potentially explosive areas (directive 94/9/EU)!**

- The certified components are to be installed in a suitable housing that guarantees a protection class of at least IP54 in accordance with EN 60529! The environmental conditions during use are thereby to be taken into account!
- If the temperatures during rated operation are higher than 70°C at the feed-in points of cables, lines or pipes, or higher than 80°C at the wire branching points, then cables must be selected whose temperature data correspond to the actual measured temperature values!
- Observe the permissible ambient temperature range of -25 to 60°C for the use of Beckhoff fieldbus components with extended temperature range (ET) in potentially explosive areas!
- Measures must be taken to protect against the rated operating voltage being exceeded by more than 40% due to short-term interference voltages!
- The individual terminals may only be unplugged or removed from the Bus Terminal system if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- The connections of the certified components may only be connected or disconnected if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- The fuses of the KL92xx/EL92xx power feed terminals may only be exchanged if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- Address selectors and ID switches may only be adjusted if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!

### Standards

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

- EN 60079-0:2012+A11:2013
- EN 60079-15:2010

### Marking

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

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

or

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

# 4 Parameterization and Commissioning

## 4.1 Start-up behavior of the Bus Terminal Controller

When the Bus Terminal Controller is switched on it checks its state, configures the K-bus, creates a configuration list based on the connected Bus Terminals and starts its local PLC.
The I/O LEDs flash when the Bus Terminal Controller starts up. If the system is in an error-free state, the I/O LEDs should stop flashing after approx. 2-3 seconds. In the event of a fault the error type determines which LED flashes (see chapter *Diagnostic LEDs*).

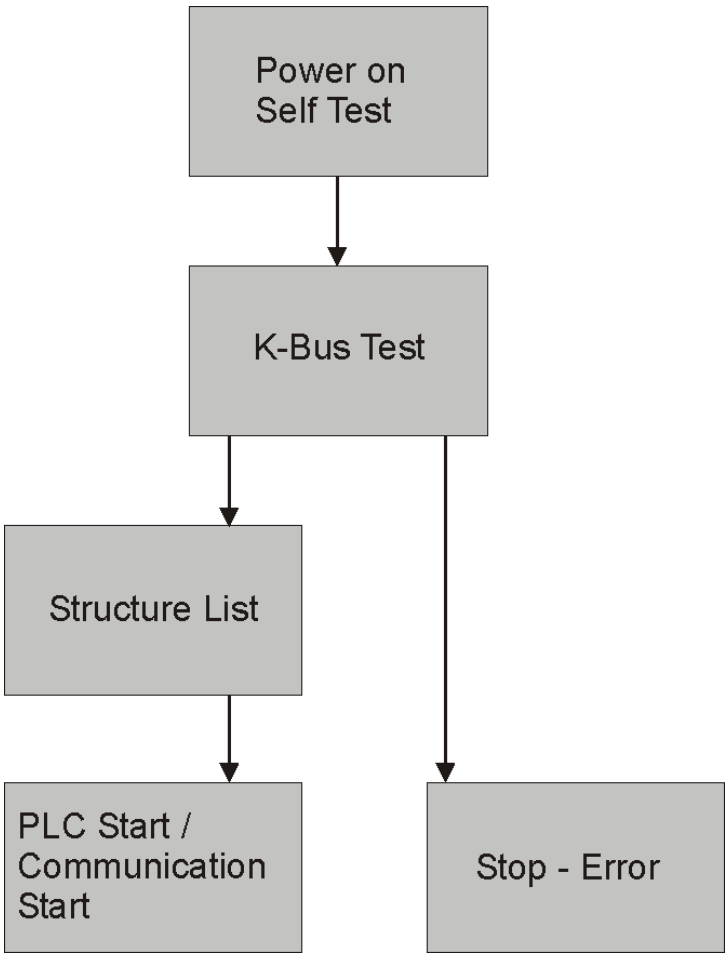



Fig. 21: **Start-up behavior of the Bus Terminal Controller**

## 4.2 Configuration

### 4.2.1 Overview

**Configuration types**

The Bus Terminal controllers of the BCxx50, BCxx20 and BXxx00 series can be configured in two different ways: DEFAULT CONFIG or TwinCAT CONFIG.

**DEFAULT-CONFIG**

Bus Terminals are mapped in the order they are inserted, i.e. first the complex Bus Terminals followed by the digital Bus Terminals.

The complex Bus Terminals are mapped as follows:

- Word Alignment
- complex representation

**⚠ CAUTION**

**The process image depends on the connected terminals!**

The process image changes when a terminal is added or removed!

The data of the fieldbus slaves interface are referred to as PLC variables. The PLC variables have addresses from %QB1000 and %IB1000

The DEFAULT CONFIG (without PLC program) can also be used for writing and testing of the Connected Bus Terminals. To this end, the Bus Terminal Controller must be scanned in the System Manager, and FreeRun mode must be enabled (to use this function, no PLC program may be active on the Bus Terminal Controller).

### TWINCAT-CONFIG

In the TwinCAT CONFIG the Bus Terminals and PLC variables can be freely linked as required (TwinCAT System Manager file required). The configuration is transferred to the coupler via the System Manager and ADS.

The following is required for the TwinCAT configuration (TC file):

- Via the fieldbus (PROFIBUS, CANopen, Ethernet)
  PROFIBUS: (BC3150, BX3100)
  - PC with FC310x from version 2.0 and TwinCAT 2.9 build 1000
  - BX3100 with CIF60 or CP5412
  - TwinCAT 2.9 build 946
    (**NOTE:** with PROFIBUS cards from Hilscher only one ADS communication is permitted, i.e. either System Manager or PLC Control)
    CANopen: (BC5150, BX5100)
  - PC with FC510x from version 1.76 TwinCAT build 1030
    DeviceNet: (BC5250, BX5200)
  - on request
    Ethernet: (BC9050, BC9020, BC9120, BX9000)
  - PC with TwinCAT 2.10 build 1322
- Via the serial ADS TwinCAT 2.9 build 1010
  - BX3100 version 1.00
  - BX5100 version 1.00
  - BX5200 version 1.10
  - BX8000 version 1.00
  - BC3150, BC5150, BC5250, BC9050, BC9020, BC9120 from firmware B0
  - For BC8150 from TwinCAT 2.10 build 1243

BCxx50 and BXxx00 can be parameterized via the System Manager of the TwinCAT program.

- Variable I/O mapping
- Type-specific PROFIBUS data (BC3150 and BX3100 only)
- RTC (real-time clock) (BX series only)
- SSB (Smart System Bus) (BX series only)
- PLC settings
- K-Bus settings

The configuration can be transferred to the BCxx50 or BXxx00 via fieldbus ADS protocol or serial ADS protocol.

The TwinCAT configuration can be used to link variables, I/Os and data. The following is possible:

- PLC - K-BUS
- PLC fieldbus (e.g. PROFIBUS slave interface to PLC)
- K-bus fieldbus (only for BX controllers)
- Support for TwinSAFE terminals (only BX controllers from firmware 1.17)

In addition, the TwinCAT configuration can be used to parameterize special behavior, for example whether data are preserved or set to "0" in the event of a fieldbus error.
The real-time clock can be set via a tab in the system manager.

**Work steps**

1. Setting the fieldbus address
2. Open the System Manager and create a TC file
3. Configure fieldbus data in the TC file
4. Save the TC file
5. Opening a new system manager, creating a PC file and reading in saved TX file
6. Creating a link to a PLC task
7. Saving the configuration
8. Starting the TwinCAT system
9. Open the TC file in the System Manager, complete the configuration and transfer it to the BCxx50, BCxx20 or BXxx00
10. Transfer the program to BCxx50, BCxx20 or BXxx00
11. Creating a boot project

## 4.2.2 Creating a TwinCAT configuration

In order to configure a Bus Terminal Controller of the BCxx50, BCxx20 or BXxx00 series, create a BX file in the System Manager. To simplify matters, files for the basic units have already been prepared. Open the corresponding Bus Terminal Controller with *New from Template*.

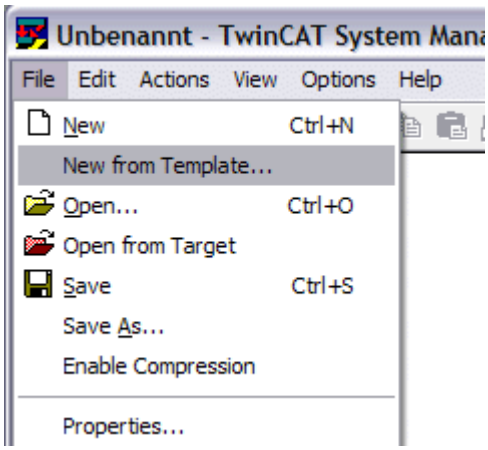



Fig. 22: Creating a TwinCAT configuration

Select the corresponding Bus Terminal Controller.

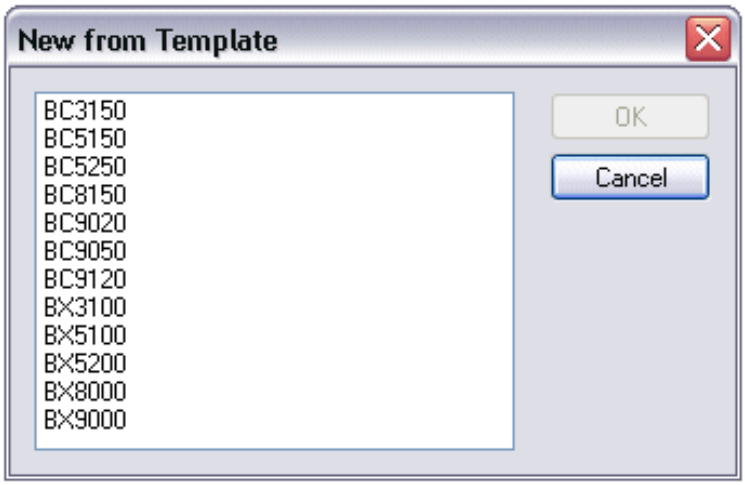



Fig. 23: Selecting the Bus Terminal Controller

All Bus Terminal Controller components are now available:

- Fieldbus interface
- K-bus interface [▶ 43]
- PLC Program [▶ 45]
- SSB (only Bus Terminal Controllers of the BX series)

Please refer to the relevant chapter for device configuration.

### 4.2.3 Downloading a TwinCAT configuration

The TwinCAT configuration is loaded into the Bus Terminal Controller via ADS protocol.

**Serial ADS protocol**

(all Bus Terminal Controllers of the BXxx00 and BCxx50 series)

Enter the serial ADS connection, as described in the chapter Serial ADS [▶ 38].

**ADS protocol via the fieldbus**

(BC3150, BC5150, BC9x20, BC9050, BX3100, BX5100, BX9000 only)

A prerequisite is that TwinCAT operates as master and is engaged in data exchange, i.e. the physical and fieldbus configuration must be complete, and data exchange must take place between the master (e.g. fieldbus master card) and the Bus Terminal Controller.

**Choose Target System**

Select the Bus Terminal Controller onto which the configuration is to be loaded. Use the function key F8 to open the dialog for downloading your file to the corresponding device.

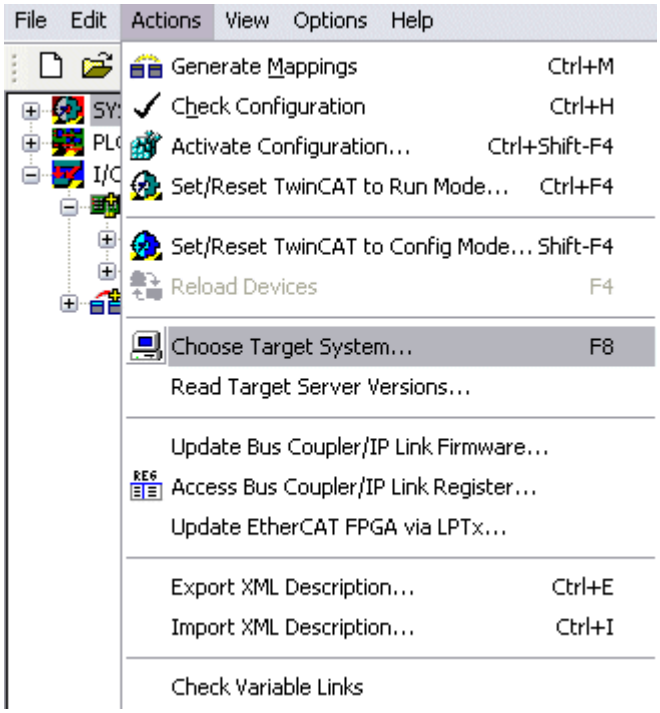



Fig. 24: Downloading a TwinCAT configuration

Select the corresponding Bus Terminal Controller.

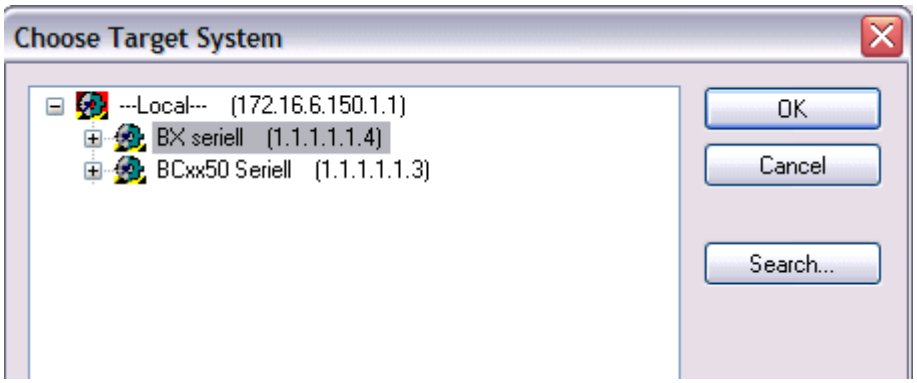



Fig. 25: Selecting the Bus Terminal Controller

The state of the Bus Terminal Controller is shown at the bottom right of the System Manager.

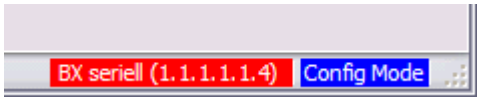



Fig. 26: State of the Bus Terminal Controller

In *Config mode / FreeRun* the configuration can now be downloaded to Bus Terminal Controller. If the Bus Terminal Controller is in *Stop mode*, ADS communication is not yet activated. In this case, it is not possible to download the configuration.

To activate the TwinCAT configuration select Ctrl+Shift+F4 or *Activate Configuration*.

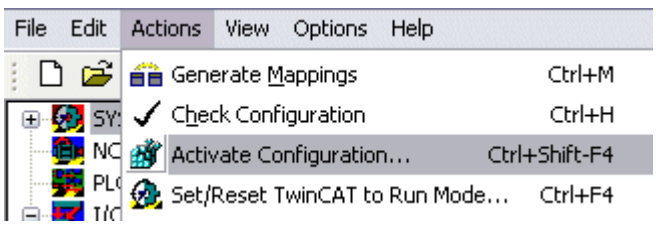



Fig. 27: Activating the TwinCAT configuration

The current configuration is loaded onto the Bus Terminal Controller. The display will show *Store Config*, and the BUS and I/O LED will flash. Once the configuration is successfully loaded onto Bus Terminal Controller, *TwinCAT Config* should appear in the display of a BXxx00. The corresponding program can now be transferred to the Bus Terminal Controller (program-download via the fieldbus).

## 4.2.4 Uploading a TwinCAT configuration

The TwinCAT configuration is loaded into the Bus Terminal Controller via ADS protocol.

### Serial ADS protocol

(all Bus Terminal Controllers of the BCxx50, BCxx20 and BXxx00 series)

Enter the serial ADS connection, as described in the chapter Serial ADS [▶ 38].

### ADS protocol via the fieldbus

(BC3150, BC5150, BC9x20, BC9050, BX3100, BX5100, BX9000 only)

A prerequisite is that TwinCAT operates as master and is engaged in data exchange, i.e. the physical and fieldbus configuration must be complete, and data exchange must take place between the master (e.g. fieldbus card) and the Bus Terminal Controller.

### Choose Target System

Select the Bus Terminal Controller onto which the configuration is to be loaded. Use the function key [F8] to open the dialog for downloading your file to the corresponding device.

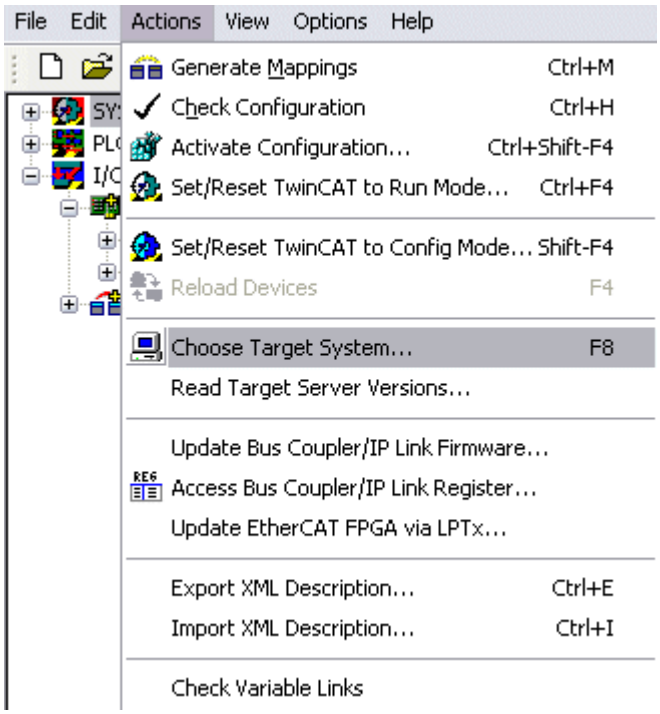



Fig. 28: Choose Target System

Select the corresponding Bus Terminal Controller.

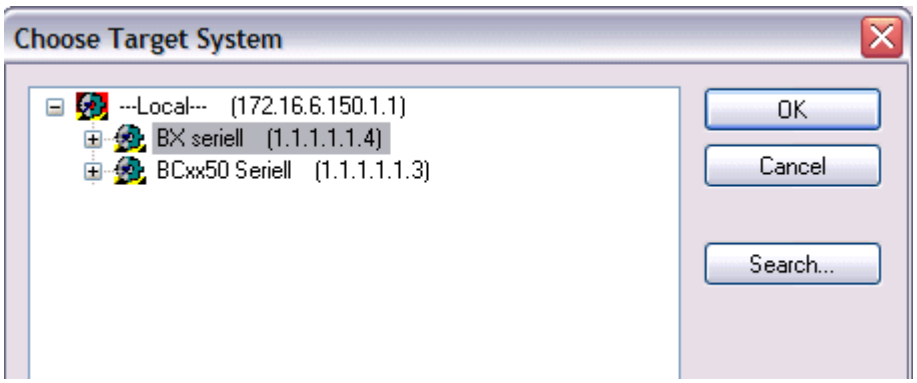



Fig. 29: Selecting the Bus Terminal Controller

The state of the Bus Terminal Controller is shown at the bottom right of the System Manager.

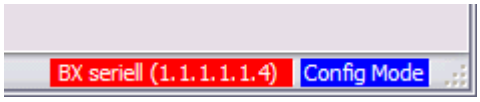



Fig. 30: State of the Bus Terminal Controller

Click on the red folder. The TwinCAT configuration will now be uploaded.

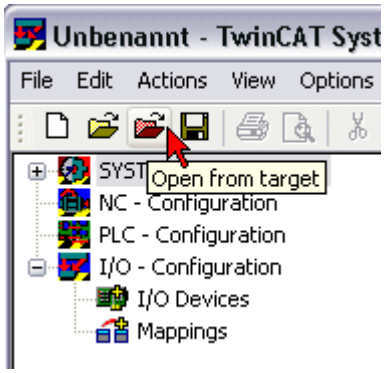



Fig. 31: Uploading the TwinCAT configuration

## 4.2.5 Resources in the Bus Terminal Controller

The memory resources assigned in the Bus Terminal Controller are shown in the System Manager in the *Resources* tab of the Bus Terminal Controller.

### Mapping code

The mapping code is required for calculating the TwinCAT configuration (see Figure *Memory for the code mapping*). The percentages are added here. In the example from Fig. *Memory for code mapping*, 8% of the memory is allocated to the mapping calculation.

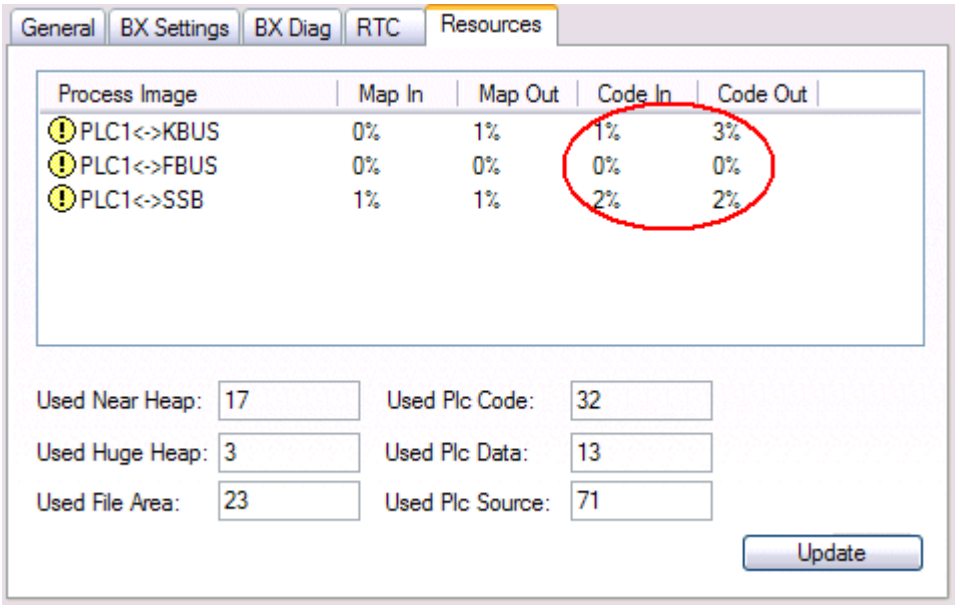



Fig. 32: Memory for code mapping

### Data memory mapping

Data memory for mapping. The values are to be considered individually, i.e. each value can be up to 100%.

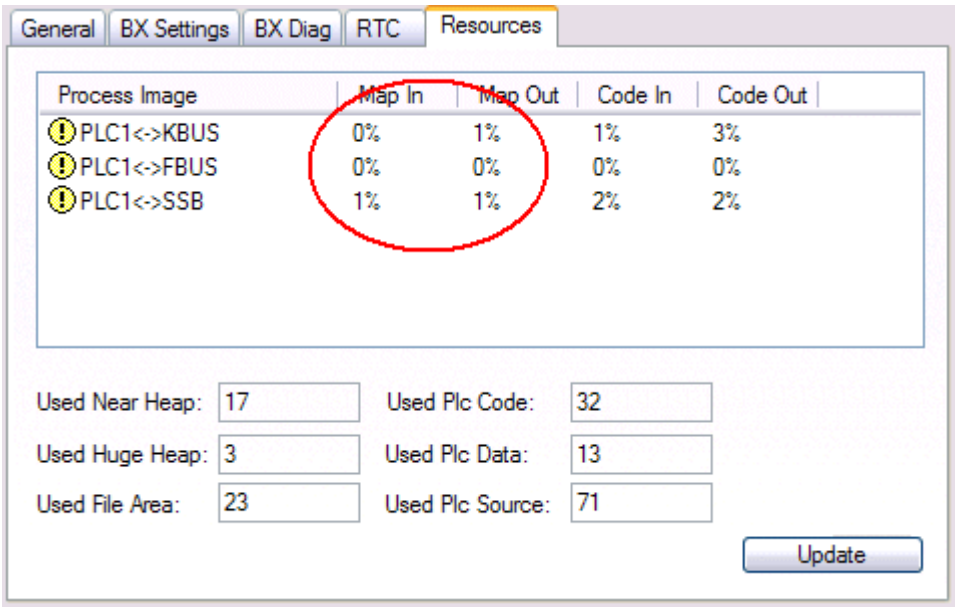



Fig. 33: Data memory mapping

### Used code and data memory

Fig. *Code and data memory* (1) "Used PLC code" in %.
Fig. *Code and data memory* (2) "Used PLC data" in %.
Fig. *Code and data memory* (3) "Used PLC source" in %.

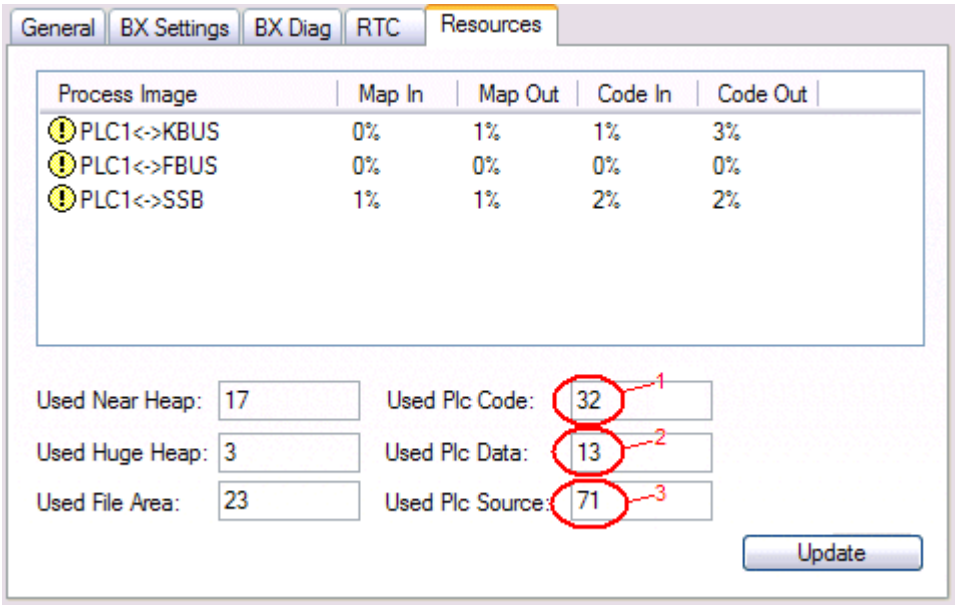



Fig. 34: Code and data memory

### Other memory

Fig. *Other Memory* (1) "Used Near Heap" is required for the COM interface and SSB. % values.
Fig. *Other Memory* (2) "Used Huge Heap" is required for the ADS communication. % values. This value should be less than 30 %.
Fig. *Other Memory* (3) "Used File Area" is required for the TwinCAT configuration, the TSM file and the 16 kbyte flash access. % values.

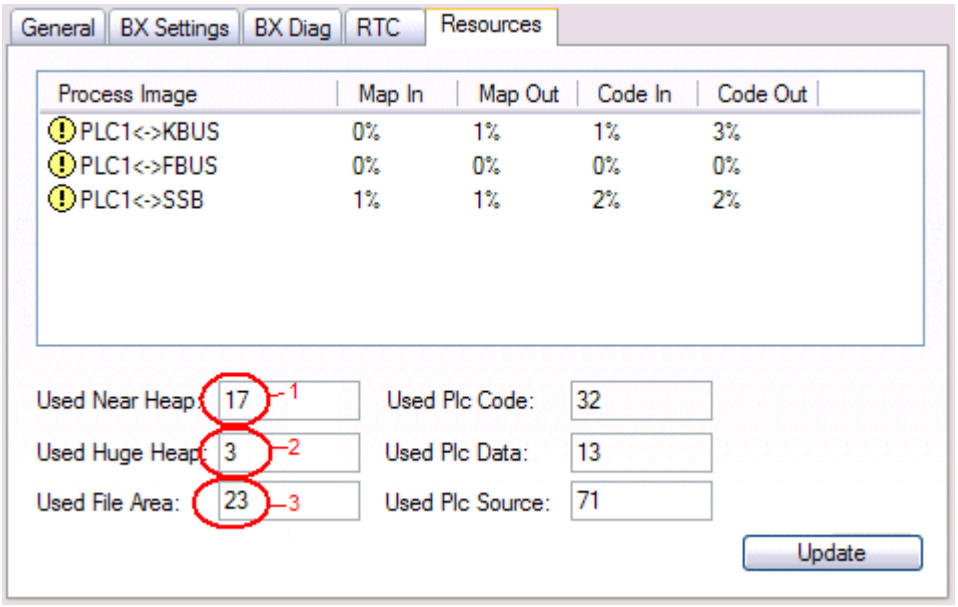



Fig. 35: Other memory

## 4.2.6 ADS connection via serial interface

(from firmware version 1.xx or 0.99x, Bus Terminal Controllers of the BX series and for all BCxx50)

From TwinCAT 2.9 build 1020 (TwnCAT level PLC, NC or NCI)

**Use only a serial connection**

To ensure trouble-free operation of the ADS link via the serial interface, only a serial connection to the BX controller is allowed.
After successful configuration via the System Manager, close the System Manager before starting programming.

**AMS Net ID in delivery state (default)**

**For BX9000**

The default AMS Net ID is 172.16.21.20.1.1. If the IP address of the BX9000 is changed, the AMS Net ID of the BX9000 also changes. There is a menu option for displaying the current AMS Net ID.
Example: If you change the IP address to 10.2.3.7, the AMS Net ID changes to 10.2.3.7.1.1.

**For BC9050, BC9020, BC9120**

The default AMS Net ID is 172.16.xxx.[DIP switch].1.1. If the IP address of the BX9xxx is changed, the AMS Net ID of the BX9xxx also changes.
Example: If you change the IP address to 10.2.3.7, the AMS Net ID changes to 10.2.3.7.1.1.
BC9050: DEFAULT 172.16.21.[DIP-Switch].1.1
BC9020: DEFAULT 172.16.22.[DIP-Switch].1.1
BC9120: DEFAULT 172.16.23.[DIP-Switch].1.1

### Initializing the ADS connection

Enter the Bus Terminal Controller in the remote connection under TwinCAT. Click on the TwinCAT icon and open the features menu. The following settings can be made under the >AMS Remote< tab.

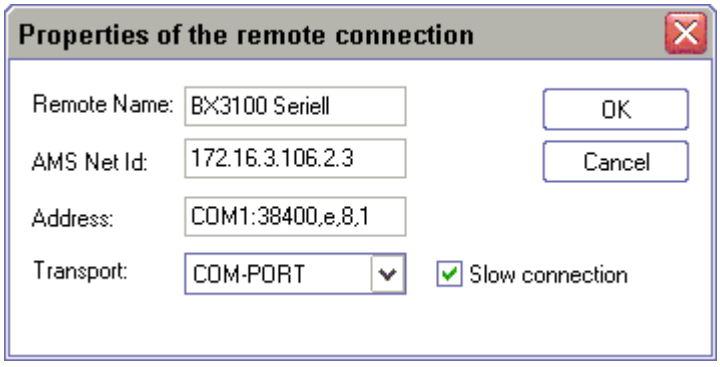



Fig. 36: Properties of the remote connection

Remote Name: Any
AMS-Net-ID: 1.1.1.1.1.1 (Default)
Address: COM Port: Baud rate, parity, data bits, stop bits
Transport: Select "COM port"

When the Bus Terminal Controller is switched on, the default AMS Net ID is always "1.1.1.1.1.1" (except all Ethernet Controllers).
The AMS Net ID can be changed as required. Please note that the new AMS Net ID cannot be changed again in this way.

If you need to change the new AMS Net ID again, you have to restart the Bus Terminal Controller, so that the AMS Net ID is reset to the default AMS Net ID, "1.1.1.1.1.1".
You can now change the AMS Net ID again.

**Strings can only be entered at the second call**

No strings can be entered under address when the dialog is first called (see above). Enter the name, AMS Net ID and transport type and close the dialog. With the second call you can enter your COM port.

The communication starts when TwinCAT is in Config mode (TwinCAT icon is blue) or RUN mode (TwinCAT icon is green). The COM interface remains open until a TwinCAT stop occurs (TwinCAT icon is red). It is then available again for other programs. No error message is issued if the COM interface is used by another program during a TwinCAT restart (e.g. by the KS2000 configuration software).

**AMS Net ID after ADS connection via the fieldbus**

If you have addressed the Bus Terminal Controller with an ADS connection via the fieldbus before the serial ADS was used, the AMS Net ID was automatically changed by the System Manager. In this case a new serial ADS connection is only possible, if the AMS Net ID is adjusted.

#### BX series: reading the AMS Net ID

The current AMS Net ID can be read from the menu via the display of BX series Bus Terminal Controller.

**AMS** AMS Net ID
1.1.1.1.1.1

## 4.2.7 CANopen slave interface

### 4.2.7.1 CANopen slave interface

There are two types of configuration. In the default configuration (delivery state) the CANopen data of the CANopen Slave interface map from the address 1000 of the BX5100/BC5150 and the first 4 PDOs are activated. In the TwinCAT configuration, any required configuration can be created via the System Manager and variables can be connected in any required combination with the CANopen slave interface.

#### Default Configuration

In this configuration, the CANopen data are mapped as follows:

#### CANopen data

| PDO number | Read/Write | BX process image |
|---|---|---|
| PDO 1 | Rx/Tx | %IB1000...%IB1007/QB1000...%QB1007 |
| PDO 2 | Rx/Tx | %IB1008...%IB1015/QB1008...%QB1015 |
| PDO 3 | Rx/Tx | %IB1016...%IB1023/QB1016...%QB1023 |
| PDO 4 | Rx/Tx | %IB1024...%IB1031/QB1024...%QB1031 |

In the default configuration the Tx PDOs are only transferred in the event of a change.

#### More than 4 PDOs

Utilization of more than 4 PDOs can be specified in the TwinCAT configuration or in the default configuration. With the TwinCAT configuration the required PDOs are created in the System Manager file, and the configuration is transferred to the BX5100/BC5150. With the default configuration objects 0x14xx (for TxPDOs) and 0x18xx (for the RxPDOs) have to be used for this purpose before the node is started. The COB ID should be entered in subindex 1, and the data transfer type in subindex 2.

Example for PDO 5 of the node with address 11:
0x1804 subindex 1, length 4, value 0x68B
0x1804 subindex 2 length 1 value 0xFF (not necessarily required for activating the PDO)
0x1404 subindex 1 length 4 value 0x78B
0x1404 subindex 2 length 1 value 0xFF (not necessarily required for activating the PDO)

With the BX5100 a maximum of 32 PDOs can be used in each direction (32 TxPDOs and 32 RxPDOs). With the BC5150 a maximum of 16 PDOs can be used in each direction (16 TxPDOs and 16 RxPDOs).

| PDO number | Read/Write | BX process image |
|---|---|---|
| PDO 5 | Rx/Tx | %IB1032...%IB1039/QB1032...%QB1039 |
| PDO 6 | Rx/Tx | %IB1040...%IB1047/QB1040...%QB1047 |
| PDO 7 | Rx/Tx | %IB1048...%IB1055/QB1048...%QB1055 |
| ... | Rx/Tx | ... |
| PDO 32 | Rx/Tx | %IB1248...%IB1255/QB1248...%QB1255 |

### Acyclic Synchronous

PDOs of transmission type 0 function synchronously, but not cyclically. An RxPDO is only evaluated after the next SYNC telegram has been received. In this way, for instance, axis groups can be given new target positions one after another, but these positions only become valid at the next SYNC - without the need to be constantly outputting reference points. A device whose TxPDO is configured for transmission type 0 acquires its input data when it receives the SYNC (synchronous process image) and then transmits it if the data correspond to an event (such as a change in input) having occurred. Transmission type 0 thus combines transmission for reasons that are event driven with a time for transmission (and, as far as possible, sampling) and processing given by the *reception of "SYNC"*.

### Cyclic Synchronous

In transmission types 1-240 the PDO is sent cyclically after each nth SYNC (n=1...240). Since transmission types can be combined on a device as well as in the network, it is possible, for example, for a fast cycle to be agreed for digital inputs (n = 1), whereas the data for analog inputs is transmitted in a slower cycle (e.g. n = 10). RxPDOs generally do not differentiate between the transmission types 0...240: a received PDO is set to valid with the next SYNC reception. The cycle time (SYNC rate) can be monitored (object 0x1006), so that if the SYNC fails the device reacts in accordance with the definition in the device profile, and switches, for example, its outputs into the error state.

The CANopen CIFx0 PC cards always transmit under event control, even if the transmission type is set in the range from 1-240. This behavior is quite similar to transmission type 0. The FC510x PC cards support cyclic synchronous transmission types completely.

### Asynchronous

The transmission types 254 and 255 are asynchronous, but may also be event-driven. In transmission type 254, the event is vendor-specific, whereas for type 255 it is defined in the device profile. In the simplest case, the event is the change of an input value - this means that every change in the value is transmitted.

### Send delay time (inhibit time)

(BC5150 and BX5100 do not support this function.)

The *Inhibit Time* parameter can be used to implement a transmit filter that does not increase the reaction time for relatively new input alterations, but is active for changes that follow immediately afterwards. The send delay time describes the minimum time interval required between sending two identical telegrams. If the inhibit time is used, the maximum bus loading can be determined, so that the worst case latency can then be found.

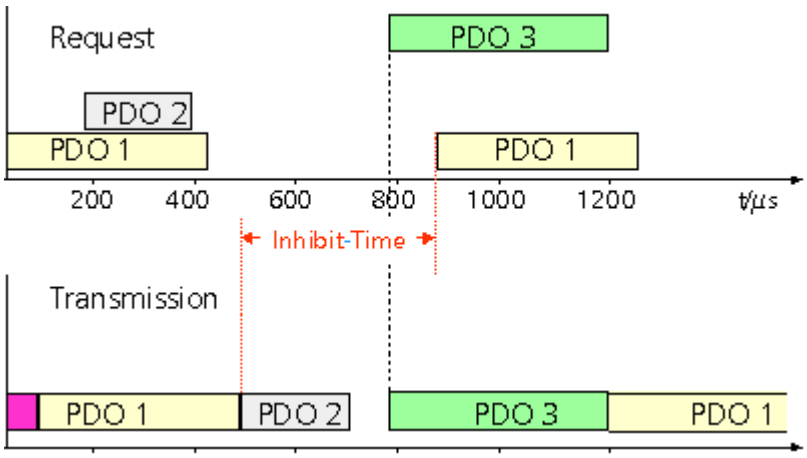



Fig. 37: Send delay time (inhibit time)

The transmitted PDOs become automatically spread out (transmit delay) as a result of the selected PLC cycle time - and there is little value in having the PLC run faster than the bus bandwidth permits. The bus loading, furthermore, can be significantly affected by the synchronous communication.

**Event Timer**

An event timer for transmit PDOs can be specified by subindex 5 in the communication parameters. Expiry of this timer is treated as an additional event for the corresponding PDO, so that the PDO will then be transmitted. If the application event occurs during a timer period, it will also be transmitted, and the timer is reset.

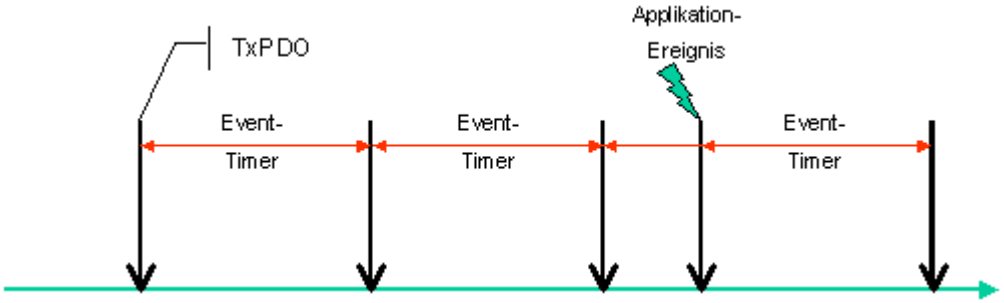



Fig. 38: Event timer

In receive PDOs the timer parameter is used to specify the monitoring time for the respective PDO: The application will be notified if no corresponding PDO was received within the set time.

### 4.2.7.2 TwinCAT configuration

The TwinCAT configuration enables free, address-independent mapping of the PLC data to the CAN slave interface.

**TwinCAT configuration**

The configuration requires the TwinCAT System Manager and an ADS connection to the Bus Terminal Controller.

The ADS connection can be realized via the serial interface (see serial ADS) or the fieldbus (FC510x PCI card and the CAN slave interface of the BX5100).

### 4.2.7.3 Setting the slave address

The address must be set via the two rotary selection switches. The default setting is 11. All addresses are permitted, although each address may only occur once within the network. For changing an address the Bus Coupler must be switched off. The switches can be set to the required position using a screwdriver. Ensure that the switches engage correctly. The lower switch is the ten-multiplier, the upper switch is the one-multiplier. The change in address is active as soon as the module is switched on.

**Example**

You want to set address 34:

- Lower rotary selection switch Sx11: 3
- Upper rotary selection switch Sx10: 4

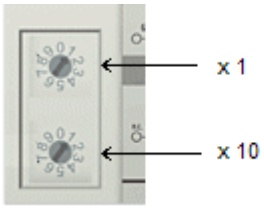



Fig. 39: Setting the node ID

**Special functions using the address switch**

In addition, the address switch can be used for following functions. To this end the associated address has to be set (as long as the coupler is switched off), only the end terminal may be connected (KL9010), and the coupler must then has to be connected to the supply voltage.

Address 99: Manufacturer's setting
Address 98: Delete boot project
Address 97: Delete TwinCAT configuration

### 4.2.7.4 Baud rate

**Auto-Baud-Rate**

In order for automatic baud rate detection to function, it is necessary for a number of valid telegrams to be present on the bus at the desired baud rate. The RUN and CAN ERR LEDs blink in rapid alternation while the baud rate search is in progress. Once a baud rate was detected and accepted, the Bus Terminal Controller continues with the initialization.

A software reset does not lead to a new baud rate search. The previously active baud rate is maintained.

**Bit Timing**

The following baud rates and entries in the bit-timing register are supported by the BECKHOFF CANopen devices:

| Baud rate [kbaud] | BTR0 | BTR1 | Sampling Point |
|---|---|---|---|
| 1000 | 0x00 | 0x14 | 75% |
| 800 | 0x00 | 0x16 | 80% |
| 500 | 0x00 | 0x1C | 87% |
| 250 | 0x01 | 0x1C | 87% |
| 125 | 0x03 | 0x1C | 87% |
| 100 | 0x04 | 0x1C | 87% |
| 50 | 0x09 | 0x1C | 87% |
| 20 | 0x18 | 0x1C | 87% |
| 10 | 0x31 | 0x1C | 87% |

The bit-timing register settings given (BTR0, BTR1) apply, e.g. for the Philips 82C200, SJA1000, Intel 80C527, Siemens 80C167 and other CAN controllers. They are optimized for the maximum bus length.

## 4.2.8 K-bus

**Bus Terminal and end terminal required**

To operate a Bus Terminal Controller of the BC or BX series, at least one Bus Terminal with process image and the end terminal must be connected to the K-bus.

### BX Settings tab

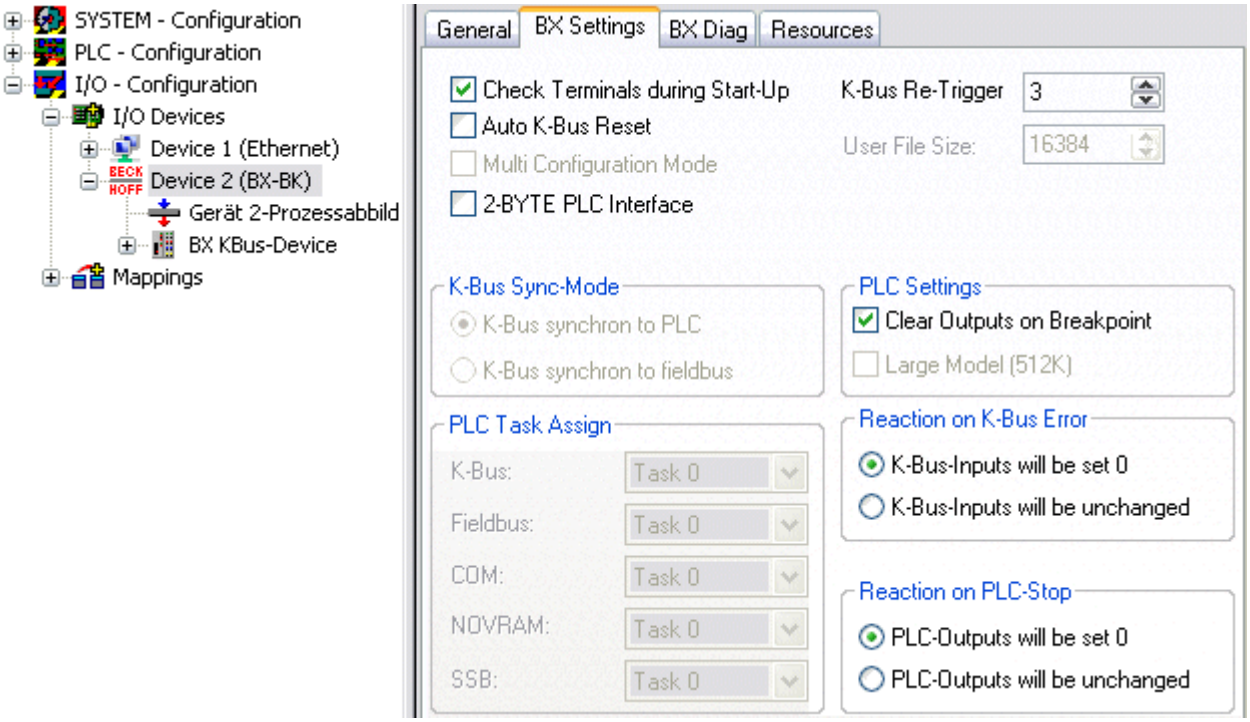



Fig. 40: BX Settings tab

#### Check Terminals during Start-up

When a boot project is created, the current Bus Terminal configuration is stored. The connected Bus Terminals are checked when the Bus Terminal Controller restarts. If this option is selected, the Bus Terminal Controller does not enter into data exchange. The PLC project will not be started.

#### Auto K-Bus Reset

Once a K-bus error has been rectified, the Bus Terminal Controller automatically resumes the data exchange.

**⚠ CAUTION**

**Once a K-Bus error has been rectified, the outputs become active again immediately!**

Ensure that the outputs are reactivated immediately and that analog outputs retain their programmed value, if this is not dealt with in your PLC project.

#### Clear Outputs on Breakpoint

If breakpoints are set in PLC Control, the K-Bus is no longer processed, i.e. the outputs are set to a safe state (zero).

### K-Bus Sync Mode

Writing and reading of the Bus Terminals can take place synchronously with task 1 or the fieldbus.

### K-Bus Re-Trigger

If the processor is busy dealing with the PLC project or the SSB, the K-Bus cannot be processed for a certain amount of time. This leads to triggering of the Bus Terminal watchdog and dropping of the outputs. The Bus Terminal Controller is set such that the K-bus watchdog is re-triggered 3 times after 85 ms. The K-Bus watchdog would then be activated.
K-Bus Re-Trigger 0: 100 ms
K-Bus Re-Trigger 1: 2 x 85 ms = 170 ms
K-Bus Re-Trigger 2: 3 x 85 ms = 255 ms
K-Bus Re-Trigger 3: 4 x 85 ms = 340 ms

### Reaction on K-Bus Error

In the event of a K-Bus error, the K-Bus inputs are set to "0" or retain their last state.

### Response on PLC-Stop

The user can set the behavior of the fieldbus output data in the event of the PLC project being stopped. The master will use these data as input data. In the event of a PLC stop, the data can be set to "0" or remain unchanged.

### BX Diag tab

Display of the cycle time for task 1, K-bus, fieldbus processing and the SSB load.

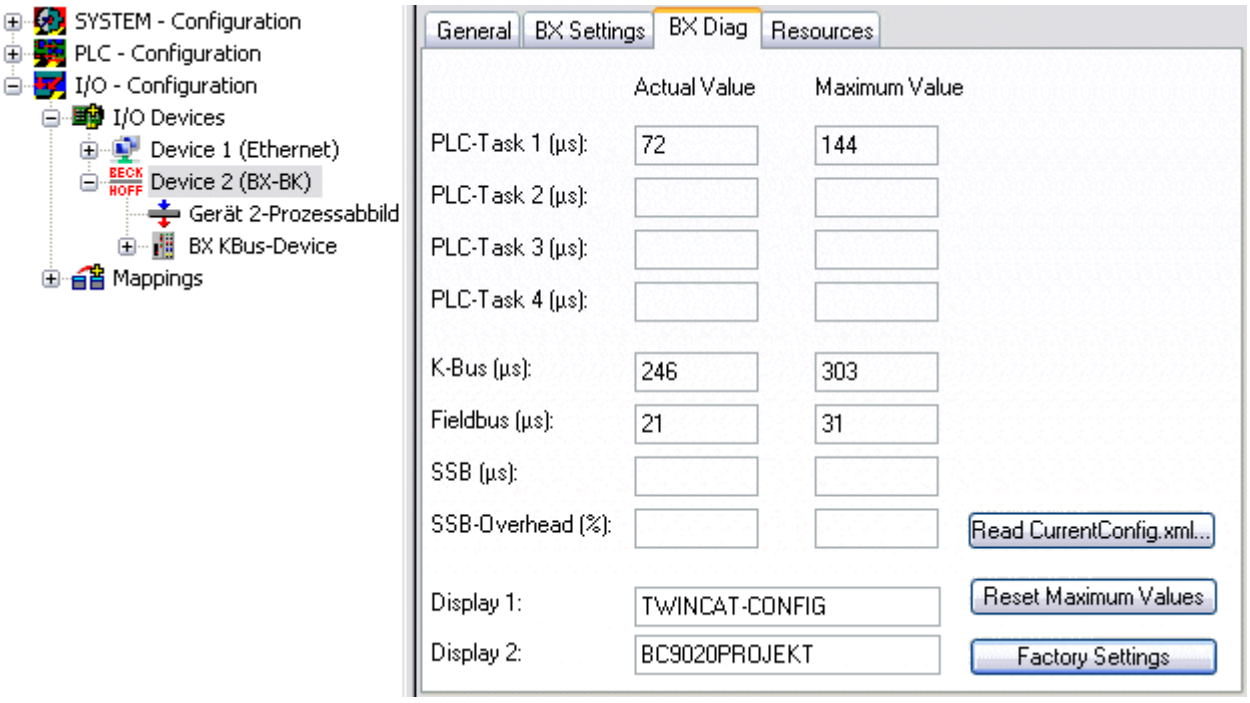
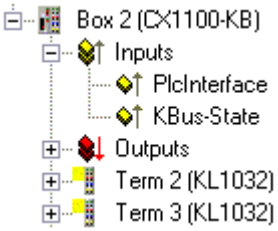



Fig. 41: BX Diag tab

*Factory Settings*: the Bus Terminal Controller is set to its delivery. These settings are reactivated via Restart System or by switching the system off and on again (display shows DEFAULT-CONFIG).
*Reset Maximum Values*: resets the maximum values

**K-Bus variables**




**PLC interface:** Not supported (only included for moving CX or BX projects)

**K-bus state:** see Diagnostics

## 4.2.9 PLC

### 4.2.9.1 Inserting a PLC project

For variable mapping, configuration has to be specified in the system manager. This is where the link between PLC and hardware is specified. The variables can process and link bit, byte, word or data structures. Automatic addressing via the System Manager is possible, but should be checked for offset.

