# **BECKHOFF** New Automation Technology

Manual | EN

CX7051

Embedded PC for CANopen responder (slave)





# **Table of contents**

| 1 | Note  | s on the   | documentationdocumentation                  | 7   |
|---|-------|------------|---|-----|
|   | 1.1   | Represe    | entation and structure of warnings          | 8   |
|   | 1.2   | Docume     | entation issue status                       | 9   |
| 2 | For y | our safe   | ty  | 10  |
|   | 2.1   | Intende    | d use                                       | 10  |
|   | 2.2   | Staff qu   | alification                                 | 10  |
|   | 2.3   | Safety ii  | nstructions                                 | 10  |
|   | 2.4   | Notes o    | n information security                      | 11  |
| 3 | Tran  | sport and  | d storage                                   | 12  |
| 4 | Prod  | luct over  | view  | 13  |
|   | 4.1   | Structur   | e   | 14  |
|   | 4.2   | Name p     | late  | 15  |
|   | 4.3   | Etherne    | t interface (X001)                          | 16  |
|   | 4.4   | USB into   | erface (X002)                               | 18  |
|   | 4.5   | D-sub c    | onnector (X003)                             | 18  |
|   | 4.6   | MicroSE    | card  | 20  |
|   | 4.7   | CANope     | en system overview                          | 21  |
|   |       | 4.7.1      | Network Management                          | 22  |
|   |       | 4.7.2      | Process Data Objects (PDO)                  | 26  |
|   |       | 4.7.3      | PDO Parameterization                        |     |
|   |       | 4.7.4      | Service Data Objects (SDO)                  | 35  |
|   |       | 4.7.5      | Objekt dictionary                           | 38  |
| 5 | Com   | missioni   | ng  | 81  |
|   | 5.1   | Mountin    | g   | 81  |
|   |       | 5.1.1      | Note the permissible installation positions | 81  |
|   |       | 5.1.2      | Fastening to the DIN rail                   | 83  |
|   |       | 5.1.3      | Changing the MicroSD card                   | 84  |
|   |       | 5.1.4      | Installing passive EtherCAT Terminals       | 85  |
|   | 5.2   | Power s    | upply                                       | 86  |
|   |       | 5.2.1      | Connect Embedded PC                         | 87  |
|   |       | 5.2.2      | UL requirements                             | 88  |
|   | 5.3   | CANope     | en: Connection and wiring                   | 89  |
|   |       | 5.3.1      | D-sub connector (X003)                      | 92  |
|   |       | 5.3.2      | Cable and shielding                         | 93  |
| 6 | Multi | ifunction  | I/Os  | 95  |
|   | 6.1   | Digital ir | nputs                                       | 97  |
|   | 6.2   | Digital o  | utputs                                      | 98  |
|   | 6.3   | Counter    | mode  | 100 |
|   |       | 6.3.1      | Select operation mode                       | 102 |
|   |       | 6.3.2      | Switching outputs                           | 103 |
|   |       | 6.3.3      | Set counter value                           | 104 |
|   |       | 6.3.4      | Setting the limit value for counters        | 105 |
|   | 6.4   | Increme    | ntal encoder mode                           | 106 |
|   |       |            |   |     |