**Word alignment, byte orientation**

With data structures, ensure that the Bus Terminal Controller saves the data in word alignment and the System Manager operates byte-oriented (see Data structures [▶ 60])

A valid project has to be compiled and saved in PLC Control. These data are saved as a *.tpy file. For inserting a PLC project, right-click on *PLC - Configuration*. Select your current PLC project.

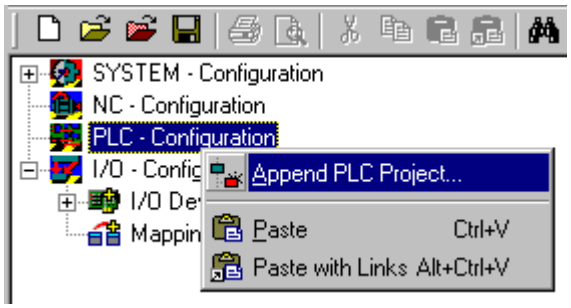



Fig. 42: Selecting the PLC project

Link the PLC variable with the hardware (e.g. digital Bus Terminal).

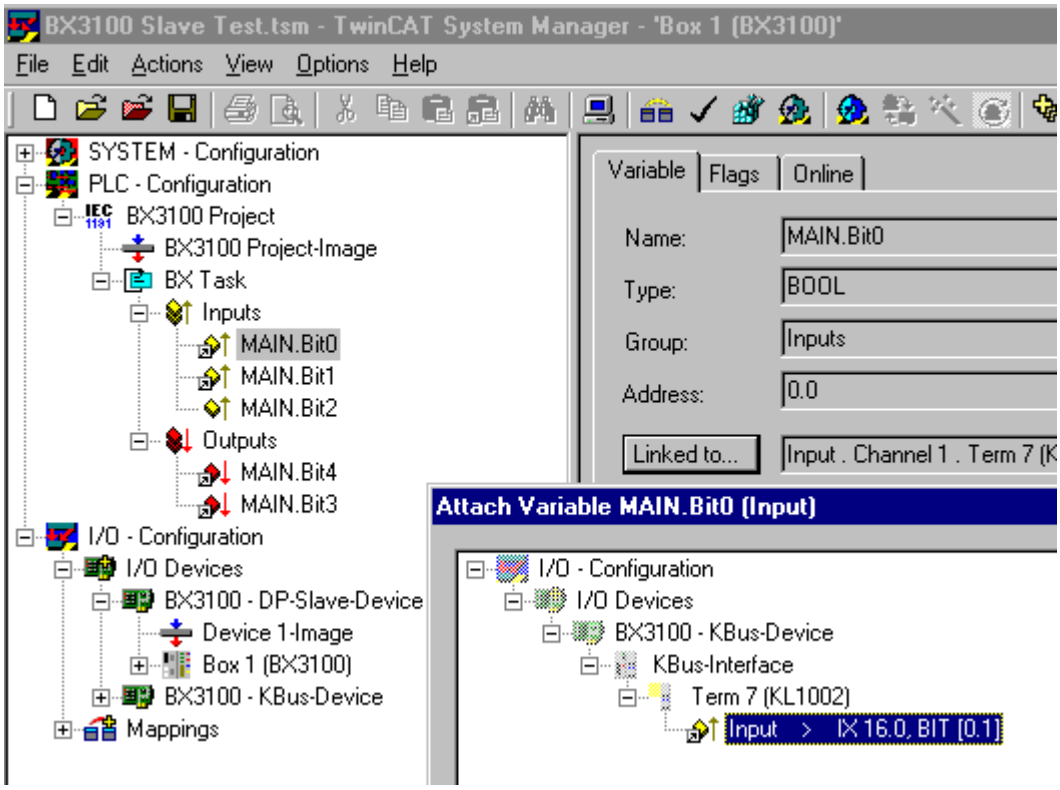



Fig. 43: Connecting PLC variable and hardware

Once all links have been created, activate the configuration *Actions/Activate Configuration* (Ctrl+Shift+F4) and start TwinCAT *Set/Reset TwinCAT to Run Mode*. Ensure that you have selected the correct target system (bottom right in the System Manager window).

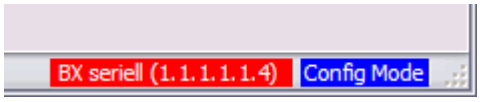



Fig. 44: Target system display

### 4.2.9.2 Measuring the PLC cycle time

The task time is set in PLC Control. The default setting is 20 ms.

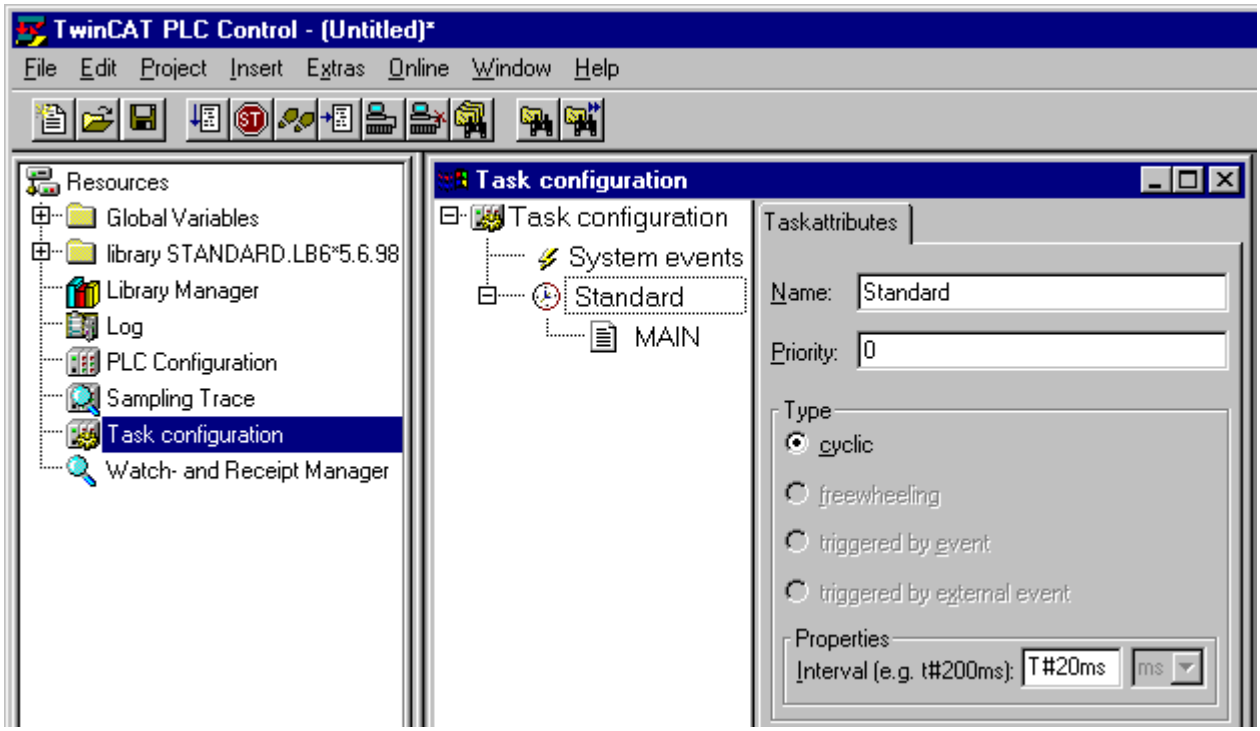



Fig. 45: Setting the task time

In the default setting, the PLC program is called every 20 ms, as long as the general cycle time is less than 20 ms. To determine the load of your system, the PLC cycle time can be measured in the System Manager. In order to ensure trouble-free operation, the set task time should be 20-30 % higher than the measured total cycle time. A precise cycle time breakdown can be found under K-Bus tab [▶ 43] description. The total cycle time is displayed with the TcBase library (see TcBase.lbx or TcBaseBCxx50.lbx).

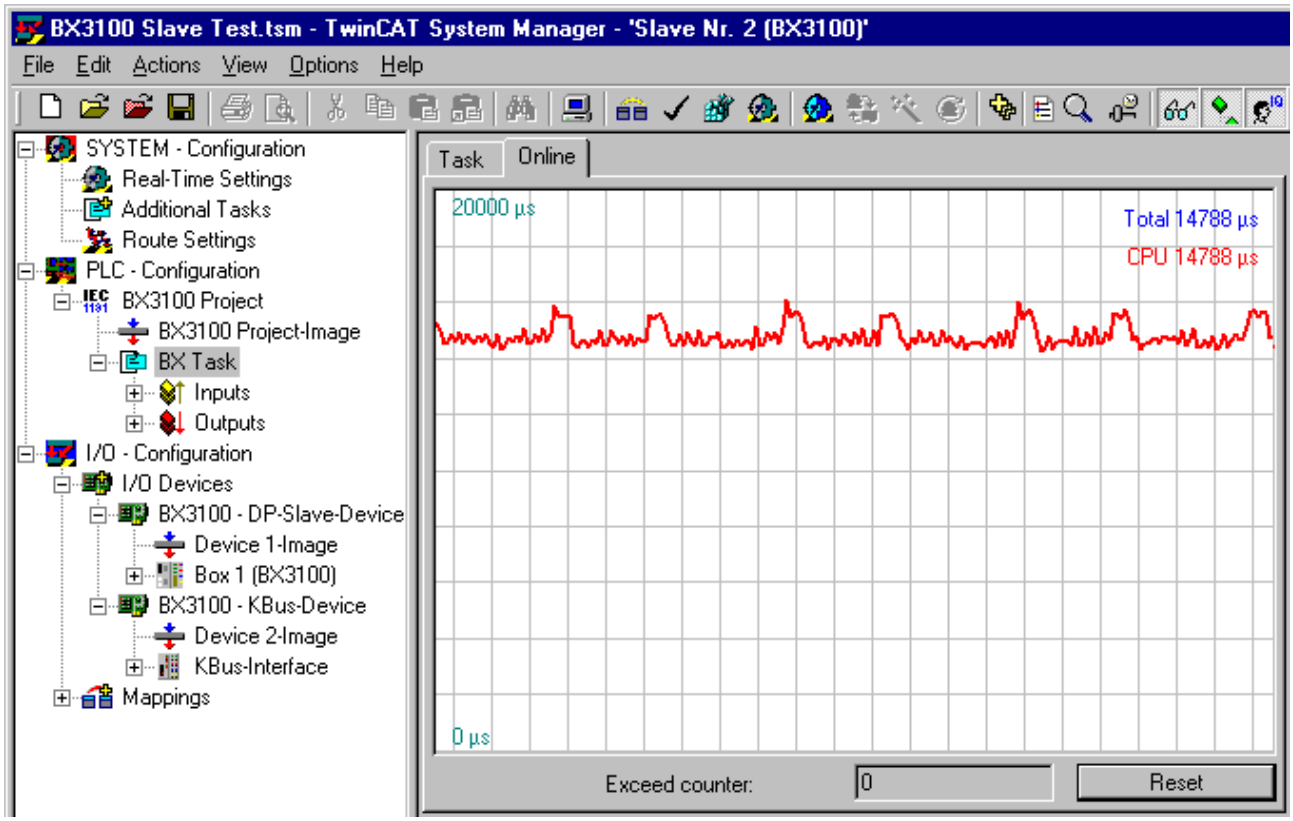



Fig. 46: Displaying the PLC cycle time

## 4.2.10 KS2000 - Introduction

The KS2000 configuration software permits configuration, commissioning and parameterization of bus couplers, of the affiliated bus terminals and of Fieldbus Box Modules. The connection between bus coupler / Fieldbus Box Module and the PC is established by means of the serial configuration cable or the fieldbus.


Fig. 47: KS2000 configuration software

### Configuration

You can configure the Fieldbus stations with the Configuration Software KS2000 offline. That means, setting up a terminal station with all settings on the couplers and terminals resp. the Fieldbus Box Modules can be prepared before the commissioning phase. Later on, this configuration can be transferred to the terminal station in the commissioning phase by means of a download. For documentation purposes, you are provided with the breakdown of the terminal station, a parts list of modules used and a list of the parameters you have modified. After an upload, existing fieldbus stations are at your disposal for further editing.

### Parameterization

KS2000 offers simple access to the parameters of a fieldbus station: specific high-level dialogs are available for all bus couplers, all intelligent bus terminals and Fieldbus Box modules with the aid of which settings can be modified easily. Alternatively, you have full access to all internal registers of the bus couplers and intelligent terminals. Refer to the register description for the meanings of the registers.

### Commissioning

The KS2000 software facilitates commissioning of machine components or their fieldbus stations: Configured settings can be transferred to the fieldbus modules by means of a download. After a *login* to the terminal station, it is possible to define settings in couplers, terminals and Fieldbus Box modules directly *online*. The same high-level dialogs and register access are available for this purpose as in the configuration phase.

The KS2000 offers access to the process images of the bus couplers and Fieldbus Box modules.

- Thus, the coupler's input and output images can be observed by monitoring.
- Process values can be specified in the output image for commissioning of the output modules.

All possibilities in the *online mode* can be used in parallel with the actual fieldbus mode of the terminal station. The fieldbus protocol always has the higher priority in this case.

## 4.2.11 Configuration software KS2000

Bus Terminal controllers of the BCxx50, BXxx20 and BXxx00 series cannot be parameterized and configured with the KS2000 configuration software. These devices must be configured with the TwinCAT System Manager.

The KS2000 configuration software offers configuration and diagnostic support for the Bus Terminals attached to the Bus Terminal Controller.
It is advisable to set the baud rate in the KS2000 configuration software and the BCxx50 BCxx20 and BXxx00 to 38400 baud (8 data bits, even, 1 stop bit).

**COM1 - automatic baud rate detection**

The COM 1 interface of the BXxx00 features automatic baud rate detection between 9.6 kbaud and 56.4 kbaud.

**Required KS2000 version**

Configuration or diagnostics of Bus Terminals at BXxx00 is supported from KS2000 version 4.3.14.

In some Bus Terminals (e.g. KL25xx, KL6811, KL6201, KL6401) the following parameters must be set in order to be able to use the configuration dialogs:

- A PLC project or boot project must be deactivated.
- The BX controller must be in its default configuration. Set the manufacturer’s setting or switch to Config Mode in the TwinCAT System Manager (blue TwinCAT icon).
- The BX controller must be in FreeRun mode. Activate it with the TwinCAT System Manager.

You can now log in with the KS2000 configuration software via ADS (port 100) or the serial cable and use the KS2000 dialogs in the Bus Terminals.

# 5 Programming

## 5.1 BCxx50 PLC features

| Description | Value |
|---|---|
| Data memory | 32 kbyte |
| Program memory | 48 kbyte minus task-configuration minus POUs during online change |
| Source code memory | 128 kbyte |
| RETAIN | 2 kbyte |
| INPUT | 2 kbyte |
| OUTPUT | 2 kbyte |
| FLAG | 4 kbyte |
| Max. variable size | 16 kbyte |
| Max. POUs | Limited by memory |

## 5.2 TwinCAT PLC

The Beckhoff TwinCAT Software System turns any compatible PC into a real-time controller with a multi-PLC system, NC axis control, programming environment and operating station. The TwinCAT programming environment is also used for programming the BC/BX. If you have TwinCAT PLC (Windows NT4/2000/XP) installed, you can use the fieldbus connection or the serial port for downloading and debugging software.

TwinCAT I/O or TwinCAT PLC can also be used as the Ethernet Master (host), in order to exchange process data with the Bus Terminal Controller. TwinCAT provides you with the System Manager as a configuration tool, as well as the drivers and the ADS protocol.

**Bus Terminal Controllers of the BCxx50, BCxx20 and BXxx00 series**

These 2nd-generation Bus Terminal Controllers are configured with the TwinCAT System Manager and programmed with TwinCAT PLC Control. TwinCAT PLC must be installed for these couplers (Windows NT4, Windows 2000, Windows XP).

**Programming and program transfer**

- via the serial interface [▶ 69]
- via the fieldbus interface (only for Bus Terminal controllers for PROFIBUS, CANopen and Ethernet)

**Online change**

The Bus Terminal Controllers of the BX series and the BCxx50 support online change. This means that the PLC program is replaced with a new program without interrupting the program. The switch-over to the new program occurs after the task is completed. This means that two versions of the PLC program have to be stored. 512 kbyte are available, which therefore have to be divided by two, leaving 256 kbyte for the actual PLC program. In addition, several kbyte are required for task configuration etc. During an online change, dynamic data are stored in memory. Should a program approach the memory limit (program size greater than 240 kbyte), the online change may no longer work, even though the program may still be written to the BX after "Rebuild all".

**When is online change not available?**

Online change is not available under certain conditions,.

- Inserting of a new library
- Changing the task setting
- "Rebuild all"

- Controller memory limit is almost reached (PLC program greater than 90%)

## 5.3 TwinCAT PLC - Error codes

| Error type | Description |
|---|---|
| PLC compiler error | Maximum number of POUs (...) exceeded |
| PLC compiler error | Out of global data memory ... |

**Error POUs**

For each function block one POU (process object unit) is created. 256 function blocks are available by default.

Error 3612: Maximum number of POUs (100) exceeded! Compile is aborted.
Data allocation
1 Error(s), 0 Warning(s).

Fig. 48: Maximum number of POUs exceeded

If libraries are integrated this value may be insufficient. In this case, the number of POUs should be increased.

To this end, open in PLC Control under Projects/Options...

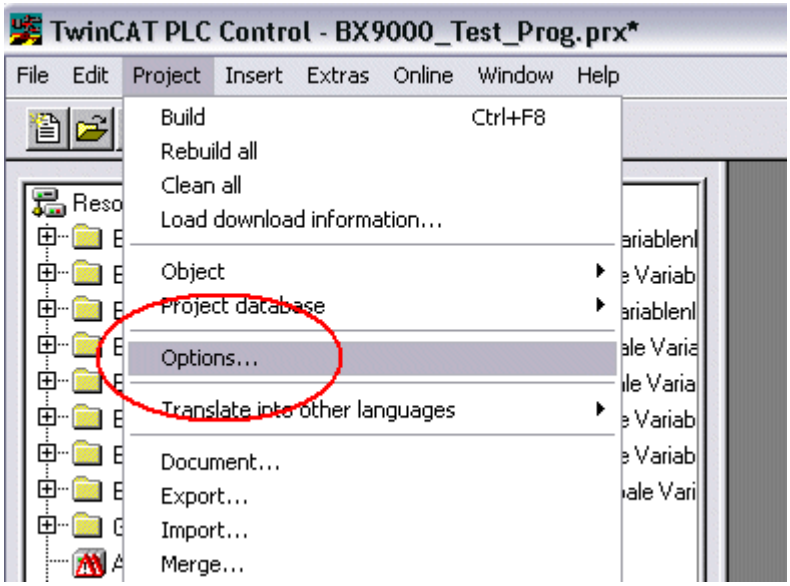



Fig. 49: Menu path Projects / Options / Controller Settings

...the controller settings.

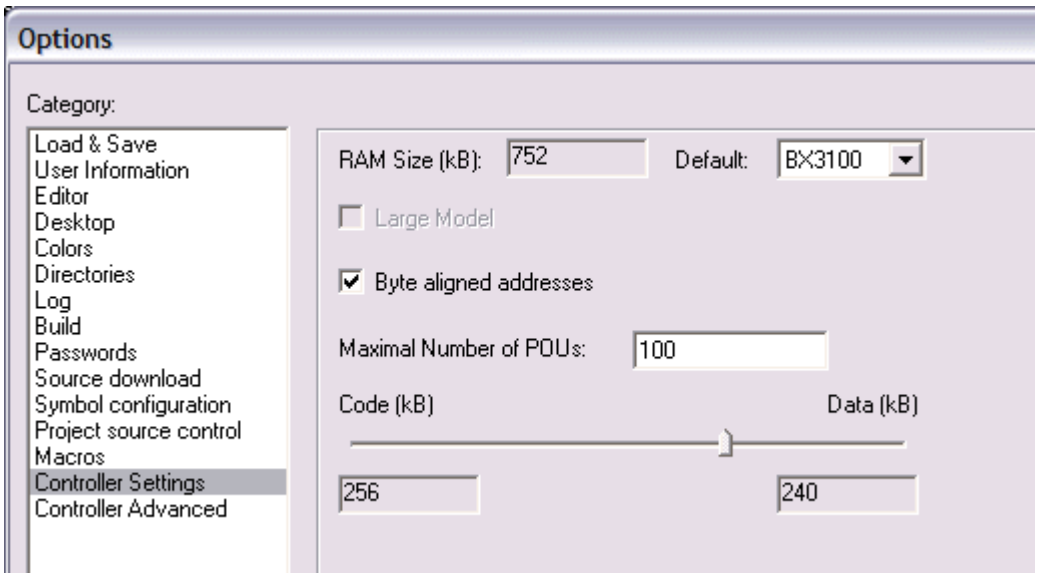



Fig. 50: Controller settings

Changing these settings will deactivate online changes.

**Global memory error**

```
Interface of POU 'MAIN'
Data allocation
Error 3803: MAIN (7): Out of global data memory. Variable 'Test_', 16002 bytes.
1 Error(s), 0 Warning(s).
```

Fig. 51: Global memory insufficient

2 x 16 kbyte of data are available by default. If large data quantities are to be used, this range should be increased. A maximum of 14 data segments are possible for the BX.

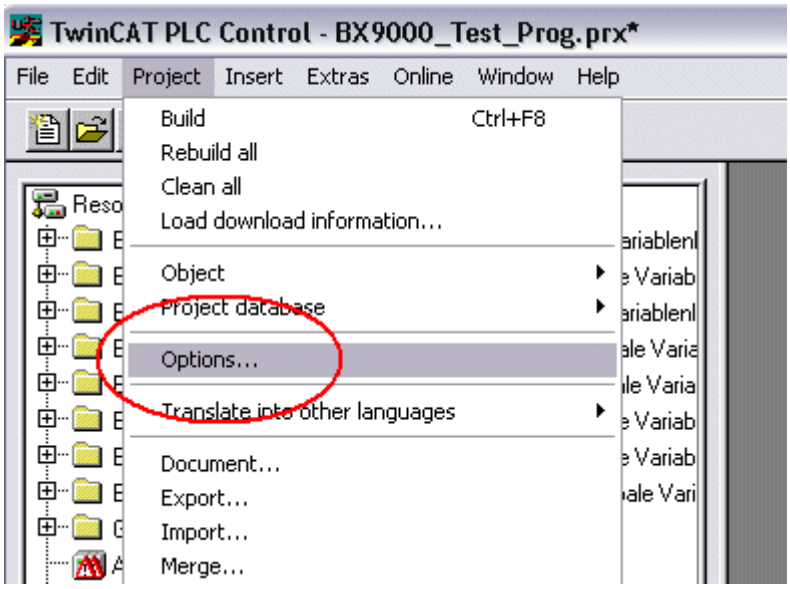



Fig. 52: Menu path Projects / Options / Build

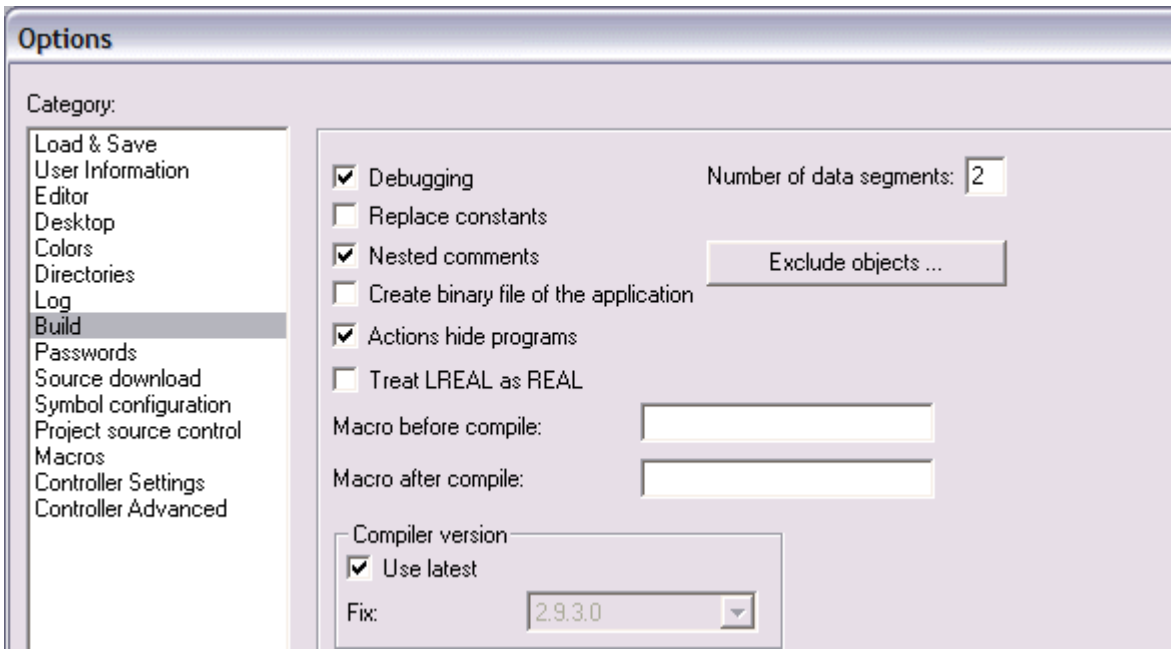



Fig. 53: Build

## 5.4 Remanent data

2000 kbyte of remanent data are available for the BX controller. These data are declared as VAR RETAIN in PLC Control:

**Example**

```
VAR RETAIN
    Test    :BOOL;
    Count   :INT;
END_VAR
```

Retain data are located between VAR RETAIN and END_VAR. These data are stored in a NOVRAM and are consistent across the whole 2 kbyte range. The RETAIN data are stored in the NOVRAM after each cycle. For 2 kbyte approx. 2 ms are required (for 1 kbyte approx. 1 ms). The variables can be configured locally or globally. Allocated variables (%MB, %QB, %IB) cannot be used as remanent data.

**Do not use VAR_RETAIN in function blocks**

VAR_RETAIN should not be used in function blocks. All FB data are copied into the retain memory. This leads to an unnecessary increase in cycle time, and the retain memory is filled with unnecessary data.

**Do not use variables with address as remanent data**

Variables that have been assigned an address (%MB, %QB, %IB) must not be used as remanent data.

**Example for remanent data in the function block**

This should be avoided, if possible, since all the data of a function block, in which even just a single remanent bit is found, are stored by default. A program sample can be found below.

**Function block test (no program code required - in ST semicolon is sufficient)**

```
FUNCTION_BLOCK Test
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
END_VAR
```

```
VAR_IN_OUT
    Counter   :INT;
END_VAR
```

### MAIN program

```
PROGRAM MAIN
VAR
    fb_Test:Test;
END_VAR
VAR RETAIN
    iCounter1:INT;
END_VAR
```

```
fb_Test(Counter:=iCounter1);
```

## 5.5 Allocated flags

4 kbyte of allocated flags are available. They can be used to assign different variable types to the same address, e.g. for converting strings to bytes. Data can also be placed here that can be read or written via ADS by the controller.

**Allocated variables are not remanent data**

For the Bus Terminal Controllers of the BX series and the BCxx50 the allocated variables are **not** saved as remanent data.

### Reading/writing of allocated flags via ADS

The flags may also be read via the controller and ADS. In PROFIBUS, the DPV-1 services are used for this purpose, in CANopen SDO communication is used.
The AmsNetID can be obtained from the System Manager, or it can be displayed via the Bus Terminal Controller menu.
The PLC port number is 800.

| Index group | Meaning | Index offset (value range) |
|---|---|---|
| 0x4020 | Flag (only BXxxx0) | 0..4096 |

### Example

BX program

```
VAR
    Flag_01 AT %MB0: WORD;
END_VAR
```

TwinCAT PC/CX master program

```
VAR
    fbADRSREAD: ADSREAD;
    Flag_M: WORD;
END_VAR
```

```
fbADRSREAD(
    NETID:='172.16.3.0.2.3' ,  (* AMSNetId BX *)
    PORT:=800 ,                (* 800 - PLC *)
    IDXGRP:=16#4020 ,          (* 0x4020hex falgs *)
    IDXOFFS:=0 ,               (* byte offset *)
    LEN:=2 ,                   (* Lenght byte *)
    DESTADDR:=ADR(Merker) ,
    READ:=TRUE ,
    TMOUT:=t#1s );
IF NOT fbADRSREAD.BUSY THEN
    fbADRSREAD(READ:=FALSE);
END_IF
```

# 5.6 Local process image in delivery state

The process image of the Bus Terminal Controller consists of input, output and flag areas. In addition, there are unallocated data without fixed address. They are created without specifying an address. 256 kB of memory space is reserved on the Bus Terminal Controller for this type of variable. For the allocated data 2048 bytes of input data and 2048 bytes of output data are available. In the default configuration, i.e. in delivery state of the Bus Terminal Controller, all connected Bus Terminals are assigned a fixed address. The data for the CANopen communication are stored from address offset 1000 dec. In the default configuration the number of CANopen data is 4 x kbyte (4 PDOs).

| INPUTS | OUTPUTS |
|---|---|
| Bus Terminal %IB0 ... | Bus Terminal %QB0 ... |
| CANopen DATA (PLC variables) %IB1000 ... | CANopen DATA (PLC variables) %QB1000 ... |
| ... %IB2047 maximal | ... %QB2047 maximal |

### CANopen data

| PDO number | Read/Write | BX process image |
|---|---|---|
| PDO 1 | Rx/Tx | %IB1000...%IB1007/QB1000...%QB1007 |
| PDO 2 | Rx/Tx | %IB1008...%IB1015/QB1008...%QB1015 |
| PDO 3 | Rx/Tx | %IB1016...%IB1023/QB1016...%QB1023 |
| PDO 4 | Rx/Tx | %IB1024...%IB1031/QB1024...%QB1031 |

Further PDO data (5-32) have to be enabled by the master (see CAN configuration).

### Addressing of the connected Bus Terminals

The default setting is for all the connected Bus Terminals to be assigned to the local process image.
Mapping within the Bus Terminal Controller is carried out according to the following rule:
First come all the complex Bus Terminals, in whatever sequence they are physically inserted, followed by the digital Bus Terminals which are padded to a whole byte. The default mapping of the complex Bus Terminals is:

- complete evaluation
- Intel format
- Word alignment

### Example

Bus Terminal Controller: 1 x BX5100
Position 1: 1 x KL1012
Position 2: 1 x KL1104
Position 3: 1 x KL2012
Position 4: 1 x KL2034
Position 5: 1 x KL1502
Position 6: 1 x KL3002
Position 7: 1 x KL4002
Position 8: 1 x KL6001
Position 9: 1 x KL9010

**Process image**

| Bus Terminal | Position | Input image | Output image |
|---|---|---|---|
| KL1501 | 5 | %IB0...%IB5 | %QB0...%QB5 |
| KL3002 | 6 | %IB6...%IB13 | %QB6...%QB13 |
| KL4002 | 7 | %IB14...%IB21 | %QB14...%QB21 |
| KL6001 | 8 | %IB22...%IB29 | %QB22...%QB29 |
| KL1012 | 1 | %IX30.0..30.1 | - |
| KL1104 | 2 | %IX30.1..30.5 | - |
| KL2012 | 3 | - | %QX30.0..30.1 |
| KL2034 | 4 | - | %QX30.2..30.5 |
| KL9010 | 9 | - | - |

**Adress of the Bus Terminals, which you have assigned to local PLCs (BCxx00)**

If you do not know the address of the Bus Terminals that you have assigned to the local PLC (BCxx00):
Perform your hardware configuration in the System Manager. After you have entered all the Bus Terminals and PLC variables, click with the right mouse button on the BCxx00 in the hardware tree, and select the menu item *Export variables information*.... A file is saved, and this file can be inserted in the System Manager under *Project - Import*. Now you will have the entry *TwinCAT import* under the global variables, and you will find here all the variables that you have assigned to the local PLC (BCxx00).

## 5.7 Mapping the Bus Terminals

The precise assignment of the byte-oriented Bus Terminals may be found in the configuration guide for the particular bus terminal. This documentation is available on the Beckhoff *Products & Solutions* CD or on the Internet under http://www.beckhoff.de.

| Byte oriented Bus Terminals | Bit oriented Bus Terminals |
|---|---|
| KL15x1 | KL10xx, KL11xx, KL12xx, KL17xx, KM1xxx |
| KL25xx | KL20xx, KL21xx, KL22xx, KL26xx, KL27xx, KM2xxx |
| KL3xxx | |
| KL4xxx | |
| KL5xxx | |
| KL6xxx | |
| KL7xxx | |
| KL8xxx | |
| | KL9110, KL9160, KL9210, KL9260 |

## 5.8 Creating a boot project

The following memory resources are available for generating the boot project

- approx. 250 kbyte flash on the Bus Terminal controllers of the BX series;
- approx. 48 kbyte flash on the Bus Terminal controllers of the BCxx50 series.

**PLC Control**

After logging into TwinCAT PLC Control, a boot project can be created.

- Opening a PLC project
- Selecting the target system (or selection the serial interface)
- Logging into the BX/BCxx50

- Creating a boot project (Online\Create boot project)

The PLC LED lights up green once a valid boot project is available on the BX/BCxx50.

In the Bus Terminal controllers of the BX series, the PLC LED flashes orange while boot project is created. The PLC LED lights up orange if no boot project is available on the BX.

### Deleting a boot project

The boot project can be deleted from the Bus Terminal Controller. The following steps must be followed:

- Opening the project
- Logging into the Bus Terminal Controller
- Deleting the boot project (Online\Delete boot project)

The PLC LED lights up orange when the boot project is deleted.

**Using the current project as boot project**

After an online change the old project is still shown as boot project. To use the current project (after the online change) as the boot project, the boot project has to be recreated.

### Bypassing the start of the boot project*

With the Bus Terminal controllers of the BX series, starting of the boot project during booting can be prevented by pressing the Navi button. This does not delete the boot project. The project is reloaded when the Bus Terminal Controller is rebooted.

* from version 0.85

## 5.9 Local process image in the TwinCAT configuration

The TwinCAT configuration (TwinCAT CONFIG) enables free mapping between fieldbus, K-bus and PLC variables. Variables can be linked independent of their address via the System Manager.

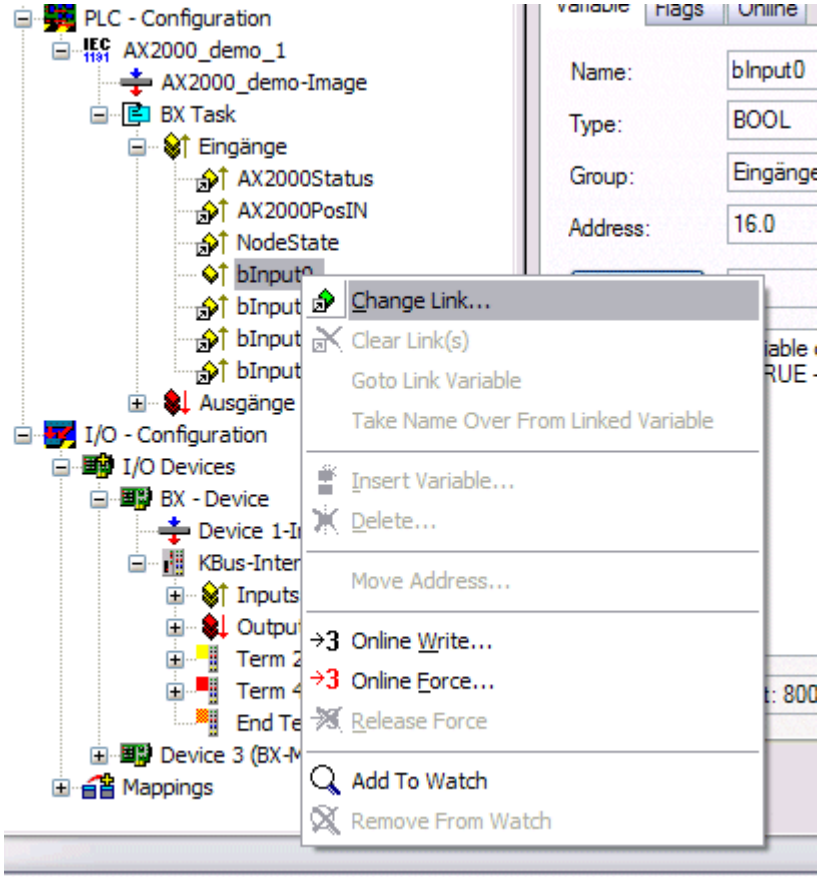



Fig. 54: Changing variable links

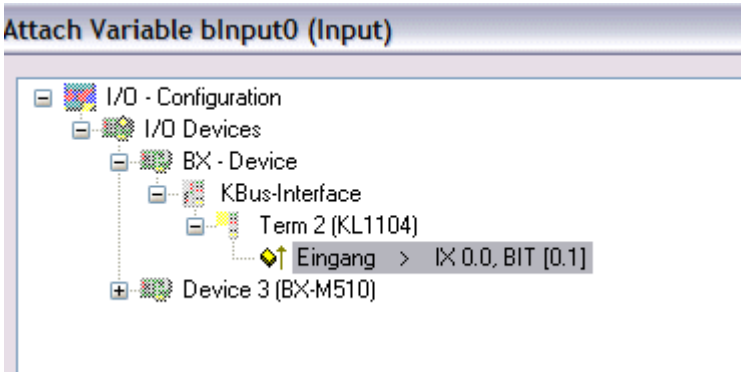



Fig. 55: Linking a variable with an input

In the default configuration all Bus Terminals are assigned fixed addresses. If a Bus Terminal is inserted, the whole address range may be shifted. The TwinCAT configuration enables allocated variables to be linked to a Bus Terminal, as required. This is parameterized in the System Manager, and the configuration is then downloaded to the Bus Terminal Controller (see TwinCAT configuration [▶ 29]). It is also possible to upload an existing TwinCAT configuration.

# 5.10 Communication between TwinCAT and BX/BCxx50

For transferring data from TwinCAT to the Bus Terminal Controller, it makes sense to organize the data in a structure. Please note the following to account for the differences in data management on the two systems.

- If two different data types are sent in sequence (e.g. byte and INT), the following variable is set to the next even address offset
- Boolean variables should never be allocated individually within a structure, since they would invariably occupy 1 byte. Boolean expressions should always be masked in a byte or word.

**Example 1: A structure on the BX/BCxx50 and on the PC**

| Variable | BX/BCxx50 memory | PC memory (TwinCAT) |
|---|---|---|
| Byte | %..B0 | %..B0 |
| INT (1) | %..B2 | %..B1 |
| INT (2) | %..B4 | %..B3 |

Due to the fact that another variable type (INT) follows the first byte, in the BX/BCxx50 it was assigned the next free even address. In order to achieve the same data structure on both systems, a dummy byte has to be inserted in the PC project (see example 2).

**Example 2: A structure on the BX/BCxx50 and on the PC with the same memory allocation**

| Variable | BX/BCxx50 memory | PC memory (TwinCAT) |
|---|---|---|
| Byte | %..B0 | %..B0 |
| Byte (dummy) | %..B1 (not necessarily required, since the system deals with this itself if the variable does not exist) | %..B1 |
| INT (1) | %..B2 | %..B2 |
| INT (2) | %..B4 | %..B4 |

**Data structure**

```
Type PB_Data
STRUCT
    wVar_1:WORD;
    iValue_1:INT;
    iValue_2:INT;
    iValue_3:INT;
END_STRUCT
END_TYPE
```

**Creating a variable structure**

```
VAR_Global
    strData_Out AT %QB1000:PB_Data; (*PLC Variables *)
    bInput_01 AT %IX0.0:BOOL; (* Input from a terminal *)
END_VAR
```

**Small programming example**

```
strData_Out.wVar_1.0:=bInput_01;
```

**Do not use real values in a mixed data structure**

A mixed data structure should not contain real values. If this is nevertheless the case, the high and low words must be swapped in the BX/BCxx50 or in the TwinCAT master project. It is better to use an array of Real values or to transfer the Real values individually.

**Larger fieldbus data blocks**

You can transfer larger fieldbus data blocks, in order to have a reserve for your structure. Disadvantage: These reserves are then transferred with each fieldbus telegram, resulting in overload of the fieldbus communication.

## 5.11 Up- and downloading of programs

The Bus Terminal Controller has a memory for the source code. It can be used for storing the program, the task configuration, and the libraries. Should the memory be insufficient, the source code may be stored without task configuration and libraries. This takes up significant less memory space!

### General settings

The timing of the source code download to the target system can be specified via Edit/Options. Open the options menu.

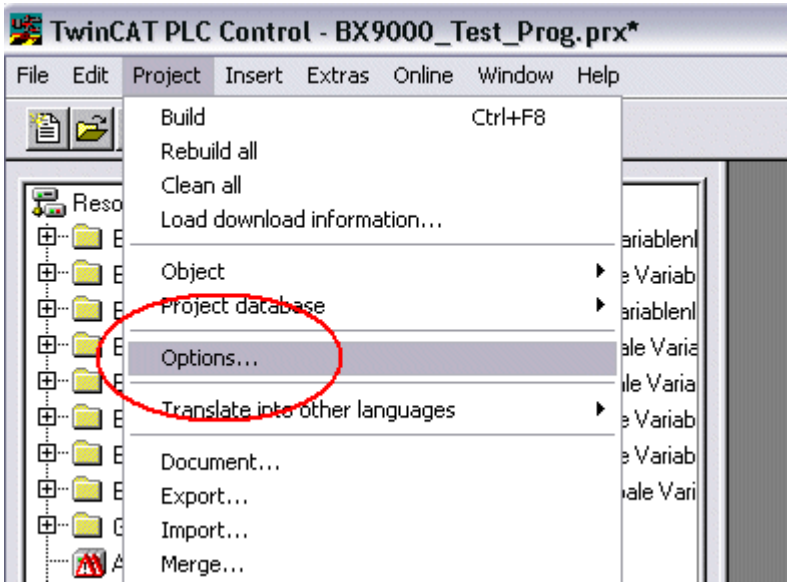



Fig. 56: Opening the options menu

Select Source Download.

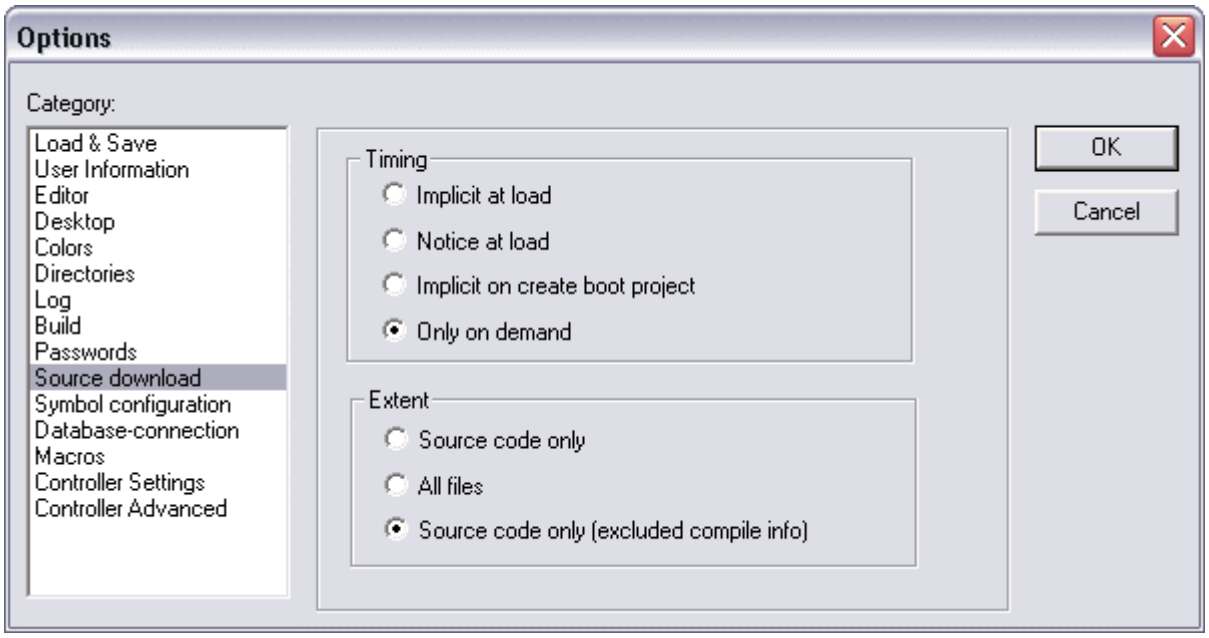



Fig. 57: Selecting Source Download

Here you can set which parts of the source code are to be downloaded to the Bus Terminal Controller, and when.

**Source code only:** the prx file with information on the online change is transferred. Login via online change is possible (the PLC does not stop).
**All files:** as *Source code only*, plus all required libraries.
**Source code only (compile info excluded):** only the prx file is transferred. Login is only possible when the PLC stops.

Which option you can use depends on the size of your projects.

### Downloading a program

The source code can be transferred to the target system on request. This requires the user to be logged in with his program. Under Online/Source code download the program code can now be transferred to the Bus Terminal Controller.

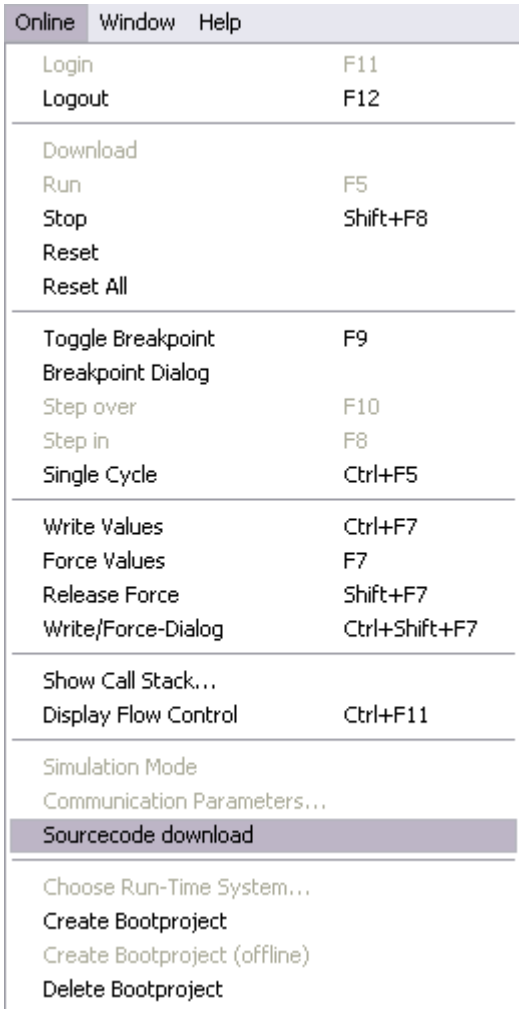



Fig. 58: Downloading the program code

After a short delay, a window will open that indicates the download progress.

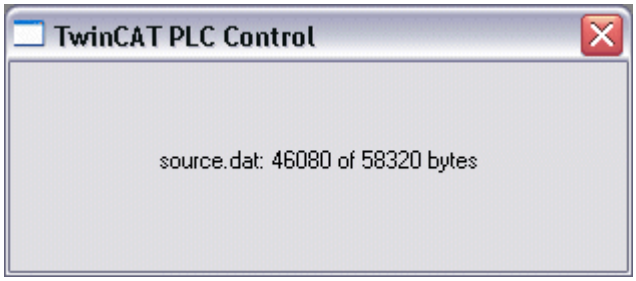



Fig. 59: Download progress

### Uploading a program

For uploading the program code again, open a new file in PLC Control. Then click on the PLC button.