|                          |      | 6.4.1     | Switching outputs                              | 108 |
|--------------------------|------|-----------|--|-----|
|                          |      | 6.4.2     | Latching the counter value                     | 109 |
|                          |      | 6.4.3     | Setting the limit value for counters           | 110 |
|                          | 6.5  | Analog    | signal mode                                    | 111 |
|                          | 6.6  | PWM si    | ignal mode                                     | 112 |
|                          |      | 6.6.1     | Setting the PWM clock frequency and duty cycle | 114 |
|                          |      | 6.6.2     | Setting the channel synchronization            | 115 |
| 7                        | Conf | iguration | n  | 116 |
|                          | 7.1  | Starting  | the Beckhoff Device Manager                    | 116 |
|                          | 7.2  | Persiste  | ent data                                       | 117 |
|                          | 7.3  | NOVRA     | \M   | 118 |
|                          |      | 7.3.1     | Creating a Retain Handler                      | 119 |
|                          |      | 7.3.2     | Creating and linking variables                 | 121 |
|                          |      | 7.3.3     | Deleting variables under the Retain Handler    | 123 |
|                          | 7.4  | Software  | e configuration                                | 124 |
|                          |      | 7.4.1     | User name and password                         | 124 |
|                          |      | 7.4.2     | Setting the IP address                         | 125 |
|                          |      | 7.4.3     | Update image                                   | 126 |
|                          |      | 7.4.4     | Updating the firmware for multifunction I/Os   | 127 |
|                          |      | 7.4.5     | Updating the ESI device description            | 128 |
| 8                        | Twin | CAT       |  | 129 |
|                          | 8.1  | First Ste | eps  | 129 |
|                          |      | 8.1.1     | Connect to the CX70x0                          | 129 |
|                          |      | 8.1.2     | Scan multifunction I/Os                        | 131 |
|                          |      | 8.1.3     | Establishing ADS communication                 | 133 |
|                          |      | 8.1.4     | Creating a PLC project                         | 135 |
|                          |      | 8.1.5     | Linking variables                              | 137 |
|                          |      | 8.1.6     | Load configuration to CX                       | 138 |
|                          | 8.2  | TwinCA    | ،T tabs  | 140 |
|                          |      | 8.2.1     | Tree view                                      | 140 |
|                          |      | 8.2.2     | CANopen slave                                  | 142 |
|                          | 8.3  | Creating  | g CX705x as slave                              | 145 |
|                          |      | 8.3.1     | Creating a virtual slave                       | 148 |
|                          |      | 8.3.2     | Setting the address                            | 149 |
|                          |      | 8.3.3     | Creating further PDOs                          | 150 |
|                          |      | 8.3.4     | Creating variables                             | 151 |
|                          |      | 8.3.5     | Setting the transmission type                  | 152 |
|                          |      | 8.3.6     | Receiving SDO data in the PLC                  | 153 |
|                          |      | 8.3.7     | Switching slave node to PreOp from the PLC     | 154 |
|                          | 8.4  | Reading   | g the CAN baud rate                            | 155 |
|                          | 8.5  | Sending   | g arbitrary CAN telegrams                      | 155 |
|                          | 8.6  | Reading   | g the IP and MAC addresses                     | 156 |
|                          | 8.7  | Virtual E | Ethernet interface                             | 156 |
| 8.8 CoE access to multi- |      | CoE acc   | cess to multi-function I/Os                    | 157 |
|                          | 8.9  | Power s   | supply terminal                                | 159 |



|    | 8.10   | Cycle an   | d processing times                           | 161 |
|----|--------|------------|--|-----|
|    |        | 8.10.1     | Measuring processing time in the PLC program | 161 |
|    |        | 8.10.2     | Real-Time Clock (RTC)                        | 161 |
|    |        | 8.10.3     | Cycle time of 250 µs                         | 162 |
|    | 8.11   | Function   | Blocks                                       | 167 |
|    |        | 8.11.1     | FB_CX70xx_RW_EEPROM                          | 167 |
|    |        | 8.11.2     | FB_CX70xx_ResetOnBoardIO                     | 168 |
|    | 8.12   | Importan   | t attribute pragmas                          | 169 |
|    |        | 8.12.1     | Attribute 'Tc2GvlVarNames'                   | 169 |
|    |        | 8.12.2     | Attribute 'pack_mode'                        | 169 |
|    |        | 8.12.3     | Attribute 'TcCallAfterOutputUpdate'          | 170 |
| 9  | Error  | handling   | g and diagnostics                            | 174 |
|    | 9.1    | Diagnost   | tic LEDs                                     | 174 |
|    |        | 9.1.1      | K-bus  | 175 |
|    |        | 9.1.2      | E-bus  | 178 |
|    | 9.2    | CANope     | n diagnostics                                | 179 |
|    |        | 9.2.1      | Status messages                              | 179 |
|    |        | 9.2.2      | Communication                                | 180 |
|    |        | 9.2.3      | PDOs   | 182 |
|    |        | 9.2.4      | Troubleshooting                              | 183 |
|    | 9.3    | Diagnosi   | s of the multi-function I/Os                 | 186 |
|    | 9.4    | Memory     | usage  | 187 |
|    | 9.5    | Real-time  | e and CPU load                               | 189 |
| 10 | Tech   | nical data | 3  | 191 |
| 11 | Appe   | ndix       |  | 193 |
| •  |        |            | ntifier list                                 |     |
|    |        |            | rty components                               |     |
|    |        |            | ries   |     |
|    |        |            | ions   |     |
|    |        |            |  |     |
|    |        |            |  |     |
|    | List o | ot figures |  | 208 |





# 1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.

For installation and commissioning of the components, it is absolutely necessary to comply with the documentation and the following notes and explanations.

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

The responsible staff must ensure that the application or use of the products described satisfies all safety requirements, 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 notice.

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

#### **Trademarks**

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

If third parties make use of designations or trademarks used in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

#### **Patents**

The EtherCAT Technology is covered by the following patent applications and patents, without this constituting an exhaustive list:

EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702 and similar applications and registrations in several other countries.



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

### Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The distribution and reproduction of this document, as well as the use and communication of its contents without express authorization, are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event that a patent, utility model, or design are registered.



# 1.1 Representation and structure of warnings

The following warnings are used in the documentation. Read and follow the warnings.

# Warnings relating to personal injury:

# **▲ DANGER**

Hazard with high risk of death or serious injury.

# **⚠ WARNING**

Hazard with medium risk of death or serious injury.

### **⚠ CAUTION**

There is a low-risk hazard that can result in minor injury.

# Warnings relating to damage to property or the environment:

# **NOTICE**

There is a potential hazard to the environment and equipment.

### Notes showing further information or tips:



This notice provides important information that will be of assistance in dealing with the product or software. There is no immediate danger to product, people or environment.



# 1.2 Documentation issue status

| Version | Comment        |
|---------|----------------|
| 1.0     | First version. |



# 2 For your safety

Read the chapter on safety and follow the instructions in order to protect from personal injury and damage to equipment.

#### Limitation of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Unauthorized modifications and changes to the hardware or software configuration, which go beyond the documented options, are prohibited and nullify the liability of Beckhoff Automation GmbH & Co. KG.

In addition, the following actions are excluded from the liability of Beckhoff Automation GmbH & Co. KG:

- · Failure to comply with this documentation.
- · Improper use.
- · Use of untrained personnel.
- · Use of unauthorized replacement parts.

# 2.1 Intended use

The embedded PC is a control system for use in machine and system engineering for automation, visualization and communication. The embedded PC is designed for installation in a control cabinet or terminal box and is used together with Bus or EtherCAT Terminals to receive digital and analog signals from sensors and output them to actuators or forward them to higher-level controllers.

The Embedded PC is designed for a working environment that meets the requirements of protection class IP20. This involves finger protection and protection against solid foreign objects up to 12.5 mm, but not protection against water. Operation of the devices in wet and dusty environments is not permitted, unless specified otherwise. The specified limits for electrical and technical data must be adhered to.

### Improper use

The Embedded PC is not suitable for operation in the following areas:

- · Potentially explosive atmospheres.
- · Areas with an aggressive environment, e.g. aggressive gases or chemicals.
- Living areas. If the devices are to be used in living areas, the relevant standards and guidelines for interference emissions must be adhered to, and the devices must be installed in housings or control boxes with suitable shielding.

# 2.2 Staff qualification

All operations involving Beckhoff software and hardware may only be carried out by qualified personnel with knowledge of control and automation engineering. The qualified personnel must have knowledge of the administration of the Industrial PC and the associated network.

All interventions must be carried out with knowledge of control programming, and the qualified personnel must be familiar with the current standards and guidelines for the automation environment.

# 2.3 Safety instructions

The following safety instructions must be followed during installation and working with networks and the software.

# **Mounting**

• Never work on live equipment. Always switch off the power supply for the device before installation, troubleshooting or maintenance. Protect the device against unintentional switching on.



- Observe the relevant accident prevention regulations for your machine (e.g. the BGV A 3, electrical systems and equipment).
- Ensure standard-compliant connection and avoid risks to personnel. Ensure that data and supply cables are laid in a standard-compliant manner and ensure correct connection.
- · Observe the relevant EMC guidelines for your application.
- · Avoid polarity reversal of the data and supply cables, as this may cause damage to the equipment.
- The devices contain electronic components, which may be destroyed by electrostatic discharge when touched. Observe the safety precautions against electrostatic discharge according to DIN EN 61340-5-1/-3.

#### Working with networks

- · Restrict access to all devices to an authorized circle of persons.
- · Change the default passwords to reduce the risk of unauthorized access.
- · Protect the devices with a firewall.
- Apply the IT security precautions according to IEC 62443, in order to limit access to and control of devices and networks.