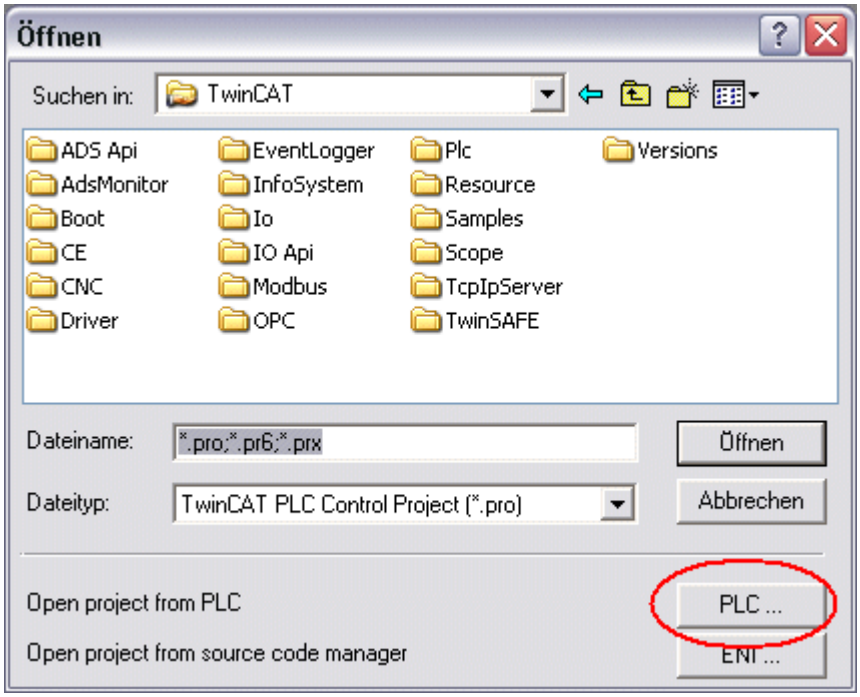



Fig. 60: Uploading a program

Select the data transfer route:

- *BCxx50 or BX via AMS*, if you are connected to the Bus Terminal Controller via the fieldbus, or
- *BCxx50 or BX via serial*, if you are connected to the Bus Terminal Controller via the serial interface.

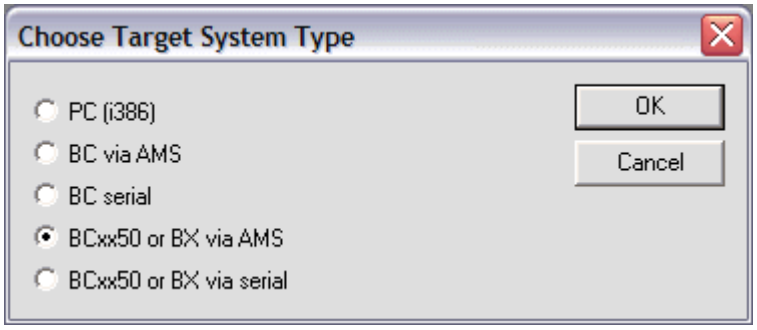



Fig. 61: Selecting the data transfer route

Then select the device and confirm with OK.

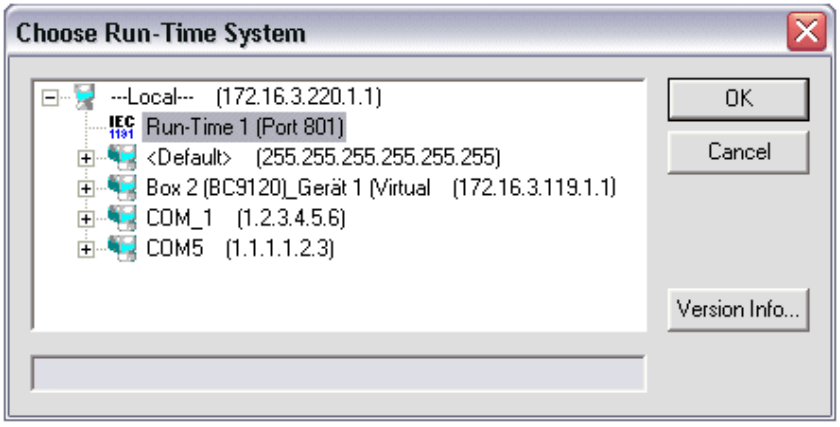



Fig. 62: Selecting the device

The source code will now be uploaded.

**Password**

You can protect your project with a password (in PLC Control Project/Options/Passwords).

# 5.12 Libraries

## 5.12.1 Libraries overview

The TwinCAT Automation Software offers various libraries for the BCxx50 series Bus Terminal Controllers (Bus Coupler with PLC functionality) (see BECKHOFF Information System).

**Download**

The libraries are also included in this documentation. To extract the libraries, left-click on the link and copy the libraries to directory TwinCAT\PLC\LIB.

- Standard (https://infosys.beckhoff.com/content/1033/bc5150/Resources/zip/3730726795.zip)

- TcSystemBCxx50 (https://infosys.beckhoff.com/content/1033/bc5150/Resources/zip/3730728971.zip)

 TcSystemBCxx50 requires the TcBaseBCxx50 library.

- TcBaseBCxx50 (https://infosys.beckhoff.com/content/1033/bc5150/Resources/zip/3730731147.zip)

- ChrAscBX.lbx (https://infosys.beckhoff.com/content/1033/bc5150/Resources/zip/3730733323.zip)

**Use the library that matches the firmware**

The latest firmware requires the latest library. If you update your Bus Terminal Controller, please also update the libraries.
Copy these libraries to the LIB directory, remove these libraries from your project and add them again.

**TcSystemBCxx50**

| ADS | Version | Firmware | | | | |
|---|---|---|---|---|---|---|
| | | BC3150 | BC5150 | BC5250 | BC8150 | - |
| ADSREAD | | B0 | B0 | B1 | B0 | - |
| ADSWRITE | | B0 | B0 | B1 | B0 | - |
| ADSRDWRT | | B0 | B0 | B1 | B0 | - |
| ADSWRTCTL | | B0 | B0 | B1 | B0 | - |
| ADSRDSTATE | | B0 | B0 | B1 | B0 | - |
| ADSRDDEVINFO | | B0 | B0 | B1 | B0 | - |

| Bit Functions | Version | Firmware | | | | |
|---|---|---|---|---|---|---|
| | | BC3150 | BC5150 | BC5250 | BC8150 | - |
| CLEARBIT32 | | B0 | B0 | B1 | B0 | - |
| CSETBIT32 | | B0 | B0 | B1 | B0 | - |
| GETBIT32 | | B0 | B0 | B1 | B0 | - |
| SETBIT32 | | B0 | B0 | B1 | B0 | - |

| Controller | Version | Firmware | | | | |
|---|---|---|---|---|---|---|
| | | BC3150 | BC5150 | BC5250 | BC8150 | - |
| FB_BasicPID | - | B0 | B0 | B1 | B0 | - |
| - | - | - | - | - | - | - |

| File Access | Version | Firmware | | | | |
|---|---|---|---|---|---|---|
| | | BC3150 | BC5150 | BC5250 | BC8150 | - |
| FB_ReadFromFile | | - | - | - | - | - |
| FB_WriteToFile | | - | - | - | - | - |
| FB_ReadWriteFile | | - | - | - | - | - |

| Memory Functions | Version | Firmware | | | | |
|---|---|---|---|---|---|---|
| | | BC3150 | BC5150 | BC5250 | BC8150 | - |
| MEMCMP | | B0 | B0 | B1 | B0 | - |
| MEMCYP | | B0 | B0 | B1 | B0 | - |
| MEMMOVE | | B0 | B0 | B1 | B0 | - |
| MEMSET | | B0 | B0 | B1 | B0 | - |

| NOVRAM Functions | Version | Firmware | | | | |
|---|---|---|---|---|---|---|
| | | BX3100 | BX5100 | BX5200 | BX8000 | - |
| - | - | - | - | - | - | - |

| SFC | Version | Firmware | | | | |
|---|---|---|---|---|---|---|
| | | BC3150 | BC5150 | BC5250 | BC8150 | - |
| AnalyzeExpression | | - | - | - | - | - |
| AppendErrorString | | - | - | - | - | - |
| SFCActionControl | | - | - | - | - | - |

| System / Time / TBus | Version | Firmware | | | | |
|---|---|---|---|---|---|---|
| | | BC3150 | BC5150 | BC5250 | BC8150 | - |
| DRAND | | B0 | B0 | B1 | B0 | - |
| SYSTEMTIME_TO_DT | | B0 | B0 | B1 | B0 | - |
| DT_TO_SYSTEMTIME | | B0 | B0 | B1 | B0 | - |
| GetSysTick | | B0 | B0 | B1 | B0 | - |
| PresetSysTick | | B0 | B0 | B1 | B0 | - |
| Reboot | | B0 | B0 | B1 | B0 | - |

| Debug | Version | Firmware | | | | |
|---|---|---|---|---|---|---|
| | | BC3150 | BC5150 | BC5250 | BC8150 | - |
| F_ReadDebugTimer | | B0 | B0 | B1 | B0 | - |
| F_StartDebugTimer | | B0 | B0 | B1 | B0 | - |

## 5.12.2 TcBaseBX

### 5.12.2.1 System task information

```
VAR_GLOBAL
    SystemTaskInfo  :  SYSTEMTASKINFOTYPE;
END_VAR
```

System flags are implicitly declared variables. Using the Input Assistant, a variable SystemTaskInfoArr can be found under system variables. This variable is a field with four structures of type SYTEMTASKINFOTYPE [▶ 66]. The structure definition can be found in the system library. The index in this field is the task ID.

| Development environment | Target platform | PLC libraries to be linked |
|---|---|---|
| TwinCAT v2.9.0 | BCxx50, BC9x20 Controller | TcBaseBCxx50.lbx |

### 5.12.2.2 System Task Info Type

```
TYPE SYSTEMTASKINFOTYPE
STRUCT
    active              :       BOOL;
    taskName            :       STRING(16);
    firstCycle          :       BOOL;
    cycleTimeExceeded   :       BOOL;
    cycleTime           :       UDINT;
    lastExecTime        :       UDINT;
    priority            :       BYTE;
    cycleCount          :       UDINT;
END_STRUCT
END_TYPE
```

**Key**

active: This variable indicates whether the task is active.
taskName: the task name.
firstCycle: During the first PLC task cycle, this variable has the value: TRUE.
cycleTimeExceeded: this variable indicates whether the set task cycle time was exceeded.
cycleTime: set task cycle time in multiples of 100 ns.
lastExecTime: cycle time required for the last cycle in multiples of 100 ns.
priority: set task priority.
cycleCount: cycle counter.

| Development environment | Target platform | PLC libraries to be linked |
|---|---|---|
| TwinCAT v2.9.0 | BCxx50, BC9x20 Controller | TcBaseBCxx50.lbx |

### 5.12.2.3 System info

```
VAR_GLOBAL
    SystemInfo     : SYSTEMINFOTYPE;
END_VAR
```

System flags are implicitly declared variables. Using the Input Assistant, a variable Systeminfo can be found under system variables. The type SYSTEMINFOTYPE [▶ 66]is declared in the system library. For accessing the variable, the system library has to be integrated in the project.

| Development environment | Target platform | PLC libraries to be linked |
|---|---|---|
| TwinCAT v2.9.0 | BCxx50, BC9x20 Controller | TcBaseBCxx50.lbx |

### 5.12.2.4 System information type

```
TYPE SYSTEMINFOTYPE
STRUCT
    runTimeNo           :       BYTE;
    projectName         :       STRING(32);
    numberOfTasks       :       BYTE;
    onlineChangeCount   :       UINT;
    bootDataFlags       :       BYTE;
    systemStateFlags    :       WORD;
END_STRUCT
END_TYPE
```

**Key**

runTimeNo: indicates the number of the runtime system (1).
projectName: project name as STRING.
numberOfTasks: number of tasks contained in the runtime system (max. 1).
onlineChangeCount: number of online changes since the last complete download.
bootDataFlags: Reserved
systemStateFlags: Reserved.

| Development environment | Target platform | PLC libraries to be linked |
|---|---|---|
| TwinCAT v2.9.0 | BCxx50, BC9x20 Controller | TcBaseBCxx50.lbx |

## 5.12.3 ADS

### 5.12.3.1 Local ADS Port Numbers

| Port number | Description |
|---|---|
| 100 [▶ 67] dec | Reading and writing of registers and tables from the coupler and the complex Bus Terminals |
| 801 [▶ 67] dec | Local PLC process image |

### 5.12.3.2 ADS services

**Local process image task 1 port 800 or 801**

Data can be read from and written to the local process image. If it is necessary for outputs to be written, it is important to ensure that they are not used by the local PLC, because the local controller will overwrite these values. The data are not associated with a watchdog, and therefore must not be used for outputs that would have to be switched off in the event of a fault.

| Index group | Meaning | Index offset (value range) |
|---|---|---|
| 0xF020 | Inputs | 0...2047 |
| 0xF030 | Outputs | 0...2047 |
| 0x4020 | Flags | 0...4095 |

**ADS services**

**AdsServerAdsState**

| Data type (read only) | Meaning |
|---|---|
| String | Start - the local PLC is running<br>Start - the local PLC is stopped |

**AdsServerDeviceState**

| Data type (read only) | Meaning |
|---|---|
| INT | 0: Start - the local PLC is running<br>1: Start - the local PLC is stopped |

**AdsServerType**

| Data type (read only) | Meaning |
|---|---|
| String | BX PLC Server |

### Register Port 100

On the Bus Terminal Controllers of the BX series, and on the BCxx50, the ADS port number for register communication is fixed at 100.

| Index Group | Index offset (value range) | | Meaning |
|---|---|---|---|
| | Hi-Word | Lo-Word | |
| 0 [READ ONLY] | 0...127 | 0..255 | Registers in the Bus Coupler<br>Hi-Word, table number of the Bus Coupler<br>Lo-Word, register number of the table |
| 1...255 | 0...3 | 1-255 | Register of the Bus Terminals<br>Hi-Word, channel number<br>Lo-Word, register number of the Bus Terminal |

**Minimum timeout**

When reading the register, the time out of the ADS block has to be set to a time longer than 1 second.

**Setting the password**

When writing to the registers, the password has to be set (see the documentation for the particular Bus Terminal).

## 5.12.4 BX debugging function

### 5.12.4.1 BX debugging function

These functions can be used for measuring command execution times in a PLC project. The unit is a tick. One tick corresponds to 5.12 µs.

#### Start Debug Timer function

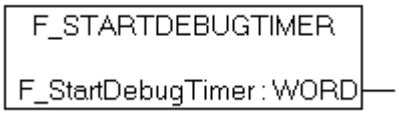



Fig. 63: Function block F_STARTDEBUGTIMER

Calling this function starts the timer. The return value is "0".

#### Read Debug Timer function

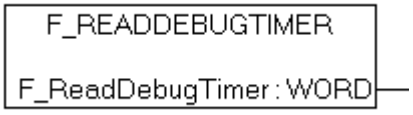



Fig. 64: Function block F_READDEBUGTIMER

This function reads the timer value. The return value has to be multiplied with 5.12 µs.

#### Example

```
VAR
    Timer_BX     :WORD;
    i            :INT;
END_VAR
```

Program

```
F_STARTDEBUGTIMER();
For i:=0 to 1000 do
    ;
END_FOR
Timer_BX:=F_READDEBUGTIMER();
```

# 5.13 Program transfer

## 5.13.1 Program transfer via the serial interface

Every Bus Terminal Controller can be programmed via the PC's RS232 interface.

Select the serial interface in TwinCAT PLC Control.

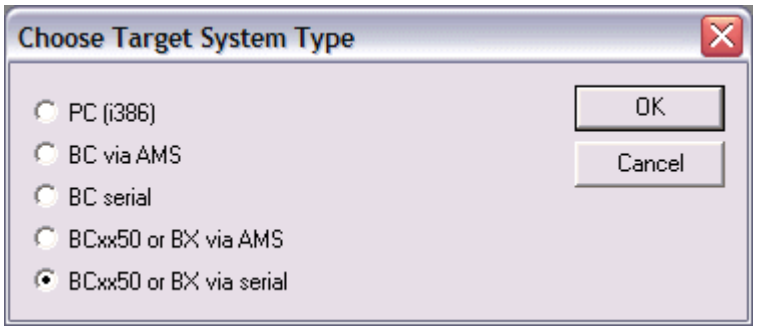



Fig. 65: *Selecting the data transfer route - serial interface*

The settings for the serial interface, port number, baud rate etc. are found under Online/Communication parameters in PLC Control.

The Bus Terminal Controller requires the following setting:

- Baud Rate: 9600/19200/38400/57600 baud (automatic baud rate detection)
- Stop bits: 1
- Parity: Straight line

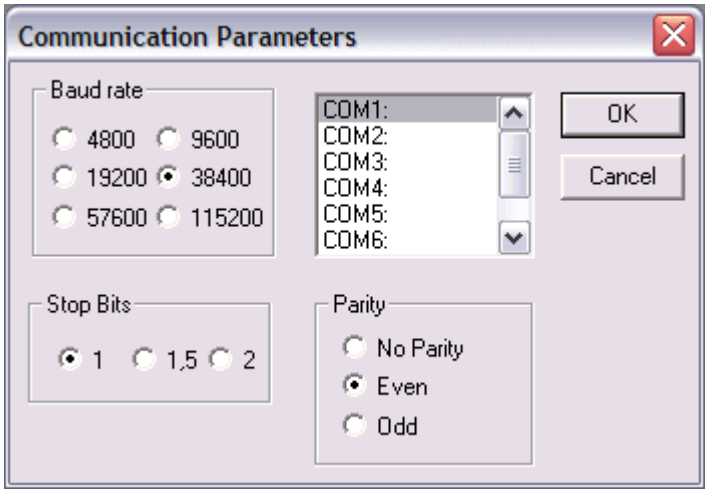



Fig. 66: Parameterization of the serial interface

**Program transfer via the serial interface and ADS**

The Bus Terminal Controller can be programmed via the PC's RS232 interface. Before you can work with the Bus Terminal Controller, TwinCAT must be notified of it (see serial ADS [▶ 38]).

Select the ADS connection in TwinCAT PLC Control.

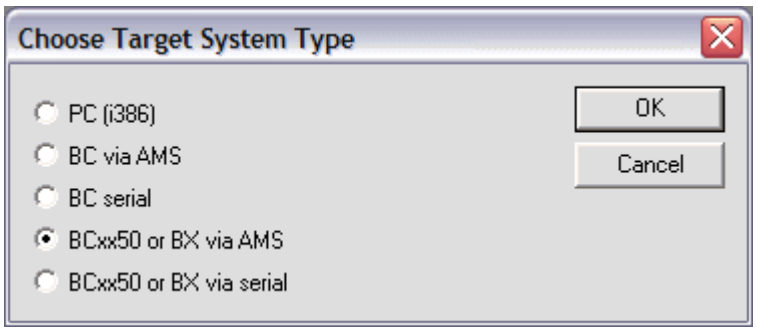



Fig. 67: *Selecting the data transfer route - AMS*

PLC Control can be accessed via *Online/Communication Parameters...*.

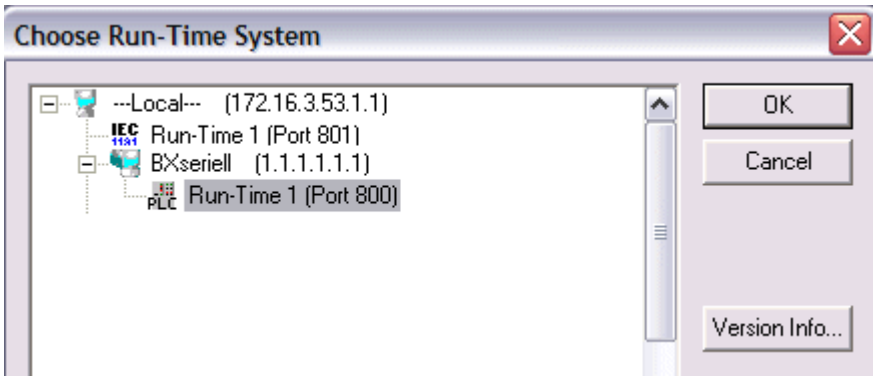



Fig. 68: Selecting the device

## 5.13.2 Programming via CANopen

TwinCAT offers the option to transfer the user program to the Bus Terminal Controller via the fieldbus. The BC/BX can be selected as the target system in PLC Control, after saving in the registry and restarting the TwinCAT system. The TwinCAT-level TwinCAT PLC is necessary.

**Minimum requirements**

FC510x with firmware from 1.55
TwinCAT 2.9 Build 948

**Initializing the Bus Terminal Controller**

The coupler must first be made known to the system before it can be selected in PLC Control.
Enter the Bus Terminal Controller in the System Manager, specify type, quantity and size of the fieldbus variables and link them with a task. For the subsequent program download via CANopen, the ADS interface must be enabled for the Bus Terminal Controller in the ADS tab. Save your settings and activate the configuration. Then start the TwinCAT system and the cyclic task.

### TwinCAT System Manager

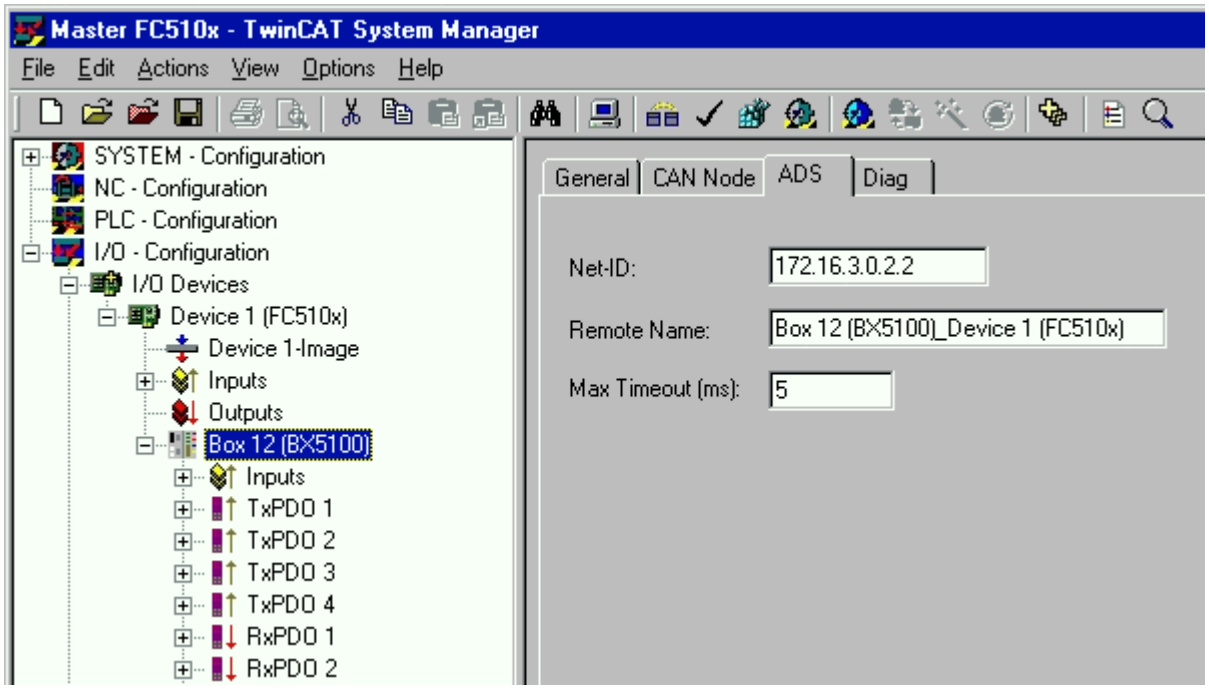



Fig. 69: Display of the BX5100 in the TwinCAT System Manager

### PLC Control

When TwinCAT PLC Control is restarted, TwinCAT asks for the target platform, i.e. the device on which the user program is later to run. TwinCAT offers two target platforms as controller, the PC or the Bus Terminal Controller.

Two options are available to you for transmission to the Bus Terminal Controller:

- AMS for BCxx00 (Bus Terminal Controller without online change, one task)
- AMS for BCxx50 and BX (Bus Terminal Controller with online change, two tasks)
- BC serial – the serial cable for communication via the RS232 interface of the PC and the programming interface of the Bus Terminal Controller

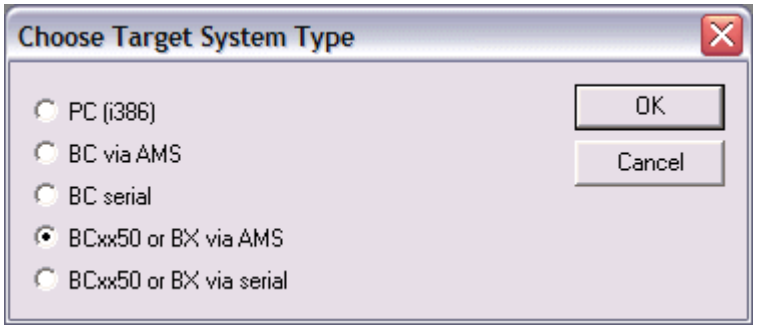



Fig. 70: Choose Target System

After your program has been created, select the target system under the *Online* toolbar. TwinCAT must be running to do this. In the sample, this is the Ethernet card with Box 2 and the Run-Time 1 of the Bus Terminal Controller.

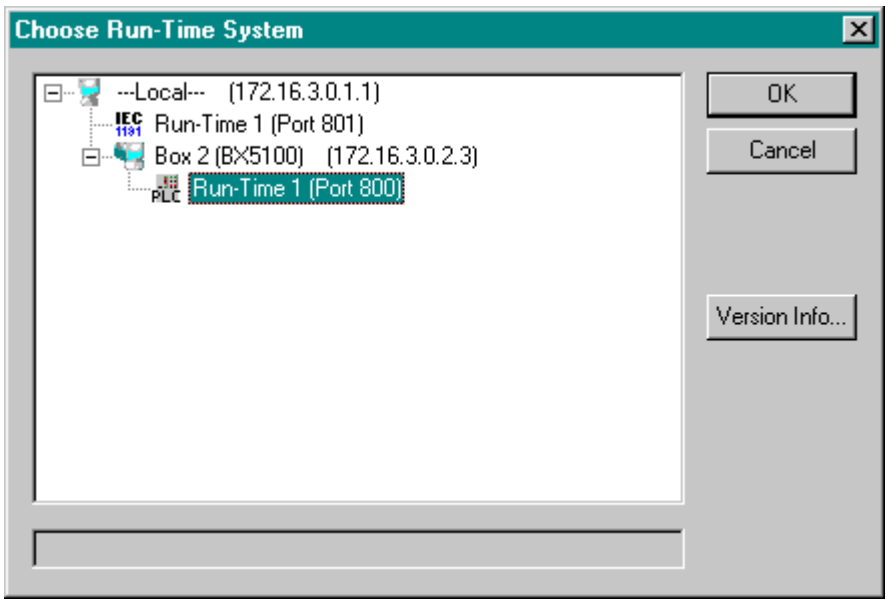



Fig. 71: Selecting the target system - box 2, runtime 1 of the Bus Terminal Controller

# 6 Automatic PDO Mapping

**BK51x0, IL23x0-B510**

PDO1 and PDO2 are occupied, as described, with digital and analog process data. For each further PDO the CANopen node uses the procedure shown in the flow diagram below and assigns process data to the PDOs in the following order:

1. Digital I/Os (if more than 64 are present)
2. 1-bytes special terminals
3. Analog I/Os
4. 2-bytes special terminals
5. 3-bytes special terminals
   …
6. 10. 8-bytes special terminals

Data types are not mixed! A new PDO is taken for each new data type (sample see below).

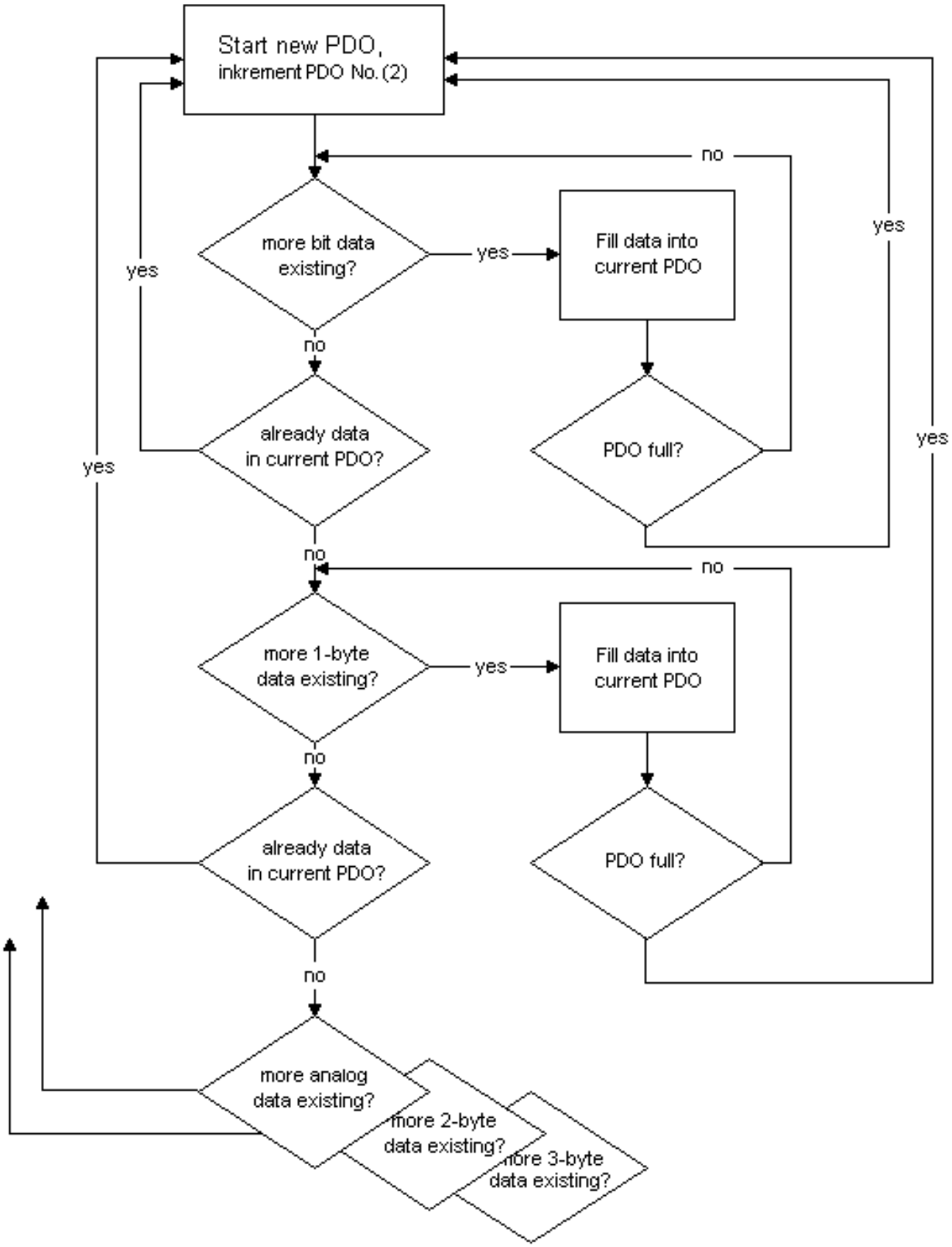



Fig. 72: Automatic PDO Mapping

**Sample**

A BK5120 (CANopen Coupler) has:

- 78 digital inputs and 48 digital outputs
- 6 analog inputs and 4 analog outputs
- a KL5001 (SSI encoder interface: 4 byte inputs by default)

- a KL6001 (serial interface: 4 byte inputs and 4 byte outputs by default)
- a KL5111 (incremental encoder interface) (6-byte inputs and 6-byte outputs)
- a KL6201 AS-i master terminal with default setting (22-byte process data interface)

| PDO | Data content (Mapping) | Object directory | PDO | Data content (Mapping) | Object directory |
|---|---|---|---|---|---|
| RxPDO1 | 5-byte digital outputs 1...48 | 0x6200, SI 1...5 | TxPDO1 | 8-byte digital inputs 1...64 | 0x6000, SI 1..8 |
| RxPDO2 | 8-byte analog outputs 1..4 | 0x6411, SI 1...4 | TxPDO2 | 4-byte analog inputs 1...4 | 0x6401, SI 1...4 |
| RxPDO3 | 4-byte serial interface | 0x2900, SI 1 | TxPDO3 | 2-byte digital inputs 65...78 | 0x6000, SI 9...10 |
| RxPDO4 | 6-byte encoder outputs | 0x2D00, SI 1 | TxPDO4 | analog inputs 5 and 6 | 0x6401, SI 5...6 |
| RxPDO5 | 8-byte ASI master 1: parameter data block | 0x3100, SI 1 | TxPDO5 | 8 bytes: 4-byte SSI and 4-byte serial interface | 0x2800, SI 1...2 |
| RxPDO6 | 8-byte ASI master 1: Process data block outputs ASI slave 1...15 | 0x3100, SI 2 | TxPDO6 | 6 bytes encoder input | 0x2C00, SI 1 |
| RxPDO7 | 8-byte ASI master 1: Process data block outputs ASI slave 16...31 | 0x3100, SI 3 | TxPDO7 | 8-byte ASI master 1: parameter data block | 0x3000, SI 1 |
| | | | TxPDO8 | 8-byte ASI master 1: Process data block inputs ASI slave 1...15 | 0x3000, SI 2 |
| | | | TxPDO9 | 8-byte ASI master 1: Process data block inputs ASI slave 16...31 | 0x3000, SI 3 |

# 7 CANopen Communication

## 7.1 CANopen Introduction

Fig. 73: CANopenLogo

CANopen is a widely used CAN application layer, developed by the CAN-in-Automation association (CiA, http://www.can-cia.org), and which has meanwhile been adopted for international standardization.

### Device Model

CANopen consists of the protocol definitions (communication profile) and of the device profiles that standardize the data contents for the various device classes. Process data objects (PDO) [▶ 82] are used for fast communication of input and output data. The CANopen device parameters and process data are stored in a structured object directory. Any data in this object directory is accessed via service data objects (SDO). There are, additionally, a few special objects (such as telegram types) for network management (NMT), synchronization, error messages and so on.

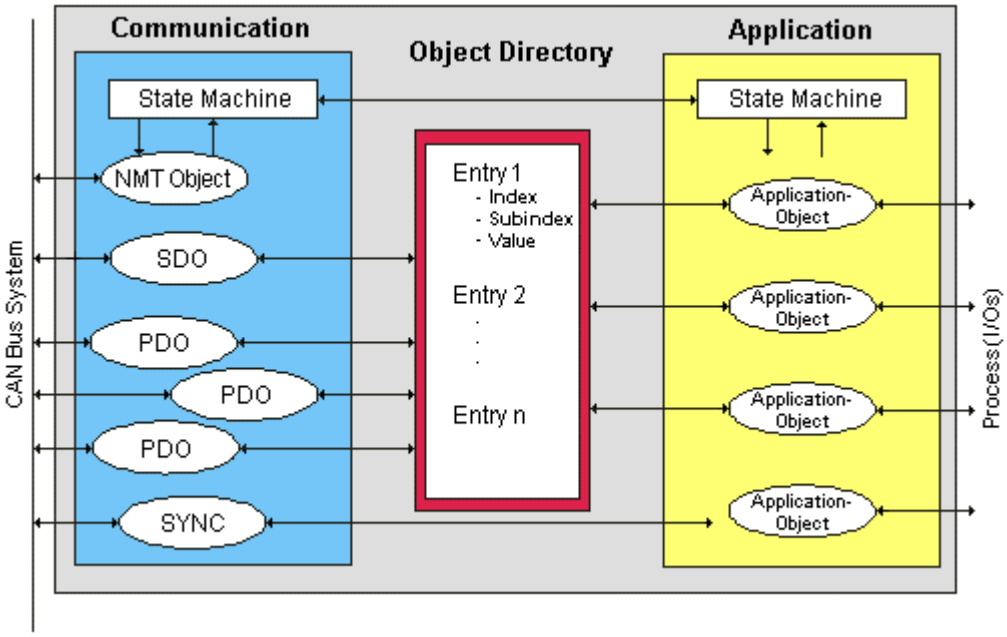



Fig. 74: CANopen Device Model

### Communication Types

CANopen defines a number of communication classes for the input and output data (process data objects):

- Event driven [▶ 84]: Telegrams are sent as soon as their contents have changed. This means that the process image as a whole is not continuously transmitted, only its changes.
- Cyclic synchronous [▶ 84]: A SYNC telegram causes the modules to accept the output data that was previously received, and to send new input data.
- Requested (polled) [▶ 82]: A CAN data request telegram causes the modules to send their input data.

The desired communication type is set by the Transmission Type [▶ 82] parameter.

#### Device Profile

The BECKHOFF CANopen devices support all types of I/O communication, and correspond to the device profile for digital and analog input/output modules (DS401 Version 1). For reasons of backwards compatibility, the default mapping was not adapted to the DS401 V2 profile version.

#### Data transfer rates

Nine transmission rates from 10 kbaud up to 1 Mbaud are available for different bus lengths. The effective utilization of the bus bandwidth allows CANopen to achieve short system reaction times at relatively low data rates.

#### Topology

CAN is based on a linear topology [▶ 20]. The number of devices participating in each network is logically limited by CANopen to 128, but physically the present generation of drivers allows up to 64 nodes in one network segment. The maximum possible size of the network for any particular data rate is limited by the signal propagation delay required on the bus medium. For 1 Mbaud, for instance, the network may extend 25 m, whereas at 50 kbaud the network may reach up to 1000 m. At low data rates the size of the network can be increased by repeaters, which also allow the construction of tree structures.

#### Bus access procedures

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

#### Configuration and parameterization

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

#### Certification

The Beckhoff CANopen devices have a powerful implementation of the protocol, and are certified by the CAN in Automation Association (http://www.can-cia.org).

## 7.2 Protocol Description

### 7.2.1 Network Management

#### Simple Boot-Up

CANopen allows the distributed network to boot in a very simple way. After initialization, the modules are automatically in the *Pre-Operational* state. In this state it is already possible to access the object directory using service data objects (SDOs) with default identifiers, so that the modules can be configured. Since default settings exist for all the entries in the object directory, it is in most cases possible to omit any explicit configuration.

Only one CAN message is then required to start the module: Start_Remote_Node: Identifier *0*, two data bytes: 0x01, 0x00. It switches the node into the *Operational* state.

### Network Status

The states and the state transitions involved as CANopen boots up can be seen from the state diagram:

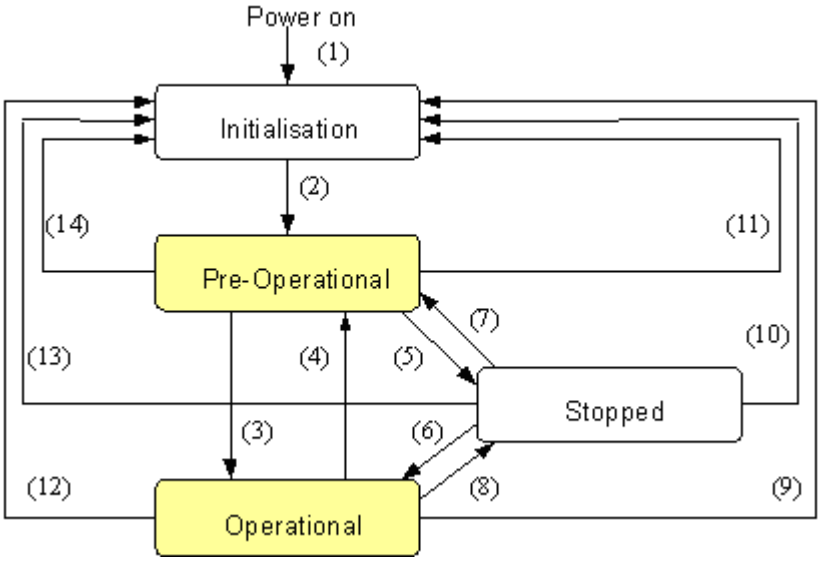



Fig. 75: CANopen bootup state diagram

### Pre-Operational

After initialization the Bus Coupler goes automatically (i.e. without the need for any external command) into the *Pre-Operational* state. In this state it can be configured, since the service data objects (SDOs) are already active. The process data objects, on the other hand, are still locked.

### Operational

In the *Operational* state the process data objects are also active.

If external influences (such as a CAN error, or absence of output voltage) or internal influences (such as a K-Bus error) mean that it is no longer possible for the Bus Coupler to set outputs, to read inputs or to communicate, it attempts to send an appropriate emergency message, goes into the error state, and thus returns to the *Pre-Operational* state. In this way the NMT status machine in the network master can also immediately detect fatal errors.

### Stopped

In the *Stopped* state (formerly: *Prepared*) data communication with the Coupler is no longer possible - only NMT messages are received. The outputs go into the fault state.

### State Transitions

The network management messages have a very simple structure: CAN identifier *0*, with two bytes of data content. The first data byte contains what is known as the command specifier (cs), and the second data byte contains the node address, the node address *0* applying to all nodes (broadcast).

| 11 bit identifier | 2 byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x00 | cs | Node ID | | | | | | |

The following table gives an overview of all the CANopen state transitions and the associated commands (command specifier in the NMT master telegram):

| Status transition | Command Specifier cs | Explanation |
|---|---|---|
| (1) | - | The initialization state is reached automatically at power-up |
| (2) | - | After initialization the pre-operational state is reached automatically - this involves sending the boot-up message. |
| (3), (6) | cs = 1 = 0x01 | Start_Remote_Node.<br>Starts the module, enables outputs, starts transmission of PDOs. |
| (4), (7) | cs = 128 = 0x80 | Enter_Pre-Operational. Stops PDO transmission, SDO still active. |
| (5), (8) | cs = 2 = 0x02 | Stop_Remote_Node.<br>Outputs go into the fault state, SDO and PDO switched off. |
| (9), (10), (11) | cs = 129 = 0x81 | Reset_Node. Carries out a reset. All objects are reset to their power-on defaults. |
| (12), (13), (14) | cs = 130 = 0x82 | Reset_Communication. Carries out a reset of the communication functions. Objects 0x1000 - 0x1FFF are reset to their power-on defaults. |

**Sample 1**

The following telegram puts all the modules in the network into the error state (outputs in a safe state):

| 11 bit identifier | 2 byte of user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x00 | 0x02 | 0x00 | | | | | | |

**Sample 2**

The following telegram resets node 17:

| 11 bit identifier | 2 byte of user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x00 | 0x81 | 0x11 | | | | | | |

**Boot-up message**

After the initialization phase and the self-test the Bus Coupler sends the boot-up message, which is a CAN message with a data byte (0) on the identifier of the guarding or heartbeat message: CAN-ID = 0x700 + node ID. In this way temporary failure of a module during operation (e.g. due to a voltage drop), or a module that is switched on at a later stage, can be reliably detected, even without Node Guarding. The sender can be determined from the message identifier (see default identifier allocation).

It is also possible, with the aid of the boot-up message, to recognize the nodes present in the network at start-up with a simple CAN monitor, without having to make write access to the bus (such as a scan of the network by reading out parameter 0x1000).

Finally, the boot-up message communicates the end of the initialization phase; the Bus Coupler signals that it can now be configured or started.

**Firmware version BA**

Up to firmware version BA the emergency identifier was used for the boot up message.

**Format of the Boot-up message**

| 11 bit identifier | 1 byte of user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x700 (=1792)+ node ID | 0x00 | | | | | | | |

### Node Monitoring

Heartbeat and guarding mechanisms are available to monitor failures in the CANopen network. These are of particular importance for CANopen, since modules do not regularly speak in the event-driven mode of operation. In the case of "guarding", the devices are cyclically interrogated about their status by means of a data request telegram (remote frame), whereas with "heartbeat" the nodes transmit their status on their own initiative.

### Guarding: Node Guarding and Life Guarding

Node Guarding is used to monitor the non-central peripheral modules, while they themselves can use Life Guarding to detect the failure of the guarding master. Guarding involves the master sending remote frames (remote transmit requests) to the guarding identifier of the slaves that are to be monitored. These reply with the guarding message. This contains the slave's status code and a toggle bit that has to change after every message. If either the status or the toggle bit do not agree with that expected by the NMT master, or if there is no answer at all, the master assumes that there is a slave fault.

### Guarding procedure

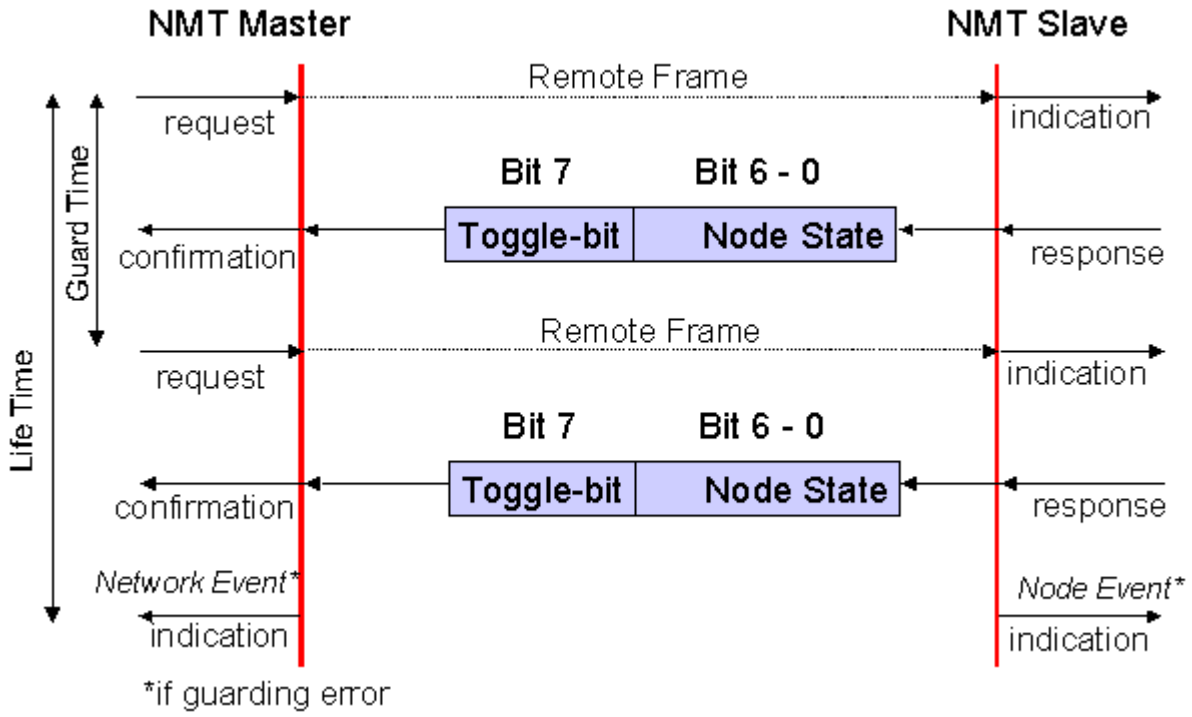



Fig. 76: Schematic diagram: "Guarding procedure"

### Protocol

The toggle bit (t) transmitted in the first guarding telegram has the value *0*. After this, the bit must change (toggle) in every guarding telegram so that the loss of a telegram can be detected. The node uses the remaining seven bits to transmit its network status (s):

| s | Status |
|---|---|
| 4 = 0x04 | Stopped (previously: Prepared) |
| 5 = 0x05 | Operational |
| 127 = 0x7F | Pre-Operational |

### Sample

The guarding message for node 27 (0x1B) must be requested by a remote frame having identifier 0x71B ($1819_{dec}$). If the node is *Operational*, the first data byte of the answer message alternates between 0x05 and 0x85, whereas in the *Pre-Operational* state it alternates between 0x7F and 0xFF.

### Guard time and life time factor

If the master requests the guard messages in a strict cycle, the slave can detect the failure of the master. In this case, if the slave fails to receive a message request from the master within the set *Node Life Time* (a guarding error), it assumes that the master has failed (the watchdog function). It then puts its outputs into the error state, sends an emergency telegram, and returns to the pre-operational state. After a guarding time-out the procedure can be re-started by transmitting a guarding telegram again.

The node life time is calculated from the guard time (object 0x100C) and life time factor (object 0x100D) parameters:

Life time = guard time x life time factor

If either of these two parameters is "0" (the default setting), the master will not be monitored (no life guarding).

### Heartbeat: Node Monitoring without Remote Frame

In the heart beat procedure, each node transmits its status message cyclically on its own initiative. There is therefore no need to use remote frames, and the bus is less heavily loaded than under the guarding procedure.

The master also regularly transmits its heartbeat telegram, so that the slaves are also able to detect failure of the master.

### Heartbeat procedure

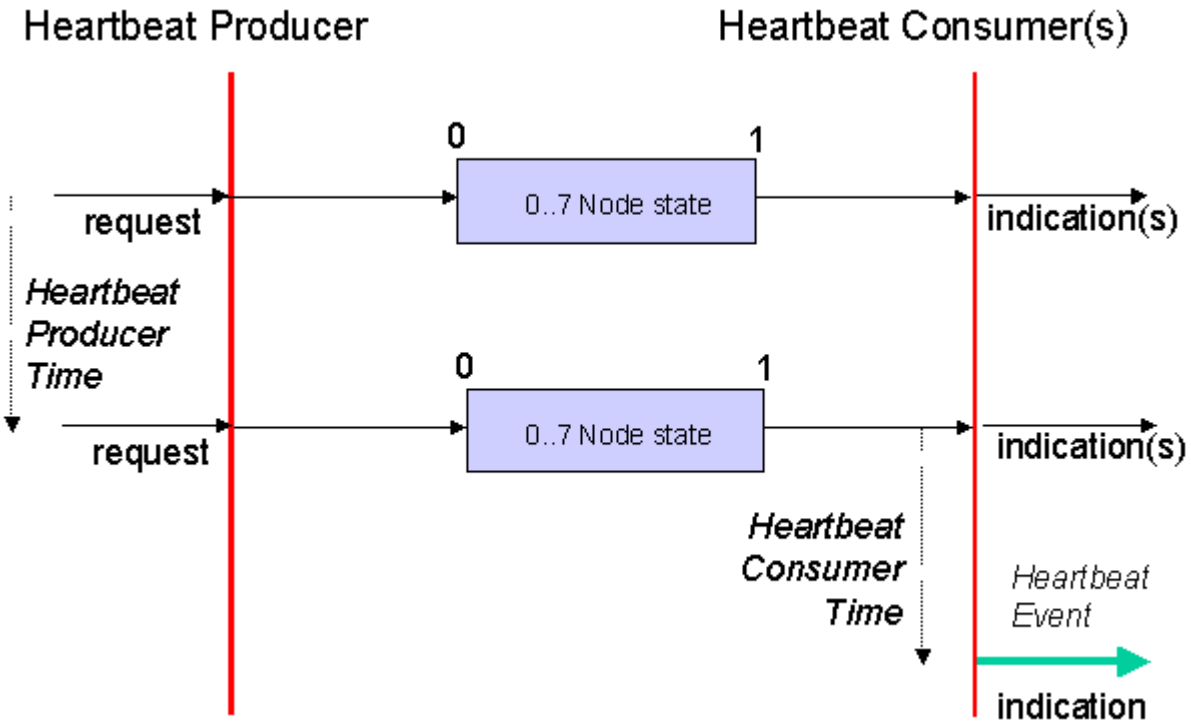



Fig. 77: Schematic diagram: "Heartbeat procedure"

### Protocol

The toggle bit is not used in the heart beat procedure. The nodes send their status cyclically (s). See Guarding [▶ 80].