# 2.4 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our <a href="https://www.beckhoff.com/secquide">https://www.beckhoff.com/secquide</a>.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at <a href="https://www.beckhoff.com/secinfo">https://www.beckhoff.com/secinfo</a>.



# 3 Transport and storage

#### **Transport**

# **NOTICE**

### Short circuit due to moisture

Moisture can form during transport in cold weather or in the event of large temperature fluctuations.

Avoid moisture formation (condensation) in the embedded PC, and leave it to adjust to room temperature slowly. If condensation has occurred, wait at least 12 hours before switching on the embedded PC.

Despite the robust design of the unit, the components are sensitive to strong vibrations and impacts. During transport the embedded PC must be protected from

- · high mechanical stress and
- · use the original packaging for shipping.

Table 1: Dimensions and weight.

|                        | CX7051                 |
|------------------------|------------------------|
| Dimensions (W x H x D) | 49 mm x 100 mm x 73 mm |
| Weight                 | approx. 142 g          |

### **Storage**

· Store the Embedded PC in the original packaging.



# 4 Product overview

The CX7051 Embedded PC has an ARM Cortex™ M7 single-core processor running at 480 MHz and the following basic configuration:

- · a microSD card slot with integrated 512 MB microSD card,
- an Ethernet interface (10/100 Mbit/s, RJ45),
- a USB interface (max. 12 Mbit/s, max. 100 mA),
- integrated multi-function I/Os.

The CX7051 is programmed with TwinCAT 3 via the Ethernet interface. In addition, the Beckhoff Device Manger is available as a web interface for configuring the CX7051.

The CX7051 has a CANopen responder (slave) interface, which is implemented on a 9-pin D-sub connector and also supports up to three virtual slaves.

#### Multi-function I/Os

Special features of the CX7000 series are the eight integrated multifunction inputs and four integrated multifunction outputs.

- 8 digital inputs, 24 V DC, filter 3 ms, type 3, 1-wire technique
- 4 digital outputs, 24 V DC, 0.5 A, 1-wire technique

The integrated multifunction I/Os of the CX7051 can be configured via TwinCAT 3 for other operation modes in order to enable fast counting or the processing of analog values:

- Counter mode: 1 x digital counter input 100 kHz, 1 x digital input for up/down counter 20 kHz, 2 x digital counter outputs
- Incremental encoder mode: 2 x digital inputs for 250 kHz encoder signal (A/B input), 2 x digital encoder output
- Analog signal mode: 2 x digital inputs configured as analog inputs 0 to 10 V, 12-bit resolution with 16bit representation
- PWM signal mode: 2 x digital outputs configured for PWM signal, 15 Hz...100 kHz

#### Power supply terminal

EtherCAT Terminals (E-bus) or Bus Terminals (K-bus) can optionally be connected directly on the right-hand side; the CX7051 automatically recognizes which system is connected during the start-up phase. If further electrical signals are to be processed, the CX7051 can be extended as required and extremely flexibly by EtherCAT Terminals or Bus Terminals in addition to the integrated I/Os.

#### **Firmware**

The real-time operating system TC/RTOS, which is based on FreeRTOS, is used as the operating system or firmware. Note that TC/RTOS is a closed system and you cannot install your own software. This provides a certain level of security, as third-party software such as viruses or similar cannot be installed and the CX7051 can be connected to a network. The CX7051 can be used from TwinCAT 3.1 Build 4024.12. The following TC 3 functions are included and licensed:

- TC1000 TC3 ADS
- TC1100 TC3 IO
- TC1200 TC3 PLC
- TF4100 TC3 Controller Toolbox
- TF4110 TC3 Temperature Controller
- TF6255 TC3 Modbus-RTU
- · TF6340 TC3 Serial Communication
- TF6701 | TwinCAT 3 IoT Communication (MQTT)\*)
- TF6730 | TwinCAT 3 IoT Communicator\*)



 $^{\mbox{\tiny †}}$  Image version 114606 and TwinCAT 3 XAE 4024.47 or higher required.

The open source licenses can be viewed as a ZIP file on the microSD card.