## 7.2.2 Process Data Objects (PDO)

### Introduction

In many fieldbus systems the entire process image is continuously transferred - usually in a more or less cyclic manner. CANopen is not limited to this communication principle, since the multi-master bus access protocol allows CAN to offer other methods. Under CANopen the process data is not transferred in a master/slave procedure, but follows instead the producer-consumer model. In this model, a bus node transmits its data, as a producer, on its own accord. This might, for example, be triggered by an event. All the other nodes listen, and use the identifier to decide whether they are interested in this telegram, and handle it accordingly. These are the consumers.

The process data in CANopen is divided into segments with a maximum of 8 bytes. These segments are known as process data objects (PDOs). The PDOs each correspond to a CAN telegram, whose specific CAN identifier is used to allocate them and to determine their priority. Receive PDOs (RxPDOs) and transmit PDOs (TxPDOs) are distinguished, the name being chosen from the point of view of the device: an input/output module sends its input data with TxPDOs and receives its output data in the RxPDOs. **This naming convention is retained in the TwinCAT System Manager.**

### Communication parameters

The PDOs can be given different communication parameters according to the requirements of the application. Like all the CANopen parameters, these are also available in the device's object directory, and can be accessed by means of the service data objects. The parameters for the receive PDOs are at index 0x1400 (RxPDO1) onwards. There can be up to 512 RxPDOs (ranging up to index 0x15FF). In the same way, the entries for the transmit PDOs are located from index 0x1800 (TxPDO1) to 0x19FF (TxPDO512).

The Beckhoff Bus Couplers or Fieldbus Coupler Box modules make 16 RxPDO and TxPDOs available for the exchange of process data (although the figure for Economy and LowCost BK5110 and LC5100 Couplers and the Fieldbus Boxes is 5 PDOs each, since these devices manage a lower quantity of process data). The FC510x CANopen master card supports up to 192 transmit and 192 receive PDOs for each channel - although this is restricted by the size of the DPRAM. The EL6751 CANopen terminal dynamically organizes the process image; i.e. the process data are written in succession, enabling a higher data transmission rate. Up to 32 TxPDOs and 32 RxPDOs can be handled in slave mode.

For each existing process data object there is an associated communication parameter object. The TwinCAT System Manager automatically assigns the set parameters to the relevant object directory entries. These entries and their significance for the communication of process data are explained below.

### PDO Identifier

The most important communication parameter in a PDO is the CAN identifier (also known as the communication object identifier, or COB-ID). It is used to identify the data, and determines their priority for bus access. For each CAN data telegram there may only be one sender node (producer), although all messages sent in the CAN broadcast procedure can be received, as described, by any number of nodes (consumers). Thus a node can make its input information available to a number of bus devices at the same time - even without transferring them through a logical bus master. The identifier is located in sub-index 1 of the communication parameter set. It is coded as a 32-bit value in which the least significant 11 bits (bits 0...10) contain the identifier itself. The data width of the object of 32 bits also allows 29-bit identifiers in accordance with CAN 2.0B to be entered, although the default identifiers always refer to the more usual 11-bit versions. Generally speaking, CANopen is economical it its use of the available identifiers, so that the use of the 29-bit versions remains limited to unusual applications. It is therefore also not supported by a Beckhoff's CANopen devices. The highest bit (bit 31) can be used to activate the process data object or to turn it off.

A complete identifier list [▶ 140] is provided in the appendix.

### PDO linking

In the system of default identifiers, all the nodes (here: slaves) communicate with one central station (the master), since slave nodes do not listen by default to the transmit identifier of any other slave node.

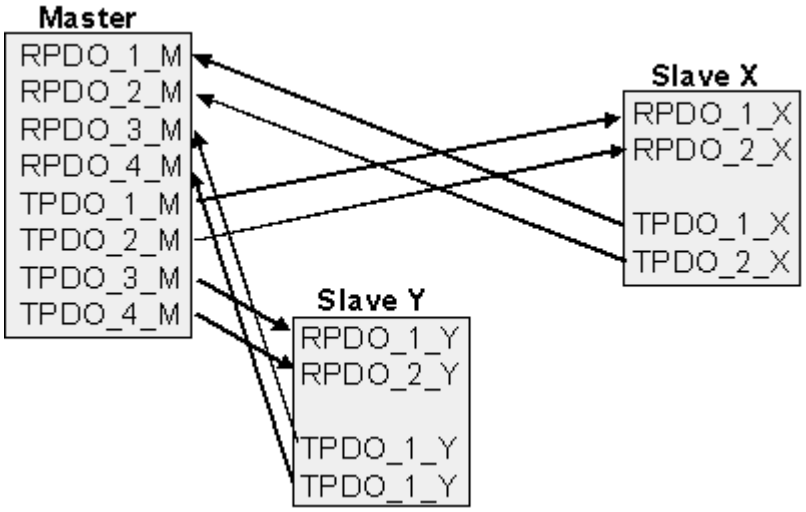



Fig. 78: Default identifier allocation: Master/Slave

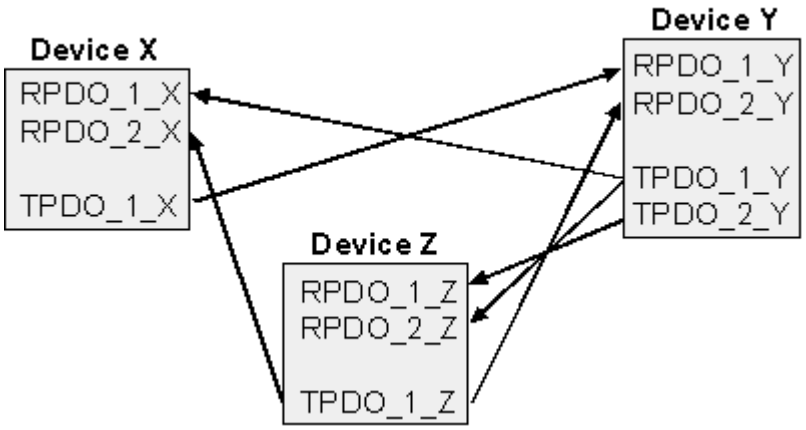



Fig. 79: PDO linking: Peer to Peer

If the consumer-producer model of CANopen PDOs is to be used for direct data exchange between nodes (without a master), the identifier allocation must be appropriately adapted, so that the TxPDO identifier of the producer agrees with the RxPDO identifier of the consumer: This procedure is known as PDO linking. It permits, for sample, easy construction of electronic drives in which several slave axes simultaneously listen to the actual value in the master axis TxPDO.

**PDO Communication Types: Overview**

CANopen offers a number of possible ways to transmit process data (see also: Notes on PDO Parameterization [▶ 89]).

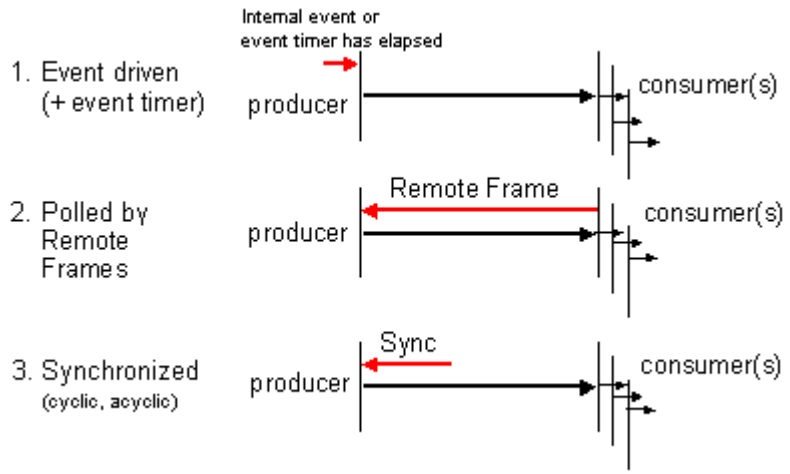



Fig. 80: Diagram: CAN process data transmission

**Event driven**

The ”event" is the alteration of an input value, the data being transmitted immediately after this change. The event-driven flow can make optimal use of the bus bandwidth, since instead of the whole process image it is only the changes in it that are transmitted. A short reaction time is achieved at the same time, since when an input value changes it is not necessary to wait for the next interrogation from a master.

As from CANopen Version 4 it is possible to combine the event driven type of communication with a cyclic update. Even if an event has not just occurred, event driven TxPDOs are sent after the event timer has elapsed. If an event does occur, the event timer is reset. For RxPDOs the event timer is used as a watchdog in order to monitor the arrival of event driven PDOs . If a PDO does not arrive within a set period of time, the bus node adopts the error state.

**Polled**

The PDOs can also be polled by data request telegrams (remote frames). In this way it is possible to get the input process image of event-driven inputs onto the bus, even when they do not change, for instance through a monitoring or diagnostic device brought into the network while it is running. The time behavior of remote frame and response telegrams depends on what CAN controller is in use. Components with full integrated message filtering ("FullCAN") usually answer a data request telegram immediately, transmitting data that is waiting in the appropriate transmit buffer - it is the responsibility of the application to see that the data there is continuously updated. CAN controllers with simple message filtering (BasicCAN) on the other hand pass the request on to the application which can now compose the telegram with the latest data. This does take longer, but does mean that the data is up-to-date. Beckhoff use CAN controllers following the principle of Basic CAN.

Since this device behavior is usually not transparent to the user, and because there are CAN controllers still in use that do not support remote frames at all, polled communication can only with reservation be recommended for operative running.

**Synchronized**

It is not only for drive applications that it is worthwhile to synchronize the determination of the input information and the setting the outputs. For this purpose CANopen provides the SYNC object, a CAN telegram of high priority but containing no user data, whose reception is used by the synchronized nodes as a trigger for reading the inputs or for setting the outputs.

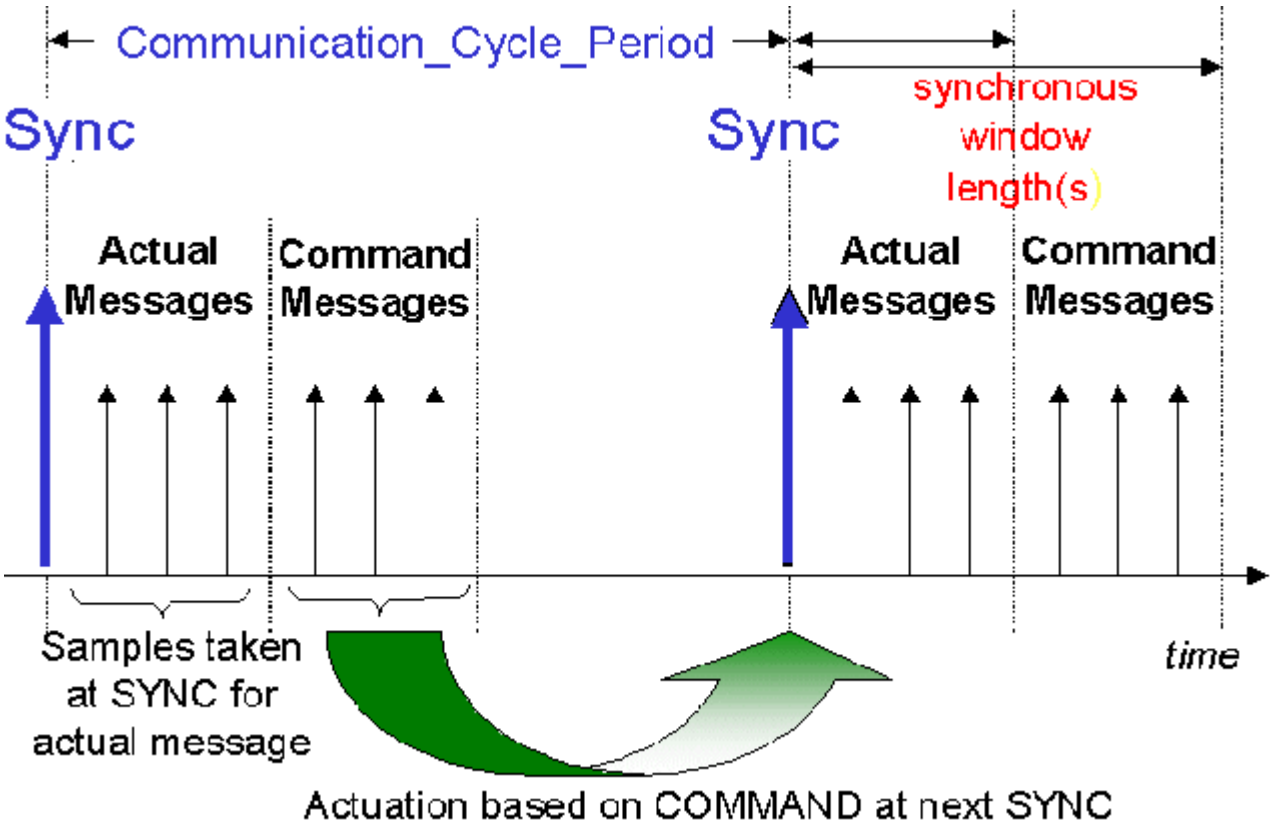



Fig. 81: Diagram: CAN "SYNC" telegram

**PDO transmission types: Parameterization**

The PDO transmission type parameter specifies how the transmission of the PDO is triggered, or how received PDOs are handled.

| Transmission type | Cyclical | Acyclical | Synchronous | Asynchronous | Only RTR |
|---|---|---|---|---|---|
| 0 | | X | X | | |
| 1-240 | X | | X | | |
| 241-251 | - reserved - | | | | |
| 252 | | | X | | X |
| 253 | | | | X | X |
| 254, 255 | | | | X | |

The type of transmission is parameterized for RxPDOs in the objects at 0x1400ff, sub-index 2, and for TxPDOs in the objects at 0x1800ff, sub-index 2.

**Acyclic Synchronous**

PDOs of transmission type 0 function synchronously, but not cyclically. An RxPDO is only evaluated after the next SYNC telegram has been received. In this way, for instance, axis groups can be given new target positions one after another, but these positions only become valid at the next SYNC - without the need to be constantly outputting reference points. A device whose TxPDO is configured for transmission type 0 acquires its input data when it receives the SYNC (synchronous process image) and then transmits it if the data correspond to an event (such as a change in input) having occurred. Transmission type 0 thus combines transmission for reasons that are event driven with a time for transmission (and, as far as possible, sampling) and processing given by the reception of "SYNC".

**Cyclic Synchronous**

In transmission types 1-240 the PDO is transmitted cyclically: after every ”nth" SYNC (n = 1...240). Since transmission types can be combined on a device as well as in the network, it is possible, for example, for a fast cycle to be agreed for digital inputs (n = 1), whereas the data for analog inputs is transmitted in a slower

cycle (e.g. n = 10). RxPDOs do not generally distinguish between transmission types 0...240: a PDO that has been received is set to valid when the next SYNC is received. The cycle time (SYNC rate) can be monitored (object 0x1006), so that if the SYNC fails the device reacts in accordance with the definition in the device profile, and switches, for sample, its outputs into the error state.

The FC510x card / EL6751 terminal fully support the synchronous communication method: transmitting the SYNC telegram is coupled to the linked task, so that new input data is available every time the task begins. If a synchronous PDO does not arrive, this is detected and reported to the application.

### Only RTR

Transmission types 252 and 253 apply to process data objects that are transmitted exclusively on request by a remote frame. 252 is synchronous: when the SYNC is received the process data is acquired. It is only transmitted on request. 253 is asynchronous. The data here is acquired continuously, and transmitted on request. This type of transmission is not generally recommended, because fetching input data from some CAN controllers is only partially supported. Because, furthermore, the CAN controllers sometimes answer remote frames automatically (without first requesting up-to-date input data), there are circumstances in which it is questionable whether the polled data is up-to-date. Transmission types 252 and 253 are for this reason not supported by the Beckhoff PC cards / terminals.

### Asynchronous

The transmission types 254 + 255 are asynchronous, but may also be event-driven. In transmission type 254, the event is specific to the manufacturer, whereas for type 255 it is defined in the device profile. In the simplest case, the event is the change of an input value - this means that every change in the value is transmitted. The asynchronous transmission type can be coupled with the event timer, thus also providing input data when no event has just occurred.

### Inhibit time

The ”inhibit time" parameter can be used to implement a ”transmit filter" that does not increase the reaction time for relatively new input alterations, but is active for changes that follow immediately afterwards. The inhibit time (transmit delay time) specifies the minimum length of time that must be allowed to elapse between the transmission of two of the same telegrams. If the inhibit time is used, the maximum bus loading can be determined, so that the worst case latency can then be found.

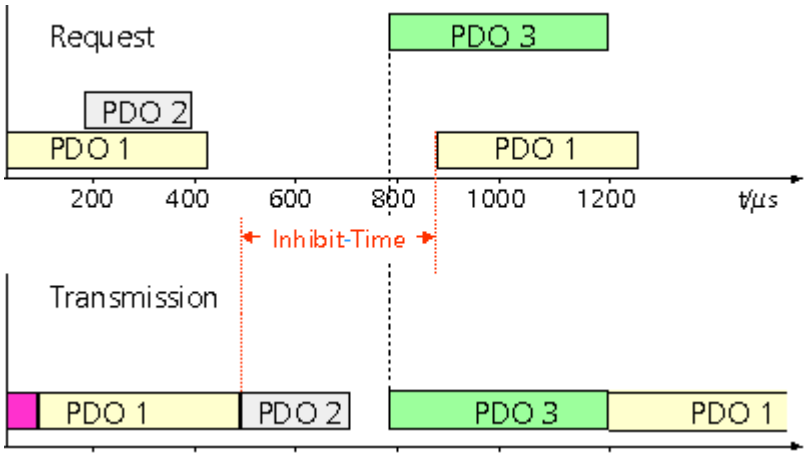



Fig. 82: Timing diagram: "Inhibit time"

Although the Beckhoff FC510x PC cards / EL6751 terminal can parameterize the inhibit time on slave devices, they do not themselves support it. The transmitted PDOs become automatically spread out (transmit delay) as a result of the selected PLC cycle time - and there is little value in having the PLC run faster than the bus bandwidth permits. The bus loading, furthermore, can be significantly affected by the synchronous communication.

### Event Timer

An event timer for transmit PDOs can be specified by sub-index 5 in the communication parameters. Expiry of this timer is treated as an additional event for the corresponding PDO, so that the PDO will then be transmitted. If the application event occurs during a timer period, it will also be transmitted, and the timer is reset.

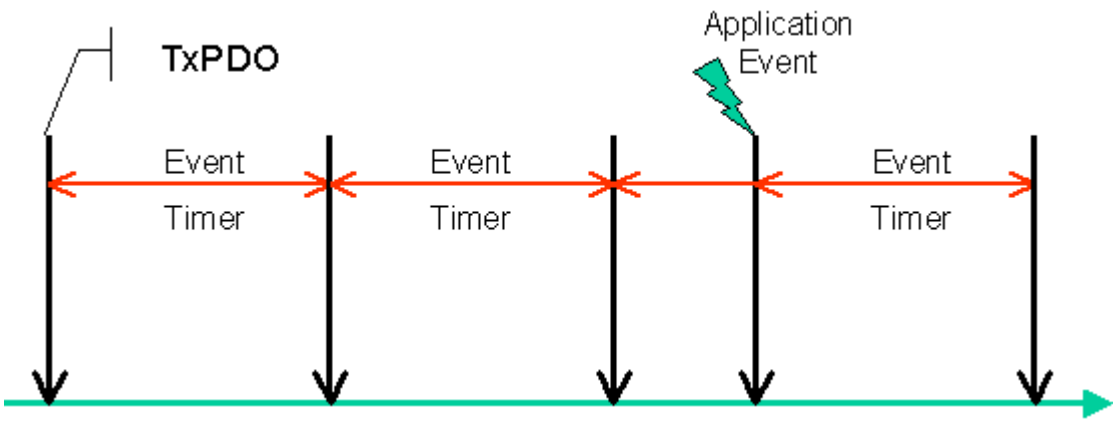



Fig. 83: Time representation of the event timer

In the case of receive PDOs, the timer is used to set a watchdog interval for the PDO: the application is informed if no corresponding PDO has been received within the set period. The FC510x / EL6751 can in this way monitor each individual PDO.

Notes on PDO Parameterization [▶ 89]

### PDO Mapping

PDO mapping refers to mapping of the application objects (real time data) from the object directory to the process data objects. The CANopen device profile provide a default mapping for every device type, and this is appropriate for most applications. Thus the default mapping for digital I/O simply represents the inputs and outputs in their physical sequence in the transmit and receive process data objects.

The default PDOs for drives contain 2 bytes each of a control and status word and a set or actual value for the relevant axis.

The current mapping can be read by means of corresponding entries in the object directory. These are known as the mapping tables. The first location in the mapping table (sub-index 0) contains the number of mapped objects that are listed after it. The tables are located in the object directory at index 0x1600ff for the RxPDOs and at 0x1A00ff for the TxPDOs.

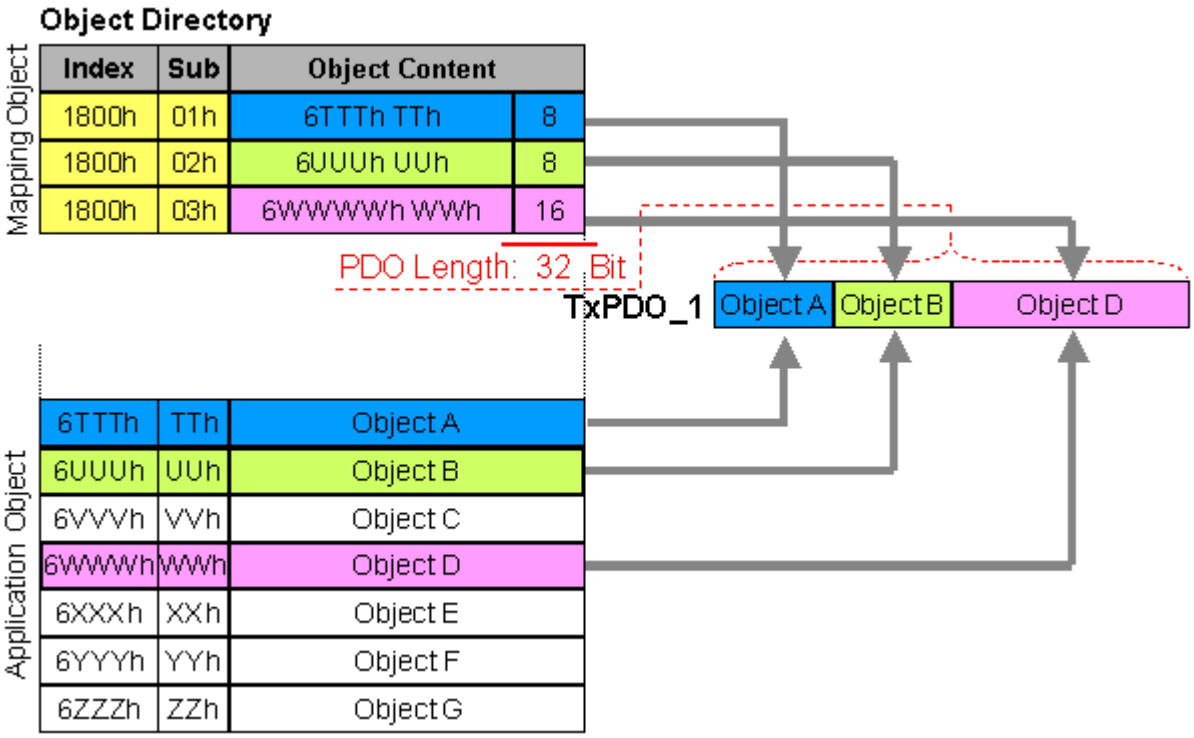



Fig. 84: Mapping representation

**Digital and analog input/output modules: Read out the I/O number**

The current number of digital and analog inputs and outputs can be determined or verified by reading out the corresponding application objects in the object directory:

| Parameter | Object directory address |
|---|---|
| Number of digital input bytes | Index 0x6000, sub-index 0 |
| Number of digital output bytes | Index 0x6200, sub-index 0 |
| Number of analog inputs | Index 0x6401, sub-index 0 |
| Number of analog outputs | Index 0x6411, sub-index 0 |

**Variable mapping**

As a rule, the default mapping of the process data objects already satisfies the requirements. For special types of application the mapping can nevertheless be altered: the Beckhoff CANopen Bus Couplers, for instance, thus support variable mapping, in which the application objects (input and output data) can be freely allocated to the PDOs. The mapping tables must be configured for this: as from Version 4 of CANopen, only the following procedure is permitted, and must be followed precisely:

1. First delete the PDO (set 0x1400ff, or 0x1800ff, sub-index 1, bit 31 to "1")
2. Set sub-index 0 in the mapping parameters (0x1600ff or 0x1A00ff) to "0"
3. Change mapping entries (0x1600ff or 0x1A00ff, SI 1..8)
4. Set sub-index 0 in the mapping parameters to the valid value. The device then checks the entries for consistency.
5. Create PDO by entering the identifier (0x1400ff or 0x1800ff, sub-index 1).

**Dummy Mapping**

A further feature of CANopen is the mapping of placeholders, or dummy entries. The data type entries stored in the object directory, which do not themselves have data, are used as placeholders. If such entries are contained in the mapping table, the corresponding data from the device is not evaluated. In this way, for instance, a number of drives can be supplied with new set values using a single CAN telegram, or outputs on a number of nodes can be set simultaneously, even in event-driven mode.

## 7.2.3 PDO Parameterization

Even though the majority of CANopen networks operate satisfactorily with the default settings, i.e. with the minimum of configuration effort, it is wise at least to check whether the existing bus loading is reasonable: 80% bus loading may be acceptable for a network operating purely in cyclic synchronous modes, but for a network with event-driven traffic this value would generally be too high, as there is hardly any bandwidth available for additional events.

**Consider the Requirements of the Application**

The communication of the process data must be optimized in the light of application requirements which are likely to be to some extent in conflict. These include

- Little work on parameterization - useable default values are optimal
- Guaranteed reaction time for specific events
- Cycle time for regulation processes over the bus
- Safety reserves for bus malfunctions (enough bandwidth for the repetition of messages)
- Maximum baud rate - depends on the maximum bus length
- Desired communication paths - who is speaking with whom

The determining factor often turns out to be the available bus bandwidth (bus load).

**Baud rate**

We generally begin by choosing the highest baud rate that the bus will permit. It should be borne in mind that serial bus systems are fundamentally more sensitive to interference as the baud rate is increased. The following rule therefore applies: just as fast as necessary. 1000 kbit/s are not usually necessary, and only to be unreservedly recommended on networks within a control cabinet where there is no electrical isolation between the bus nodes. Experience also tends to show that estimates of the length of bus cable laid are often over-optimistic - the length actually laid tends to be longer.

**Determine the Communication Type**

Once the baud rate has been chosen it is appropriate to specify the PDO communication type(s). These have different advantages and disadvantages:

- Cyclic synchronous communication provides an accurately predictable bus loading, and therefore a defined time behavior - you could say that the standard case is the worst case. It is easy to configure: The SYNC rate parameter sets the bus loading globally. The process images are synchronized: Inputs are read at the same time, output data is set valid simultaneously, although the quality of the synchronization depends on the implementation. The BECKHOFF FC510x PC cards / EL6751 CANopen terminal are capable of synchronizing the CANopen bus system with the cycles of the application program (PLC or NC).

  The guaranteed reaction time under cyclic synchronous communication is always at least as long as the cycle time, and the bus bandwidth is not exploited optimally, since old data, i.e. data that has not changed, is continuously transmitted. It is however possible to optimize the network through the selection of different SYNC multiples (transmission types 1...240), so that data that changes slowly is transmitted less often than, for instance, time-critical inputs. It must, however, be borne in mind that input states that last for a time that is shorter than the cycle time will not necessarily be communicated. If it is necessary for such conditions to be registered, the associated PDOs for asynchronous communication should be provided.
- Event-driven asynchronous communication is optimal from the point of view of reaction time and the exploitation of bus bandwidth - it can be described as "pure CAN". Your choice must, however, also take account of the fact that it is not impossible for a large number of events to occur simultaneously, leading to corresponding delays before a PDO with a relatively low priority can be sent. Proper network planning therefore necessitates a worst-case analysis. Through the use of, for instance, inhibit time [▶ 82], it is also necessary to prevent a constantly changing input with a high PDO priority from blocking the bus (technically known as a "babbling idiot"). It is for this reason that event driving is switched off by

default in the device profile of analog inputs, and must be turned on specifically. Time windows for the transmit PDOs can be set using progress timers: the telegram is not sent again before the inhibit time [▶ 82] has elapsed, and not later than the time required for the progress timer to complete.

- The communication type is parameterized by means of the transmission type [▶ 82].

It is also possible to combine the two PDO principles. It can, for instance, be helpful to exchange the set and actual values of an axis controller synchronously, while limit switches, or motor temperatures with limit values are monitored with event-driven PDOs. This combines the advantages of the two principles: synchronicity for the axis communication and short reaction times for limit switches. In spite of being event-driven, the distributed limit value monitoring avoids a constant addition to the bus load from the analog temperature value.

In this sample it can also be of value to deliberately manipulate the identifier allocation, in order to optimize bus access by means of priority allocation: the highest priority is given to the PDO with the limit switch data, and the lowest to that with the temperature values.

Optimization of bus access latency time through modification of the identifier allocation is not, however, normally required. On the other hand the identifiers must be altered if masterless communication is to be made possible (PDO linking [▶ 82]). In this sample it would be possible for one RxPDO for each axis to be allocated the same identifier as the limit switch TxPDO, so that alterations of the input value can be received without delay.

**Determining the Bus Loading**

It is always worth determining the bus loading. But what bus loading values are permitted, or indeed sensible? It is first necessary to distinguish a short burst of telegrams in which a number of CAN messages follow one another immediately - a temporary 100% bus loading. This is only a problem if the sequence of receive interrupts that it caused at the CAN nodes cannot be handled. This would constitute a data overflow (or CAN queue overrun). This can occur at very high baud rates (> 500 kbit/s) at nodes with software telegram filtering and relatively slow or heavily loaded microcontrollers if, for instance, a series of remote frames (which do not contain data bytes, and are therefore very short) follow each other closely on the bus (at 1 Mbit/s this can generate an interrupt every 40 µs; for example, an NMT master might transmit all its guarding requests in an unbroken sequence). This can be avoided through skilled implementation, and the user should be able to assume that the device suppliers have taken the necessary trouble. A burst condition is entirely normal immediately after the SYNC telegram, for instance: triggered by the SYNC, all the nodes that are operating synchronously try to send their data at almost the same time. A large number of arbitration processes take place, and the telegrams are sorted in order of priority for transmission on the bus. This is not usually critical, since these telegrams do contain some data bytes, and the telegrams trigger a sequence of receive interrupts at the CAN nodes which is indeed rapid, but is nevertheless manageable.

Bus loading most often refers to the value averaged over several primary cycles, that is the mean value over 100-500 ms. CAN, and therefore CANopen, is indeed capable of managing a bus loading of close to 100% over long periods, but this implies that no bandwidth is available for any repetitions that may be necessitated by interference, for asynchronous error messages, parameterization and so on. Clearly, the dominant type of communication will have a large influence on the appropriate level of bus loading: a network with entirely cyclic synchronous operation is always in any case near to the worst case state, and can therefore be operated with values in the 70-80% range. The figure is very hard to state for an entirely event-driven network: an estimate must be made of how many events additional to the current state of the system might occur, and of how long the resulting burst might last - in other words, for how long the lowest priority message will be delayed. If this value is acceptable to the application, then the current bus loading is acceptable. As a rule of thumb it can usually be assumed that an event-driven network running with a base loading of 30-40% has enough reserve for worst-case scenarios, but this assumption does not obviate the need for a careful analysis if delays could have critical results for the plant.

The BECKHOFF FC510x CANopen master cards / EL6751 CANopen master terminal display the bus load via the System Manager. This variable can also be processed in the PLC, or can be displayed in the visualization system.

The amount data in the process data objects is of course as relevant as the communication parameters: the PDO mapping. [▶ 87]

## 7.2.4 Service Data Objects (SDO)

The parameters listed in the object directory are read and written by means of service data objects. These SDOs are *Multiplexed Domains*, i.e. data structures of any size that have a multiplexer (address). The multiplexer consists of a 16-bit index and an 8-bit sub-index that address the corresponding entries in the object directory.

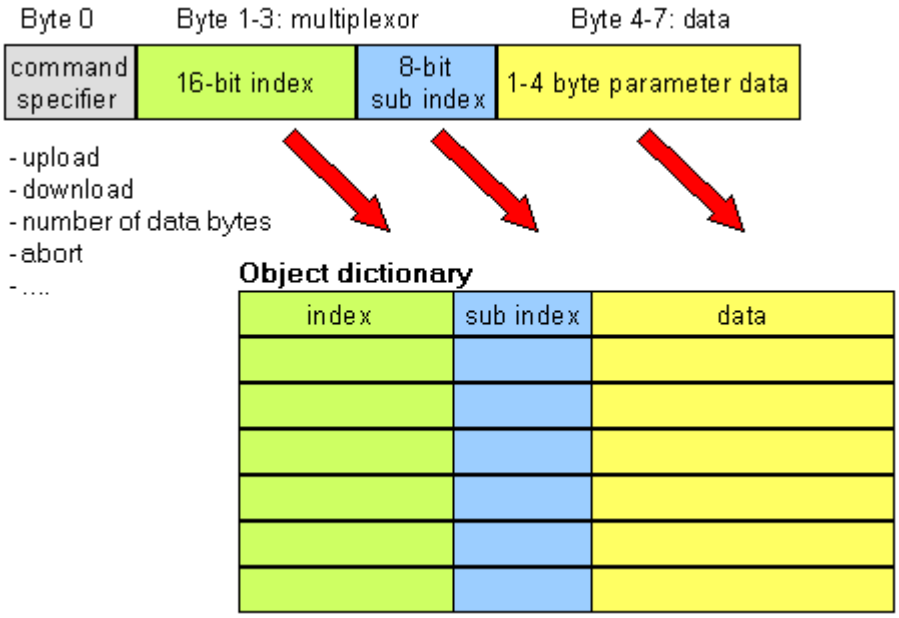



Fig. 85: SDO protocol: access to the object directory

The CANopen Bus Couplers are servers for the SDO, which means that at the request of a client (e.g. of the IPC or the PLC) they make data available (upload), or they receive data from the client (download). This involves a handshake between the client and the server.

When the size of the parameter to be transferred is not more than 4 bytes, a single handshake is sufficient (one telegram pair): For a download, the client sends the data together with its index and sub-index, and the server confirms reception. For an upload, the client requests the data by transmitting the index and sub-index of the desired parameter, and the server sends the parameter (including index and sub-index) in its answer telegram.

The same pair of identifiers is used for both upload and download. The telegrams, which are always 8 bytes long, encode the various services in the first data byte. All parameters with the exception of objects 1008h, 1009h and 100Ah (device name, hardware and software versions) are only at most 4 bytes long, so this description is restricted to transmission in expedited transfer.

### Protocol

The structure of the SDO telegrams is described below.

#### Client -> Server, Upload Request

| 11 bit identifier | 8 byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x600 (=1536dec) + node ID | 0x40 | Index0 | Index1 | SubIdx | 0x00 | 0x00 | 0x00 | 0x00 |

| Parameter | Explanation |
|---|---|
| Index0 | Index low byte (Unsigned16, LSB) |
| Index1 | Index high byte (Unsigned16, MSB) |
| SubIdx | Sub-index (Unsigned8) |

### Client -> Server, Upload Response

| 11 bit identifier | 8 byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x580 (=1408dec) + node ID | 0x4x | Index0 | Index1 | SubIdx | Data0 | Data1 | Data2 | Data3 |

| Parameter | Explanation |
|---|---|
| Index0 | Index low byte (Unsigned16, LSB) |
| Index1 | Index high byte (Unsigned16, MSB) |
| SubIdx | Sub-index (Unsigned8) |
| Data0 | Data low low byte (LLSB) |
| Data3 | Data high high byte (MMSB) |

Parameters whose data type is Unsigned8 are transmitted in byte D0, parameters whose type is Unsigned16 use D0 and D1.

The number of valid data bytes is coded as follows in the first CAN data byte (0x4x):

| Number of parameter bytes | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| First CAN data byte | 0x4F | 0x4B | 0x47 | 0x43 |

### Client -> Server, Download Request

| 11 bit identifier | 8 byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x600 (=1536dec) + node ID | 0x22 | Index0 | Index1 | SubIdx | Data0 | Data1 | Data2 | Data3 |

| Parameter | Explanation |
|---|---|
| Index0 | Index low byte (Unsigned16, LSB) |
| Index1 | Index high byte (Unsigned16, MSB) |
| SubIdx | Sub-index (Unsigned8) |
| Data0 | Data low low byte (LLSB) |
| Data3 | Data high high byte (MMSB) |

It is optionally possible to give the number of valid parameter data bytes in the first CAN data byte

| Number of parameter bytes | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| First CAN data byte | 0x2F | 0x2B | 0x27 | 0x23 |

This is, however, not generally necessary, since only the less significant data bytes up to the length of the object directory entry that is to be written are evaluated. A download of data up to 4 bytes in length can therefore always be achieved in BECKHOFF bus nodes with 22 h in the first CAN data byte.

### Client -> Server, Download Response

| 11 bit identifier | 8 byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x580 (=1408dec) + node ID | 0x60 | Index0 | Index1 | SubIdx | 0x00 | 0x00 | 0x00 | 0x00 |

| Parameter | Explanation |
|---|---|
| Index0 | Index low byte (Unsigned16, LSB) |
| Index1 | Index high byte (Unsigned16, MSB) |
| SubIdx | Sub-index (Unsigned8) |

### Breakdown of Parameter Communication

Parameter communication is interrupted if it is faulty. The client or server send an SDO telegram with the following structure for this purpose:

| 11 bit identifier | 8 byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x580 (client) or 0x600(server) + node ID | 0x80 | Index0 | Index1 | SubIdx | Error0 | Error1 | Error2 | Error3 |

| Parameter | Explanation |
|---|---|
| Index0 | Index low byte (Unsigned16, LSB) |
| Index1 | Index high byte (Unsigned16, MSB) |
| SubIdx | Sub-index (Unsigned8) |
| Error0 | SDO error code low low byte (LLSB) |
| Error3 | SDO error code high high byte (MMSB) |

List of SDO error codes (reason for abortion of the SDO transfer):

| SDO error code | Explanation |
|---|---|
| 0x05 03 00 00 | Toggle bit not changed |
| 0x05 04 00 01 | SDO command specifier invalid or unknown |
| 0x06 01 00 00 | Access to this object is not supported |
| 0x06 01 00 02 | Attempt to write to a Read_Only parameter |
| 0x06 02 00 00 | The object is not found in the object directory |
| 0x06 04 00 41 | The object cannot be mapped into the PDO |
| 0x06 04 00 42 | The number and/or length of mapped objects would exceed the PDO length |
| 0x06 04 00 43 | General parameter incompatibility |
| 0x06 04 00 47 | General internal error in device |
| 0x06 06 00 00 | Access interrupted due to hardware error |
| 0x06 07 00 10 | Data type or parameter length do not agree or are unknown |
| 0x06 07 00 12 | Data type does not agree, parameter length too great |
| 0x06 07 00 13 | Data type does not agree, parameter length too short |
| 0x06 09 00 11 | Sub-index not present |
| 0x06 09 00 30 | General value range error |
| 0x06 09 00 31 | Value range error: parameter value too great |
| 0x06 09 00 32 | Value range error: parameter value too small |
| 0x06 0A 00 23 | Resource not available |
| 0x08 00 00 21 | Access not possible due to local application |
| 0x08 00 00 22 | Access not possible due to current device status |

Further, manufacturer-specific error codes have been introduced for register communication (index 0x4500, 0x4501):

| SDO error code | Explanation |
|---|---|
| 0x06 02 00 11 | Invalid table: Table or channel not present |
| 0x06 02 00 10 | Invalid register: table not present |
| 0x06 01 00 22 | Write protection still set |
| 0x06 07 00 43 | Incorrect number of function arguments |
| 0x06 01 00 21 | Function still active, try again later |
| 0x05 04 00 40 | General routing error |
| 0x06 06 00 21 | Error accessing BC table |
| 0x06 09 00 10 | General error communicating with terminal |
| 0x05 04 00 47 | Time-out communicating with terminal |

## 7.2.5 Identifier Allocation

### Default Identifier

CANopen provides default identifiers for the most important communication objects, and these are derived from the 7-bit node address (the node ID) and a 4-bit function code in accordance with the following scheme:

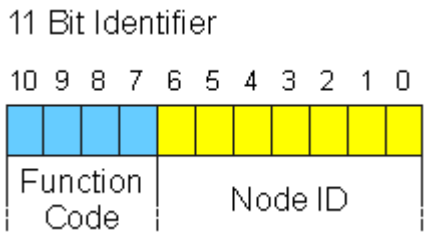



Fig. 86: Default Identifier

For broadcast objects the node ID is set to *0*. This gives rise to the following default identifiers:

### Broadcast objects

| Object | Function | Function Code | resulting COB ID | | Object for Comm. Parameter / Mapping |
|---|---|---|---|---|---|
| | | | hex | dec | |
| NMT | Boot-Up | 0 | 0x00 | 0 | - / - |
| SYNC | Synchronization | 1 | 0x80 | 128 | 0x1005+0x1006 / - |

### Peer-to-peer objects

| Object | Function | Function Code | resulting COB ID | | Object for Comm. Parameter / Mapping |
|---|---|---|---|---|---|
| | | | hex | dec | |
| Emergency | Status / error | 1 | 0x81 - 0xFF | 129 - 255 | - / - |
| PDO1 (tx) | dig. inputs | 11 | 0x181 - 0x1FF | 385 - 511 | 0x1800 / 0x1A00 |
| PDO1 (rx) | digital outputs | 100 | 0x201 - 0x27F | 513 - 639 | 0x1400 / 0x1600 |
| PDO2 (tx) | analog inputs | 101 | 0x281 - 0x2FF | 641 - 767 | 0x1801 / 0x1A01 |
| PDO2 (rx) | analog outputs | 110 | 0x301 - 0x37F | 769 - 895 | 0x1401 / 0x1601 |
| PDO3 (tx) | analog inputs* | 111 | 0x381 - 0x3FF | 897 - 1023 | 0x1802 / 0x1A02 |
| PDO3 (rx) | analog outputs* | 1000 | 0x401 - 0x47F | 1025 - 1151 | 0x1402 / 0x1602 |
| PDO4 (tx) | analog inputs* | 1001 | 0x481 - 0x4FF | 1153 - 1279 | 0x1803 / 0x1A03 |
| PDO4 (rx) | analog outputs* | 1010 | 0x501 - 0x57F | 1281 - 1407 | 0x1403 / 0x1603 |
| SDO (tx) | Parameter | 1011 | 0x581 - 0x5FF | 1409 - 1535 | - / - |
| SDO (rx) | Parameter | 1100 | 0x601 - 0x67F | 1537 - 1663 | - / - |
| Guarding | Life and node guarding, heartbeat, boot-up message | 1110 | 0x701 - 0x77F | 1793 - 1919 | (0x100C, 0x100D, 0x100E, 0x1016, 0x1017) |

*) The Beckhoff Default Mapping applies to PDO3 and PDO4. In most configurations, PDOs 3+4 contain data related to analog inputs and outputs, but there can also be "excess" data from digital I/Os, or data from special terminals. Details may be found in the section covering PDO Mapping [▶ 82].

Up until version 3 of the CANopen specification, default identifiers were assigned to 2 PDOs at a time. The BECKHOFF Bus Couplers up to firmware version BA correspond to this issue of the specification. After firmware version C0 (CANopen version 4), default identifiers are provided for up to 4 PDOs.

**Manufacture-Specific Default Identifiers for Additional PDOs**

Default identifier for additional PDOs

Identifiers are not assigned to the additional PDOs that are filled by the Beckhoff Bus Couplers in accordance with the standard scheme. The user must enter an identifier for these PDOs in the object directory. It is easier to activate the occupied PDOs by means of object 0x5500.

This entry in the object directory extends the default identifier allocation up to 11 PDOs. This creates the following identifiers:

| Object | Function Code | Resulting COB ID (hex) | Resulting COB ID (dec) |
|---|---|---|---|
| PDO5 (tx) | 1101 | 0x681 - 0x6BF | 1665 - 1727 |
| PDO5 (rx) | 1111 | 0x781 - 0x7BF | 1921- 1983 |
| PDO6 (tx) | 111 | 0x1C1 - 0x1FF | 449 - 511 |
| PDO6 (rx) | 1001 | 0x241 - 0x27F | 577 - 639 |
| PDO7 (tx) | 1011 | 0x2C1 - 0x2FF | 705 - 767 |
| PDO7 (rx) | 1101 | 0x341 - 0x37F | 833 - 895 |
| PDO8 (tx) | 1111 | 0x3C1- 0x3FF | 961 - 1023 |
| PDO8 (rx) | 10001 | 0x441 - 0x47F | 1089 - 1151 |
| PDO9 (tx) | 10011 | 0x4C1 - 0x4FF | 1217 - 1279 |
| PDO9 (rx) | 10101 | 0x541 - 0x57F | 1345 - 1407 |
| PDO10 (tx) | 10111 | 0x5C1 - 0x5FF | 1473 - 1535 |
| PDO10 (rx) | 11001 | 0x641 - 0x67F | 1601- 1663 |
| PDO11 (tx) | 11011 | 0x6C1 - 0x6FF | 1729 - 1791 |
| PDO11 (rx) | 11101 | 0x741 - 0x77F | 1857 - 1919 |

**NOTE**

**Index 0x5500**

Index 0x5500 must not be used if Bus Couplers with more than 5 PDOs are present in networks with node numbers greater than 64, otherwise identifier overlaps can occur.

# 7.3 Object directory

## 7.3.1 Object directory structure

All the CANopen objects relevant for the Bus Coupler are entered into the CANopen object directory. The object directory is divided into three different regions:

1. communication-specific profile region (index 0x1000 – 0x1FFF).
   This contains the description of all the parameters specific to communication.
2. manufacturer-specific profile region (index 0x2000 – 0x5FFF).
   Contains the description of the manufacturer-specific entries.
3. standardized device profile region (0x6000 – 0x9FFF).
   Contains the objects for a device profile according to DS-401.

Every entry in the object directory is identified by a 16 bit index. If an object consists of several components (e.g. object type array or record), the components are identified by an 8-bit subindex. The object name describes the function of an object, while the data type attribute specifies the data type of the entry. The access attribute specifies whether an entry may only be read, only written, or may be both read and written.

### Communication-specific region

All the parameters and objects necessary for the CANopen Bus Coupler's communication are in this region of the object directory. The region from 0x1000 to 0x1018 contains various general communication-specific parameters (e.g. the device name).

The communication parameters (e.g. identifiers) for the receive PDOs are located in the region from 0x1400 to 0x140F (plus subindex). The mapping parameters of the receive PDOs are in the region from 0x1600 to 0x160F (plus subindex). The mapping parameters contain the cross-references to the application objects that are mapped into the PDOs and the data width of the corresponding object (see also the section dealing with PDO Mapping).

The communication and mapping parameters for the transmit PDOs are located in the regions from 0x1800 to 0x180F and from 0x1A00 to 0x1A0F.