# 4.1 Structure

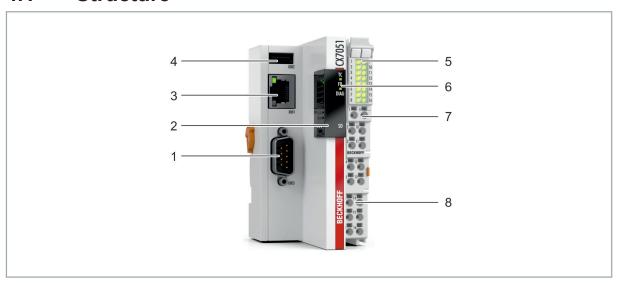


Fig. 1: Sample configuration of a CX7051 Embedded PC.

Table 2: Legend for the configuration of the basic CPU module

| No. | Component                                 | Description  |
|-----|---|--|
| 1   | D-sub connector (X003).                   | CANopen interface of the CX705x.   |
| 2   | MicroSD card slot (under the cover).      | Slot for industrial MicroSD cards. Memory space for firmware and TwinCAT 3 projects.   |
| 3   | Ethernet interface (X001)                 | For the connection to local networks. Serves as a programming interface.   |
| 4   | USB interface (X002)                      | Interface for additional USB data storage device.  |
| 5   | I/O Status LEDs                           | Diagnosis of the power supply for the Embedded PC and the terminal bus. Status of the E-bus or K-bus communication and multifunction I/Os. |
| 6   | Diagnostic LEDs                           | 1 x TwinCAT Status, 1 x Flash access, 1 x Error LED.   |
| 7   | Spring-loaded terminals, +24 V and 0 V    | Power supply (Us) for Embedded PC.   |
| 8   | Spring-loaded terminals,<br>+24 V and 0 V | Power supply (Up) for integrated multifunction I/Os and Bus Terminals via the power contacts.  |



# 4.2 Name plate

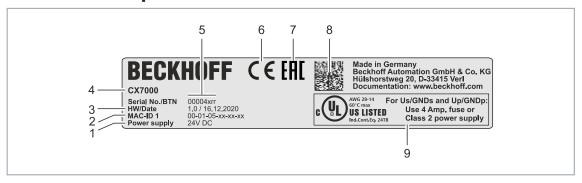


Fig. 2: Name plate example.

Table 3: Information on the name plate.

| No. | Description   |
|-----|---|
| 1   | Power supply 24 V DC.   |
| 2   | MAC addresses of the built-in Ethernet interface.   |
| 3   | Hardware version and date of manufacture.   |
| 4   | Product designation for identification of the Embedded PC.  |
| 5   | Serial number/ Beckhoff Traceability Number (BTN) for the unambiguous identification of the product. The host name is formed from BTN and the serial number/ Beckhoff Traceability Number (BTN). Example: the BTN 00004xrr results in the host name BTN-00004xrr. |
| 6   | CE marking  |
| 7   | EAC marking   |
| 8   | Machine-readable information in the form of a Data Matrix Code (DMC, code scheme ECC200) that can be used for better identification and management.   |
| 9   | UL marking with prescribed information on power supply, fuse, temperature and cable cross-sections.   |



# 4.3 Ethernet interface (X001)

You can program and commission the CX7051 Embedded PC via the X001 Ethernet interface. The Ethernet interface achieves speeds of 10 / 100 Mbit/s.

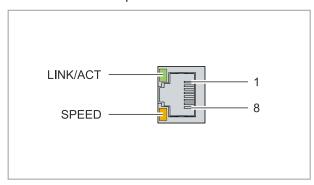


Fig. 3: Ethernet interface X001.

The LEDs on the left of the interface indicate the connection status. The upper LED (LINK/ACT) indicates whether the interface is connected to a network. If this is the case, the LED lights up green. The LED flashes when data transfer on the interface is in progress.

The lower LED (SPEED) indicates the connection speed. The LED is not lit if the speed is 10 Mbit/s. At 100 Mbit/s the LED lights up orange.

Table 4: Ethernet interface X001, pin assignment.

| PIN | Signal    | Description |
|-----|-----------|-------------|
| 1   | TD +      | Transmit +  |
| 2   | TD -      | Transmit -  |
| 3   | RD +      | Receive +   |
| 4   | connected | reserved    |
| 5   |           |             |
| 6   | RD -      | Receive -   |
| 7   | connected | reserved    |
| 8   |           |             |

#### **Transmission standards**

### 10Base5

The transmission medium for 10Base5 consists of a thick coaxial cable ("yellow cable") with a max. data transfer rate of 10 Mbaud arranged in a line topology with branches (drops) each of which is connected to one network device. Because all the devices are in this case connected to a common transmission medium, it is inevitable that collisions occur often in 10Base5.

#### 10Base2

10Base2 (Cheaper net) is a further development of 10Base5, and has the advantage that the coaxial cable is cheaper and, being more flexible, is easier to lay. It is possible for several devices to be connected to one 10Base2 cable. It is frequent for branches from a 10Base5 backbone to be implemented in 10Base2.

#### 10BaseT

Describes a twisted pair cable for 10 Mbaud. The network here is constructed as a star. It is no longer the case that every device is attached to the same medium. This means that a broken cable no longer results in failure of the entire network. The use of switches as star couplers enables collisions to be reduced. Using full-duplex connections they can even be entirely avoided.



#### 100BaseT

Twisted pair cable for 100 Mbaud. It is necessary to use a higher cable quality and to employ appropriate hubs or switches in order to achieve the higher data rate.

#### 10BaseF

The 10BaseF standard describes several optical fiber versions.

#### Short description of the 10BaseT and 100BaseT cable types

Twisted-pair copper cable for star topologies, where the distance between two devices may not exceed 100 meters.

#### **UTP**

Unshielded twisted-pair

This type of cable belongs to category 3, and is not recommended for use in an industrial environment.

### S/UTP

Screened/unshielded twisted-pair (shielded with copper braid)

Has an overall shield of copper braid to reduce influence of external interference. This cable is recommended for use with Bus Couplers.

#### **FTP**

Foiled shielded twisted-pair (shielded with aluminum foil)

This cable has an outer shield of laminated aluminum and plastic foil.

#### S/FTP

Screened/foiled shielded twisted-pair (shielded with copper braid and aluminum foil)

Has a laminated aluminum shield with a copper braid on top. Such cables can provide up to 70 dB reduction in interference power.

#### **STP**

Shielded twisted-pair

Describes a cable with overall shielding without further specification of the type of shielding.

#### S/STP

Screened/shielded twisted-pair (wires are individually shielded)

This identification refers to a cable with a shield for each of the two wires as well as an outer shield.

# **ITP**

Industrial Twisted-Pair

The structure is similar to that of S/STP, but, in contrast to S/STP, it has only two pairs of conductors.



# 4.4 USB interface (X002)

A USB flash drive can be connected to the USB interface and used as an additional memory. The USB interface supports transfer speeds of up to 12 Mbit/s and no more than 100 mA. The file is accessed from TwinCAT or the PLC program with the help of the associated function blocks. No other devices can be connected to the USB interface and used.

The same functional mode can be used for accessing files on the MicroSD card. Use C:\ as the drive letter for accessing the MicroSD card and D:\ for accessing the USB flash drive.

#### Function blocks for data access

The function blocks can be used to process files from the PLC locally on the PC. The TwinCAT target system is identified by the AMS network address. This mechanism makes it possible, amongst other things, to store or to edit files on other TwinCAT systems in the network. Access to files consists of three sequential phases:

- 1. Opening the file.
- 2. Read or write access to the opened file.
- 3. Closing the file.

Opening the file has the purpose of establishing a temporary connection between the external file, whose name is all that initially is known, and the running program. Closing the file has the purpose of indicating the end of the processing and placing it in a defined output state for processing by other programs.

| Name          | Description                   |
|---------------|-------------------------------|
| FB_EOF        | Check the end of file         |
| FB_FileOpen   | Open a file                   |
| FB_FileClose  | Close a file                  |
| FB_FileGets   | Get string from a file        |
| FB_FilePuts   | Put string to a file          |
| FB_FileRead   | Read from a file              |
| FB_FileWrite  | Write to a file               |
| FB_FileSeek   | Move the file pointer         |
| FB_FileTell   | Get the file pointer position |
| FB_FileDelete | Delete a file                 |
| FB_FileRename | Rename a file                 |
| FB_CreateDir  | Create new directory          |
| FB_RemoveDir  | Remove directory              |

# Requirements

| Development environment | , | PLC libraries to include (Category group) |
|-------------------------|---|---|
| TwinCAT v3.1.0          | PC or CX (x86, x64, ARM)                | Tc2_System (System)                       |

# 4.5 D-sub connector (X003)

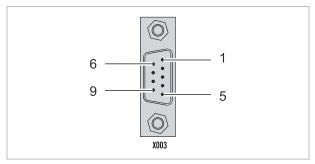


Fig. 4: CANopen interface X003.