### Manufacturer-specific region

This region contains entries that are specific to BECKHOFF, e.g.:

- data objects for special terminals
- objects for register communication providing access to all the Bus Couplers' and Bus Terminals' internal registers
- objects for simplified configuration of the PDOs

### Standardized device profile region

The standardized device profile region supports the device profile of CANopen DS-401, Version 1. Functions are available for analog inputs that can adapt communication in the event-driven operating mode to the requirements of the application and to minimize the loading of the bus:

- Limit value monitoring
- Delta function
- activation/deactivation of event-driven mode

## 7.3.2 Object list

**Accessibility of the objects and registers**

The objects in the object directory can be reached by SDO access, but not generally through the KS2000 configuration software. On the other hand, all the registers that can be configured with KS2000 can also be reached using SDO access to the object directory (objects 0x4500 and 0x4501) - even though this does not offer the same convenience as the KS2000 software.

| Parameter | Index | BK5120/BK515x | BK5110 | LC5100 | BX5100/BC5150 | CX8051/B510 |
|---|---|---|---|---|---|---|
| Device type [▶ 100] | 0x1000 | x | x | x | | x |
| Error register [▶ 100] | 0x1001 | x | x | x | x | x * |
| Error memory [▶ 101] | 0x1003 | x | x | x | | x * |
| Sync Identifier [▶ 101] | 0x1005 | x | x | x | x | x |
| Sync Interval [▶ 101] | 0x1006 | x | x | x | x | x |
| Device name [▶ 102] | 0x1008 | x | x | x | x | x * |
| Hardware-Version [▶ 102] | 0x1009 | x | x | x | | x * |
| Software version [▶ 102] | 0x100A | x | x | x | x | x * |
| Node number [▶ 102] | 0x100B | x | x | x | | |
| Guard Time [▶ 102] | 0x100C | x | x | x | x | x |
| Life Time Factor [▶ 103] | 0x100D | x | x | x | x | x |
| Guarding Identifier [▶ 103] | 0x100E | x | x | x | | |
| Save parameters [▶ 103] | 0x1010 | x | x | x | | |
| Load default values [▶ 104] | 0x1011 | x | x | x | | |
| Emergency Identifier [▶ 104] | 0x1014 | x | x | x | | |
| Consumer Heartbeat Time [▶ 104] | 0x1016 | x | x | x | x | x |
| Producer Heartbeat Time [▶ 105] | 0x1017 | x | x | x | x | x |
| Device identifier (identity object) [▶ 105] | 0x1018 | x | x | x | x | x * |
| Server SDO parameters [▶ 105] | 0x1200 | x | x | x | | |

| Parameter | Index | BK5120/BK515x | BK5110 | LC5100 | BX5100/BC5150 | CX8051/B510 |
|---|---|---|---|---|---|---|
| Communication parameters for the 1st - 5th RxPDOs [▶ 106] | 0x1400 - 0x1404 | x | x | x | x | x |
| Communication parameters for the 6th - 16nd RxPDOs [▶ 108] | 0x1405 - 0x140F | x | | | x | x |
| Communication parameters for the 17th - 32nd RxPDOs [▶ 108] | 0x1410 - 0x141F | | | | x only BX5100 | x |
| Mapping 1st - 5th RxPDO [▶ 109] | 0x1600 - 0x1604 | x | x | x | x | x |
| Mapping 6th - 16nd RxPDO [▶ 110] | 0x1605 - 0x160F | x | | | x | x |
| Mapping 17th - 32nd RxPDO [▶ 110] | 0x1610 - 0x161F | | | | x only BX5100 | x |
| Communication parameters for the 1st - 5th TxPDOs [▶ 111] | 0x1800 - 0x1804 | x | x | x | x | x |
| Communication parameters for the 6st - 16th TxPDOs [▶ 112] | 0x1805 - 0x180F | x | | | x | x |
| Communication parameters for the 17st - 32th TxPDOs [▶ 113] | 0x1810 - 0x181F | | | | x only BX5100 | x |
| Mapping 1rd - 5th TxPDO [▶ 114] | 0x1A00 - 0x1A04 | x | x | x | x | x |
| Mapping 6th - 16nd TxPDO [▶ 114] | 0x1A05 - 0x1A0F | x | | | x | x |
| Mapping 17th - 32nd TxPDO [▶ 115] | 0x1A10 - 0x1A1F | | | | x only BX5100 | x |
| Flag area %MB0-511 [▶ 131] | 0x2F00 | | | | x | |
| Flag area %MB511-1023 [▶ 131] | 0x2F01 | | | | x | |
| Flag area %MB1024-1535 [▶ 131] | 0x2F02 | | | | x | |
| Flag area %MB1536-2047 [▶ 131] | 0x2F03 | | | | x | |
| Flag area %MB2048-2559 [▶ 131] | 0x2F04 | | | | x | |
| Flag area %MB2560-3071 [▶ 131] | 0x2F05 | | | | x | |

| Parameter | Index | BK5120/BK515x | BK5110 | LC5100 | BX5100/BC5150 | CX8051/B510 |
|---|---|---|---|---|---|---|
| Flag area %MB3072-3584 [▶ 131] | 0x2F06 | | | | x | |
| Flag area %MB3585-4095 [▶ 131] | 0x2F07 | | | | x | |
| 3-byte special terminals, input data [▶ 116] | 0x2600 | x | | | | |
| 3-byte special terminals, output data [▶ 116] | 0x2700 | x | | | | |
| 4-byte special terminals, input data [▶ 117] | 0x2800 | x | | | | |
| 4-byte special terminals, output data [▶ 117] | 0x2900 | x | | | | |
| 5-byte special terminals, input data [▶ 117] | 0x2A00 | x | | | | |
| 5-byte special terminals, output data [▶ 118] | 0x2B00 | x | | | | |
| 6-byte special terminals, input data [▶ 118] | 0x2C00 | x | | | | |
| 6-byte special terminals, output data [▶ 118] | 0x2D00 | x | | | | |
| 8-byte special terminals, input data [▶ 119] | 0x3000 | x | | | | |
| 8-byte special terminals, output data [▶ 119] | 0x3100 | x | | | | |
| Bus node register communication [▶ 119] | 0x4500 | x | x | x | | |
| Bus Terminal / Extension Box register communication [▶ 123] | 0x4501 | x | x | x | | |
| Activate PDOs [▶ 125] | 0x5500 | x | x | x | | |
| NetId [▶ 131] | 0x5FFE | | | | x | |
| Digital inputs [▶ 127] | 0x6000 | x | x | x | | |
| Interrupt mask [▶ 127] | 0x6126 | x | x | x | | |
| Digital outputs [▶ 128] | 0x6200 | x | x | x | | |
| Analog inputs [▶ 129] | 0x6401 | x | | | | |
| Analog outputs [▶ 129] | 0x6411 | x | | | | |
| Event control analog inputs [▶ 129] | 0x6423 | x | | | | |
| Upper limit value analog inputs [▶ 130] | 0x6424 | x | | | | |
| Lower limit value analog inputs [▶ 130] | 0x6425 | x | | | | |
| Delta function for analog inputs [▶ 130] | 0x6426 | x | | | | |

*) When an ADS server is registered, these objects are relayed to the PLC via ADS notification and have to be answered there.

## 7.3.3 Objects and Data

### Device type

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1000** | 0 | Device Type | Unsigned32 | ro | N | 0x00000000 | Statement of device type |

The 32-bit value is divided into two 16-bit fields:

| MSB | LSB |
|---|---|
| Additional Information | Device profile number |
| 0000 0000 0000 wxyz | 0x191 ($401_{dec}$) |

The *additional information* contains data related to the signal type of the I/O device:
z=1 signifies digital inputs,
y=1 signifies digital outputs,
x=1 signifies analog inputs,
w=1 signifies analog outputs.
A BK5120 with digital and analog inputs, but with no outputs, thus returns 0x00 05 01 91.

Special terminals (such as serial interfaces, PWM outputs, incremental encoder inputs) are not considered. A Coupler that, e.g. only has KL6001 serial interface terminals plugged in, thus returns 0x00 00 01 91.

The device type supplies only a rough classification of the device. The terminal identifier register of the Bus Coupler can be read for detailed identification of the Bus Couplers and the attached terminals (for details see Register communication index 0x4500).

### Error register

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1001** | 0 | Error Register | Unsigned8 | ro | N | 0x00 | Error register |

The 8-bit value is coded as follows:

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| ManSpec. | reserved | reserved | Comm. | reserved | reserved | reserved | Generic |

ManSpec. Manufacturer-specific error, specified more precisely in object 0x1003.

Comm.: Communication error (CAN overrun)

Generic: An error that is not more precisely specified has occurred (the flag is set at every error message)

### Error memory

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1003** | 0x00 | Predefined error field (error memory) | Unsigned8 | rw | N | 0x00 | Object 1003h contains a description of the error that has occurred in the device - subindex 0 has the number of error states stored. |
| | 1 | Actual error | Unsigned32 | ro | N | none | Last error state to have occurred |
| | ... | ... | ... | -- | ... | ... | ... |
| | 10 | Standard error field | Unsigned32 | ro | N | none | A maximum of 10 error states are stored. |

The 32 bit value in the error memory is divided into two 16 bit fields:

| MSB | LSB |
|---|---|
| Additional Code | Error Code |

The additional code contains the error trigger (see emergency object) and thereby a detailed error description.

New errors are always saved at subindex 1, all the other subindices being appropriately incremented. The whole error memory is cleared by writing a 0 to subindex 0.

If there has not been an error since power up, then object 0x1003 only consists of subindex 0 with a 0 entered into it. The error memory is cleared by a reset or a power cycle.

As is usual in CANopen, the LSB is transferred first, followed by the MSB.

### Sync Identifier

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1005** | 0 | COB-ID Sync Message | Unsigned32 | rw | N | 0x80000080 | Identifier of the SYNC message |

The bottom 11 bits of the 32-bit value contain the identifier (0x80=128 dec). Bit 30 indicates whether the device sends the SYNC telegram (1) or not (0). The CANopen I/O devices receive the SYNC telegram, and accordingly bit 30=0. For reasons of backwards compatibility, bit 31 has no significance.

### Sync Interval

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1006** | 0 | Communication cycle period | Unsigned32 | rw | N | 0x00000000 | Length of the SYNC interval in µs. |

If a value other than zero is entered here, the bus node will go into the error state if, during synchronous PDO operation, no SYNC telegram is received within the watchdog time. The watchdog time corresponds here to 1,5 times the communication cycle period that has been set - the planned SYNC interval can therefore be entered.

The I/O update is carried out at the Beckhoff CANopen bus nodes immediately after reception of the SYNC telegram, provided the following conditions are satisfied:
- Firmware status C0 or above (CANopen Version 4.01 or higher).

- All PDOs that have data are set to synchronous communication (0..240).
- Sync interval has been entered in object 0x1006 and (sync interval x lowest PDO transmission type) is less than 90ms.

The modules are then synchronized throughout.

**Device name**

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1008** | 0 | Manufacturer Device Name | Visible String | ro | N | BK51x0, LC5100, IPxxxx-B510 or ILxxxx-B510 | Device name of the bus node |

Since the returned value is longer than 4 bytes, the segmented SDO protocol is used for transmission.

**Hardware-Version**

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1009** | 0 | Manufacturer Hardware-Version | Visible String | ro | N | - | Hardware version number of the bus node |

Since the returned value is longer than 4 bytes, the segmented SDO protocol is used for transmission.

**Software version**

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x100A** | 0 | Manufacturer Software-Version | Visible String | ro | N | - | Software version number of the bus node |

Since the returned value is longer than 4 bytes, the segmented SDO protocol is used for transmission.

**Node number**

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x100B** | 0 | Node-ID | Unsigned32 | ro | N | none | Set node number |

The node number is supported for reasons of compatibility.

**Guard Time**

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x100C** | 0 | Guard Time [ms] | Unsigned16 | rw | N | 0 | Interval between two guard telegrams. Is set by the NMT master or configuration tool. |

### Life Time Factor

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x100D** | 0 | Life Time Factor | Unsigned8 | rw | N | 0 | Life time factor x guard time = life time (watchdog for life guarding) |

If a guarding telegram is not received within the life time, the node enters the error state. If the life time factor and/or guard time = 0, the node does not carry out any life guarding, but can itself be monitored by the master (node guarding).

### Guarding Identifier

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x100 E** | 0 | COB-ID guarding protocol | Unsigned32 | ro | N | 0x000007xy, xy = NodeID | Identifier of the guarding protocol |

The guarding identifier is supported for reasons of compatibility. Changing the guarding identifier has no longer been permitted since version 4 of CANopen.

### Save parameters

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1010** | 0 | Store Parameter | Unsigned8 | ro | N | 1 | Number of store options |
| | 1 | store all parameters | Unsigned32 | rw | N | 1 | Stores all (storable) parameters |

By writing the string *save* in ASCII code (hexadecimal 0x65766173) to subindex 1, the current parameters are placed into non-volatile storage. (The byte sequence on the bus including the SDO protocol: 0x23 0x10 0x10 0x01 0x73 0x61 0x76 0x65).

The storage process takes about 3 seconds, and is confirmed, if successful, by the corresponding TxSDO (0x60 in the first byte). Since the Bus Coupler is unable to send or receive any CAN telegrams during the storage process, saving is only possible when the node is in the pre-operational state. It is recommended that the entire network is placed into the pre-operational state before such storage. This avoids a buffer overflow.

Data saved includes:

- The terminals currently inserted (the number of each terminal category)
- All PDO parameters (identifier, transmission type, inhibit time, mapping).

**Valid identifiers**

The stored identifiers apply afterwards, not the default identifiers derived from the node addresses. Changes to the DIP switch setting no longer affects the PDOs!

- All SYNC parameters
- All guarding parameters
- Limit values, delta values and interrupt enables for analog inputs

Parameters directly stored in the terminals by way of register communication are immediately stored there in non-volatile form.

### Load default values

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1011** | 0 | Restore Parameter | Unsigned8 | ro | N | 4 | Number of reset options |
| | 1 | Restore all parameters | Unsigned32 | rw | N | 1 | Resets all parameters to their default values |
| | 4 | Set manufacturer Defaults | Unsigned32 | rw | N | 1 | Resets all coupler parameters to manufacturer's settings (including registers) |

Writing the string *load* in ASCII code (hexadecimal 0x64616F6C) into subindex 1 resets all parameters to default values (as initially supplied) **at the next boot (reset)**.

(The byte sequence on the bus including the SDO protocol: 0x23 0x11 0x10 0x01 0x6C 0x6F 0x61 0x64).

This makes the default identifiers for the PDOs active again.

### Emergency Identifier

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1014** | 0 | COB-ID Emergency | Unsigned32 | rw | N | 0x00000080, + NodeID | Identifier of the emergency telegram |

The bottom 11 bits of the 32-bit value contain the identifier (0x80=128 dec). The MSBit can be used to set whether the device sends (1) the emergency telegram or not (0).

Alternatively, the bus node's diagnostic function can also be switched off using the *Device diagnostics* bit in the K-bus configuration (see object 0x4500).

### Consumer Heartbeat Time

| Index | Subindex | Name | Type | Attribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1016** | 0 | Number of items | Unsigned8 | ro | N | 2 | The consumer heartbeat time describes the expected heartbeat cycle time and the node ID of the monitored node |
| | 1 | Consumer Heartbeat Time | Unsigned32 | rw | N | 0 | Watchdog time in ms and node ID of the monitored node |

The 32-bit value is used as follows:

| MSB | | LSB |
|---|---|---|
| Bit 31...24 | Bit 23...16 | Bit 15...0 |
| Reserved (0) | Node-ID (Unsigned8) | heartbeat time in ms (Unsigned16) |

The monitored identifier can be obtained from the node ID by means of the default identifier allocation: Guard-ID = 0x700 + Node-ID.

As is usual in CANopen, the LSB is transferred first, followed by the MSB.

### Producer Heartbeat Time

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1017** | 0 | Producer Heartbeat Time | Unsigned16 | rw | N | 0 | Time interval in ms between two transmitted heartbeat telegrams |

### Device identifier (identity object)

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1018** | 0 | Identity Object: Number of items | Unsigned8 | ro | N | 4 | The identity object contains general information about the type and version of the device. |
| | 1 | Vendor ID | Unsigned32 | ro | N | 0x00000002 | Manufacturer identifier. Beckhoff has vendor ID 2 |
| | 2 | Product Code | Unsigned32 | ro | N | Depends on the product | Device identifier |
| | 3 | Revision Number | Unsigned32 | ro | N | - | Version number |
| | 4 | Serial Number | Unsigned32 | ro | N | - | Date of manufacture Low word, high byte: calendar week (dec), low word, low byte: calendar year |

| Product | Product Code |
|---|---|
| BK5120 | 0x11400 |
| BK5110 | 0x113F6 |
| LC5100 | 0x113EC |
| IPwxyz-B510 | 0x2wxyz |
| IL2301-B510 | 0x2008FD |

### Server SDO parameters

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1200** | 0 | Number of items | Unsigned8 | ro | N | 2 | Communication parameters of the server SDO. Subindex 0: Number of following parameters |
| | 1 | COB-ID Client ->Server | Unsigned32 | ro | N | 0x000006xy, xy=Node-ID | COB-ID RxSDO (Client -> Server) |
| | 2 | COB-ID Server ->Client | Unsigned32 | ro | N | 0x00000580 + Node-ID | COB-ID TxSDO (Client -> Server) |

This is contained in the object directory for reasons of backwards compatibility.

**Communication parameter 1st RxPDO**

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1400** | 0 | Number of items | Unsigned8 | ro | N | 5 | Communication parameters for the first receive PDO. Subindex 0: Number of following parameters |
| | 1 | COB-ID | Unsigned32 | rw | N | 0x000002xy, xy=Node-ID | COB-ID (Communication Object Identifier) RxPDO1 |
| | 2 | Transmissio n Type | Unsigned8 | rw | N | 255 | Transmission type of the PDO |
| | 3 | Inhibit Time | Unsigned16 | rw | N | 0 | Present for reasons of backwards compatibility, but not used in the RxPDO. |
| | 4 | CMS Priority Group | Unsigned8 | rw | N | - | Present for reasons of backwards compatibility, but not used. |
| | 5 | Event Timer | Unsigned16 | rw | N | 0 | Event-Timer. Watchdog time defined for monitoring reception of the PDO. |

Subindex 1 (COB-ID): The bottom 11 bits of the 32 bit value (bits 0-10) contain the CAN identifier. The MSB (bit 31) indicates whether the PDO exists currently (0) or not (1). Bit 30 indicates whether an RTR access to this PDO is permissible (0) or not (1). Changing the identifier (bits 0-10) is not allowed while the object exists (bit 31=0). Subindex 2 contains the transmission type (see introduction to PDOs).

### Communication parameter 2nd RxPDO

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1401** | 0 | Number of items | Unsigned8 | ro | N | 5 | Communication parameter for the second receive PDO. |
| | 1 | COB-ID | Unsigned32 | rw | N | 0x000003xy, xy=Node-ID | COB-ID (Communication Object Identifier) RxPDO2 |
| | 2 | Transmissio n Type | Unsigned8 | rw | N | 255 | Transmission type of the PDO |
| | 3 | Inhibit Time | Unsigned16 | rw | N | 0 | Present for reasons of backwards compatibility, but not used in the RxPDO. |
| | 4 | CMS Priority Group | Unsigned8 | rw | N | - | Present for reasons of backwards compatibility, but not used. |
| | 5 | Event Timer | Unsigned16 | rw | N | 0 | Event-Timer. Watchdog time defined for monitoring reception of the PDO. |

### Communication parameter 3rd RxPDO

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1402** | 0 | Number of items | Unsigned8 | ro | N | 5 | Communication parameter for the third receive PDO. |
| | 1 | COB-ID | Unsigned32 | rw | N | 0x000004xy, xy=Node-ID | COB-ID (Communication Object Identifier) RxPDO3 |
| | 2 | Transmissio n Type | Unsigned8 | rw | N | 255 | Transmission type of the PDO |
| | 3 | Inhibit Time | Unsigned16 | rw | N | 0 | Present for reasons of backwards compatibility, but not used in the RxPDO. |
| | 4 | CMS Priority Group | Unsigned8 | rw | N | - | Present for reasons of backwards compatibility, but not used. |
| | 5 | Event Timer | Unsigned16 | rw | N | 0 | Event-Timer. Watchdog time defined for monitoring reception of the PDO. |

### Communication parameter 4th RxPDO

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1403** | 0 | Number of items | Unsigned8 | ro | N | 5 | Communication parameters for the fourth receive PDO. |
| | 1 | COB-ID | Unsigned32 | rw | N | 0x000005xy, xy=Node-ID | COB-ID (Communication Object Identifier) RxPDO4 |
| | 2 | Transmissio n Type | Unsigned8 | rw | N | 255 | Transmission type of the PDO |
| | 3 | Inhibit Time | Unsigned16 | rw | N | 0 | Present for reasons of backwards compatibility, but not used in the RxPDO. |
| | 4 | CMS Priority Group | Unsigned8 | rw | N | - | Present for reasons of backwards compatibility, but not used. |
| | 5 | Event Timer | Unsigned16 | rw | N | 0 | Event-Timer. Watchdog time defined for monitoring reception of the PDO. |

### Communication parameters 5th-16th RxPDO

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1404 - 0x140F (depending on the device type)** | 0 | Number of items | Unsigned8 | ro | N | 5 | Communication parameter for the 5th to 16th receive PDOs. |
| | 1 | COB-ID | Unsigned32 | rw | N | 0x8000000 | COB-ID (Communication Object Identifier) RxPDO5...16 |
| | 2 | Transmissio n Type | Unsigned8 | rw | N | 255 | Transmission type of the PDO |
| | 3 | Inhibit Time | Unsigned16 | rw | N | 0 | Present for reasons of backwards compatibility, but not used in the RxPDO. |
| | 4 | CMS Priority Group | Unsigned8 | rw | N | - | Present for reasons of backwards compatibility, but not used. |
| | 5 | Event Timer | Unsigned16 | rw | N | 0 | Event-Timer. Watchdog time defined for monitoring reception of the PDO. |

The number of RxPDOs for each bus node type can be found in the technical data.

**Parameter / mapping RxPDO**

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1600** | 0 | Number of items | Unsigned8 | rw | N | Depending on type and fittings | Mapping parameter of the first receive PDO; subindex 0: number of mapped objects. |
| | 1 | 1st mapped object | Unsigned32 | rw | N | 0x62000108 | 1st mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |
| | 2 | 2nd mapped object | Unsigned32 | rw | N | 0x62000208 | 2nd mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |
| | ... | ... | ... | ... | ... | ... | ... |
| | 8 | 8th mapped object | Unsigned32 | rw | N | 0x62000808 | 8th mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |

The first receive PDO (RxPDO1) is provided by default for digital output data. Depending on the number of outputs inserted, the necessary length of the PDO is automatically determined, and the corresponding objects are mapped. Since the digital outputs are organized in bytes, the length of the PDO in bytes can be found directly at subindex 0.

**Changes to the mapping**

The following sequence must be observed in order to change the mapping (specified as from CANopen, version 4):

1. Delete PDO (set bit 31 in the identifier entry (subindex 1) of the communication parameters to 1)
2. Deactivate mapping (set subindex 0 of the mapping entry to 0)
3. Change mapping entries (subindices 1...8)
4. Activate mapping (set subindex 0 of the mapping entry to the correct number of mapped objects)
5. Create PDO (set bit 31 in the identifier entry (subindex 1) of the communication parameters to 0)

### Mapping parameter 2nd RxPDO

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1601** | 0 | Number of items | Unsigned8 | rw | N | Depending on type and fittings | Mapping parameter of the second receive PDO; subindex 0: number of mapped objects. |
| | 1 | 1st mapped object | Unsigned32 | rw | N | 0x64110110 | 1st mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |
| | 2 | 2st mapped object | Unsigned32 | rw | N | 0x64110210 | 2nd mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |
| | ... | ... | ... | ... | ... | ... | ... |
| | 8 | 8th mapped object | Unsigned32 | rw | N | 0x00000000 | 8th mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |

The second receive PDO (RxPDO2) is provided by default for analog outputs. Depending on the number of outputs inserted, the necessary length of the PDO is automatically determined, and the corresponding objects are mapped. Since the analog outputs are organized in words, the length of the PDO in bytes can be found directly at subindex 0.

A specific sequence must be observed in order to change the mapping (see object index 0x1600).

### Mapping-parameters 3rd - 16th RxPDO

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1602-0x160F (depending on the device type)** | 0 | Number of items | Unsigned8 | rw | N | Depending on type and fittings | Mapping parameters for the 3rd to 16th receive PDOs; subindex 0: number of mapped objects. |
| | 1 | 1st mapped object | Unsigned32 | rw | N | 0x00000000 (see text) | 1st mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |
| | 2 | 2nd mapped object | Unsigned32 | rw | N | 0x00000000 (see text) | 2nd mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |
| | ... | ... | ... | ... | ... | ... | ... |
| | 8 | 8th mapped object | Unsigned32 | rw | N | 0x00000000 (see text) | 8th mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |

The 3rd to 16th receive PDOs (RxPDO3ff) are automatically given a default mapping by the bus node depending on the attached terminals (or depending on the extension modules). The procedure is described in the section on PDO Mapping [▶ 73].

A specific sequence must be observed in order to change the mapping (see object index 0x1600).

### Beckhoff Default Mapping

DS401 V2 specifies analog input and/or output data as the default mapping for PDOs 3 and 4. This corresponds to Beckhoff's default mapping when less than 65 digital inputs or outputs are present. In order to ensure backwards compatibility, the Beckhoff default mapping is retained - the mapping behavior of the devices therefore corresponds to DS401 V1, where in all other respects they accord with DS401 V2.

**Communication parameter 1st TxPDO**

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1800** | 0 | Number of items | Unsigned8 | ro | N | 5 | Communication parameters for the first transmit PDO. Subindex 0: Number of following parameters |
| | 1 | COB-ID | Unsigned32 | rw | N | 0x00000180 + Node-ID | COB-ID (Communication Object Identifier) TxPDO1 |
| | 2 | Transmissio n Type | Unsigned8 | rw | N | 255 | Transmission type of the PDO |
| | 3 | Inhibit Time | Unsigned16 | rw | N | 0 | Repetition delay [value x 100 µs] |
| | 4 | CMS Priority Group | Unsigned8 | rw | N | - | Present for reasons of backwards compatibility, but not used. |
| | 5 | Event Timer | Unsigned16 | rw | N | 0 | Event-Timer |

Subindex 1 (COB-ID): The bottom 11 bits of the 32 bit value (bits 0-10) contain the CAN identifier. The MSB (bit 31) indicates whether the PDO exists currently (0) or not (1). Bit 30 indicates whether an RTR access to this PDO is permissible (0) or not (1). Changing the identifier (bits 0-10) is not allowed while the object exists (bit 31=0). Subindex 2 contains the transmission type, subindex 3 the repetition delay between two PDOs of the same type, while subindex 5 contains the event timer. Subindex 4 is retained for reasons of compatibility, but is not used. (See also Introduction to PDOs.)

### Communication parameter 2nd TxPDO

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1801** | 0 | Number of items | Unsigned8 | ro | N | 5 | Communication parameters for the second transmit PDO. Subindex 0: Number of following parameters |
| | 1 | COB-ID | Unsigned32 | rw | N | 0x00000280 + Node-ID | COB-ID (Communication Object Identifier) TxPDO1 |
| | 2 | Transmissio n Type | Unsigned8 | rw | N | 255 | Transmission type of the PDO |
| | 3 | Inhibit Time | Unsigned16 | rw | N | 0 | Repetition delay [value x 100 µs] |
| | 4 | CMS Priority Group | Unsigned8 | rw | N | - | Present for reasons of backwards compatibility, but not used. |
| | 5 | Event Timer | Unsigned16 | rw | N | 0 | Event-Timer |

The second transmit PDO is provided by default for analog inputs, and is configured for event-driven transmission (transmission type 255). Event-driven mode must first be activated (see object 0x6423), otherwise the inputs can only be interrogated (polled) by Remote Transmission Request (RTR).

### Communication parameter 3rd TxPDO

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1802** | 0 | Number of items | Unsigned8 | ro | N | 5 | Communication parameters for the third transmit PDO. Subindex 0: Number of following parameters |
| | 1 | COB-ID | Unsigned32 | rw | N | 0x00000380 + Node-ID | COB-ID (Communication Object Identifier) TxPDO1 |
| | 2 | Transmissio n Type | Unsigned8 | rw | N | 255 | Transmission type of the PDO |
| | 3 | Inhibit Time | Unsigned16 | rw | N | 0 | Repetition delay [value x 100 µs] |
| | 4 | CMS Priority Group | Unsigned8 | rw | N | - | Present for reasons of backwards compatibility, but not used. |
| | 5 | Event Timer | Unsigned16 | rw | N | 0 | Event-Timer |

The third transmit PDO contains analog input data as a rule (see Mapping [▶ 73]). It is configured for event-driven transmission (transmission type 255). Event-driven mode must first be activated (see object 0x6423), otherwise the inputs can only be interrogated (polled) by Remote Transmission Request (RTR).

### Communication parameter 4th TxPDO

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1803** | 0 | Number of items | Unsigned8 | ro | N | 5 | Communication parameters for the fourth transmit PDO. Subindex 0: Number of following parameters |
| | 1 | COB-ID | Unsigned32 | rw | N | 0x00000480 + Node-ID | COB-ID (Communication Object Identifier) TxPDO1 |
| | 2 | Transmission Type | Unsigned8 | rw | N | 255 | Transmission type of the PDO |
| | 3 | Inhibit Time | Unsigned16 | rw | N | 0 | Repetition delay [value x 100 µs] |
| | 4 | CMS Priority Group | Unsigned8 | rw | N | - | Present for reasons of backwards compatibility, but not used. |
| | 5 | Event Timer | Unsigned16 | rw | N | 0 | Event-Timer |

The fourth transmit PDO contains analog input data as a rule (see Mapping [▶ 73]). It is configured for event-driven transmission (transmission type 255). Event-driven mode must first be activated (see object 0x6423), otherwise the inputs can only be interrogated (polled) by Remote Transmission Request (RTR).

### Communication parameters 5th - 16th RxPDO

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1804-0x180F (depending on the device type)** | 0 | Number of items | Unsigned8 | ro | N | 5 | Communication parameters for the 5th - 16th transmit PDOs. Subindex 0: Number of following parameters |
| | 1 | COB-ID | Unsigned32 | rw | N | 0x0000000 | COB-ID (Communication Object Identifier) TxPDO1 |
| | 2 | Transmission Type | Unsigned8 | rw | N | 255 | Transmission type of the PDO |
| | 3 | Inhibit Time | Unsigned16 | rw | N | 0 | Repetition delay [value x 100 µs] |
| | 4 | CMS Priority Group | Unsigned8 | rw | N | - | Present for reasons of backwards compatibility, but not used. |
| | 5 | Event Timer | Unsigned16 | rw | N | 0 | Event-Timer |

### Mapping 1nd TxPDO

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1A00** | 0 | Number of items | Unsigned8 | rw | N | Depending on type and fittings | Mapping parameter of the first transmit PDO; subindex 0: number of mapped objects. |
| | 1 | 1st mapped object | Unsigned32 | rw | N | 0x60000108 | 1st mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |
| | 2 | 2nd mapped object | Unsigned32 | rw | N | 0x60000208 | 2nd mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |
| | ... | ... | ... | ... | ... | ... | ... |
| | 8 | 8th mapped object | Unsigned32 | rw | N | 0x60000808 | 8th mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |

The first transmit PDO (TxPDO1) is provided by default for digital input data. Depending on the number of inputs inserted, the necessary length of the PDO is automatically determined, and the corresponding objects are mapped. Since the digital inputs are organized in bytes, the length of the PDO in bytes can be found directly at subindex 0.

A specific sequence must be observed in order to change the mapping (see object index 0x1600).

### Mapping 2nd TxPDO

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1A01** | 0 | Number of items | Unsigned8 | rw | N | Depending on type and fittings | Mapping parameter of the second transmit PDO; subindex 0: number of mapped objects. |
| | 1 | 1st mapped object | Unsigned32 | rw | N | 0x64010110 | 1st mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |
| | 2 | 2nd mapped object | Unsigned32 | rw | N | 0x64010210 | 2nd mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |
| | ... | ... | ... | ... | ... | ... | ... |
| | 8 | 8th mapped object | Unsigned32 | rw | N | | 8th mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |

The second transmit PDO (TxPDO2) is provided by default for analog input data. Depending on the number of inputs inserted, the necessary length of the PDO is automatically determined, and the corresponding objects are mapped. Since the analog inputs are organized in words, the length of the PDO in bytes can be found directly at subindex 0.

A specific sequence must be observed in order to change the mapping (see object index 0x1600).

**Mapping 3rd - 16th TxPDO**

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x1A02-0x1A0F (depending on the device type)** | 0 | Number of items | Unsigned8 | rw | N | Depending on type and fittings | Mapping parameters for the 3rd to 16th transmit PDOs; subindex 0: number of mapped objects. |
| | 1 | 1st mapped object | Unsigned32 | rw | N | 0x00000000 (see text) | 1st mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |
| | 2 | 2nd mapped object | Unsigned32 | rw | N | 0x00000000 (see text) | 2nd mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |
| | ... | ... | ... | ... | ... | ... | ... |
| | 8 | 8th mapped object | Unsigned32 | rw | N | 0x00000000 (see text) | 8th mapped application object (2 bytes index, 1 byte subindex, 1 byte bit width) |

The 3rd to 16th transmit PDOs (TxPDO3ff) are automatically given a default mapping by the bus node depending on the attached terminals (or depending on the extension modules). The procedure is described in the section on PDO Mapping [▶ 73].

A specific sequence must be observed in order to change the mapping (see object index 0x1600).

**Beckhoff Default Mapping**

DS401 V2 specifies analog input and/or output data as the default mapping for PDOs 3 and 4. This corresponds to Beckhoff's default mapping when less than 65 digital inputs or outputs are present. In order to ensure backwards compatibility, the Beckhoff default mapping is retained - the mapping behavior of the devices therefore corresponds to DS401 V1, where in all other respects they accord with DS401 V2.

For the sake of completeness, the following object entries are also contained in the object directory (and therefore also in the eds files):

| Index | Meaning |
|---|---|
| **0x2000** | Digital inputs (function identical to object 0x6000) |
| **0x2100** | Digital outputs (function identical with object 0x6200) |
| **0x2200** | 1-byte special terminals, inputs (at present no terminals corresponding to this type are included in the product range) |
| **0x2300** | 1-byte special terminals, outputs (at present no terminals corresponding to this type are included in the product range) |
| **0x2400** | 2-byte special terminals, inputs (at present no terminals corresponding to this type are included in the product range) |
| **0x2500** | 2-byte special terminals, outputs (at present no terminals corresponding to this type are included in the product range) |
| **0x2E00** | 7-byte special terminals, inputs (at present no terminals corresponding to this type are included in the product range) |
| **0x2F00** | 7-byte special terminals, outputs (at present no terminals corresponding to this type are included in the product range) |

**3-byte special terminals, input data**

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x2600** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of available 3-byte special channels, inputs |
| | 1 | 1st input block | Unsigned24 | ro | Y | 0x000000 | 1st input channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0X80 | 128. input block | Unsigned24 | ro | Y | 0x000000 | 128st input channel |

Example of special terminals with 3-byte input data (in the default setting): KL2502 (PWM outputs, 2 x 3 bytes)

**3-byte special terminals, output data**

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x2700** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of available 3-byte special channels, outputs |
| | 1 | 1st output block | Unsigned24 | rww | Y | 0x000000 | 1st output channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0X80 | 128. output block | Unsigned24 | rww | Y | 0x000000 | 128st output channel |

Example of special terminals with 3-byte output data (in the default setting): KL2502 (PWM outputs, 2 x 3 bytes)

### 4-byte special terminals, input data

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x2800** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of available 4-byte special channels, inputs |
| | 1 | 1st input block | Unsigned32 | ro | Y | 0x00000000 | 1st input channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0X80 | 128. input block | Unsigned32 | ro | Y | 0x00000000 | 128st input channel |

Examples of special terminals with 4-byte input data (in the default setting): KL5001, KL6001, KL6021, KL6051

### 4-byte special terminals, output data

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x2900** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of available 4-byte special channels, outputs |
| | 1 | 1st output block | Unsigned32 | rww | Y | 0x00000000 | 1st output channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0X80 | 128. output block | Unsigned32 | rww | Y | 0x00000000 | 128st output channel |

Examples of special terminals with 4-byte output data (in the default setting): KL5001, KL6001, KL6021, KL6051

### 5-byte special terminals, input data

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x2A00** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of available 5-byte special channels, inputs |
| | 1 | 1st input block | Unsigned40 | ro | Y | 0x0000000000 | 1st input channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0X40 | 64. input block | Unsigned40 | ro | Y | 0x0000000000 | 64st input channel |

Example of special terminals with 5-byte input data (in the default setting): KL1501

**5-byte special terminals, output data**

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x2B00** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of available 5-byte special channels, outputs |
| | 1 | 1st output block | Unsigned40 | rww | Y | 0x0000000000 | 1st output channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0X40 | 64. output block | Unsigned40 | rww | Y | 0x0000000000 | 64st output channel |

Example of special terminals with 5-byte output data (in the default setting): KL1501

**6-byte special terminals, input data**

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x2C00** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of available 6-byte special channels, inputs |
| | 1 | 1st input block | Unsigned48 | ro | Y | 0x0000000000 | 1st input channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0X40 | 64. input block | Unsigned48 | ro | Y | 0x0000000000 | 64st input channel |

Example of special terminals with 6-byte input data (in the default setting): KL5051, KL5101, KL5111

**6-byte special terminals, output data**

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x2D00** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of available 6-byte special channels, outputs |
| | 1 | 1st output block | Unsigned48 | rww | Y | 0x0000000000 | 1st output channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0X40 | 64. output block | Unsigned48 | rww | Y | 0x0000000000 | 64st output channel |

Example of special terminals with 6-byte output data (in the default setting): KL5051, KL5101, KL5111

### 8-byte special terminals, input data

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x3000** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of available 6-byte special channels, inputs |
| | 1 | 1st input block | Unsigned64 | ro | Y | 0x0000000000 | 1st input channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0x40 | 64. input block | Unsigned64 | ro | Y | 0x0000000000 | 64st input channel |

Example for special terminals with 8-byte input data: KL5101 (with word alignment, not according to the default setting)

### 8-byte special terminals, output data

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x3100** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of available 6-byte special channels, outputs |
| | 1 | 1st output block | Unsigned64 | rww | Y | 0x0000000000 | 1st output channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0X40 | 64. output block | Unsigned64 | rww | Y | 0x0000000000 | 64st output channel |

Example for special terminals with 8-byte output data: KL5101 (with word alignment, not according to the default setting)

### Bus node register communication

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x4500** | 0 | Register Access | Unsigned32 | rw | N | none | Access to internal bus node registers |

The 32 bit value is composed as follows:

| MSB | | | LSB |
|---|---|---|---|
| Access (bit 7) + table number (bit 6...0) | Register number | High byte register value | Low byte register value |
| [0..1] + [0...0x7F] | [0...0xFF] | [0...0xFF] | [0...0xFF] |

As is usual in CANopen, the LSB is transferred first, followed by the MSB.

Accessing index 0x4500 allows any registers in the bus station to be written or read. The channel number and the register are addressed here with a 32 bit data word.

### Reading the register value

The coupler must first be informed of which register is to be read. This requires an SDO write access to the appropriate index/subindex combination, with:
- table number (access bit = 0) in byte 3
- register address in byte 2 of the data value 32 bit.

Bytes 1 and 0 are not evaluated if the access bit (MSB of byte 3) equals 0. The register value can then be read with the same combination of index and subindex.

After the writing of the register address to be read, the coupler sets the access bit to 1 until the correct value is available. Thus an SDO read access must check that the table number lies in the range from 0...0x7F.

An access error during register communication is indicated by the corresponding return value in the SDO protocol (see the SDO section, Breakdown of parameter communication).

### An example of reading register values

It is necessary to determine which baud rate index has been assigned to switch setting 1,1 (DIP 7,8). (See the section covering *Network addresses and baud rates*). To do this, the value in table 100, register 3, must be read. This means that the following SDO telegrams must be sent:

Write access (download request) to index 0x4500, subindex 0 with data value 32 bit 00x64 03 00 00.

*Id=0x600+Node-ID DLC=8; Data=23 00 45 00 00 00 03 64*

Then a read access (upload request) to the same index/subindex. The data value sent here is irrelevant (00 is used here).

*Id=0x600+Node-ID DLC=8; Data=40 00 45 00 00 00 00 00*

The coupler responds with the upload response telegram:

*Id=0x580+Node-ID DLC=8; Data=43 00 45 00 04 00 03 64*

This tells us that the value contained in this register is 4, and this baud rate index corresponds to 125 kbit/s (the default value).

### Writing register values

SDO write access to the corresponding combination of index and subindex with:
- table number + 0x80 (access bit = 1) in byte 3
- register address in byte 2
- high byte register value in byte 1
- low byte register value in byte 0 of the 32-bit data value.

### Remove coupler write protection

Before the registers of the Bus Coupler can be written, the write protection must first be removed. In order to do this, the following values must be written in the given sequence to the corresponding registers:

| Step | Table | Register | Value | corresponding SDO download value (0x4500/0) |
|---|---|---|---|---|
| **1.** | 99 | 2 | 45054 (0xAFFE) | 0xE3 02 AF FE (0xE3=0x63(=99)+0x80) |
| **2.** | 99 | 1 | 1 (0x0001) | 0xE3 01 00 01 |
| **3.** | 99 | 0 | 257 (0x0101) | 0xE3 00 01 01 |

### Remove coupler write protection (CAN representation)

In order to remove the coupler write protection, the following SDO telegrams (download requests) must thus be sent to the coupler:

*Id=0x600+Node-ID DLC=8; Data=23 00 45 00 FE AF 02 E3*

*Id=0x600+Node-ID DLC=8; Data=23 00 45 00 01 00 01 E3*

*Id=0x600+Node-ID DLC=8; Data=23 00 45 00 01 01 00 E3*

### An example of writing register values

After the write protection has been removed, the baud rate index for DIP switch setting 1,1 is to be set to the value 7. This will assign a baud rate of 20 kbaud to this switch setting.

This requires the value 7 to be written into table 100, register 3. This is done with an SDO write access (download request) to index 0x4500, subindex 0 with the 32 bit value E4 03 00 07 (0xE4 = 0x64+0x80):

*Id=0x600+Node-ID DLC=8; Data=23 00 45 00 07 00 03 E4*

### Identify terminals

The identifier of the coupler (or of the bus station) and of the attached Bus Terminals can be read from the Bus Coupler's table 9. Register 0 then contains the identifier of the Bus Coupler itself, register 1 the identifier of the first terminal and register n the identification of the nth terminal:

| Table number | Register number | Description | Range of values |
|---|---|---|---|
| **9** | 0 | Bus station identifier | 0 - 65535 |
| **9** | 1-255 | Identifier of the extension module/bus terminal | 0 - 65535 |

The Bus Coupler description in register number 0 contains 5120 = 0x1400 for the BK5120, 5110 = 0x13F6 for the BK5110 and 5100 = 0x13EC for the LC5100. For the Fieldbus Box modules, register 0 contains the ID 510dec = 0x1FE or 518dec = 0x206.

In the case of analog and special terminals, the terminal identifier (dec) is contained in the extension module identifier or the terminal description.
Example: if a KL3042 is plugged in as the third terminal, then register 3 contains the value $3042_{dec}$ (0x0BE2).

The following bit identifier is used for digital terminals:

| MSB | | | | | | | | LSB | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | s6 | s5 | s4 | s3 | s2 | s1 | s0 | 0 | 0 | 0 | 0 | 0 | 0 | a | e |

s6...s1: data width in bits; a=1: output terminal; e=1: Input terminal

This ID leads to the following terminal descriptions:

| Identifier terminals | Meaning |
|---|---|
| 0x8201 | 2 bit digital input terminal, e.g. KL1002, KL1052, Kl9110, KL9260 |
| 0x8202 | 2 bit digital output terminal, e.g. KL2034, KL2612, KL2702 |
| 0x8401 | 4 bit digital input terminal, e.g. KL1104, KL1124, KL1194 |
| 0x8402 | 4 bit digital output terminal, e.g. KL2124, KL2134, KL2184 |
| 0x8403 | 4 bit digital in/output terminal, e.g. KL2212 |

and the following ID for extension box modules:

| Identifier extension box modules | Meaning |
|---|---|
| 0x000A | 4 bit input and 4 bit output module |
| 0x0011 | 8 bit input and 8 bit output module |
| 0x0014 | 8 bit digital input module |
| 0x0015 | 8 bit digital output module |

### General coupler configuration (table 0)

Table 0 of the Bus Coupler contains the data for the general coupler configuration. It is not, as a general rule, necessary to change this; however, for special applications it is possible to change the settings using the KS2000 configuration software, or through direct access via register communication. The write protection must first be removed in order to do this (see above).

The relevant register entries are described below:

### K-bus configuration

Table 0, register 2 contains the K-bus configuration and is coded as follows (default value: 0x0006):

| MSB | | | | | | | | LSB | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | D | G | A |

**A: Auto-reset**

If there is a K-bus error, attempts are made cyclically to start the K-bus up again through a reset. If emergency telegrams and guarding are not evaluated, activation of auto-reset can lead to output and input information being lost without that loss being noticed.

0: No auto-reset (default)

1: Auto-reset active

**G: Device diagnostics**

Message (via emergency), e.g. that
- open circuit at current inputs (with diagnostics)
- 10 V exceeded at 1-10 V input terminal

0: Device diagnostics switched off

1: Device diagnostics active (default)

**D: Diagnostic data**

show digital terminals into the process image (e.g. KL2212). This flag is only evaluated when device diagnostics is active (see above).

0: Do not display

1: Display (default)

**Process image description**

Table 0, register 3 contains the process image description and is coded as follows (default value: 0x0903):

| MSB | | | | | | | | LSB | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | k1 | k0 | f1 | f0 | 0 | 0 | a | 0 | d | k | 1 | 1 |

**k0...k1: Reaction to K-bus errors**

0,2: Inputs remain unchanged (default = 2);

1: Set inputs to 0 (TxPDO with zeros is sent)

**f0...f1: Reaction to fieldbus error**

0: Stop the K-Bus cycles, watchdog in the terminals triggers, fault output values become active. The old output values are initially set during a restart.

1: Set outputs to 0, then stop the K-Bus cycles (default). 2: Outputs remain unchanged.

**a: Word alignment of analog and special terminals**

0: no alignment (default)

1: Map data to word boundaries (process data always starts on an even address in the PDO)

**d: Data format for complex terminals (analog and special terminals)**

0: Intel format (default)

1: Motorola format

**k: Evaluation of complex terminals (analog and special terminals)**

0: only user data (default)

1: Complete evaluation (Attention: analog channels then, for example, need 3 input and 3 output bytes instead of, e.g., 2 input bytes; instead of 4 channels per PDO, 2 channels require a RxPDO and a TxPDO)

**Bus Terminal / Extension Box register communication**

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x4501** | 0 | Access Terminal Register | Unsigned8 | ro | N | none | Index 0x4501 allows access to all the registers in the bus terminal or extension module. Subindex 0 contains the number of attached bus terminals. |
| | 1 | Access Reg. Terminal 1 | Unsigned32 | rw | N | none | Access to bus terminal or extension module register 1 |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0XFE | Access Reg. Terminal 254 | Unsigned32 | rw | N | none | Access to bus terminal or extension module register 254 |

The 32 bit value is composed as follows:

| MSB | | | LSB |
|---|---|---|---|
| Access (bit 7 + channel number (bit 6...0) | Register number | High byte register value | Low byte register value |
| [0..1] + [0...0x7F] | [0...0xFF] | [0...0xFF] | [0...0xFF] |

As is usual in CANopen, the LSB is transferred first, followed by the MSB.

Accessing index 0x4501 allows the user registers in the bus terminal or extension module to be written or read. The modules have a set of registers for each input or output channel. The modules are addressed by means of the subindex; the channel number and register are addressed in the 32-bit data value. Channel number 0 corresponds here to the first channel, 1 to the second channel, and so forth.

**Reading the register value**

The coupler must first be informed of which register is to be read. This requires an SDO write access to the appropriate index/subindex combination, with:
- channel number (access bit = 0) in byte 3
- register address in byte 2 of the data value 32 bit.

Bytes 1 and 0 are not evaluated if the access bit (MSB of byte 3) equals 0. The register value can then be read with the same combination of index and subindex.

After the writing of the register address to be read, the coupler sets the access bit to 1 until the correct value is available. Thus an SDO read access must check that the table number lies in the range from 0...0x7F.

An access error during register communication is indicated by the corresponding return value in the SDO protocol (see the SDO section, Breakdown of parameter communication).

**An example of reading register values**

The thermocouple type to which the second input channel of a KL3202 Thermocouple Input Terminal has been set is to be determined. This requires feature register 32 to be read. The terminal is located in the fifth slot, next to the Bus Coupler. This means that the following SDO telegrams must be sent:

Write access (download request) to index 0x4501, subindex 5 with 32 bit data value 01 20 00 00 (0x01 = 2nd channel, 00x20 = register 32)
*Id=0x600+Node-ID DLC=8; Data=23 01 45 05 00 00 20 01*

Then a read access (upload request) to the same index/subindex. The data value sent here is irrelevant (0x00 is used here).
*Id=0x600+Node-ID DLC=8; Data=40 01 45 05 00 00 00 00*

The coupler responds with the upload response telegram:
*Id=0x580+Node-ID DLC=8; Data=43 01 45 05 06 31 20 01*

This means that the feature register contains the value 31 06. The upper 4 bits indicate the thermocouple type. Their value here is 3, which means that PT500 is the type that has been set for this channel (see the KL3202 documentation).

### Writing register values

SDO write access to the corresponding combination of index and subindex with:
- channel number + 0x80 (access bit = 1) in byte 3
- register address in byte 2
- high byte register value in byte 1
- low byte register value in byte 0 of the 32-bit data value.

**NOTE**

**Check the value that was written**

If the write protection is not removed (as a result, for instance, of a faulty code word), then although a write access to the terminal register will be confirmed (SDO download response), the value is not in fact entered into the register. It is therefore recommended that the value is read back after writing and compared.

### Remove terminal write protection

Before the user registers in the Bus Terminal (register 32-xx, depending on terminal type or extension module) can be written to, it is first necessary for write protection to be removed. The following code word is written for this purpose into register 31 of the channel concerned:

| Write protection | Channel | Register | Value | Corresponding SDO download value (0x4500/0) |
|---|---|---|---|---|
| | 1, 2, 3 or 4 | 31 (0x1F) | 4661 (0x1235) | 8y 1F 12 35 (y = channel number) |

### Remove terminal write protection (CAN representation)

In order to remove the terminal's write protection, the following SDO telegram must thus be sent to the coupler:

*Id=600 + Node-ID DLC=8; Data=23 01 45 xx 35 12 1F 8y*

where xx is the terminal's slot, and y indicates the channel.

### An example of removing write protection

Suppose that a KL3202 Thermocouple Input Terminal is inserted into slot 5 of a BK5120 that has node address 3, then the write protection for the first channel can be removed as follows:

*Id=0x603 DLC=8; Data=23 01 45 05 35 12 1F 80*

The following telegram is sent for the second channel:

*Id=0x603 DLC=8; Data=23 01 45 05 35 12 1F 81*

### An example of writing register values

The type of thermocouple attached to the second channel of the KL3202 Terminal in slot 5 is now to be changed to PT1000. For this purpose, the value 2 must be written into the upper 4 bits (the upper nibble) of the feature register. It is assumed to that the default values are to be supplied for all the other bits in the

feature register. Once the write protection has been removed, SDO write access (download request) is used to write the following 32 bit value into index 0x4501, subindex 05: 81 20 21 06 (0x81=01+0x80; 0x20=32;0x2106 = register value).

The corresponding telegram on the bus looks like this:

*Id=0x600+Node-ID DLC=8; Data=23 01 45 05 06 21 20 81*

### Activate PDOs

| Index | Subindex | Name | Type | At-tribute | Mapping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x5500** | 0 | Activate PDO Defaults | Unsigned32 | rw | N | 0x00000000 | sets PDO communication parameters for PDOs 2...11 |

CANopen defines default identifiers for 4 transmit (Tx) and receive (Rx) PDOs, all other PDOs being initially deactivated after the nodes have started up. Index 0x5500 can activate all the PDOs that, in accordance with the terminals inserted, are filled with process data (manufacturer-specific default mapping). A manufacturer-specific default identifier allocation is carried out here for PDO5…11, while the transmission type and a uniform inhibit time is set for PDO 2…11. PDOs that do not have process data (and which are thus superfluous in the present configuration) are not activated.

**Pre-operational required**

This object can only be written in the pre-operational state!

The 32-bit value is used as follows:

| MSB | | | LSB |
|---|---|---|---|
| Transmission Type RxPDOs | Transmission Type TxPDOs | High byte inhibit time | Low byte inhibit time |

As is usual in CANopen, the LSB is transferred first, followed by the MSB.

### Example:

Activate PDOs for bus node number 1, set inhibit time to 10 ms (=100 x 100 µs), set transmission type for TxPDOs to 255, and set transmission type for RxPDOs to 1. The following telegram must be sent:
*Id=0x601 DLC=8; Data=23 00 55 00 64 00 FF 01*

The node responds with the following telegram:
*Id=0x601 DLC=8; Data=60 00 55 00 00 00 00 00*

### Identifiers used

The default identifier allocation for the additional PDOs leaves the pre-defined regions for guarding, SDOs etc. free, assumes a maximum of 64 nodes in the network with PDO6 as the next node, and proceeds according to the following scheme:

| Object | Function Code | Resulting COB ID (hex) | Resulting COB ID (dec) |
|---|---|---|---|
| **TxPDO5** | 1101 | 0x681 - 0x6BF | 1665 - 1727 |
| **RxPDO5** | 1111 | 0x781 - 0x7BF | 1921 - 1983 |
| **TxPDO6** | 00111 | 0x1C1 - 0x1FF | 449 - 511 |
| **RxPDO6** | 01001 | 0x241 - 0x27F | 577 - 639 |
| **TxDPO7** | 01011 | 0x2C1 - 0x2FF | 705 - 767 |
| **RxPDO7** | 01101 | 0x341 - 0x37F | 833 - 895 |
| **TxPDO8** | 01111 | 0x3C1 - 0x3FF | 961 - 1023 |
| **RxPDO8** | 10001 | 0x441 - 0x47F | 1089 - 1151 |
| **TxPDO9** | 10011 | 0x4C1 - 0x4FF | 1217 - 1279 |
| **RxPDO9** | 10101 | 0x541 - 0x57F | 1345 - 1407 |
| **TxDPO10** | 10111 | 0x5C1 - 0x5FF | 1473 - 1535 |
| **RxPDO10** | 11001 | 0x641 - 0x67F | 1601 - 1663 |
| **TxPDO11** | 11011 | 0x6C1 - 0x6FF | 1729 - 1791 |
| **RxPDO11** | 11101 | 0x741 - 0x77F | 1857 - 1919 |

**NOTE**

**Index 0x5500**

Ensure that index 0x5500 is not used if Bus Couplers with more than 5 PDOs are present in networks with node addresses > 64, otherwise identification overlaps can occur. In that case, the PDO identifiers must be set individually.

For the sake of clarity, the default identifiers defined according to CANopen are also listed here:

| Object | Function Code | Resulting COB ID (hex) | Resulting COB ID (dec) |
|---|---|---|---|
| **Emergency** | 0001 | 0x81 - 0xBF [0xFF] | 129 - 191 [255] |
| **TxPDO1** | 0011 | 0x181 - 0x1BF [0x1FF] | 385 - 447 [511] |
| **RxPDO1** | 0100 | 0x201 - 0x23F [0x27F] | 513 - 575 [639] |
| **TxPDO2** | 0101 | 0x281 - 0x2BF [0x2FF] | 641 - 676 [767] |
| **RxPDO2** | 0110 | 0x301 - 0x33F [0x37F] | 769 - 831 [895] |
| **TxDPO3** | 0111 | 0x381 - 0x3BF [0x3FF] | 897 - 959 [1023] |
| **RxPDO3** | 1000 | 0x401 - 0x43F [0x47F] | 1025 - 1087 [1151] |
| **TxPDO4** | 1001 | 0x481 - 0x4BF [0x4FF] | 1153 - 1215 [1279] |
| **RxPDO4** | 1010 | 0x501 - 0x53F [0x57F] | 1281- 1343 [1407] |
| **SDO (Tx)** | 1011 | 0x581 - 0x5BF [0x5FF] | 1409 - 1471 [1535] |
| **SDO (Rx)** | 1100 | 0x601 - 0x63F [0x67F] | 1537 - 1599 [1663] |
| **Guarding / Heartbeat/ Bootup** | 1110 | 0x701 - 0x73F [0x77F] | 1793 - 1855 [1919] |

The identifiers that result from the DIP switch settings on the coupler are given, as are the identifier regions for the node addresses 64...127 (not settable in Bus Couplers BK5110, BK5120 and LC5100) in square brackets. Addresses 1…99 can be set for the Fieldbus Box modules and the BK515x Bus Couplers.

The appendix contains a tabular summary of all the identifiers.

### Digital inputs

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x6000** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of available digital 8-bit input data blocks |
| | 1 | 1st input block | Unsigned8 | ro | Y | 0x00 | 1st input channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0XFE | 254. input block | Unsigned8 | ro | Y | 0x00 | 254st input channel |

### Interrupt mask

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x6126** | 0 | Number of items | Unsigned8 | ro | N | Depending on type | The number of 32-bit interrupt masks = 2 x the number of TxDPOs |
| | 1 | IR-Mask0 TxPDO1 | Unsigned32 | rw | N | 0xFFFFFFFF | IR mask bytes 0...3 TxPDO1 |
| | 2 | IR-Mask1 TxPDO1 | Unsigned32 | rw | N | 0xFFFFFFFF | IR-mask bytes 4...7 TxPDO1 |
| | 3 | IR-Mask0 TxPDO2 | Unsigned32 | rw | N | 0xFFFFFFFF | IR-mask bytes 0...3 TxPDO2 |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0x20 | IR-Mask1 TxPDO16 | Unsigned32 | rw | N | 0xFFFFFFFF | IR-mask bytes 4...7 TxPDO16 |

By default, every change in the value in an event-driven PDO causes a telegram to be sent. The interrupt mask makes it possible to determine which data changes are evaluated for this purpose. By clearing the appropriate ranges within the PDOs they are masked out for event-driving purposes (interrupt control). The interrupt mask does not just govern all the PDOs with digital inputs, but all the TxPDOs that are present. If the TxPDOs are shorter than 8 bytes, then the superfluous part of the IR mask is not evaluated.

The interrupt mask only has an effect on TxPDOs with transmission types 254 and 255. It is not stored in the device (not even through object 0x1010). Changes to the mask at runtime (when the status is operational) are possible, and are evaluated starting from the next change of input data.

The interrupt mask for TxPDOs with analog input data is not evaluated if either limit values (0x6424, 0x6425) or the delta function (0x6426) have been activated for the inputs.

This entry has been implemented in firmware C3 and above.

### Sample for assignment of data

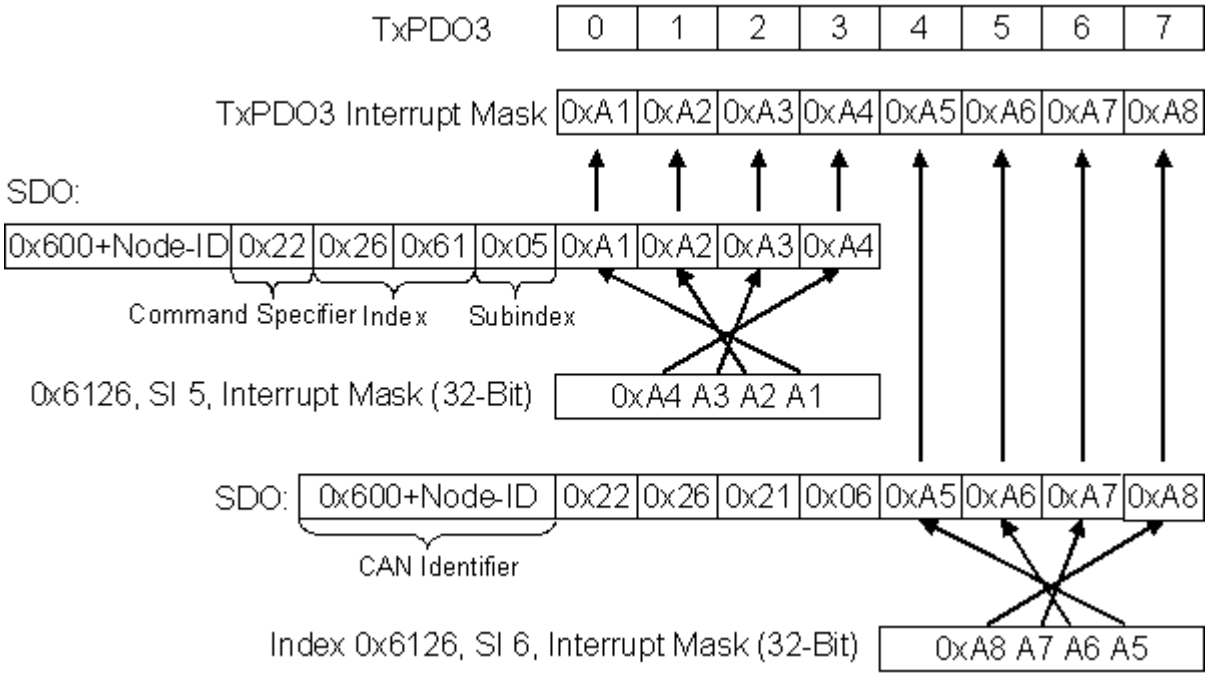



Fig. 87: Example for data assignment

### Application example

The value contained in a fast counter input is only to be transmitted when bits in the status word (the latch input, for instance) have changed. This requires the 32 bit counter value to be masked out (zeroed) in the interrupt mask. The status is located in byte 0, while the counter value is, by default, contained in bytes or 1..4 of the corresponding PDOs (TxPDO3 in this example, because < 65 digital and < 5 analog inputs are present).
This means that index 0x6126, subindex5 must receive the value 0x0000 00FF and that subindex6 must have 0xFFFF FF00 written into it.

The corresponding SDOs therefore appear as follows:

| 11-bit Identifier | 8-byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x600+ Node-ID | 0x22 | 0x26 | 0x61 | 0x05 | 0xFF | 0x00 | 0x00 | 0x00 |

| 11-bit Identifier | 8-byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x600+ Node-ID | 0x22 | 0x26 | 0x61 | 0x06 | 0x00 | 0xFF | 0xFF | 0xFF |

### Digital outputs

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x6200** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of available digital 8-bit output data blocks |
| | 1 | 1st input block | Unsigned8 | rw | Y | 0x00 | 1st output channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0XFE | 254. input block | Unsigned8 | rw | Y | 0x00 | 254st output channel |

### Analog inputs

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x6401** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of analog input channels available |
| | 1 | 1st input | Unsigned16 | ro | Y | 0x0000 | 1st input channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0XFE | 254. input | Unsigned16 | ro | Y | 0x0000 | 254st input channel |

The analog signals are displayed left aligned. The representation in the process image is therefore independent of the actual resolution. Detailed information on the data format can be found at the relevant signal type.

### Analog outputs

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x6411** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of analog output channels available |
| | 1 | 1st input block | Unsigned16 | rw | Y | 0x0000 | 1st output channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0XFE | 254. input block | Unsigned16 | rw | Y | 0x0000 | 254st output channel |

The analog signals are displayed left aligned. The representation in the process image is therefore independent of the actual resolution. Detailed information on the data format can be found at the relevant signal type.

### Event driven analog inputs

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x6423** | 0 | Global Interrupt Enable | Boolean | rw | N | FALSE (0) | Activates the event-driven transmission of PDOs with analog inputs. |

Although, in accordance with CANopen, the analog inputs in TxPDO2..4 are by default set to transmission type 255 (event driven), the event (the alteration of an input value) is suppressed by the event control in object 0x6423, in order to prevent the bus from being swamped with analog signals. It is recommended that the flow of data associated with the analog PDOs is controlled either through synchronous communication or through using the event timer. In event-driven operation, the transmission behavior of the analog PDOs can be parameterized before activation by setting the inhibit time (object 0x1800ff, subindex 3) and/or limit value monitoring (objects 0x6424 + 0x6425) and/or delta function (object 0x6426).

### Upper limit value analog inputs

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x6424** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of analog input channels available |
| | 1 | upper limit 1st input | Unsigned16 | rw | Y | 0x0000 | Upper limit value for 1st input channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0XFE | upper limit 254. input | Unsigned16 | rw | Y | 0x0000 | Upper limit value for 254st input channel |

Values different from 0 activate the upper limit value for this channel. A PDO is then transmitted if this limit value is exceeded. In addition, the event driven mode must be activated (object 0x6423). The data format corresponds to that of the analog inputs.

### Lower limit value analog inputs

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x6425** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of analog input channels available |
| | 1 | lower limit 1st input | Unsigned16 | rw | Y | 0x0000 | Lower limit value for 1st input channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0XFE | lower limit 254. input | Unsigned16 | rw | Y | 0x0000 | Lower limit value for 254st input channel |

Values different from 0 activate the lower limit value for this channel. A PDO is then transmitted if the value falls below this limit value. In addition, the event driven mode must be activated (object 0x6423). The data format corresponds to that of the analog inputs.

### Delta function for analog inputs

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x6426** | 0 | Number of items | Unsigned8 | ro | N | Depending on type and fittings | Number of analog input channels available |
| | 1 | delta value 1st input | Unsigned16 | rw | Y | 0x0000 | Delta value for the 1st input channel |
| | ... | ... | ... | ... | ... | ... | ... |
| | 0XFE | delta value 254. input | Unsigned16 | rw | Y | 0x0000 | Delta value for the 254st input channel |

Values different from 0 activate the delta function for this channel. A PDO is then transmitted if the value has changed by more than the delta value since the last transmission. In addition, the event driven mode must be activated (object 0x6423). The data format corresponds to that of the analog inputs (delta value: can only have positive values).

### 7.3.4 Objects and data of BX5100/BC5150

**Access to allocated flags**

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x2F00** | 0 | Number of the subindex | Unsigned8 | ro | N | 128 | Number of the subindex |
| | 1 | Flags | Unsigned32 | rw | N | none | Flag %MB0..3 |
| | 2 | Flags | Unsigned32 | rw | N | none | Flag %MB4..7 |
| | ... | ... | ... | ... | ... | ... | ... |
| | 128 | Flags | Unsigned32 | rw | N | None | Flag %MB508..511 |

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x2F01-0x2F07** | 0 | Number of the subindex | Unsigned8 | ro | N | 128 | Number of the subindex |
| | 1 | Flags | Unsigned32 | rw | N | none | Flag %MBx..x+4 |
| | 2 | Flags | Unsigned32 | rw | N | none | Flag %MBy..y+4 |
| | ... | ... | ... | ... | ... | ... | ... |
| | 128 | Flags | Unsigned32 | rw | N | None | Flag %MBz...z+14 |

**AMS NetID**

| Index | Subindex | Name | Type | At-tribute | Map-ping | Default value | Meaning |
|---|---|---|---|---|---|---|---|
| **0x5FFE** | 0 | NetId | String | rw | N | 1.1.1.1.1.1 | AMS NetID |

## 7.4 ADS-Communication

### 7.4.1 ADS services

**Local process image task 1 port 800 or 801**

Data can be read from and written to the local process image. If it is necessary for outputs to be written, it is important to ensure that they are not used by the local PLC, because the local controller will overwrite these values. The data are not associated with a watchdog, and therefore must not be used for outputs that would have to be switched off in the event of a fault.

| Index Group | Meaning | Index offset (value range) |
|---|---|---|
| 0xF020 | Inputs | 0...2047 |
| 0xF030 | Outputs | 0...2047 |
| 0x4020 | Flags | 0...4095 |

**ADS services**

**AdsServerAdsState**

| Data type (read only) | Meaning |
|---|---|
| String | Start - the local PLC is running<br>Start - the local PLC is stopped |

### AdsServerDeviceState

| Data type (read only) | Meaning |
|---|---|
| INT | 0: Start - the local PLC is running<br>1: Start - the local PLC is stopped |

### AdsServerType

| Data type (read only) | Meaning |
|---|---|
| String | BX PLC Server |

### Register port 100

On the Bus Terminal Controllers of the BX series, and on the BCxx50, the ADS port number for register communication is fixed at 100.

| Index Group | Index offset (value range) | | Meaning |
|---|---|---|---|
| | Hi-Word | Low Word | |
| 0 [READ ONLY] | 0...127 | 0...255 | Registers in the Bus Coupler<br>Hi-Word, table number of the Bus Coupler<br>Lo-Word, register number of the table |
| 1...255 | 0...3 | 1-255 | Register of the Bus Terminals<br>Hi-Word, channel number<br>Lo-Word, register number of the Bus Terminal |

**Minimum timeout**

When reading the register, the time out for the ADS block has to be set to a time longer than 1 second.

**Setting the password**

When writing to the registers, the password has to be set (see the documentation for the particular Bus Terminal).

# 8 Error handling and diagnosis

## 8.1 Diagnostics

### CANopen state

In many cases it is important to know whether the communication with the higher-level master is still OK. To this end, link the *NodeState* variable with your PLC program. A TwinCAT configuration is required for this purpose.

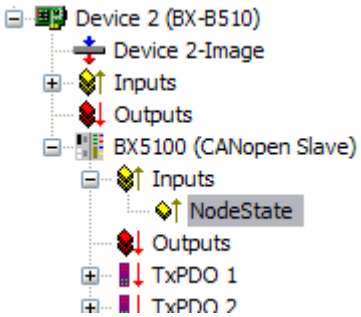



Fig. 88: CANopen diagnostic byte in the System Manager

| Error number | Description | Remedy |
|---|---|---|
| 0 | No error | - |
| 2 | Guarding error | Check the connection |
| 20 | Too few PDOs (only TwinCAT configuration) | Check the configuration |
| 22 | Sync error | Check the connection |
| 129 | Node is pre-operational | Switch the node to operational |
| 130 | Node is stopped | Start the node |

### Example

If the CANopen is interrupted, e.g. if the cable is pulled or the PLC is switched off, the Bus Terminal Controller indicates this by reporting ??? in the node state.

### Reading fieldbus state by ADS

You can read the fieldbus state via ADSREAD in the default configuration or in the TwinCAT configuration.

| Parameter ADSREAD function block | Description |
|---|---|
| NetID | local – empty string |
| Port | 1 |
| IndexGroup | 16#0006 |
| IndexOffset | 16#000C_AE00 |
| LEN | 1 |

### State of the K-bus

An internal K-bus or Bus Terminal error is indicated in the K-bus state. A more precise fault description can be obtained via a function block (in preparation). To this end, link the *K-bus state* variable with your PLC program.

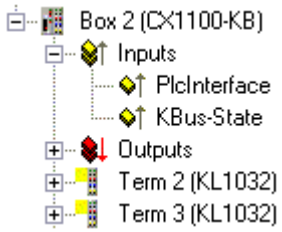



Fig. 89: State of the K-bus

| Error bit | Description | Error type |
|---|---|---|
| 0 | No error | No error |
| Bit 0 | K-bus error | Error |
| Bit 2 | K-bus is re-triggered | Note |

### Reading K-bus state by ADS

You can read the fieldbus state via ADSREAD in the default configuration or in the TwinCAT configuration.

| Parameter ADSREAD function block | Description |
|---|---|
| NetID | local – empty string |
| Port | 1 |
| IndexGroup | 16#0006 |
| IndexOffset | 16#000C_9000 |
| LEN | 1 |

## 8.2 Diagnostic LEDs

The Bus Terminal Controllers have status LEDs. The row of LEDs on the left describes the status of the fieldbus and of the PLC. The row of LEDs on the right indicates the supply voltage and the K-Bus state.

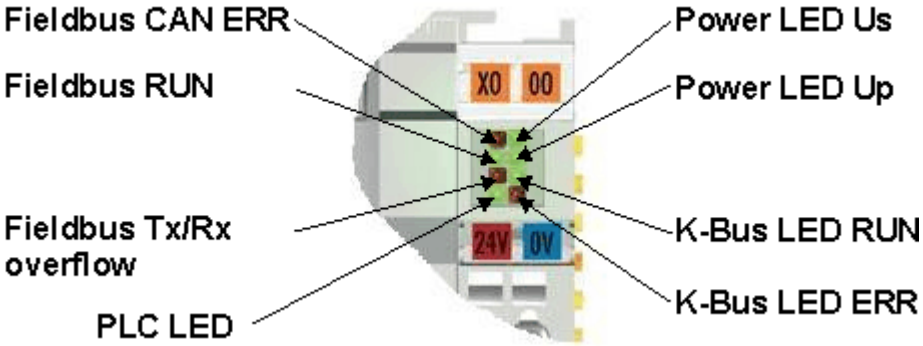



Fig. 90: LEDs

### LEDs for power supply diagnostics

| LED (Power LEDs) | Meaning |
|---|---|
| LED Us | LED off: Bus Terminal Controller does not have 24 $V_{DC}$ voltage |
| LED Up | LED off: No power supply 24 $V_{DC}$ connected at the power contacts |

### LEDs for K-Bus diagnostics

| LED (Power LEDs) | Meaning |
|---|---|
| LED RUN | LED off: No K-Bus Update<br>LED on, flashing: K-bus running |
| LED ERR | LED off: No error<br>LED flashing: see K-Bus error code |

## K-Bus error code diagnostics

| Error code | Error argument | Description | Remedy |
|---|---|---|---|
| 0 | - | EMC problems | • Check power supply for undervoltage or overvoltage peaks<br>• Implement EMC measures<br>• If a K-bus error is present, it can be localized by a restart of the coupler (by switching it off and then on again) |
| 1 | 0 | EEPROM checksum error | Enter factory settings with the KS2000 configuration software |
| | 1 | Code buffer overflow | Insert fewer Bus Terminals. The programmed configuration has too many entries in the table |
| | 2 | Unknown data type | Software update required for the Bus Terminal Controller |
| 2 | - | Reserve | - |
| 3 | 0 | K-bus command error | • No Bus Terminal inserted<br>• One of the Bus Terminals is defective; halve the number of Bus Terminals attached and check whether the error is still present with the remaining Bus Terminals. Repeat until the defective Bus Terminal is located. |
| 4 | 0 | K-bus data error, break behind the Bus Terminal Controller | Check whether the n+1 Bus Terminal is correctly connected; replace if necessary. |
| | n | Break behind Bus Terminal n | Check whether the KL9010 Bus End Terminal is connected |
| 5 | n | K-bus error in register communication with Bus Terminal n | Exchange the nth Bus Terminal |
| 6 | 0 | Error at initialization | Exchange Bus Terminal Controller |
| | 1 | Internal data error | Perform a hardware reset on the Bus Terminal Controller (switch off and on again) |
| | 2 | DIP switch changed after a software reset | Perform a hardware reset on the Bus Terminal Controller (switch off and on again) |
| 7 | 0 | Note: cycle time was exceeded | Warning: the set cycle time was exceeded. This indication (flashing LEDs) can only be cleared by booting the Bus Terminal Controller again.<br>Remedy: increase the cycle time |
| 9 | 0 | Checksum error in Flash program | Transmit program to the BX again |
| | 1 | Incorrect or faulty library implemented | Remove the faulty library |
| 10 | n | Bus Terminal n is not consistent with the configuration that existed when the boot project was created | Check the nth Bus Terminal. The boot project must be deleted if the insertion of an nth Bus Terminal is intentional. |
| 14 | n | nth Bus Terminal has the wrong format | Start the Bus Terminal Controller again, and if the error occurs again then exchange the Bus Terminal. |
| 15 | n | Number of Bus Terminals is no longer correct | Start the Bus Terminal Controller again. If the error occurs again, restore the manufacturers setting using the KS2000 configuration software |
| 16 | n | Length of the K-bus data is no longer correct | Start the Bus Terminal Controller again. If the error occurs again, restore the manufacturers setting using the KS2000 configuration software |

## LED bus - fieldbus diagnostics

| LED RUN | LED BF CAN ERR | LED DIA CAN RxTx Overflow | Meaning |
|---|---|---|---|
| off | off | off | no fieldbus connected, Bus Terminal Controller searches for baud rate |
| on | on | off | Stopped, pre-operational, no error |
| on | off | off | Operational, no error |
| off | flashes | off | Bus-Off |
| off | on | on | Warning Level |
| on | on | Flashes quickly, then 1 x slowly | Guarding error |
| on | on | Flashes quickly, then 2 x slowly | Sync error |
| on | on | Flashes quickly, then 3 x slowly | Event-Timeout |

### LED PLC - PLC diagnostics

| LED | Meaning |
|---|---|
| PLC LED | LED on: PLC is running<br>LED off: PLC stopped |

# 9 Appendix

## 9.1 Quick Start for Experienced Users

### Target group

This brief introduction is intended for users who are already familiar with CAN. It clarifies the CAN messages that are required in order to work with Beckhoff CANopen input/output groups in the initial configuration (default settings).

It remains necessary to read and use the full documentation.

### Hardware configuration

The DIP switches must be used to set a consistent data transfer rate and differing node addresses (node ID) on the Bus Couplers. The switch assignments are printed on the modules. It should be noted that CANopen uses address "0" to address all modules (broadcast), so that this cannot be set as the address of a particular module.

### Starting the modules

CANopen allows the modules to be started with a single network management [▶ 77] telegram:

| 11 bit identifier | 2-byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x00 | 0x01 | 0x00 | | | | | | |

The first data byte here contains the start command (Start_Remote_Node), while the second data byte contains the node address (here: 0, which addresses all nodes).

The inputs and outputs are enabled after the modules have been started. In the default setting the modules communicate in event-driven mode, so that changes at the digital inputs are immediately transmitted and outputs are immediately set in accordance with received telegrams containing output data.

### CAN identifier

The CAN identifiers for the input and output data are derived from the node address (1-63):

| Data type | Default CAN identifier |
|---|---|
| digital inputs 1...64 | 0x180 (=$384_{dec}$) + node address |
| digital outputs 1...64 | 0x200 (=$512_{dec}$) + node address |
| analog inputs 1...4 | 0x280 (=$640_{dec}$) + node address |
| analog outputs 1...4 | 0x300 (=$768_{dec}$) + node address |
| analog inputs 5...8* | 0x380 (=$896_{dec}$) + node address |
| analog outputs 5...8* | 0x400 (=$1024_{dec}$) + node address |
| analog inputs 9...12* | 0x480 (=$1152_{dec}$) + node address |
| analog outputs 9...12* | 0x500 (=$1280_{dec}$) + node address |

* If more than 64 digital inputs or outputs are present, the range is offset accordingly (see section Default mapping).

### Digital inputs

The CAN messages with digital input data are composed as follows:

| 11 bit identifier | 1-8 bytes of user data (depending on the number of input terminals or extension modules) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x180(=$384_{dec}$) + node ID | I0 | I1 | I2 | I3 | I4 | I5 | I6 | I7 |

E0: Input bytes on input terminals (or Fieldbus Box modules), from left to right.

### Digital outputs

The CAN messages with digital output data have the following structure:

| 11 bit identifier | 1-8 bytes of user data (depending on the number of output terminals or extension modules) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x200(=$512_{dec}$) + node ID | O0 | O1 | O2 | O3 | O4 | O5 | O6 | O7 |

A0: Output bytes on output terminals (or Fieldbus Box modules), from left to right.

### Analog inputs

CAN messages with analog input data look like this:

| 11 bit identifier | 4-8 bytes of user data (depending on the number of analog inputs) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x280(=$640_{dec}$) + node ID | I0.0 | I0.1 | I1.0 | I1.1 | I2.0 | I2.1 | I3.0 | I3.1 |

E x.0...E x.1: analog input x. A more detailed description of the data format can be found in the object directory under object 0x6401.

### The transmission behavior of analog inputs

To avoid "swamping" the bus with constantly changing analog input values, the analog CANopen input modules do not generate any data telegrams in the default state. The analog data can be read out by means of a remote access (Remote Transmit Request, a CAN message with no data and with the RTR bit set) to the analog input telegrams. Alternatively, of course, the module can be re-configured in such a way that an alteration of the input value does trigger the sending of a telegram. For this purpose a value > 0 is written into index 0x6423 of the object directory. The corresponding SDO telegram looks like this:

| 11 bit identifier | 8-byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x600(=$768_{dec}$) + node ID | 0x22 | 0x23 | 0x64 | 0x00 | 0x01 | 0x00 | 0x00 | 0x00 |

It is recommended that event control (where every change in the LSB is considered an event, resulting in the corresponding telegram being transmitted) is not used for the transmission of input data, but that either cyclic, synchronous transmission or the event timer is used to send the data. If event control is indeed used, then the quantity of data should be reduced by setting a delta value (object directory index 0x6426), limit values (0x6424+0x6425) or an inhibit time (no new data transmission until the inhibit time has elapsed, 0x1801ff). Details of parameter communication are found in the section on Service data: SDO [▶ 91].

### Analog outputs

CAN messages with analog output data look like this:

| 11 bit identifier | 4-8 bytes of user data (depending on the number of analog outputs) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x300(=$768_{dec}$) + node ID | O0.0 | O0.1 | O1.0 | O1.1 | O2.0 | O2.1 | O3.0 | O3.1 |

A x.0...A x.1: analog output x. A more detailed description of the data format can be found in the object directory under object 0x6411.

### Default Identifier

The appendix contains a tabular summary of all the default identifiers. The CAN messages displayed on a CAN monitor can quickly and easily be identified with the help of that overview.

### Stopping the modules

If necessary, the process data communication from the modules can be stopped with the following telegram:

| 11 bit identifier | 2-byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x00 | 0x80 | 0xYZ | | | | | | |

0xXX: node address; 0xYZ=0x00 addresses all modules

**Guarding**

The telegrams described above are themselves adequate for many applications. Since, however, the modules operate in event-driven mode by default (no cyclical data exchange), the failure of a module is not necessarily detected. A remedy for this is provided here through monitoring the modules by cyclically polling their status, a process known as node guarding.

For this purpose a status telegram is requested cyclically by means of remote transmit request (RTR):

| 11 bit identifier | No user data in the request telegram (RTR) |
|---|---|
| 0x700(=$1792_{dec}$) + node ID | (RTR bit set in the header) |

The modules answer with a telegram that includes a status byte.

| 11 bit identifier | 1-byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x700(=$1792_{dec}$) + node ID | 0xYZ | | | | | | | |

0xYZ: Status byte:
bits 6...0 contain the node status (0x7F=127: pre-operational, 0x05=operational; 0x04= stopped or prepared).
Bit 7 = toggle bit (inverts with every transmission).

So that the Bus Coupler can detect failure of the network master (watchdog function), the guard time (object 0x100C) and the life time factor (object 0x100D) must be set to have value different from 0. (response time in the event of a fault: Guard Time X Life Time Factor).

**Heartbeat**

As an alternative to guarding, the module can also be monitored by means of what is called the heartbeat. This involves a status telegram (the heartbeat) being issued cyclically by the node. Data request telegrams (remote frames) are not required.

In order to activate the heartbeat telegram, the producer heartbeat time must be set. This is done with the following SDO [▶ 91] telegram:

| 11 bit identifier | 8-byte user data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x600(=$768_{dec}$) + node ID | 0x22 | 0x17 | 0x10 | 0x00 | 0xcd | 0xab | 0x00 | 0x00 |

where 0xabcd is the desired heartbeat cycle time, expressed in milliseconds.

With the telegrams that have now been described you are in a position to start and stop the modules, read inputs, write outputs and to monitor the modules. Do not neglect to read the manual with attention. Only by doing so can you properly use the many features of the BECKHOFF CANopen Bus Coupler.

## 9.2 CAN Identifier List

The list provided here should assist in identifying and assigning CANopen messages. All the identifiers allocated by the CANopen default identifier allocation are listed, as well as the manufacturer-specific default identifiers issued by BECKHOFF via object 0x5500 (only to be used in networks with node addresses less than 64).

The following values can be used as search aids and "entry points" in the extensive identifier table in the *chm edition of the documentation:

Decimal: 400 [▶ 142], 500 [▶ 149], 600 [▶ 149], 700 [▶ 144], 800 [▶ 145], 900 [▶ 146], 1000 [▶ 151], 1100 [▶ 151], 1200 [▶ 147], 1300 [▶ 147], 1400 [▶ 152], 1500 [▶ 153], 1600 [▶ 153], 1700 [▶ 148], 1800 [▶ 156], 1900 [▶ 148]

Hexadecimal: 0x181 [▶ 142], 0x1C1 [▶ 149], 0x201 [▶ 143], 0x301 [▶ 145], 0x401 [▶ 146], 0x501 [▶ 147], 0x601 [▶ 155], 0x701 [▶ 156]

The identifier distribution via object 0x5500 follows this pattern:

| Object | Resulting COB ID (dec) | Resulting COB ID (hex) |
|---|---|---|
| Emergency [▶ 142] | 129 to 191 [255] | 0x81 to 0xBF [0xFF] |
| TxPDO1 [▶ 142] | 385 to 447 [511] | 0x181 to 0x1BF [0x1FF] |
| RxPDO1 [▶ 143] | 513 to 575 [639] | 0x201 to 0x23F [0x27F] |
| TxPDO2 [▶ 144] | 641 to 676 [767] | 0x281 to 0x2BF [0x2FF] |
| RxPDO2 [▶ 145] | 769 to 831 [895] | 0x301 to 0x33F [0x37F] |
| TxDPO3 [▶ 146] | 897 to 959 [1023] | 0x381 to 0x3BF [0x3FF] |
| RxPDO3 [▶ 146] | 1025 to 1087 [1151] | 0x401 to 0x43F [0x47F] |
| TxPDO4 [▶ 147] | 1153 to 1215 [1279] | 0x481 to 0x4BF [0x4FF] |
| RxPDO4 [▶ 147] | 1281 to 1343 [1407] | 0x501 to 0x53F [0x57F] |
| TxPDO5 [▶ 148] | 1665 to 1727 | 0x681 to 0x6BF |
| RxPDO5 [▶ 148] | 1921 to 1983 | 0x781 to 0x7BF |
| TxPDO6 [▶ 149] | 449 to 511 | 0x1C1 to 0x1FF |
| RxPDO6 [▶ 149] | 577 to 639 | 0x241 to 0x27F |
| TxDPO7 [▶ 150] | 705 to 767 | 0x2C1 to 0x2FF |
| RxPDO7 [▶ 150] | 833 to 895 | 0x341 to 0x37F |
| TxPDO8 [▶ 151] | 961 to 1023 | 0x3C1 to 0x3FF |
| RxPDO8 [▶ 151] | 1089 to 1151 | 0x441 to 0x47F |
| TxPDO9 [▶ 152] | 1217 to 1279 | 0x4C1 to 0x4FF |
| RxPDO9 [▶ 152] | 1345 to 1407 | 0x541 to 0x57F |
| TxDPO10 [▶ 153] | 1473 to 1535 | 0x5C1 to 0x5FF |
| RxPDO10 [▶ 153] | 1601 to 1663 | 0x641 to 0x67F |
| TxPDO11 [▶ 154] | 1729 to 1791 | 0x6C1 to 0x6FF |
| RxPDO11 [▶ 154] | 1857 to 1919 | 0x741 to 0x77F |
| SDO (Tx) [▶ 155] | 1409 to 1471 [1535] | 0x581 to 0x5BF [0x5FF] |
| SDO (Rx) [▶ 155] | 1537 to 1599 [1663] | 0x601 to 0x63F [0x67F] |
| Guarding / Heartbeat/ Bootup [▶ 156] | 1793 to 1855 [1919] | 0x701 to 0x73F [0x77F] |

**Identifier List**

Identifiers marked with * are given manufacturer-specific assignments on the Bus Couplers after writing index 0x5500

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 0 | 0x00 | NMT | 149 | 0x95 | EMCY Nd.21 | 171 | 0xAB | EMCY Nd.43 |
| 128 | 0x80 | SYNC | 150 | 0x96 | EMCY Nd.22 | 172 | 0xAC | EMCY Nd.44 |
| 129 | 0x81 | EMCY Nd.1 | 151 | 0x97 | EMCY Nd.23 | 173 | 0xAD | EMCY Nd.45 |
| 130 | 0x82 | EMCY Nd.2 | 152 | 0x98 | EMCY Nd.24 | 174 | 0xAE | EMCY Nd.46 |
| 131 | 0x83 | EMCY Nd.3 | 153 | 0x99 | EMCY Nd.25 | 175 | 0xAF | EMCY Nd.47 |
| 132 | 0x84 | EMCY Nd.4 | 154 | 0x9A | EMCY Nd.26 | 176 | 0xB0 | EMCY Nd.48 |
| 133 | 0x85 | EMCY Nd.5 | 155 | 0x9B | EMCY Nd.27 | 177 | 0xB1 | EMCY Nd.49 |
| 134 | 0x86 | EMCY Nd.6 | 156 | 0x9C | EMCY Nd.28 | 178 | 0xB2 | EMCY Nd.50 |
| 135 | 0x87 | EMCY Nd.7 | 157 | 0x9D | EMCY Nd.29 | 179 | 0xB3 | EMCY Nd.51 |
| 136 | 0x88 | EMCY Nd.8 | 158 | 0x9E | EMCY Nd.30 | 180 | 0xB4 | EMCY Nd.52 |
| 137 | 0x89 | EMCY Nd.9 | 159 | 0x9F | EMCY Nd.31 | 181 | 0xB5 | EMCY Nd.53 |
| 138 | 0x8A | EMCY Nd.10 | 160 | 0xA0 | EMCY Nd.32 | 182 | 0xB6 | EMCY Nd.54 |
| 139 | 0x8B | EMCY Nd.11 | 161 | 0xA1 | EMCY Nd.33 | 183 | 0xB7 | EMCY Nd.55 |
| 140 | 0x8C | EMCY Nd.12 | 162 | 0xA2 | EMCY Nd.34 | 184 | 0xB8 | EMCY Nd.56 |
| 141 | 0x8D | EMCY Nd.13 | 163 | 0xA3 | EMCY Nd.35 | 185 | 0xB9 | EMCY Nd.57 |
| 142 | 0x8E | EMCY Nd.14 | 164 | 0xA4 | EMCY Nd.36 | 186 | 0xBA | EMCY Nd.58 |
| 143 | 0x8F | EMCY Nd.15 | 165 | 0xA5 | EMCY Nd.37 | 187 | 0xBB | EMCY Nd.59 |
| 144 | 0x90 | EMCY Nd.16 | 166 | 0xA6 | EMCY Nd.38 | 188 | 0xBC | EMCY Nd.60 |
| 145 | 0x91 | EMCY Nd.17 | 167 | 0xA7 | EMCY Nd.39 | 189 | 0xBD | EMCY Nd.61 |
| 146 | 0x92 | EMCY Nd.18 | 168 | 0xA8 | EMCY Nd.40 | 190 | 0xBE | EMCY Nd.62 |
| 147 | 0x93 | EMCY Nd.19 | 169 | 0xA9 | EMCY Nd.41 | 191 | 0xBF | EMCY Nd.63 |
| 148 | 0x94 | EMCY Nd.20 | 170 | 0xAA | EMCY Nd.42 | | | |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 385 | 0x181 | TxPDO1, DI, Nd.1 | 406 | 0x196 | TxPDO1, DI, Nd.22 | 427 | 0x1AB | TxPDO1, DI, Nd.43 |
| 386 | 0x182 | TxPDO1, DI, Nd.2 | 407 | 0x197 | TxPDO1, DI, Nd.23 | 428 | 0x1AC | TxPDO1, DI, Nd.44 |
| 387 | 0x183 | TxPDO1, DI, Nd.3 | 408 | 0x198 | TxPDO1, DI, Nd.24 | 429 | 0x1AD | TxPDO1, DI, Nd.45 |
| 388 | 0x184 | TxPDO1, DI, Nd.4 | 409 | 0x199 | TxPDO1, DI, Nd.25 | 430 | 0x1AE | TxPDO1, DI, Nd.46 |
| 389 | 0x185 | TxPDO1, DI, Nd.5 | 410 | 0x19A | TxPDO1, DI, Nd.26 | 431 | 0x1AF | TxPDO1, DI, Nd.47 |
| 390 | 0x186 | TxPDO1, DI, Nd.6 | 411 | 0x19B | TxPDO1, DI, Nd.27 | 432 | 0x1B0 | TxPDO1, DI, Nd.48 |
| 391 | 0x187 | TxPDO1, DI, Nd.7 | 412 | 0x19C | TxPDO1, DI, Nd.28 | 433 | 0x1B1 | TxPDO1, DI, Nd.49 |
| 392 | 0x188 | TxPDO1, DI, Nd.8 | 413 | 0x19D | TxPDO1, DI, Nd.29 | 434 | 0x1B2 | TxPDO1, DI, Nd.50 |
| 393 | 0x189 | TxPDO1, DI, Nd.9 | 414 | 0x19E | TxPDO1, DI, Nd.30 | 435 | 0x1B3 | TxPDO1, DI, Nd.51 |
| 394 | 0x18A | TxPDO1, DI, Nd.10 | 415 | 0x19F | TxPDO1, DI, Nd.31 | 436 | 0x1B4 | TxPDO1, DI, Nd.52 |
| 395 | 0x18B | TxPDO1, DI, Nd.11 | 416 | 0x1A0 | TxPDO1, DI, Nd.32 | 437 | 0x1B5 | TxPDO1, DI, Nd.53 |
| 396 | 0x18C | TxPDO1, DI, Nd.12 | 417 | 0x1A1 | TxPDO1, DI, Nd.33 | 438 | 0x1B6 | TxPDO1, DI, Nd.54 |
| 397 | 0x18D | TxPDO1, DI, Nd.13 | 418 | 0x1A2 | TxPDO1, DI, Nd.34 | 439 | 0x1B7 | TxPDO1, DI, Nd.55 |
| 398 | 0x18E | TxPDO1, DI, Nd.14 | 419 | 0x1A3 | TxPDO1, DI, Nd.35 | 440 | 0x1B8 | TxPDO1, DI, Nd.56 |
| 399 | 0x18F | TxPDO1, DI, Nd.15 | 420 | 0x1A4 | TxPDO1, DI, Nd.36 | 441 | 0x1B9 | TxPDO1, DI, Nd.57 |
| 400 | 0x190 | TxPDO1, DI, Nd.16 | 421 | 0x1A5 | TxPDO1, DI, Nd.37 | 442 | 0x1BA | TxPDO1, DI, Nd.58 |
| 401 | 0x191 | TxPDO1, DI, Nd.17 | 422 | 0x1A6 | TxPDO1, DI, Nd.38 | 443 | 0x1BB | TxPDO1, DI, Nd.59 |
| 402 | 0x192 | TxPDO1, DI, Nd.18 | 423 | 0x1A7 | TxPDO1, DI, Nd.39 | 444 | 0x1BC | TxPDO1, DI, Nd.60 |
| 403 | 0x193 | TxPDO1, DI, Nd.19 | 424 | 0x1A8 | TxPDO1, DI, Nd.40 | 445 | 0x1BD | TxPDO1, DI, Nd.61 |
| 404 | 0x194 | TxPDO1, DI, Nd.20 | 425 | 0x1A9 | TxPDO1, DI, Nd.41 | 446 | 0x1BE | TxPDO1, DI, Nd.62 |
| 405 | 0x195 | TxPDO1, DI, Nd.21 | 426 | 0x1AA | TxPDO1, DI, Nd.42 | 447 | 0x1BF | TxPDO1, DI, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 513 | 0x201 | RxPDO1, DO, Nd.1 | 534 | 0x216 | RxPDO1, DO, Nd.22 | 555 | 0x22B | RxPDO1, DO, Nd.43 |
| 514 | 0x202 | RxPDO1, DO, Nd.2 | 535 | 0x217 | RxPDO1, DO, Nd.23 | 556 | 0x22C | RxPDO1, DO, Nd.44 |
| 515 | 0x203 | RxPDO1, DO, Nd.3 | 536 | 0x218 | RxPDO1, DO, Nd.24 | 557 | 0x22D | RxPDO1, DO, Nd.45 |
| 516 | 0x204 | RxPDO1, DO, Nd.4 | 537 | 0x219 | RxPDO1, DO, Nd.25 | 558 | 0x22E | RxPDO1, DO, Nd.46 |
| 517 | 0x205 | RxPDO1, DO, Nd.5 | 538 | 0x21A | RxPDO1, DO, Nd.26 | 559 | 0x22F | RxPDO1, DO, Nd.47 |
| 518 | 0x206 | RxPDO1, DO, Nd.6 | 539 | 0x21B | RxPDO1, DO, Nd.27 | 560 | 0x230 | RxPDO1, DO, Nd.48 |
| 519 | 0x207 | RxPDO1, DO, Nd.7 | 540 | 0x21C | RxPDO1, DO, Nd.28 | 561 | 0x231 | RxPDO1, DO, Nd.49 |
| 520 | 0x208 | RxPDO1, DO, Nd.8 | 541 | 0x21D | RxPDO1, DO, Nd.29 | 562 | 0x232 | RxPDO1, DO, Nd.50 |
| 521 | 0x209 | RxPDO1, DO, Nd.9 | 542 | 0x21E | RxPDO1, DO, Nd.30 | 563 | 0x233 | RxPDO1, DO, Nd.51 |
| 522 | 0x20A | RxPDO1, DO, Nd.10 | 543 | 0x21F | RxPDO1, DO, Nd.31 | 564 | 0x234 | RxPDO1, DO, Nd.52 |
| 523 | 0x20B | RxPDO1, DO, Nd.11 | 544 | 0x220 | RxPDO1, DO, Nd.32 | 565 | 0x235 | RxPDO1, DO, Nd.53 |
| 524 | 0x20C | RxPDO1, DO, Nd.12 | 545 | 0x221 | RxPDO1, DO, Nd.33 | 566 | 0x236 | RxPDO1, DO, Nd.54 |
| 525 | 0x20D | RxPDO1, DO, Nd.13 | 546 | 0x222 | RxPDO1, DO, Nd.34 | 567 | 0x237 | RxPDO1, DO, Nd.55 |
| 526 | 0x20E | RxPDO1, DO, Nd.14 | 547 | 0x223 | RxPDO1, DO, Nd.35 | 568 | 0x238 | RxPDO1, DO, Nd.56 |
| 527 | 0x20F | RxPDO1, DO, Nd.15 | 548 | 0x224 | RxPDO1, DO, Nd.36 | 569 | 0x239 | RxPDO1, DO, Nd.57 |
| 528 | 0x210 | RxPDO1, DO, Nd.16 | 549 | 0x225 | RxPDO1, DO, Nd.37 | 570 | 0x23A | RxPDO1, DO, Nd.58 |
| 529 | 0x211 | RxPDO1, DO, Nd.17 | 550 | 0x226 | RxPDO1, DO, Nd.38 | 571 | 0x23B | RxPDO1, DO, Nd.59 |
| 530 | 0x212 | RxPDO1, DO, Nd.18 | 551 | 0x227 | RxPDO1, DO, Nd.39 | 572 | 0x23C | RxPDO1, DO, Nd.60 |
| 531 | 0x213 | RxPDO1, DO, Nd.19 | 552 | 0x228 | RxPDO1, DO, Nd.40 | 573 | 0x23D | RxPDO1, DO, Nd.61 |
| 532 | 0x214 | RxPDO1, DO, Nd.20 | 553 | 0x229 | RxPDO1, DO, Nd.41 | 574 | 0x23E | RxPDO1, DO, Nd.62 |
| 533 | 0x215 | RxPDO1, DO, Nd.21 | 554 | 0x22A | RxPDO1, DO, Nd.42 | 575 | 0x23F | RxPDO1, DO, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 641 | 0x281 | TxPDO2, AI, Nd.1 | 662 | 0x296 | TxPDO2, AI, Nd.22 | 683 | 0x2AB | TxPDO2, AI, Nd.43 |
| 642 | 0x282 | TxPDO2, AI, Nd.2 | 663 | 0x297 | TxPDO2, AI, Nd.23 | 684 | 0x2AC | TxPDO2, AI, Nd.44 |
| 643 | 0x283 | TxPDO2, AI, Nd.3 | 664 | 0x298 | TxPDO2, AI, Nd.24 | 685 | 0x2AD | TxPDO2, AI, Nd.45 |
| 644 | 0x284 | TxPDO2, AI, Nd.4 | 665 | 0x299 | TxPDO2, AI, Nd.25 | 686 | 0x2AE | TxPDO2, AI, Nd.46 |
| 645 | 0x285 | TxPDO2, AI, Nd.5 | 666 | 0x29A | TxPDO2, AI, Nd.26 | 687 | 0x2AF | TxPDO2, AI, Nd.47 |
| 646 | 0x286 | TxPDO2, AI, Nd.6 | 667 | 0x29B | TxPDO2, AI, Nd.27 | 688 | 0x2B0 | TxPDO2, AI, Nd.48 |
| 647 | 0x287 | TxPDO2, AI, Nd.7 | 668 | 0x29C | TxPDO2, AI, Nd.28 | 689 | 0x2B1 | TxPDO2, AI, Nd.49 |
| 648 | 0x288 | TxPDO2, AI, Nd.8 | 669 | 0x29D | TxPDO2, AI, Nd.29 | 690 | 0x2B2 | TxPDO2, AI, Nd.50 |
| 649 | 0x289 | TxPDO2, AI, Nd.9 | 670 | 0x29E | TxPDO2, AI, Nd.30 | 691 | 0x2B3 | TxPDO2, AI, Nd.51 |
| 650 | 0x28A | TxPDO2, AI, Nd.10 | 671 | 0x29F | TxPDO2, AI, Nd.31 | 692 | 0x2B4 | TxPDO2, AI, Nd.52 |
| 651 | 0x28B | TxPDO2, AI, Nd.11 | 672 | 0x2A0 | TxPDO2, AI, Nd.32 | 693 | 0x2B5 | TxPDO2, AI, Nd.53 |
| 652 | 0x28C | TxPDO2, AI, Nd.12 | 673 | 0x2A1 | TxPDO2, AI, Nd.33 | 694 | 0x2B6 | TxPDO2, AI, Nd.54 |
| 653 | 0x28D | TxPDO2, AI, Nd.13 | 674 | 0x2A2 | TxPDO2, AI, Nd.34 | 695 | 0x2B7 | TxPDO2, AI, Nd.55 |
| 654 | 0x28E | TxPDO2, AI, Nd.14 | 675 | 0x2A3 | TxPDO2, AI, Nd.35 | 696 | 0x2B8 | TxPDO2, AI, Nd.56 |
| 655 | 0x28F | TxPDO2, AI, Nd.15 | 676 | 0x2A4 | TxPDO2, AI, Nd.36 | 697 | 0x2B9 | TxPDO2, AI, Nd.57 |
| 656 | 0x290 | TxPDO2, AI, Nd.16 | 677 | 0x2A5 | TxPDO2, AI, Nd.37 | 698 | 0x2BA | TxPDO2, AI, Nd.58 |
| 657 | 0x291 | TxPDO2, AI, Nd.17 | 678 | 0x2A6 | TxPDO2, AI, Nd.38 | 699 | 0x2BB | TxPDO2, AI, Nd.59 |
| 658 | 0x292 | TxPDO2, AI, Nd.18 | 679 | 0x2A7 | TxPDO2, AI, Nd.39 | 700 | 0x2BC | TxPDO2, AI, Nd.60 |
| 659 | 0x293 | TxPDO2, AI, Nd.19 | 680 | 0x2A8 | TxPDO2, AI, Nd.40 | 701 | 0x2BD | TxPDO2, AI, Nd.61 |
| 660 | 0x294 | TxPDO2, AI, Nd.20 | 681 | 0x2A9 | TxPDO2, AI, Nd.41 | 702 | 0x2BE | TxPDO2, AI, Nd.62 |
| 661 | 0x295 | TxPDO2, AI, Nd.21 | 682 | 0x2AA | TxPDO2, AI, Nd.42 | 703 | 0x2BF | TxPDO2, AI, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 769 | 0x301 | RxPDO2, AO, Nd.1 | 790 | 0x316 | RxPDO2, AO, Nd.22 | 811 | 0x32B | RxPDO2, AO, Nd.43 |
| 770 | 0x302 | RxPDO2, AO, Nd.2 | 791 | 0x317 | RxPDO2, AO, Nd.23 | 812 | 0x32C | RxPDO2, AO, Nd.44 |
| 771 | 0x303 | RxPDO2, AO, Nd.3 | 792 | 0x318 | RxPDO2, AO, Nd.24 | 813 | 0x32D | RxPDO2, AO, Nd.45 |
| 772 | 0x304 | RxPDO2, AO, Nd.4 | 793 | 0x319 | RxPDO2, AO, Nd.25 | 814 | 0x32E | RxPDO2, AO, Nd.46 |
| 773 | 0x305 | RxPDO2, AO, Nd.5 | 794 | 0x31A | RxPDO2, AO, Nd.26 | 815 | 0x32F | RxPDO2, AO, Nd.47 |
| 774 | 0x306 | RxPDO2, AO, Nd.6 | 795 | 0x31B | RxPDO2, AO, Nd.27 | 816 | 0x330 | RxPDO2, AO, Nd.48 |
| 775 | 0x307 | RxPDO2, AO, Nd.7 | 796 | 0x31C | RxPDO2, AO, Nd.28 | 817 | 0x331 | RxPDO2, AO, Nd.49 |
| 776 | 0x308 | RxPDO2, AO, Nd.8 | 797 | 0x31D | RxPDO2, AO, Nd.29 | 818 | 0x332 | RxPDO2, AO, Nd.50 |
| 777 | 0x309 | RxPDO2, AO, Nd.9 | 798 | 0x31E | RxPDO2, AO, Nd.30 | 819 | 0x333 | RxPDO2, AO, Nd.51 |
| 778 | 0x30A | RxPDO2, AO, Nd.10 | 799 | 0x31F | RxPDO2, AO, Nd.31 | 820 | 0x334 | RxPDO2, AO, Nd.52 |
| 779 | 0x30B | RxPDO2, AO, Nd.11 | 800 | 0x320 | RxPDO2, AO, Nd.32 | 821 | 0x335 | RxPDO2, AO, Nd.53 |
| 780 | 0x30C | RxPDO2, AO, Nd.12 | 801 | 0x321 | RxPDO2, AO, Nd.33 | 822 | 0x336 | RxPDO2, AO, Nd.54 |
| 781 | 0x30D | RxPDO2, AO, Nd.13 | 802 | 0x322 | RxPDO2, AO, Nd.34 | 823 | 0x337 | RxPDO2, AO, Nd.55 |
| 782 | 0x30E | RxPDO2, AO, Nd.14 | 803 | 0x323 | RxPDO2, AO, Nd.35 | 824 | 0x338 | RxPDO2, AO, Nd.56 |
| 783 | 0x30F | RxPDO2, AO, Nd.15 | 804 | 0x324 | RxPDO2, AO, Nd.36 | 825 | 0x339 | RxPDO2, AO, Nd.57 |
| 784 | 0x310 | RxPDO2, AO, Nd.16 | 805 | 0x325 | RxPDO2, AO, Nd.37 | 826 | 0x33A | RxPDO2, AO, Nd.58 |
| 785 | 0x311 | RxPDO2, AO, Nd.17 | 806 | 0x326 | RxPDO2, AO, Nd.38 | 827 | 0x33B | RxPDO2, AO, Nd.59 |
| 786 | 0x312 | RxPDO2, AO, Nd.18 | 807 | 0x327 | RxPDO2, AO, Nd.39 | 828 | 0x33C | RxPDO2, AO, Nd.60 |
| 787 | 0x313 | RxPDO2, AO, Nd.19 | 808 | 0x328 | RxPDO2, AO, Nd.40 | 829 | 0x33D | RxPDO2, AO, Nd.61 |
| 788 | 0x314 | RxPDO2, AO, Nd.20 | 809 | 0x329 | RxPDO2, AO, Nd.41 | 830 | 0x33E | RxPDO2, AO, Nd.62 |
| 789 | 0x315 | RxPDO2, AO, Nd.21 | 810 | 0x32A | RxPDO2, AO, Nd.42 | 831 | 0x33F | RxPDO2, AO, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 897 | 0x381 | TxPDO3*, Nd.1 | 918 | 0x396 | TxPDO3*, Nd.22 | 939 | 0x3AB | TxPDO3*, Nd.43 |
| 898 | 0x382 | TxPDO3*, Nd.2 | 919 | 0x397 | TxPDO3*, Nd.23 | 940 | 0x3AC | TxPDO3*, Nd.44 |
| 899 | 0x383 | TxPDO3*, Nd.3 | 920 | 0x398 | TxPDO3*, Nd.24 | 941 | 0x3AD | TxPDO3*, Nd.45 |
| 900 | 0x384 | TxPDO3*, Nd.4 | 921 | 0x399 | TxPDO3*, Nd.25 | 942 | 0x3AE | TxPDO3*, Nd.46 |
| 901 | 0x385 | TxPDO3*, Nd.5 | 922 | 0x39A | TxPDO3*, Nd.26 | 943 | 0x3AF | TxPDO3*, Nd.47 |
| 902 | 0x386 | TxPDO3*, Nd.6 | 923 | 0x39B | TxPDO3*, Nd.27 | 944 | 0x3B0 | TxPDO3*, Nd.48 |
| 903 | 0x387 | TxPDO3*, Nd.7 | 924 | 0x39C | TxPDO3*, Nd.28 | 945 | 0x3B1 | TxPDO3*, Nd.49 |
| 904 | 0x388 | TxPDO3*, Nd.8 | 925 | 0x39D | TxPDO3*, Nd.29 | 946 | 0x3B2 | TxPDO3*, Nd.50 |
| 905 | 0x389 | TxPDO3*, Nd.9 | 926 | 0x39E | TxPDO3*, Nd.30 | 947 | 0x3B3 | TxPDO3*, Nd.51 |
| 906 | 0x38A | TxPDO3*, Nd.10 | 927 | 0x39F | TxPDO3*, Nd.31 | 948 | 0x3B4 | TxPDO3*, Nd.52 |
| 907 | 0x38B | TxPDO3*, Nd.11 | 928 | 0x3A0 | TxPDO3*, Nd.32 | 949 | 0x3B5 | TxPDO3*, Nd.53 |
| 908 | 0x38C | TxPDO3*, Nd.12 | 929 | 0x3A1 | TxPDO3*, Nd.33 | 950 | 0x3B6 | TxPDO3*, Nd.54 |
| 909 | 0x38D | TxPDO3*, Nd.13 | 930 | 0x3A2 | TxPDO3*, Nd.34 | 951 | 0x3B7 | TxPDO3*, Nd.55 |
| 910 | 0x38E | TxPDO3*, Nd.14 | 931 | 0x3A3 | TxPDO3*, Nd.35 | 952 | 0x3B8 | TxPDO3*, Nd.56 |
| 911 | 0x38F | TxPDO3*, Nd.15 | 932 | 0x3A4 | TxPDO3*, Nd.36 | 953 | 0x3B9 | TxPDO3*, Nd.57 |
| 912 | 0x390 | TxPDO3*, Nd.16 | 933 | 0x3A5 | TxPDO3*, Nd.37 | 954 | 0x3BA | TxPDO3*, Nd.58 |
| 913 | 0x391 | TxPDO3*, Nd.17 | 934 | 0x3A6 | TxPDO3*, Nd.38 | 955 | 0x3BB | TxPDO3*, Nd.59 |
| 914 | 0x392 | TxPDO3*, Nd.18 | 935 | 0x3A7 | TxPDO3*, Nd.39 | 956 | 0x3BC | TxPDO3*, Nd.60 |
| 915 | 0x393 | TxPDO3*, Nd.19 | 936 | 0x3A8 | TxPDO3*, Nd.40 | 957 | 0x3BD | TxPDO3*, Nd.61 |
| 916 | 0x394 | TxPDO3*, Nd.20 | 937 | 0x3A9 | TxPDO3*, Nd.41 | 958 | 0x3BE | TxPDO3*, Nd.62 |
| 917 | 0x395 | TxPDO3*, Nd.21 | 938 | 0x3AA | TxPDO3*, Nd.42 | 959 | 0x3BF | TxPDO3*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1025 | 0x401 | RxPDO3*, Nd.1 | 1046 | 0x416 | RxPDO3*, Nd.22 | 1067 | 0x42B | RxPDO3*, Nd.43 |
| 1026 | 0x402 | RxPDO3*, Nd.2 | 1047 | 0x417 | RxPDO3*, Nd.23 | 1068 | 0x42C | RxPDO3*, Nd.44 |
| 1027 | 0x403 | RxPDO3*, Nd.3 | 1048 | 0x418 | RxPDO3*, Nd.24 | 1069 | 0x42D | RxPDO3*, Nd.45 |
| 1028 | 0x404 | RxPDO3*, Nd.4 | 1049 | 0x419 | RxPDO3*, Nd.25 | 1070 | 0x42E | RxPDO3*, Nd.46 |
| 1029 | 0x405 | RxPDO3*, Nd.5 | 1050 | 0x41A | RxPDO3*, Nd.26 | 1071 | 0x42F | RxPDO3*, Nd.47 |
| 1030 | 0x406 | RxPDO3*, Nd.6 | 1051 | 0x41B | RxPDO3*, Nd.27 | 1072 | 0x430 | RxPDO3*, Nd.48 |
| 1031 | 0x407 | RxPDO3*, Nd.7 | 1052 | 0x41C | RxPDO3*, Nd.28 | 1073 | 0x431 | RxPDO3*, Nd.49 |
| 1032 | 0x408 | RxPDO3*, Nd.8 | 1053 | 0x41D | RxPDO3*, Nd.29 | 1074 | 0x432 | RxPDO3*, Nd.50 |
| 1033 | 0x409 | RxPDO3*, Nd.9 | 1054 | 0x41E | RxPDO3*, Nd.30 | 1075 | 0x433 | RxPDO3*, Nd.51 |
| 1034 | 0x40A | RxPDO3*, Nd.10 | 1055 | 0x41F | RxPDO3*, Nd.31 | 1076 | 0x434 | RxPDO3*, Nd.52 |
| 1035 | 0x40B | RxPDO3*, Nd.11 | 1056 | 0x420 | RxPDO3*, Nd.32 | 1077 | 0x435 | RxPDO3*, Nd.53 |
| 1036 | 0x40C | RxPDO3*, Nd.12 | 1057 | 0x421 | RxPDO3*, Nd.33 | 1078 | 0x436 | RxPDO3*, Nd.54 |
| 1037 | 0x40D | RxPDO3*, Nd.13 | 1058 | 0x422 | RxPDO3*, Nd.34 | 1079 | 0x437 | RxPDO3*, Nd.55 |
| 1038 | 0x40E | RxPDO3*, Nd.14 | 1059 | 0x423 | RxPDO3*, Nd.35 | 1080 | 0x438 | RxPDO3*, Nd.56 |
| 1039 | 0x40F | RxPDO3*, Nd.15 | 1060 | 0x424 | RxPDO3*, Nd.36 | 1081 | 0x439 | RxPDO3*, Nd.57 |
| 1040 | 0x410 | RxPDO3*, Nd.16 | 1061 | 0x425 | RxPDO3*, Nd.37 | 1082 | 0x43A | RxPDO3*, Nd.58 |
| 1041 | 0x411 | RxPDO3*, Nd.17 | 1062 | 0x426 | RxPDO3*, Nd.38 | 1083 | 0x43B | RxPDO3*, Nd.59 |
| 1042 | 0x412 | RxPDO3*, Nd.18 | 1063 | 0x427 | RxPDO3*, Nd.39 | 1084 | 0x43C | RxPDO3*, Nd.60 |
| 1043 | 0x413 | RxPDO3*, Nd.19 | 1064 | 0x428 | RxPDO3*, Nd.40 | 1085 | 0x43D | RxPDO3*, Nd.61 |
| 1044 | 0x414 | RxPDO3*, Nd.20 | 1065 | 0x429 | RxPDO3*, Nd.41 | 1086 | 0x43E | RxPDO3*, Nd.62 |
| 1045 | 0x415 | RxPDO3*, Nd.21 | 1066 | 0x42A | RxPDO3*, Nd.42 | 1087 | 0x43F | RxPDO3*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1153 | 0x481 | TxPDO4*, Nd.1 | 1174 | 0x496 | TxPDO4*, Nd.22 | 1195 | 0x4AB | TxPDO4*, Nd.43 |
| 1154 | 0x482 | TxPDO4*, Nd.2 | 1175 | 0x497 | TxPDO4*, Nd.23 | 1196 | 0x4AC | TxPDO4*, Nd.44 |
| 1155 | 0x483 | TxPDO4*, Nd.3 | 1176 | 0x498 | TxPDO4*, Nd.24 | 1197 | 0x4AD | TxPDO4*, Nd.45 |
| 1156 | 0x484 | TxPDO4*, Nd.4 | 1177 | 0x499 | TxPDO4*, Nd.25 | 1198 | 0x4AE | TxPDO4*, Nd.46 |
| 1157 | 0x485 | TxPDO4*, Nd.5 | 1178 | 0x49A | TxPDO4*, Nd.26 | 1199 | 0x4AF | TxPDO4*, Nd.47 |
| 1158 | 0x486 | TxPDO4*, Nd.6 | 1179 | 0x49B | TxPDO4*, Nd.27 | 1200 | 0x4B0 | TxPDO4*, Nd.48 |
| 1159 | 0x487 | TxPDO4*, Nd.7 | 1180 | 0x49C | TxPDO4*, Nd.28 | 1201 | 0x4B1 | TxPDO4*, Nd.49 |
| 1160 | 0x488 | TxPDO4*, Nd.8 | 1181 | 0x49D | TxPDO4*, Nd.29 | 1202 | 0x4B2 | TxPDO4*, Nd.50 |
| 1161 | 0x489 | TxPDO4*, Nd.9 | 1182 | 0x49E | TxPDO4*, Nd.30 | 1203 | 0x4B3 | TxPDO4*, Nd.51 |
| 1162 | 0x48A | TxPDO4*, Nd.10 | 1183 | 0x49F | TxPDO4*, Nd.31 | 1204 | 0x4B4 | TxPDO4*, Nd.52 |
| 1163 | 0x48B | TxPDO4*, Nd.11 | 1184 | 0x4A0 | TxPDO4*, Nd.32 | 1205 | 0x4B5 | TxPDO4*, Nd.53 |
| 1164 | 0x48C | TxPDO4*, Nd.12 | 1185 | 0x4A1 | TxPDO4*, Nd.33 | 1206 | 0x4B6 | TxPDO4*, Nd.54 |
| 1165 | 0x48D | TxPDO4*, Nd.13 | 1186 | 0x4A2 | TxPDO4*, Nd.34 | 1207 | 0x4B7 | TxPDO4*, Nd.55 |
| 1166 | 0x48E | TxPDO4*, Nd.14 | 1187 | 0x4A3 | TxPDO4*, Nd.35 | 1208 | 0x4B8 | TxPDO4*, Nd.56 |
| 1167 | 0x48F | TxPDO4*, Nd.15 | 1188 | 0x4A4 | TxPDO4*, Nd.36 | 1209 | 0x4B9 | TxPDO4*, Nd.57 |
| 1168 | 0x490 | TxPDO4*, Nd.16 | 1189 | 0x4A5 | TxPDO4*, Nd.37 | 1210 | 0x4BA | TxPDO4*, Nd.58 |
| 1169 | 0x491 | TxPDO4*, Nd.17 | 1190 | 0x4A6 | TxPDO4*, Nd.48 | 1211 | 0x4BB | TxPDO4*, Nd.59 |
| 1170 | 0x492 | TxPDO4*, Nd.18 | 1191 | 0x4A7 | TxPDO4*, Nd.49 | 1212 | 0x4BC | TxPDO4*, Nd.60 |
| 1171 | 0x493 | TxPDO4*, Nd.19 | 1192 | 0x4A8 | TxPDO4*, Nd.40 | 1213 | 0x4BD | TxPDO4*, Nd.61 |
| 1172 | 0x494 | TxPDO4*, Nd.20 | 1193 | 0x4A9 | TxPDO4*, Nd.41 | 1214 | 0x4BE | TxPDO4*, Nd.62 |
| 1173 | 0x495 | TxPDO4*, Nd.21 | 1194 | 0x4AA | TxPDO4*, Nd.42 | 1215 | 0x4BF | TxPDO4*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1281 | 0x501 | RxPDO4*, Nd.1 | 1302 | 0x516 | RxPDO4*, Nd.22 | 1323 | 0x52B | RxPDO4*, Nd.43 |
| 1282 | 0x502 | RxPDO4*, Nd.2 | 1303 | 0x517 | RxPDO4*, Nd.23 | 1324 | 0x52C | RxPDO4*, Nd.44 |
| 1283 | 0x503 | RxPDO4*, Nd.3 | 1304 | 0x518 | RxPDO4*, Nd.24 | 1325 | 0x52D | RxPDO4*, Nd.45 |
| 1284 | 0x504 | RxPDO4*, Nd.4 | 1305 | 0x519 | RxPDO4*, Nd.25 | 1326 | 0x52E | RxPDO4*, Nd.46 |
| 1285 | 0x505 | RxPDO4*, Nd.5 | 1306 | 0x51A | RxPDO4*, Nd.26 | 1327 | 0x52F | RxPDO4*, Nd.47 |
| 1286 | 0x506 | RxPDO4*, Nd.6 | 1307 | 0x51B | RxPDO4*, Nd.27 | 1328 | 0x530 | RxPDO4*, Nd.48 |
| 1287 | 0x507 | RxPDO4*, Nd.7 | 1308 | 0x51C | RxPDO4*, Nd.28 | 1329 | 0x531 | RxPDO4*, Nd.49 |
| 1288 | 0x508 | RxPDO4*, Nd.8 | 1309 | 0x51D | RxPDO4*, Nd.29 | 1330 | 0x532 | RxPDO4*, Nd.50 |
| 1289 | 0x509 | RxPDO4*, Nd.9 | 1310 | 0x51E | RxPDO4*, Nd.30 | 1331 | 0x533 | RxPDO4*, Nd.51 |
| 1290 | 0x50A | RxPDO4*, Nd.10 | 1311 | 0x51F | RxPDO4*, Nd.31 | 1332 | 0x534 | RxPDO4*, Nd.52 |
| 1291 | 0x50B | RxPDO4*, Nd.11 | 1312 | 0x520 | RxPDO4*, Nd.32 | 1333 | 0x535 | RxPDO4*, Nd.53 |
| 1292 | 0x50C | RxPDO4*, Nd.12 | 1313 | 0x521 | RxPDO4*, Nd.33 | 1334 | 0x536 | RxPDO4*, Nd.54 |
| 1293 | 0x50D | RxPDO4*, Nd.13 | 1314 | 0x522 | RxPDO4*, Nd.34 | 1335 | 0x537 | RxPDO4*, Nd.55 |
| 1294 | 0x50E | RxPDO4*, Nd.14 | 1315 | 0x523 | RxPDO4*, Nd.35 | 1336 | 0x538 | RxPDO4*, Nd.56 |
| 1295 | 0x50F | RxPDO4*, Nd.15 | 1316 | 0x524 | RxPDO4*, Nd.36 | 1337 | 0x539 | RxPDO4*, Nd.57 |
| 1296 | 0x510 | RxPDO4*, Nd.16 | 1317 | 0x525 | RxPDO4*, Nd.37 | 1338 | 0x53A | RxPDO4*, Nd.58 |
| 1297 | 0x511 | RxPDO4*, Nd.17 | 1318 | 0x526 | RxPDO4*, Nd.38 | 1339 | 0x53B | RxPDO4*, Nd.59 |
| 1298 | 0x512 | RxPDO4*, Nd.18 | 1319 | 0x527 | RxPDO4*, Nd.39 | 1340 | 0x53C | RxPDO4*, Nd.60 |
| 1299 | 0x513 | RxPDO4*, Nd.19 | 1320 | 0x528 | RxPDO4*, Nd.40 | 1341 | 0x53D | RxPDO4*, Nd.61 |
| 1300 | 0x514 | RxPDO4*, Nd.20 | 1321 | 0x529 | RxPDO4*, Nd.41 | 1342 | 0x53E | RxPDO4*, Nd.62 |
| 1301 | 0x515 | RxPDO4*, Nd.21 | 1322 | 0x52A | RxPDO4*, Nd.42 | 1343 | 0x53F | RxPDO4*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1665 | 0x681 | TxPDO5*, Nd.1 | 1686 | 0x696 | TxPDO5*, Nd.22 | 1707 | 0x6AB | TxPDO5*, Nd.43 |
| 1666 | 0x682 | TxPDO5*, Nd.2 | 1687 | 0x697 | TxPDO5*, Nd.23 | 1708 | 0x6AC | TxPDO5*, Nd.44 |
| 1667 | 0x683 | TxPDO5*, Nd.3 | 1688 | 0x698 | TxPDO5*, Nd.24 | 1709 | 0x6AD | TxPDO5*, Nd.45 |
| 1668 | 0x684 | TxPDO5*, Nd.4 | 1689 | 0x699 | TxPDO5*, Nd.25 | 1710 | 0x6AE | TxPDO5*, Nd.46 |
| 1669 | 0x685 | TxPDO5*, Nd.5 | 1690 | 0x69A | TxPDO5*, Nd.26 | 1711 | 0x6AF | TxPDO5*, Nd.47 |
| 1670 | 0x686 | TxPDO5*, Nd.6 | 1691 | 0x69B | TxPDO5*, Nd.27 | 1712 | 0x6B0 | TxPDO5*, Nd.48 |
| 1671 | 0x687 | TxPDO5*, Nd.7 | 1692 | 0x69C | TxPDO5*, Nd.28 | 1713 | 0x6B1 | TxPDO5*, Nd.49 |
| 1672 | 0x688 | TxPDO5*, Nd.8 | 1693 | 0x69D | TxPDO5*, Nd.29 | 1714 | 0x6B2 | TxPDO5*, Nd.50 |
| 1673 | 0x689 | TxPDO5*, Nd.9 | 1694 | 0x69E | TxPDO5*, Nd.30 | 1715 | 0x6B3 | TxPDO5*, Nd.51 |
| 1674 | 0x68A | TxPDO5*, Nd.10 | 1695 | 0x69F | TxPDO5*, Nd.31 | 1716 | 0x6B4 | TxPDO5*, Nd.52 |
| 1675 | 0x68B | TxPDO5*, Nd.11 | 1696 | 0x6A0 | TxPDO5*, Nd.32 | 1717 | 0x6B5 | TxPDO5*, Nd.53 |
| 1676 | 0x68C | TxPDO5*, Nd.12 | 1697 | 0x6A1 | TxPDO5*, Nd.33 | 1718 | 0x6B6 | TxPDO5*, Nd.54 |
| 1677 | 0x68D | TxPDO5*, Nd.13 | 1698 | 0x6A2 | TxPDO5*, Nd.34 | 1719 | 0x6B7 | TxPDO5*, Nd.55 |
| 1678 | 0x68E | TxPDO5*, Nd.14 | 1699 | 0x6A3 | TxPDO5*, Nd.35 | 1720 | 0x6B8 | TxPDO5*, Nd.56 |
| 1679 | 0x68F | TxPDO5*, Nd.15 | 1700 | 0x6A4 | TxPDO5*, Nd.36 | 1721 | 0x6B9 | TxPDO5*, Nd.57 |
| 1680 | 0x690 | TxPDO5*, Nd.16 | 1701 | 0x6A5 | TxPDO5*, Nd.37 | 1722 | 0x6BA | TxPDO5*, Nd.58 |
| 1681 | 0x691 | TxPDO5*, Nd.17 | 1702 | 0x6A6 | TxPDO5*, Nd.38 | 1723 | 0x6BB | TxPDO5*, Nd.59 |
| 1682 | 0x692 | TxPDO5*, Nd.18 | 1703 | 0x6A7 | TxPDO5*, Nd.39 | 1724 | 0x6BC | TxPDO5*, Nd.60 |
| 1683 | 0x693 | TxPDO5*, Nd.19 | 1704 | 0x6A8 | TxPDO5*, Nd.40 | 1725 | 0x6BD | TxPDO5*, Nd.61 |
| 1684 | 0x694 | TxPDO5*, Nd.20 | 1705 | 0x6A9 | TxPDO5*, Nd.41 | 1726 | 0x6BE | TxPDO5*, Nd.62 |
| 1685 | 0x695 | TxPDO5*, Nd.21 | 1706 | 0x6AA | TxPDO5*, Nd.42 | 1727 | 0x6BF | TxPDO5*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1921 | 0x781 | RxPDO5*, Nd.1 | 1942 | 0x796 | RxPDO5*, Nd.22 | 1963 | 0x7AB | RxPDO5*, Nd.43 |
| 1922 | 0x782 | RxPDO5*, Nd.2 | 1943 | 0x797 | RxPDO5*, Nd.23 | 1964 | 0x7AC | RxPDO5*, Nd.44 |
| 1923 | 0x783 | RxPDO5*, Nd.3 | 1944 | 0x798 | RxPDO5*, Nd.24 | 1965 | 0x7AD | RxPDO5*, Nd.45 |
| 1924 | 0x784 | RxPDO5*, Nd.4 | 1945 | 0x799 | RxPDO5*, Nd.25 | 1966 | 0x7AE | RxPDO5*, Nd.46 |
| 1925 | 0x785 | RxPDO5*, Nd.5 | 1946 | 0x79A | RxPDO5*, Nd.26 | 1967 | 0x7AF | RxPDO5*, Nd.47 |
| 1926 | 0x786 | RxPDO5*, Nd.6 | 1947 | 0x79B | RxPDO5*, Nd.27 | 1968 | 0x7B0 | RxPDO5*, Nd.48 |
| 1927 | 0x787 | RxPDO5*, Nd.7 | 1948 | 0x79C | RxPDO5*, Nd.28 | 1969 | 0x7B1 | RxPDO5*, Nd.49 |
| 1928 | 0x788 | RxPDO5*, Nd.8 | 1949 | 0x79D | RxPDO5*, Nd.29 | 1970 | 0x7B2 | RxPDO5*, Nd.50 |
| 1929 | 0x789 | RxPDO5*, Nd.9 | 1950 | 0x79E | RxPDO5*, Nd.30 | 1971 | 0x7B3 | RxPDO5*, Nd.51 |
| 1930 | 0x78A | RxPDO5*, Nd.10 | 1951 | 0x79F | RxPDO5*, Nd.31 | 1972 | 0x7B4 | RxPDO5*, Nd.52 |
| 1931 | 0x78B | RxPDO5*, Nd.11 | 1952 | 0x7A0 | RxPDO5*, Nd.32 | 1973 | 0x7B5 | RxPDO5*, Nd.53 |
| 1932 | 0x78C | RxPDO5*, Nd.12 | 1953 | 0x7A1 | RxPDO5*, Nd.33 | 1974 | 0x7B6 | RxPDO5*, Nd.54 |
| 1933 | 0x78D | RxPDO5*, Nd.13 | 1954 | 0x7A2 | RxPDO5*, Nd.34 | 1975 | 0x7B7 | RxPDO5*, Nd.55 |
| 1934 | 0x78E | RxPDO5*, Nd.14 | 1955 | 0x7A3 | RxPDO5*, Nd.35 | 1976 | 0x7B8 | RxPDO5*, Nd.56 |
| 1935 | 0x78F | RxPDO5*, Nd.15 | 1956 | 0x7A4 | RxPDO5*, Nd.36 | 1977 | 0x7B9 | RxPDO5*, Nd.57 |
| 1936 | 0x790 | RxPDO5*, Nd.16 | 1957 | 0x7A5 | RxPDO5*, Nd.37 | 1978 | 0x7BA | RxPDO5*, Nd.58 |
| 1937 | 0x791 | RxPDO5*, Nd.17 | 1958 | 0x7A6 | RxPDO5*, Nd.38 | 1979 | 0x7BB | RxPDO5*, Nd.59 |
| 1938 | 0x792 | RxPDO5*, Nd.18 | 1959 | 0x7A7 | RxPDO5*, Nd.39 | 1980 | 0x7BC | RxPDO5*, Nd.60 |
| 1939 | 0x793 | RxPDO5*, Nd.19 | 1960 | 0x7A8 | RxPDO5*, Nd.40 | 1981 | 0x7BD | RxPDO5*, Nd.61 |
| 1940 | 0x794 | RxPDO5*, Nd.20 | 1961 | 0x7A9 | RxPDO5*, Nd.41 | 1982 | 0x7BE | RxPDO5*, Nd.62 |
| 1941 | 0x795 | RxPDO5*, Nd.21 | 1962 | 0x7AA | RxPDO5*, Nd.42 | 1983 | 0x7BF | RxPDO5*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 449 | 0x1C1 | TxPDO6*, Nd.1 | 470 | 0x1D6 | TxPDO6*, Nd.22 | 491 | 0x1EB | TxPDO6*, Nd.43 |
| 450 | 0x1C2 | TxPDO6*, Nd.2 | 471 | 0x1D7 | TxPDO6*, Nd.23 | 492 | 0x1EC | TxPDO6*, Nd.44 |
| 451 | 0x1C3 | TxPDO6*, Nd.3 | 472 | 0x1D8 | TxPDO6*, Nd.24 | 493 | 0x1ED | TxPDO6*, Nd.45 |
| 452 | 0x1C4 | TxPDO6*, Nd.4 | 473 | 0x1D9 | TxPDO6*, Nd.25 | 494 | 0x1EE | TxPDO6*, Nd.46 |
| 453 | 0x1C5 | TxPDO6*, Nd.5 | 474 | 0x1DA | TxPDO6*, Nd.26 | 495 | 0x1EF | TxPDO6*, Nd.47 |
| 454 | 0x1C6 | TxPDO6*, Nd.6 | 475 | 0x1DB | TxPDO6*, Nd.27 | 496 | 0x1F0 | TxPDO6*, Nd.48 |
| 455 | 0x1C7 | TxPDO6*, Nd.7 | 476 | 0x1DC | TxPDO6*, Nd.28 | 497 | 0x1F1 | TxPDO6*, Nd.49 |
| 456 | 0x1C8 | TxPDO6*, Nd.8 | 477 | 0x1DD | TxPDO6*, Nd.29 | 498 | 0x1F2 | TxPDO6*, Nd.50 |
| 457 | 0x1C9 | TxPDO6*, Nd.9 | 478 | 0x1DE | TxPDO6*, Nd.30 | 499 | 0x1F3 | TxPDO6*, Nd.51 |
| 458 | 0x1CA | TxPDO6*, Nd.10 | 479 | 0x1DF | TxPDO6*, Nd.31 | 500 | 0x1F4 | TxPDO6*, Nd.52 |
| 459 | 0x1CB | TxPDO6*, Nd.11 | 480 | 0x1E0 | TxPDO6*, Nd.32 | 501 | 0x1F5 | TxPDO6*, Nd.53 |
| 460 | 0x1CC | TxPDO6*, Nd.12 | 481 | 0x1E1 | TxPDO6*, Nd.33 | 502 | 0x1F6 | TxPDO6*, Nd.54 |
| 461 | 0x1CD | TxPDO6*, Nd.13 | 482 | 0x1E2 | TxPDO6*, Nd.34 | 503 | 0x1F7 | TxPDO6*, Nd.55 |
| 462 | 0x1CE | TxPDO6*, Nd.14 | 483 | 0x1E3 | TxPDO6*, Nd.35 | 504 | 0x1F8 | TxPDO6*, Nd.56 |
| 463 | 0x1CF | TxPDO6*, Nd.15 | 484 | 0x1E4 | TxPDO6*, Nd.36 | 505 | 0x1F9 | TxPDO6*, Nd.57 |
| 464 | 0x1D0 | TxPDO6*, Nd.16 | 485 | 0x1E5 | TxPDO6*, Nd.37 | 506 | 0x1FA | TxPDO6*, Nd.58 |
| 465 | 0x1D1 | TxPDO6*, Nd.17 | 486 | 0x1E6 | TxPDO6*, Nd.38 | 507 | 0x1FB | TxPDO6*, Nd.59 |
| 466 | 0x1D2 | TxPDO6*, Nd.18 | 487 | 0x1E7 | TxPDO6*, Nd.39 | 508 | 0x1FC | TxPDO6*, Nd.60 |
| 467 | 0x1D3 | TxPDO6*, Nd.19 | 488 | 0x1E8 | TxPDO6*, Nd.40 | 509 | 0x1FD | TxPDO6*, Nd.61 |
| 468 | 0x1D4 | TxPDO6*, Nd.20 | 489 | 0x1E9 | TxPDO6*, Nd.41 | 510 | 0x1FE | TxPDO6*, Nd.62 |
| 469 | 0x1D5 | TxPDO6*, Nd.21 | 490 | 0x1EA | TxPDO6*, Nd.42 | 511 | 0x1FF | TxPDO6*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 577 | 0x241 | RxPDO6*, Nd.1 | 598 | 0x256 | RxPDO6*, Nd.22 | 619 | 0x26B | RxPDO6* Nd.43 |
| 578 | 0x242 | RxPDO6*, Nd.2 | 599 | 0x257 | RxPDO6*, Nd.23 | 620 | 0x26C | RxPDO6, Nd.44 |
| 579 | 0x243 | RxPDO6*, Nd.3 | 600 | 0x258 | RxPDO6*, Nd.24 | 621 | 0x26D | RxPDO6*, Nd.45 |
| 580 | 0x244 | RxPDO6*, Nd.4 | 601 | 0x259 | RxPDO6*, Nd.25 | 622 | 0x26E | RxPDO6*, Nd.46 |
| 581 | 0x245 | RxPDO6*, Nd.5 | 602 | 0x25A | RxPDO6*, Nd.26 | 623 | 0x26F | RxPDO6*, Nd.47 |
| 582 | 0x246 | RxPDO6*, Nd.6 | 603 | 0x25B | RxPDO6*, Nd.27 | 624 | 0x270 | RxPDO6*, Nd.48 |
| 583 | 0x247 | RxPDO6*, Nd.7 | 604 | 0x25C | RxPDO6*, Nd.28 | 625 | 0x271 | RxPDO6*, Nd.49 |
| 584 | 0x248 | RxPDO6*, Nd.8 | 605 | 0x25D | RxPDO6*, Nd.29 | 626 | 0x272 | RxPDO6*, Nd.50 |
| 585 | 0x249 | RxPDO6*, Nd.9 | 606 | 0x25E | RxPDO6*, Nd.30 | 627 | 0x273 | RxPDO6*, Nd.51 |
| 586 | 0x24A | RxPDO6*, Nd.10 | 607 | 0x25F | RxPDO6*, Nd.31 | 628 | 0x274 | RxPDO6*, Nd.52 |
| 587 | 0x24B | RxPDO6*, Nd.11 | 608 | 0x260 | RxPDO6*, Nd.32 | 629 | 0x275 | RxPDO6*, Nd.53 |
| 588 | 0x24C | RxPDO6*, Nd.12 | 609 | 0x261 | RxPDO6*, Nd.33 | 630 | 0x276 | RxPDO6*, Nd.54 |
| 589 | 0x24D | RxPDO6*, Nd.13 | 610 | 0x262 | RxPDO6*, Nd.34 | 631 | 0x277 | RxPDO6*, Nd.55 |
| 590 | 0x24E | RxPDO6*, Nd.14 | 611 | 0x263 | RxPDO6*, Nd.35 | 632 | 0x278 | RxPDO6*, Nd.56 |
| 591 | 0x24F | RxPDO6*, Nd.15 | 612 | 0x264 | RxPDO6*, Nd.36 | 633 | 0x279 | RxPDO6*, Nd.57 |
| 592 | 0x250 | RxPDO6*, Nd.16 | 613 | 0x265 | RxPDO6*, Nd.3 | 634 | 0x27A | RxPDO6*, Nd.58 |
| 593 | 0x251 | RxPDO6*, Nd.17 | 614 | 0x266 | RxPDO6*, Nd.8 | 635 | 0x27B | RxPDO6*, Nd.59 |
| 594 | 0x252 | RxPDO6*, Nd.18 | 615 | 0x267 | RxPDO6*, Nd39 | 636 | 0x27C | RxPDO6*, Nd.60 |
| 595 | 0x253 | RxPDO6*, Nd.19 | 616 | 0x268 | RxPDO6*, N.40 | 637 | 0x27D | RxPDO6*, Nd.61 |
| 596 | 0x254 | RxPDO6*, Nd.20 | 617 | 0x269 | RxPDO6*, d.41 | 638 | 0x27E | RxPDO6*, Nd.62 |
| 597 | 0x255 | RxPDO6*, Nd.21 | 618 | 0x26A | RxPDO6*,Nd.42 | 639 | 0x27F | RxPDO6*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 705 | 0x2C1 | TxPDO7*, Nd.1 | 726 | 0x2D6 | TxPDO7*, Nd.22 | 747 | 0x2EB | TxPDO7*, Nd.43 |
| 706 | 0x2C2 | TxPDO7*, Nd.2 | 727 | 0x2D7 | TxPDO7*, Nd.23 | 748 | 0x2EC | TxPDO7*, Nd.44 |
| 707 | 0x2C3 | TxPDO7*, Nd.3 | 728 | 0x2D8 | TxPDO7*, Nd.24 | 749 | 0x2ED | TxPDO7*, Nd.45 |
| 708 | 0x2C4 | TxPDO7*, Nd.4 | 729 | 0x2D9 | TxPDO7*, Nd.25 | 750 | 0x2EE | TxPDO7*, Nd.46 |
| 709 | 0x2C5 | TxPDO7*, Nd.5 | 730 | 0x2DA | TxPDO7*, Nd.26 | 751 | 0x2EF | TxPDO7*, Nd.47 |
| 710 | 0x2C6 | TxPDO7*, Nd.6 | 731 | 0x2DB | TxPDO7*, Nd.27 | 752 | 0x2F0 | TxPDO7*, Nd.48 |
| 711 | 0x2C7 | TxPDO7*, Nd.7 | 732 | 0x2DC | TxPDO7*, Nd.28 | 753 | 0x2F1 | TxPDO7*, Nd.49 |
| 712 | 0x2C8 | TxPDO7*, Nd.8 | 733 | 0x2DD | TxPDO7*, Nd.29 | 754 | 0x2F2 | TxPDO7*, Nd.50 |
| 713 | 0x2C9 | TxPDO7*, Nd.9 | 734 | 0x2DE | TxPDO7*, Nd.30 | 755 | 0x2F3 | TxPDO7*, Nd.51 |
| 714 | 0x2CA | TxPDO7*, Nd.10 | 735 | 0x2DF | TxPDO7*, Nd.31 | 756 | 0x2F4 | TxPDO7*, Nd.52 |
| 715 | 0x2CB | TxPDO7*, Nd.11 | 736 | 0x2E0 | TxPDO7*, Nd.32 | 757 | 0x2F5 | TxPDO7*, Nd.53 |
| 716 | 0x2CC | TxPDO7*, Nd.12 | 737 | 0x2E1 | TxPDO7*, Nd.33 | 758 | 0x2F6 | TxPDO7*, Nd.54 |
| 717 | 0x2CD | TxPDO7*, Nd.13 | 738 | 0x2E2 | TxPDO7*, Nd.34 | 759 | 0x2F7 | TxPDO7*, Nd.55 |
| 718 | 0x2CE | TxPDO7*, Nd.14 | 739 | 0x2E3 | TxPDO7*, Nd.35 | 760 | 0x2F8 | TxPDO7*, Nd.56 |
| 719 | 0x2CF | TxPDO7*, Nd.15 | 740 | 0x2E4 | TxPDO7*, Nd.36 | 761 | 0x2F9 | TxPDO7*, Nd.57 |
| 720 | 0x2D0 | TxPDO7*, Nd.16 | 741 | 0x2E5 | TxPDO7*, Nd.37 | 762 | 0x2FA | TxPDO7*, Nd.58 |
| 721 | 0x2D1 | TxPDO7*, Nd.17 | 742 | 0x2E6 | TxPDO7*, Nd.38 | 763 | 0x2FB | TxPDO7*, Nd.59 |
| 722 | 0x2D2 | TxPDO7*, Nd.18 | 743 | 0x2E7 | TxPDO7*, Nd.39 | 764 | 0x2FC | TxPDO7*, Nd.60 |
| 723 | 0x2D3 | TxPDO7*, Nd.19 | 744 | 0x2E8 | TxPDO7*, Nd.40 | 765 | 0x2FD | TxPDO7*, Nd.61 |
| 724 | 0x2D4 | TxPDO7*, Nd.20 | 745 | 0x2E9 | TxPDO7*, Nd.41 | 766 | 0x2FE | TxPDO7*, Nd.62 |
| 725 | 0x2D5 | TxPDO7*, Nd.21 | 746 | 0x2EA | TxPDO7*, Nd.42 | 767 | 0x2FF | TxPDO7*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 833 | 0x341 | RxPDO7*, Nd.1 | 854 | 0x356 | RxPDO7*, Nd.22 | 875 | 0x36B | RxPDO7*, Nd.43 |
| 834 | 0x342 | RxPDO7*, Nd.2 | 855 | 0x357 | RxPDO7*, Nd.23 | 876 | 0x36C | RxPDO7*, Nd.44 |
| 835 | 0x343 | RxPDO7*, Nd.3 | 856 | 0x358 | RxPDO7*, Nd.24 | 877 | 0x36D | RxPDO7*, Nd.45 |
| 836 | 0x344 | RxPDO7*, Nd.4 | 857 | 0x359 | RxPDO7*, Nd.25 | 878 | 0x36E | RxPDO7*, Nd.46 |
| 837 | 0x345 | RxPDO7*, Nd.5 | 858 | 0x35A | RxPDO7*, Nd.26 | 879 | 0x36F | RxPDO7*, Nd.47 |
| 838 | 0x346 | RxPDO7*, Nd.6 | 859 | 0x35B | RxPDO7*, Nd.27 | 880 | 0x370 | RxPDO7*, Nd.48 |
| 839 | 0x347 | RxPDO7*, Nd.7 | 860 | 0x35C | RxPDO7*, Nd.28 | 881 | 0x371 | RxPDO7*, Nd.49 |
| 840 | 0x348 | RxPDO7*, Nd.8 | 861 | 0x35D | RxPDO7*, Nd.29 | 882 | 0x372 | RxPDO7*, Nd.50 |
| 841 | 0x349 | RxPDO7*, Nd.9 | 862 | 0x35E | RxPDO7*, Nd.30 | 883 | 0x373 | RxPDO7*, Nd.51 |
| 842 | 0x34A | RxPDO7*, Nd.10 | 863 | 0x35F | RxPDO7*, Nd.31 | 884 | 0x374 | RxPDO7*, Nd.52 |
| 843 | 0x34B | RxPDO7*, Nd.11 | 864 | 0x360 | RxPDO7*, Nd.32 | 885 | 0x375 | RxPDO7*, Nd.53 |
| 844 | 0x34C | RxPDO7*, Nd.12 | 865 | 0x361 | RxPDO7*, Nd.33 | 886 | 0x376 | RxPDO7*, Nd.54 |
| 845 | 0x34D | RxPDO7*, Nd.13 | 866 | 0x362 | RxPDO7*, Nd.34 | 887 | 0x377 | RxPDO7*, Nd.55 |
| 846 | 0x34E | RxPDO7*, Nd.14 | 867 | 0x363 | RxPDO7*, Nd.35 | 888 | 0x378 | RxPDO7*, Nd.56 |
| 847 | 0x34F | RxPDO7*, Nd.15 | 868 | 0x364 | RxPDO7*, Nd.36 | 889 | 0x379 | RxPDO7*, Nd.57 |
| 848 | 0x350 | RxPDO7*, Nd.16 | 869 | 0x365 | RxPDO7*, Nd.37 | 890 | 0x37A | RxPDO7*, Nd.58 |
| 849 | 0x351 | RxPDO7*, Nd.17 | 870 | 0x366 | RxPDO7*, Nd.38 | 891 | 0x37B | RxPDO7*, Nd.59 |
| 850 | 0x352 | RxPDO7*, Nd.18 | 871 | 0x367 | RxPDO7*, Nd.39 | 892 | 0x37C | RxPDO7*, Nd.60 |
| 851 | 0x353 | RxPDO7*, Nd.19 | 872 | 0x368 | RxPDO7*, Nd.40 | 893 | 0x37D | RxPDO7*, Nd.61 |
| 852 | 0x354 | RxPDO7*, Nd.20 | 873 | 0x369 | RxPDO7*, Nd.41 | 894 | 0x37E | RxPDO7*, Nd.62 |
| 853 | 0x355 | RxPDO7*, Nd.21 | 874 | 0x36A | RxPDO7*, Nd.42 | 895 | 0x37F | RxPDO7*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 961 | 0x3C1 | TxPDO8*, Nd.1 | 982 | 0x3D6 | TxPDO8*, Nd.22 | 1003 | 0x3EB | TxPDO8*, Nd.43 |
| 962 | 0x3C2 | TxPDO8*, Nd.2 | 983 | 0x3D7 | TxPDO8*, Nd.23 | 1004 | 0x3EC | TxPDO8*, Nd.44 |
| 963 | 0x3C3 | TxPDO8*, Nd.3 | 984 | 0x3D8 | TxPDO8*, Nd.24 | 1005 | 0x3ED | TxPDO8*, Nd.45 |
| 964 | 0x3C4 | TxPDO8*, Nd.4 | 985 | 0x3D9 | TxPDO8*, Nd.25 | 1006 | 0x3EE | TxPDO8*, Nd.46 |
| 965 | 0x3C5 | TxPDO8*, Nd.5 | 986 | 0x3DA | TxPDO8*, Nd.26 | 1007 | 0x3EF | TxPDO8*, Nd.47 |
| 966 | 0x3C6 | TxPDO8*, Nd.6 | 987 | 0x3DB | TxPDO8*, Nd.27 | 1008 | 0x3F0 | TxPDO8*, Nd.48 |
| 967 | 0x3C7 | TxPDO8*, Nd.7 | 988 | 0x3DC | TxPDO8*, Nd.28 | 1009 | 0x3F1 | TxPDO8*, Nd.49 |
| 968 | 0x3C8 | TxPDO8*, Nd.8 | 989 | 0x3DD | TxPDO8*, Nd.29 | 1010 | 0x3F2 | TxPDO8*, Nd.50 |
| 969 | 0x3C9 | TxPDO8*, Nd.9 | 990 | 0x3DE | TxPDO8*, Nd.30 | 1011 | 0x3F3 | TxPDO8*, Nd.51 |
| 970 | 0x3CA | TxPDO8*, Nd.10 | 991 | 0x3DF | TxPDO8*, Nd.31 | 1012 | 0x3F4 | TxPDO8*, Nd.52 |
| 971 | 0x3CB | TxPDO8*, Nd.11 | 992 | 0x3E0 | TxPDO8*, Nd.32 | 1013 | 0x3F5 | TxPDO8*, Nd.53 |
| 972 | 0x3CC | TxPDO8*, Nd.12 | 993 | 0x3E1 | TxPDO8*, Nd.33 | 1014 | 0x3F6 | TxPDO8*, Nd.54 |
| 973 | 0x3CD | TxPDO8*, Nd.13 | 994 | 0x3E2 | TxPDO8*, Nd.34 | 1015 | 0x3F7 | TxPDO8*, Nd.55 |
| 974 | 0x3CE | TxPDO8*, Nd.14 | 995 | 0x3E3 | TxPDO8*, Nd.35 | 1016 | 0x3F8 | TxPDO8*, Nd.56 |
| 975 | 0x3CF | TxPDO8*, Nd.15 | 996 | 0x3E4 | TxPDO8*, Nd.36 | 1017 | 0x3F9 | TxPDO8*, Nd.57 |
| 976 | 0x3D0 | TxPDO8*, Nd.16 | 997 | 0x3E5 | TxPDO8*, Nd.37 | 1018 | 0x3FA | TxPDO8*, Nd.58 |
| 977 | 0x3D1 | TxPDO8*, Nd.17 | 998 | 0x3E6 | TxPDO8*, Nd.38 | 1019 | 0x3FB | TxPDO8*, Nd.59 |
| 978 | 0x3D2 | TxPDO8*, Nd.18 | 999 | 0x3E7 | TxPDO8*, Nd.39 | 1020 | 0x3FC | TxPDO8*, Nd.60 |
| 979 | 0x3D3 | TxPDO8*, Nd.19 | 1000 | 0x3E8 | TxPDO8*, Nd.40 | 1021 | 0x3FD | TxPDO8*, Nd.61 |
| 980 | 0x3D4 | TxPDO8*, Nd.20 | 1001 | 0x3E9 | TxPDO8*, Nd.41 | 1022 | 0x3FE | TxPDO8*, Nd.62 |
| 981 | 0x3D5 | TxPDO8*, Nd.21 | 1002 | 0x3EA | TxPDO8*, Nd.42 | 1023 | 0x3FF | TxPDO8*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1089 | 0x441 | RxPDO8*, Nd.1 | 1110 | 0x456 | RxPDO8*, Nd.22 | 1131 | 0x46B | RxPDO8*, Nd.43 |
| 1090 | 0x442 | RxPDO8*, Nd.2 | 1111 | 0x457 | RxPDO8*, Nd.23 | 1132 | 0x46C | RxPDO8*, Nd.44 |
| 1091 | 0x443 | RxPDO8*, Nd.3 | 1112 | 0x458 | RxPDO8*, Nd.24 | 1133 | 0x46D | RxPDO8*, Nd.45 |
| 1092 | 0x444 | RxPDO8*, Nd.4 | 1113 | 0x459 | RxPDO8*, Nd.25 | 1134 | 0x46E | RxPDO8*, Nd.46 |
| 1093 | 0x445 | RxPDO8*, Nd.5 | 1114 | 0x45A | RxPDO8*, Nd.26 | 1135 | 0x46F | RxPDO8*, Nd.47 |
| 1094 | 0x446 | RxPDO8*, Nd.6 | 1115 | 0x45B | RxPDO8*, Nd.27 | 1136 | 0x470 | RxPDO8*, Nd.48 |
| 1095 | 0x447 | RxPDO8*, Nd.7 | 1116 | 0x45C | RxPDO8*, Nd.28 | 1137 | 0x471 | RxPDO8*, Nd.49 |
| 1096 | 0x448 | RxPDO8*, Nd.8 | 1117 | 0x45D | RxPDO8*, Nd.29 | 1138 | 0x472 | RxPDO8*, Nd.50 |
| 1097 | 0x449 | RxPDO8*, Nd.9 | 1118 | 0x45E | RxPDO8*, Nd.30 | 1139 | 0x473 | RxPDO8*, Nd.51 |
| 1098 | 0x44A | RxPDO8*, Nd.10 | 1119 | 0x45F | RxPDO8*, Nd.31 | 1140 | 0x474 | RxPDO8*, Nd.52 |
| 1099 | 0x44B | RxPDO8*, Nd.11 | 1120 | 0x460 | RxPDO8*, Nd.32 | 1141 | 0x475 | RxPDO8*, Nd.53 |
| 1100 | 0x44C | RxPDO8*, Nd.12 | 1121 | 0x461 | RxPDO8*, Nd.33 | 1142 | 0x476 | RxPDO8*, Nd.54 |
| 1101 | 0x44D | RxPDO8*, Nd.13 | 1122 | 0x462 | RxPDO8*, Nd.34 | 1143 | 0x477 | RxPDO8*, Nd.55 |
| 1102 | 0x44E | RxPDO8*, Nd.14 | 1123 | 0x463 | RxPDO8*, Nd.35 | 1144 | 0x478 | RxPDO8*, Nd.56 |
| 1103 | 0x44F | RxPDO8*, Nd.15 | 1124 | 0x464 | RxPDO8*, Nd.36 | 1145 | 0x479 | RxPDO8*, Nd.57 |
| 1104 | 0x450 | RxPDO8*, Nd.16 | 1125 | 0x465 | RxPDO8*, Nd.37 | 1146 | 0x47A | RxPDO8*, Nd.58 |
| 1105 | 0x451 | RxPDO8*, Nd.17 | 1126 | 0x466 | RxPDO8*, Nd.38 | 1147 | 0x47B | RxPDO8*, Nd.59 |
| 1106 | 0x452 | RxPDO8*, Nd.18 | 1127 | 0x467 | RxPDO8*, Nd.39 | 1148 | 0x47C | RxPDO8*, Nd.60 |
| 1107 | 0x453 | RxPDO8*, Nd.19 | 1128 | 0x468 | RxPDO8*, Nd.40 | 1149 | 0x47D | RxPDO8*, Nd.61 |
| 1108 | 0x454 | RxPDO8*, Nd.20 | 1129 | 0x469 | RxPDO8*, Nd.41 | 1150 | 0x47E | RxPDO8*, Nd.62 |
| 1109 | 0x455 | RxPDO8*, Nd.21 | 1130 | 0x46A | RxPDO8*, Nd.42 | 1151 | 0x47F | RxPDO8*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1217 | 0x4C1 | TxPDO9*, Nd.1 | 1238 | 0x4D6 | TxPDO9*, Nd.22 | 1259 | 0x4EB | TxPDO9*, Nd.43 |
| 1218 | 0x4C2 | TxPDO9*, Nd.2 | 1239 | 0x4D7 | TxPDO9*, Nd.23 | 1260 | 0x4EC | TxPDO9*, Nd.44 |
| 1219 | 0x4C3 | TxPDO9*, Nd.3 | 1240 | 0x4D8 | TxPDO9*, Nd.24 | 1261 | 0x4ED | TxPDO9*, Nd.45 |
| 1220 | 0x4C4 | TxPDO9*, Nd.4 | 1241 | 0x4D9 | TxPDO9*, Nd.25 | 1262 | 0x4EE | TxPDO9*, Nd.46 |
| 1221 | 0x4C5 | TxPDO9*, Nd.5 | 1242 | 0x4DA | TxPDO9*, Nd.26 | 1263 | 0x4EF | TxPDO9*, Nd.47 |
| 1222 | 0x4C6 | TxPDO9*, Nd.6 | 1243 | 0x4DB | TxPDO9*, Nd.27 | 1264 | 0x4F0 | TxPDO9*, Nd.48 |
| 1223 | 0x4C7 | TxPDO9*, Nd.7 | 1244 | 0x4DC | TxPDO9*, Nd.28 | 1265 | 0x4F1 | TxPDO9*, Nd.49 |
| 1224 | 0x4C8 | TxPDO9*, Nd.8 | 1245 | 0x4DD | TxPDO9*, Nd.29 | 1266 | 0x4F2 | TxPDO9*, Nd.50 |
| 1225 | 0x4C9 | TxPDO9*, Nd.9 | 1246 | 0x4DE | TxPDO9*, Nd.30 | 1267 | 0x4F3 | TxPDO9*, Nd.51 |
| 1226 | 0x4CA | TxPDO9*, Nd.10 | 1247 | 0x4DF | TxPDO9*, Nd.31 | 1268 | 0x4F4 | TxPDO9*, Nd.52 |
| 1227 | 0x4CB | TxPDO9*, Nd.11 | 1248 | 0x4E0 | TxPDO9*, Nd.32 | 1269 | 0x4F5 | TxPDO9*, Nd.53 |
| 1228 | 0x4CC | TxPDO9*, Nd.12 | 1249 | 0x4E1 | TxPDO9*, Nd.33 | 1270 | 0x4F6 | TxPDO9*, Nd.54 |
| 1229 | 0x4CD | TxPDO9*, Nd.13 | 1250 | 0x4E2 | TxPDO9*, Nd.34 | 1271 | 0x4F7 | TxPDO9*, Nd.55 |
| 1230 | 0x4CE | TxPDO9*, Nd.14 | 1251 | 0x4E3 | TxPDO9*, Nd.35 | 1272 | 0x4F8 | TxPDO9*, Nd.56 |
| 1231 | 0x4CF | TxPDO9*, Nd.15 | 1252 | 0x4E4 | TxPDO9*, Nd.36 | 1273 | 0x4F9 | TxPDO9*, Nd.57 |
| 1232 | 0x4D0 | TxPDO9*, Nd.16 | 1253 | 0x4E5 | TxPDO9*, Nd.37 | 1274 | 0x4FA | TxPDO9*, Nd.58 |
| 1233 | 0x4D1 | TxPDO9*, Nd.17 | 1254 | 0x4E6 | TxPDO9*, Nd.38 | 1275 | 0x4FB | TxPDO9*, Nd.59 |
| 1234 | 0x4D2 | TxPDO9*, Nd.18 | 1255 | 0x4E7 | TxPDO9*, Nd.39 | 1276 | 0x4FC | TxPDO9*, Nd.60 |
| 1235 | 0x4D3 | TxPDO9*, Nd.19 | 1256 | 0x4E8 | TxPDO9*, Nd.40 | 1277 | 0x4FD | TxPDO9*, Nd.61 |
| 1236 | 0x4D4 | TxPDO9*, Nd.20 | 1257 | 0x4E9 | TxPDO9*, Nd.41 | 1278 | 0x4FE | TxPDO9*, Nd.62 |
| 1237 | 0x4D5 | TxPDO9*, Nd.21 | 1258 | 0x4EA | TxPDO9*, Nd.42 | 1279 | 0x4FF | TxPDO9*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1345 | 0x541 | RxPDO9*, Nd.1 | 1366 | 0x556 | RxPDO9*, Nd.22 | 1387 | 0x56B | RxPDO9*, Nd.43 |
| 1346 | 0x542 | RxPDO9*, Nd.2 | 1367 | 0x557 | RxPDO9*, Nd.23 | 1388 | 0x56C | RxPDO9*, Nd.44 |
| 1347 | 0x543 | RxPDO9*, Nd.3 | 1368 | 0x558 | RxPDO9*, Nd.24 | 1389 | 0x56D | RxPDO9*, Nd.45 |
| 1348 | 0x544 | RxPDO9*, Nd.4 | 1369 | 0x559 | RxPDO9*, Nd.25 | 1390 | 0x56E | RxPDO9*, Nd.46 |
| 1349 | 0x545 | RxPDO9*, Nd.5 | 1370 | 0x55A | RxPDO9*, Nd.26 | 1391 | 0x56F | RxPDO9*, Nd.47 |
| 1350 | 0x546 | RxPDO9*, Nd.6 | 1371 | 0x55B | RxPDO9*, Nd.27 | 1392 | 0x570 | RxPDO9*, Nd.48 |
| 1351 | 0x547 | RxPDO9*, Nd.7 | 1372 | 0x55C | RxPDO9*, Nd.28 | 1393 | 0x571 | RxPDO9*, Nd.49 |
| 1352 | 0x548 | RxPDO9*, Nd.8 | 1373 | 0x55D | RxPDO9*, Nd.29 | 1394 | 0x572 | RxPDO9*, Nd.50 |
| 1353 | 0x549 | RxPDO9*, Nd.9 | 1374 | 0x55E | RxPDO9*, Nd.30 | 1395 | 0x573 | RxPDO9*, Nd.51 |
| 1354 | 0x54A | RxPDO9*, Nd.10 | 1375 | 0x55F | RxPDO9*, Nd.31 | 1396 | 0x574 | RxPDO9*, Nd.52 |
| 1355 | 0x54B | RxPDO9*, Nd.11 | 1376 | 0x560 | RxPDO9*, Nd.32 | 1397 | 0x575 | RxPDO9*, Nd.53 |
| 1356 | 0x54C | RxPDO9*, Nd.12 | 1377 | 0x561 | RxPDO9*, Nd.33 | 1398 | 0x576 | RxPDO9*, Nd.54 |
| 1357 | 0x54D | RxPDO9*, Nd.13 | 1378 | 0x562 | RxPDO9*, Nd.34 | 1399 | 0x577 | RxPDO9*, Nd.55 |
| 1358 | 0x54E | RxPDO9*, Nd.14 | 1379 | 0x563 | RxPDO9*, Nd.35 | 1400 | 0x578 | RxPDO9*, Nd.56 |
| 1359 | 0x54F | RxPDO9*, Nd.15 | 1380 | 0x564 | RxPDO9*, Nd.36 | 1401 | 0x579 | RxPDO9*, Nd.57 |
| 1360 | 0x550 | RxPDO9*, Nd.16 | 1381 | 0x565 | RxPDO9*, Nd.37 | 1402 | 0x57A | RxPDO9*, Nd.58 |
| 1361 | 0x551 | RxPDO9*, Nd.17 | 1382 | 0x566 | RxPDO9*, Nd.38 | 1403 | 0x57B | RxPDO9*, Nd.59 |
| 1362 | 0x552 | RxPDO9*, Nd.18 | 1383 | 0x567 | RxPDO9*, Nd.39 | 1404 | 0x57C | RxPDO9*, Nd.60 |
| 1363 | 0x553 | RxPDO9*, Nd.19 | 1384 | 0x568 | RxPDO9*, Nd.40 | 1405 | 0x57D | RxPDO9*, Nd.61 |
| 1364 | 0x554 | RxPDO9*, Nd.20 | 1385 | 0x569 | RxPDO9*, Nd.41 | 1406 | 0x57E | RxPDO9*, Nd.62 |
| 1365 | 0x555 | RxPDO9*, Nd.21 | 1386 | 0x56A | RxPDO9*, Nd.42 | 1407 | 0x57F | RxPDO9*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1473 | 0x5C1 | TxPDO10*, Nd.1 | 1494 | 0x5D6 | TxPDO10*, Nd.22 | 1515 | 0x5EB | TxPDO10*, Nd.43 |
| 1474 | 0x5C2 | TxPDO10*, Nd.2 | 1495 | 0x5D7 | TxPDO10*, Nd.23 | 1516 | 0x5EC | TxPDO10*, Nd.44 |
| 1475 | 0x5C3 | TxPDO10*, Nd.3 | 1496 | 0x5D8 | TxPDO10*, Nd.24 | 1517 | 0x5ED | TxPDO10*, Nd.45 |
| 1476 | 0x5C4 | TxPDO10*, Nd.4 | 1497 | 0x5D9 | TxPDO10*, Nd.25 | 1518 | 0x5EE | TxPDO10*, Nd.46 |
| 1477 | 0x5C5 | TxPDO10*, Nd.5 | 1498 | 0x5DA | TxPDO10*, Nd.26 | 1519 | 0x5EF | TxPDO10*, Nd.47 |
| 1478 | 0x5C6 | TxPDO10*, Nd.6 | 1499 | 0x5DB | TxPDO10*, Nd.27 | 1520 | 0x5F0 | TxPDO10*, Nd.48 |
| 1479 | 0x5C7 | TxPDO10*, Nd.7 | 1500 | 0x5DC | TxPDO10*, Nd.28 | 1521 | 0x5F1 | TxPDO10*, Nd.49 |
| 1480 | 0x5C8 | TxPDO10*, Nd.8 | 1501 | 0xDE | TxPDO10*, Nd.29 | 1522 | 0x5F2 | TxPDO10*, Nd.50 |
| 1481 | 0x5C9 | TxPDO10*, Nd.9 | 1502 | 0x5DE | TxPDO10*, Nd.30 | 1523 | 0x5F3 | TxPDO10*, Nd.51 |
| 1482 | 0x5CA | TxPDO10*, Nd.10 | 1503 | 0x5DF | TxPDO10*, Nd.31 | 1524 | 0x5F4 | TxPDO10*, Nd.52 |
| 1483 | 0x5CB | TxPDO10*, Nd.11 | 1504 | 0x5E0 | TxPDO10*, Nd.32 | 1525 | 0x5F5 | TxPDO10*, Nd.53 |
| 1484 | 0x5CC | TxPDO10*, Nd.12 | 1505 | 0x5E1 | TxPDO10*, Nd.33 | 1526 | 0x5F6 | TxPDO10*, Nd.54 |
| 1485 | 0x5CD | TxPDO10*, Nd.13 | 1506 | 0x5E2 | TxPDO10*, Nd.34 | 1527 | 0x5F7 | TxPDO10*, Nd.55 |
| 1486 | 0x5CE | TxPDO10*, Nd.14 | 1507 | 0x5E3 | TxPDO10*, Nd.35 | 1528 | 0x5F8 | TxPDO10*, Nd.56 |
| 1487 | 0x5CF | TxPDO10*, Nd.15 | 1508 | 0x5E4 | TxPDO10*, Nd.36 | 1529 | 0x5F9 | TxPDO10*, Nd.57 |
| 1488 | 0x5D0 | TxPDO10*, Nd.16 | 1509 | 0x5E5 | TxPDO10*, Nd.37 | 1530 | 0x5FA | TxPDO10*, Nd.58 |
| 1489 | 0x5D1 | TxPDO10*, Nd.17 | 1510 | 0x5E6 | TxPDO10*, Nd.38 | 1531 | 0x5FB | TxPDO10*, Nd.59 |
| 1490 | 0x5D2 | TxPDO10*, Nd.18 | 1511 | 0x5E7 | TxPDO10*, Nd.39 | 1532 | 0x5FC | TxPDO10*, Nd.60 |
| 1491 | 0x5D3 | TxPDO10*, Nd.19 | 1512 | 0x5E8 | TxPDO10*, Nd.40 | 1533 | 0x5FD | TxPDO10*, Nd.61 |
| 1492 | 0x5D4 | TxPDO10*, Nd.20 | 1513 | 0x5E9 | TxPDO10*, Nd.41 | 1534 | 0x5FE | TxPDO10*, Nd.62 |
| 1493 | 0x5D5 | TxPDO10*, Nd.21 | 1514 | 0x5EA | TxPDO10*, Nd.42 | 1535 | 0x5FF | TxPDO10*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1601 | 0x641 | RxPDO10*, Nd.1 | 1622 | 0x656 | RxPDO10*, Nd.22 | 1643 | 0x66B | RxPDO10*, Nd.43 |
| 1602 | 0x642 | RxPDO10*, Nd.2 | 1623 | 0x657 | RxPDO10*, Nd.23 | 1644 | 0x66C | RxPDO10*, Nd.44 |
| 1603 | 0x643 | RxPDO10*, Nd.3 | 1624 | 0x658 | RxPDO10*, Nd.24 | 1645 | 0x66D | RxPDO10*, Nd.45 |
| 1604 | 0x644 | RxPDO10*, Nd.4 | 1625 | 0x659 | RxPDO10*, Nd.25 | 1646 | 0x66E | RxPDO10*, Nd.46 |
| 1605 | 0x645 | RxPDO10*, Nd.5 | 1626 | 0x65A | RxPDO10*, Nd.26 | 1647 | 0x66F | RxPDO10*, Nd.47 |
| 1606 | 0x646 | RxPDO10*, Nd.6 | 1627 | 0x65B | RxPDO10*, Nd.27 | 1648 | 0x670 | RxPDO10*, Nd.48 |
| 1607 | 0x647 | RxPDO10*, Nd.7 | 1628 | 0x65C | RxPDO10*, Nd.28 | 1649 | 0x671 | RxPDO10*, Nd.49 |
| 1608 | 0x648 | RxPDO10*, Nd.8 | 1629 | 0x65D | RxPDO10*, Nd.29 | 1650 | 0x672 | RxPDO10*, Nd.50 |
| 1609 | 0x649 | RxPDO10*, Nd.9 | 1630 | 0x65E | RxPDO10*, Nd.30 | 1651 | 0x673 | RxPDO10*, Nd.51 |
| 1610 | 0x64A | RxPDO10*, Nd.10 | 1631 | 0x65F | RxPDO10*, Nd.31 | 1652 | 0x674 | RxPDO10*, Nd.52 |
| 1611 | 0x64B | RxPDO10*, Nd.11 | 1632 | 0x660 | RxPDO10*, Nd.32 | 1653 | 0x675 | RxPDO10*, Nd.53 |
| 1612 | 0x64C | RxPDO10*, Nd.12 | 1633 | 0x661 | RxPDO10*, Nd.33 | 1654 | 0x676 | RxPDO10*, Nd.54 |
| 1613 | 0x64D | RxPDO10*, Nd.13 | 1634 | 0x662 | RxPDO10*, Nd.34 | 1655 | 0x677 | RxPDO10*, Nd.55 |
| 1614 | 0x64E | RxPDO10*, Nd.14 | 1635 | 0x663 | RxPDO10*, Nd.35 | 1656 | 0x678 | RxPDO10*, Nd.56 |
| 1615 | 0x64F | RxPDO10*, Nd.15 | 1636 | 0x664 | RxPDO10*, Nd.36 | 1657 | 0x679 | RxPDO10*, Nd.57 |
| 1616 | 0x650 | RxPDO10*, Nd.16 | 1637 | 0x665 | RxPDO10*, Nd.37 | 1658 | 0x67A | RxPDO10*, Nd.58 |
| 1617 | 0x651 | RxPDO10*, Nd.17 | 1638 | 0x666 | RxPDO10*, Nd.38 | 1659 | 0x67B | RxPDO10*, Nd.59 |
| 1618 | 0x652 | RxPDO10*, Nd.18 | 1639 | 0x667 | RxPDO10*, Nd.39 | 1660 | 0x67C | RxPDO10*, Nd.60 |
| 1619 | 0x653 | RxPDO10*, Nd.19 | 1640 | 0x668 | RxPDO10*, Nd.40 | 1661 | 0x67D | RxPDO10*, Nd.61 |
| 1620 | 0x654 | RxPDO10*, Nd.20 | 1641 | 0x669 | RxPDO10*, Nd.41 | 1662 | 0x67E | RxPDO10*, Nd.62 |
| 1621 | 0x655 | RxPDO10*, Nd.21 | 1642 | 0x66A | RxPDO10*, Nd.42 | 1663 | 0x67F | RxPDO10*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1729 | 0x6C1 | TxPDO11*, Nd.1 | 1750 | 0x6D6 | TxPDO11*, Nd.22 | 1771 | 0x6EB | TxPDO11*, Nd.43 |
| 1730 | 0x6C2 | TxPDO11*, Nd.2 | 1751 | 0x6D7 | TxPDO11*, Nd.23 | 1772 | 0x6EC | TxPDO11*, Nd.44 |
| 1731 | 0x6C3 | TxPDO11*, Nd.3 | 1752 | 0x6D8 | TxPDO11*, Nd.24 | 1773 | 0x6ED | TxPDO11*, Nd.45 |
| 1732 | 0x6C4 | TxPDO11*, Nd.4 | 1753 | 0x6D9 | TxPDO11*, Nd.25 | 1774 | 0x6EE | TxPDO11*, Nd.46 |
| 1733 | 0x6C5 | TxPDO11*, Nd.5 | 1754 | 0x6DA | TxPDO11*, Nd.26 | 1775 | 0x6EF | TxPDO11*, Nd.47 |
| 1734 | 0x6C6 | TxPDO11*, Nd.6 | 1755 | 0x6DB | TxPDO11*, Nd.27 | 1776 | 0x6F0 | TxPDO11*, Nd.48 |
| 1735 | 0x6C7 | TxPDO11*, Nd.7 | 1756 | 0x6DC | TxPDO11*, Nd.28 | 1777 | 0x6F1 | TxPDO11*, Nd.49 |
| 1736 | 0x6C8 | TxPDO11*, Nd.8 | 1757 | 0x6DD | TxPDO11*, Nd.29 | 1778 | 0x6F2 | TxPDO11*, Nd.50 |
| 1737 | 0x6C9 | TxPDO11*, Nd.9 | 1758 | 0x6DE | TxPDO11*, Nd.30 | 1779 | 0x6F3 | TxPDO11*, Nd.51 |
| 1738 | 0x6CA | TxPDO11*, Nd.10 | 1759 | 0x6DF | TxPDO11*, Nd.31 | 1780 | 0x6F4 | TxPDO11*, Nd.52 |
| 1739 | 0x6CB | TxPDO11*, Nd.11 | 1760 | 0x6E0 | TxPDO11*, Nd.32 | 1781 | 0x6F5 | TxPDO11*, Nd.53 |
| 1740 | 0x6CC | TxPDO11*, Nd.12 | 1761 | 0x6E1 | TxPDO11*, Nd.33 | 1782 | 0x6F6 | TxPDO11*, Nd.54 |
| 1741 | 0x6CD | TxPDO11*, Nd.13 | 1762 | 0x6E2 | TxPDO11*, Nd.34 | 1783 | 0x6F7 | TxPDO11*, Nd.55 |
| 1742 | 0x6CE | TxPDO11*, Nd.14 | 1763 | 0x6E3 | TxPDO11*, Nd.35 | 1784 | 0x6F8 | TxPDO11*, Nd.56 |
| 1743 | 0x6CF | TxPDO11*, Nd.15 | 1764 | 0x6E4 | TxPDO11*, Nd.36 | 1785 | 0x6F9 | TxPDO11*, Nd.57 |
| 1744 | 0x6D0 | TxPDO11*, Nd.16 | 1765 | 0x6E5 | TxPDO11*, Nd.37 | 1786 | 0x6FA | TxPDO11*, Nd.58 |
| 1745 | 0x6D1 | TxPDO11*, Nd.17 | 1766 | 0x6E6 | TxPDO11*, Nd.38 | 1787 | 0x6FB | TxPDO11*, Nd.59 |
| 1746 | 0x6D2 | TxPDO11*, Nd.18 | 1767 | 0x6E7 | TxPDO11*, Nd.39 | 1788 | 0x6FC | TxPDO11*, Nd.60 |
| 1747 | 0x6D3 | TxPDO11*, Nd.19 | 1768 | 0x6E8 | TxPDO11*, Nd.40 | 1789 | 0x6FD | TxPDO11*, Nd.61 |
| 1748 | 0x6D4 | TxPDO11*, Nd.20 | 1769 | 0x6E9 | TxPDO11*, Nd.41 | 1790 | 0x6FE | TxPDO11*, Nd.62 |
| 1749 | 0x6D5 | TxPDO11*, Nd.21 | 1770 | 0x6EA | TxPDO11*, Nd.42 | 1791 | 0x6FF | TxPDO11*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1857 | 0x741 | RxPDO11*, Nd.1 | 1878 | 0x756 | RxPDO11*, Nd.22 | 1899 | 0x76B | RxPDO11*, Nd.43 |
| 1858 | 0x742 | RxPDO11*, Nd.2 | 1879 | 0x757 | RxPDO11*, Nd.23 | 1900 | 0x76C | RxPDO11*, Nd.44 |
| 1859 | 0x743 | RxPDO11*, Nd.3 | 1880 | 0x758 | RxPDO11*, Nd.24 | 1901 | 0x76D | RxPDO11*, Nd.45 |
| 1860 | 0x744 | RxPDO11*, Nd.4 | 1881 | 0x759 | RxPDO11*, Nd.25 | 1902 | 0x76E | RxPDO11*, Nd.46 |
| 1861 | 0x745 | RxPDO11*, Nd.5 | 1882 | 0x75A | RxPDO11*, Nd.26 | 1903 | 0x76F | RxPDO11*, Nd.47 |
| 1862 | 0x746 | RxPDO11*, Nd.6 | 1883 | 0x75B | RxPDO11*, Nd.27 | 1904 | 0x770 | RxPDO11*, Nd.48 |
| 1863 | 0x747 | RxPDO11*, Nd.7 | 1884 | 0x75C | RxPDO11*, Nd.28 | 1905 | 0x771 | RxPDO11*, Nd.49 |
| 1864 | 0x748 | RxPDO11*, Nd.8 | 1885 | 0x75D | RxPDO11*, Nd.29 | 1906 | 0x772 | RxPDO11*, Nd.50 |
| 1865 | 0x749 | RxPDO11*, Nd.9 | 1886 | 0x75E | RxPDO11*, Nd.30 | 1907 | 0x773 | RxPDO11*, Nd.51 |
| 1866 | 0x74A | RxPDO11*, Nd.10 | 1887 | 0x75F | RxPDO11*, Nd.31 | 1908 | 0x774 | RxPDO11*, Nd.52 |
| 1867 | 0x74B | RxPDO11*, Nd.11 | 1888 | 0x760 | RxPDO11*, Nd.32 | 1909 | 0x775 | RxPDO11*, Nd.53 |
| 1868 | 0x74C | RxPDO11*, Nd.12 | 1889 | 0x761 | RxPDO11*, Nd.33 | 1910 | 0x776 | RxPDO11*, Nd.54 |
| 1869 | 0x74D | RxPDO11*, Nd.13 | 1890 | 0x762 | RxPDO11*, Nd.34 | 1911 | 0x777 | RxPDO11*, Nd.55 |
| 1870 | 0x74E | RxPDO11*, Nd.14 | 1891 | 0x763 | RxPDO11*, Nd.35 | 1912 | 0x778 | RxPDO11*, Nd.56 |
| 1871 | 0x74F | RxPDO11*, Nd.15 | 1892 | 0x764 | RxPDO11*, Nd.36 | 1913 | 0x779 | RxPDO11*, Nd.57 |
| 1872 | 0x750 | RxPDO11*, Nd.16 | 1893 | 0x765 | RxPDO11*, Nd.37 | 1914 | 0x77A | RxPDO11*, Nd.58 |
| 1873 | 0x751 | RxPDO11*, Nd.17 | 1894 | 0x766 | RxPDO11*, Nd.38 | 1915 | 0x77B | RxPDO11*, Nd.59 |
| 1874 | 0x752 | RxPDO11*, Nd.18 | 1895 | 0x767 | RxPDO11*, Nd.39 | 1916 | 0x77C | RxPDO11*, Nd.60 |
| 1875 | 0x753 | RxPDO11*, Nd.19 | 1896 | 0x768 | RxPDO11*, Nd.40 | 1917 | 0x77D | RxPDO11*, Nd.61 |
| 1876 | 0x754 | RxPDO11*, Nd.20 | 1897 | 0x769 | RxPDO11*, Nd.41 | 1918 | 0x77E | RxPDO11*, Nd.62 |
| 1877 | 0x755 | RxPDO11*, Nd.21 | 1898 | 0x76A | RxPDO11*, Nd.42 | 1919 | 0x77F | RxPDO11*, Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1409 | 0x581 | SDO Tx Nd.1 | 1430 | 0x596 | SDO Tx Nd.22 | 1451 | 0x5AB | SDO Tx Nd.43 |
| 1410 | 0x582 | SDO Tx Nd.2 | 1431 | 0x597 | SDO Tx Nd.23 | 1452 | 0x5AC | SDO Tx Nd.44 |
| 1411 | 0x583 | SDO Tx Nd.3 | 1432 | 0x598 | SDO Tx Nd.24 | 1453 | 0x5AD | SDO Tx Nd.45 |
| 1412 | 0x584 | SDO Tx Nd.4 | 1433 | 0x599 | SDO Tx Nd.25 | 1454 | 0x5AE | SDO Tx Nd.46 |
| 1413 | 0x585 | SDO Tx Nd.5 | 1434 | 0x59A | SDO Tx Nd.26 | 1455 | 0x5AF | SDO Tx Nd.47 |
| 1414 | 0x586 | SDO Tx Nd.6 | 1435 | 0x59B | SDO Tx Nd.27 | 1456 | 0x5B0 | SDO Tx Nd.48 |
| 1415 | 0x587 | SDO Tx Nd.7 | 1436 | 0x59C | SDO Tx Nd.28 | 1457 | 0x5B1 | SDO Tx Nd.49 |
| 1416 | 0x588 | SDO Tx Nd.8 | 1437 | 0x59D | SDO Tx Nd.29 | 1458 | 0x5B2 | SDO Tx Nd.50 |
| 1417 | 0x589 | SDO Tx Nd.9 | 1438 | 0x59E | SDO Tx Nd.30 | 1459 | 0x5B3 | SDO Tx Nd.51 |
| 1418 | 0x58A | SDO Tx Nd.10 | 1439 | 0x59F | SDO Tx Nd.31 | 1460 | 0x5B4 | SDO Tx Nd.52 |
| 1419 | 0x58B | SDO Tx Nd.11 | 1440 | 0x5A0 | SDO Tx Nd.32 | 1461 | 0x5B5 | SDO Tx Nd.53 |
| 1420 | 0x58C | SDO Tx Nd.12 | 1441 | 0x5A1 | SDO Tx Nd.33 | 1462 | 0x5B6 | SDO Tx Nd.54 |
| 1421 | 0x58D | SDO Tx Nd.13 | 1442 | 0x5A2 | SDO Tx Nd.34 | 1463 | 0x5B7 | SDO Tx Nd.55 |
| 1422 | 0x58E | SDO Tx Nd.14 | 1443 | 0x5A3 | SDO Tx Nd.35 | 1464 | 0x5B8 | SDO Tx Nd.56 |
| 1423 | 0x58F | SDO Tx Nd.15 | 1444 | 0x5A4 | SDO Tx Nd.36 | 1465 | 0x5B9 | SDO Tx Nd.57 |
| 1424 | 0x590 | SDO Tx Nd.16 | 1445 | 0x5A5 | SDO Tx Nd.37 | 1466 | 0x5BA | SDO Tx Nd.58 |
| 1425 | 0x591 | SDO Tx Nd.17 | 1446 | 0x5A6 | SDO Tx Nd.38 | 1467 | 0x5BB | SDO Tx Nd.59 |
| 1426 | 0x592 | SDO Tx Nd.18 | 1447 | 0x5A7 | SDO Tx Nd.39 | 1468 | 0x5BC | SDO Tx Nd.60 |
| 1427 | 0x593 | SDO Tx Nd.19 | 1448 | 0x5A8 | SDO Tx Nd.40 | 1469 | 0x5BD | SDO Tx Nd.61 |
| 1428 | 0x594 | SDO Tx Nd.20 | 1449 | 0x5A9 | SDO Tx Nd.41 | 1470 | 0x5BE | SDO Tx Nd.62 |
| 1429 | 0x595 | SDO Tx Nd.21 | 1450 | 0x5AA | SDO Tx Nd.42 | 1471 | 0x5BF | SDO Tx Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1537 | 0x601 | SDO Rx Nd.1 | 1558 | 0x616 | SDO Rx Nd.22 | 1579 | 0x62B | SDO Rx Nd.43 |
| 1538 | 0x602 | SDO Rx Nd.2 | 1559 | 0x617 | SDO Rx Nd.23 | 1580 | 0x62C | SDO Rx Nd.44 |
| 1539 | 0x603 | SDO Rx Nd.3 | 1560 | 0x618 | SDO Rx Nd.24 | 1581 | 0x62D | SDO Rx Nd.45 |
| 1540 | 0x604 | SDO Rx Nd.4 | 1561 | 0x619 | SDO Rx Nd.25 | 1582 | 0x62E | SDO Rx Nd.46 |
| 1541 | 0x605 | SDO Rx Nd.5 | 1562 | 0x61A | SDO Rx Nd.26 | 1583 | 0x62F | SDO Rx Nd.47 |
| 1542 | 0x606 | SDO Rx Nd.6 | 1563 | 0x61B | SDO Rx Nd.27 | 1584 | 0x630 | SDO Rx Nd.48 |
| 1543 | 0x607 | SDO Rx Nd.7 | 1564 | 0x61C | SDO Rx Nd.28 | 1585 | 0x631 | SDO Rx Nd.49 |
| 1544 | 0x608 | SDO Rx Nd.8 | 1565 | 0x61D | SDO Rx Nd.29 | 1586 | 0x632 | SDO Rx Nd.50 |
| 1545 | 0x609 | SDO Rx Nd.9 | 1566 | 0x61E | SDO Rx Nd.30 | 1587 | 0x633 | SDO Rx Nd.51 |
| 1546 | 0x60A | SDO Rx Nd.10 | 1567 | 0x61F | SDO Rx Nd.31 | 1588 | 0x634 | SDO Rx Nd.52 |
| 1547 | 0x60B | SDO Rx Nd.11 | 1568 | 0x620 | SDO Rx Nd.32 | 1589 | 0x635 | SDO Rx Nd.53 |
| 1548 | 0x60C | SDO Rx Nd.12 | 1569 | 0x621 | SDO Rx Nd.33 | 1590 | 0x636 | SDO Rx Nd.54 |
| 1549 | 0x60D | SDO Rx Nd.13 | 1570 | 0x622 | SDO Rx Nd.34 | 1591 | 0x637 | SDO Rx Nd.55 |
| 1550 | 0x60E | SDO Rx Nd.14 | 1571 | 0x623 | SDO Rx Nd.35 | 1592 | 0x638 | SDO Rx Nd.56 |
| 1551 | 0x60F | SDO Rx Nd.15 | 1572 | 0x624 | SDO Rx Nd.36 | 1593 | 0x639 | SDO Rx Nd.57 |
| 1552 | 0x610 | SDO Rx Nd.16 | 1573 | 0x625 | SDO Rx Nd.37 | 1594 | 0x63A | SDO Rx Nd.58 |
| 1553 | 0x611 | SDO Rx Nd.17 | 1574 | 0x626 | SDO Rx Nd.38 | 1595 | 0x63B | SDO Rx Nd.59 |
| 1554 | 0x612 | SDO Rx Nd.18 | 1575 | 0x627 | SDO Rx Nd.39 | 1596 | 0x63C | SDO Rx Nd.60 |
| 1555 | 0x613 | SDO Rx Nd.19 | 1576 | 0x628 | SDO Rx Nd.40 | 1597 | 0x63D | SDO Rx Nd.61 |
| 1556 | 0x614 | SDO Rx Nd.20 | 1577 | 0x629 | SDO Rx Nd.41 | 1598 | 0x63E | SDO Rx Nd.62 |
| 1557 | 0x615 | SDO Rx Nd.21 | 1578 | 0x62A | SDO Rx Nd.42 | 1599 | 0x63F | SDO Rx Nd.63 |

| dec | hex | Telegram type | dec | hex | Telegram type | dec | hex | Telegram type |
|---|---|---|---|---|---|---|---|---|
| 1793 | 0x701 | Guarding Nd.1 | 1814 | 0x716 | Guarding Nd.22 | 1835 | 0x72B | Guarding Nd.43 |
| 1794 | 0x702 | Guarding Nd.2 | 1815 | 0x717 | Guarding Nd.23 | 1836 | 0x72C | Guarding Nd.44 |
| 1795 | 0x703 | Guarding Nd.3 | 1816 | 0x718 | Guarding Nd.24 | 1837 | 0x72D | Guarding Nd.45 |
| 1796 | 0x704 | Guarding Nd.4 | 1817 | 0x719 | Guarding Nd.25 | 1838 | 0x72E | Guarding Nd.46 |
| 1797 | 0x705 | Guarding Nd.5 | 1818 | 0x71A | Guarding Nd.26 | 1839 | 0x72F | Guarding Nd.47 |
| 1798 | 0x706 | Guarding Nd.6 | 1819 | 0x71B | Guarding Nd.27 | 1840 | 0x730 | Guarding Nd.48 |
| 1799 | 0x707 | Guarding Nd.7 | 1820 | 0x71C | Guarding Nd.28 | 1841 | 0x731 | Guarding Nd.49 |
| 1800 | 0x708 | Guarding Nd.8 | 1821 | 0x71D | Guarding Nd.29 | 1842 | 0x732 | Guarding Nd.50 |
| 1801 | 0x709 | Guarding Nd.9 | 1822 | 0x71E | Guarding Nd.30 | 1843 | 0x733 | Guarding Nd.51 |
| 1802 | 0x70A | Guarding Nd.10 | 1823 | 0x71F | Guarding Nd.31 | 1844 | 0x734 | Guarding Nd.52 |
| 1803 | 0x70B | Guarding Nd.11 | 1824 | 0x720 | Guarding Nd.32 | 1845 | 0x735 | Guarding Nd.53 |
| 1804 | 0x70C | Guarding Nd.12 | 1825 | 0x721 | Guarding Nd.33 | 1846 | 0x736 | Guarding Nd.54 |
| 1805 | 0x70D | Guarding Nd.13 | 1826 | 0x722 | Guarding Nd.34 | 1847 | 0x737 | Guarding Nd.55 |
| 1806 | 0x70E | Guarding Nd.14 | 1827 | 0x723 | Guarding Nd.35 | 1848 | 0x738 | Guarding Nd.56 |
| 1807 | 0x70F | Guarding Nd.15 | 1828 | 0x724 | Guarding Nd.36 | 1849 | 0x739 | Guarding Nd.57 |
| 1808 | 0x710 | Guarding Nd.16 | 1829 | 0x725 | Guarding Nd.37 | 1850 | 0x73A | Guarding Nd.58 |
| 1809 | 0x711 | Guarding Nd.17 | 1830 | 0x726 | Guarding Nd.38 | 1851 | 0x73B | Guarding Nd.59 |
| 1810 | 0x712 | Guarding Nd.18 | 1831 | 0x727 | Guarding Nd.39 | 1852 | 0x73C | Guarding Nd.60 |
| 1811 | 0x713 | Guarding Nd.19 | 1832 | 0x728 | Guarding Nd.40 | 1853 | 0x73D | Guarding Nd.61 |
| 1812 | 0x714 | Guarding Nd.20 | 1833 | 0x729 | Guarding Nd.41 | 1854 | 0x73E | Guarding Nd.62 |
| 1813 | 0x715 | Guarding Nd.21 | 1834 | 0x72A | Guarding Nd.42 | 1855 | 0x73F | Guarding Nd.63 |

## 9.3 General operating conditions

The following conditions must be met in order to ensure flawless operation of the fieldbus components.

### Environmental conditions

### Operation

The components may not be used without additional protection in the following locations:

- in difficult environments, such as where there are corrosive vapors or gases, or high dust levels
- in the presence of high levels of ionizing radiation

| Condition | Permissible range |
|---|---|
| Permissible ambient temperature during operation | see technical data |
| Installation position | variable |
| Vibration resistance | conforms to EN 60068-2-6 |
| Shock resistance | conforms to EN 60068-2-27 |
| EMC immunity | conforms to EN 61000-6-2 |
| Emission | conforms to EN 61000-6-4 |

### Transport and storage

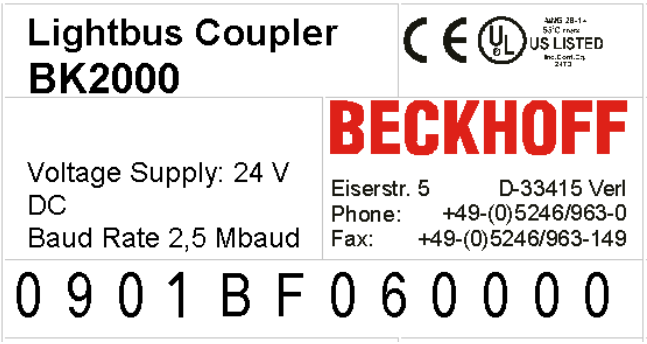
| Condition | Permissible range |
|---|---|
| Permissible ambient temperature during storage | -25°C... +85°C |
| Relative humidity | 95 %, no condensation |
| Free fall | up to 1 m in the original packaging |

### Protection classes and types

| Condition | Permissible range |
|---|---|
| Protection class in accordance with IEC 536 (VDE 0106, Part 1) | A protective conductor connection to the profile rail is necessary! |
| Protection class conforms to IEC 529 | IP20 (protection against contact with a standard test finger) |
| Protection against foreign objects | Less than 12 mm in diameter |
| Protection against water | no protection |

### Component identification

Every supplied component includes an adhesive label providing information about the product's approvals. For sample, on the BK2000 Bus Coupler:




The following information is printed on the label:

| Printed item | Meaning for this label |
|---|---|
| Precise product identification | Lightbus Coupler BK2000 |
| Supply voltage Us | 24 $V_{DC}$ (Use a 4 A fuse or a Class 2 power supply to meet UL requirements) |
| Data transfer rate | 2.5 Mbaud |
| Manufacturer | Beckhoff Automation GmbH & Co. KG |
| CE mark | Conformity mark |
| UL mark<br>  | Mark for UL approval. UL stands for the Underwriters Laboratories Inc., the leading certification organization for North America, based in the USA.<br>C = Canada, US = USA,<br>UL file number: E172151 |
| Production identification | From left to right, this sequence of characters indicates the week of production (2 characters), the year of production (2 characters), the software version (2 characters) and hardware version (2 characters), along with any special indications (4 characters).<br>In this case the device is a BK2000<br>- produced in calendar week 9<br>- of 2001<br>- with firmware version BF<br>- in hardware version 6<br>- without special designation |

## 9.4 Approvals

### Conformity mark

CE

### Protection class

IP20 conforms to EN60529

## 9.5 Test standards for device testing

### EMC

### EMC immunity

EN 61000-6-2

### Electromagnetic emission

EN 61000-6-4

### Vibration / shock resistance

### Vibration resistance

EN 60068-2-6

### Shock resistance

EN 60068-2-27

## 9.6 Bibliography

### German books


- Holger Zeltwander (ed.):
  **CANopen**,
  VDE Verlag, 2001, 197 pages,
  ISBN 3-800-72448-0

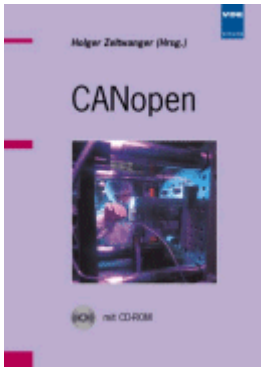




Fig. 91: CANopen

- Konrad Etschberger:
  **Controller Area Network**, Grundlagen, Protokolle, Bausteine, Anwendungen (**Controller Area Network**, principles, protocols, devices, applications), in German.
  Hanser Verlag, 2000. 431 pages.
  ISBN 3-446-19431-2

**General fieldbus technology**

- Gerhard Gruhler (ed.):
  **Feldbusse und Geräte-Kommunikationssysteme**, Praktisches Know-How mit Vergleichsmöglichkeiten, (**Fieldbuses and device communication systems**, practical knowledge, with comparison options), in German.
  Franzis Verlag 2001. 244 pages.
  ISBN 3-7723-5745-8

**English books**

- Konrad Etschberger:
  **Controller Area Network**,
  Ixxat Press, 2001. 440 pages.
  ISBN 3-00-007376-0
- M. Farsi, M. Barbosa:
  **CANopen Implementation**,
  RSP 2000. 210 pages.
  ISBN 0-86380-247-8

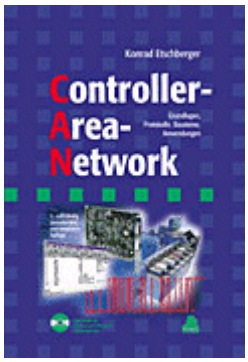



Fig. 92: CAN - Controller Area Network

**Standards**

- ISO 11898:
  Road Vehicles - Interchange of digital information - Controller Area Network (CAN) for high speed communication.
- CiA DS 301:
  CANopen Application Layer and Communication Profile.
  Available from the CAN in Automation Association.
- CiA DS 401:
  CANopen Device Profile for Generic E/A Modules.
  Available from the CAN in Automation Association.

## 9.7 List of Abbreviations

**CAN**

Controller Area Network. Serial bus system standardized in ISO 11898 that is used as the basic technology for CANopen

**CiA**

CAN in Automation e.V.. An international association of manufacturers and users based in Erlangen, Germany.

**COB**

Communication Object. A CAN telegram with up to 8 data bytes.

**COB-ID**

Communication Object Identifier. Telegram address (not to be confused with the node address). CANopen uses the 11-bit identifier according to CAN 2.0A.

**NMT**

Network Management. One of the service primitives of the CANopen specification. Network management is used to initialize the network and to monitor nodes.

**PDO**

Process Data Object. A CAN telegram for the transfer of process data (e.g. I/O data).

**RxPDO**

Receive PDO. PDOs are always identified from the point of view of the device under consideration. Thus a TxPDO with input data from an I/O module becomes an RxPDO from the controller's point of view.

**SDO**

Service Data Object. A CAN telegram with a protocol for communication with data in the object directory (typically parameter data).

**TxPDO**

Transmit PDO (named from the point of view of the CAN node).

## 9.8 Support and Service

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

**Beckhoff's branch offices and representatives**

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

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

You will also find further documentation for Beckhoff components there.

**Beckhoff Headquarters**

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

| | |
|---|---|
| Phone: | +49(0)5246/963-0 |
| Fax: | +49(0)5246/963-198 |
| e-mail: | info@beckhoff.com |

**Beckhoff Support**

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

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

| | |
|---|---|
| Hotline: | +49(0)5246/963-157 |
| Fax: | +49(0)5246/963-9157 |
| e-mail: | support@beckhoff.com |

**Beckhoff Service**

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

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

| | |
|---|---|
| Hotline: | +49(0)5246/963-460 |
| Fax: | +49(0)5246/963-479 |
| e-mail: | service@beckhoff.com |

# List of Illustrations