The CAN bus line is connected via a 9-pin D-sub connector with the following pin assignment:

| Pin | Connection                                 |
|-----|--|
| 1   | not used                                   |
| 2   | CAN low (CAN-)                             |
| 3   | CAN Ground (internally connected to pin 6) |
| 4   | not used                                   |
| 5   | Shield                                     |
| 6   | CAN Ground (internally connected to pin 3) |
| 7   | CAN high (CAN+)                            |
| 8   | not used                                   |
| 9   | not used                                   |

The DIN rail contact spring and the connector shield are connected together. An auxiliary voltage of up to  $30~V_{DC}$  may be connected to pin 9, which is used by some CAN devices to supply the transceivers.



# 4.6 MicroSD card

The basic equipment of the CX7051 includes a 512 MB microSD card. You can optionally order the embedded PC with a larger microSD card (1 GB, 2 GB, 4 GB, 8 GB or 16 GB).

The cards employed are SLC memory with extended temperature range for industrial applications. Use exclusively microSD cards approved by Beckhoff.

| Order identifier | Capacity | Description  |
|------------------|----------|--|
| CX1900-0123      | 1 GB     | microSD card (SLC memory) with                                       |
| CX1900-0125      | 2 GB     | extended temperature range for                                       |
| CX1900-0127      | 4 GB     | industrial applications instead of the 512 MB card (ordering option) |
| CX1900-0129      | 8 GB     | the 312 Mb card (ordering option)                                    |
| CX1900-0131      | 16 GB    |  |

| Order identifier | Capacity | Description                            |
|------------------|----------|--|
| CX1900-0122      | 512 MB   | microSD card (SLC memory) with         |
| CX1900-0124      | 1 GB     | extended temperature range for         |
| CX1900-0126      | 2 GB     | industrial applications as spare part. |
| CX1900-0128      | 4 GB     | part                                   |
| CX1900-0130      | 8 GB     |  |
| CX1900-0132      | 16 GB    |  |



# 4.7 CANopen system overview

CANopen is a widely used CAN application layer, developed by the CAN-in-Automation association (CiA, <a href="http://www.can-cia.org">http://www.can-cia.org</a>), 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) [▶ 26] 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.

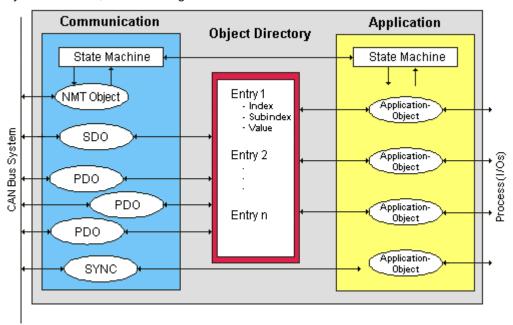


Fig. 5: CANopen Device Model

#### **Communication Types**

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

- Event driven [ 29]: 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 [ 29]: A SYNC telegram causes the modules to accept the output data that was previously received, and to send new input data.
- Requested (polled) [▶ 26]: A CAN data request telegram causes the modules to send their input data.

The desired communication type is set by the <u>Transmission Type [ 26]</u> 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 kbit/s up to 1 Mbit/s 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. 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 Mbit/s, for instance, the network may extend 25 m, whereas at 50 kbit/s 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 (<a href="http://www.beckhoff.de">http://www.beckhoff.de</a>) 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).

# 4.7.1 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:



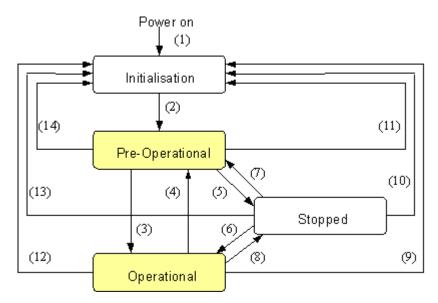


Fig. 6: CANopen bootup state diagram

# **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. Starts the module, enables outputs, starts transmission of PDOs.                                  |



| Status transition | Command Specifier cs | Explanation  |
|-------------------|----------------------|--|
| (4), (7)          | cs = 128 = 0x80      | Enter_Pre-Operational. Stops PDO transmission, SDO still active.   |
| (5), (8)          | cs = 2 = 0x02        | Stop_Remote_Node. 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 | 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**

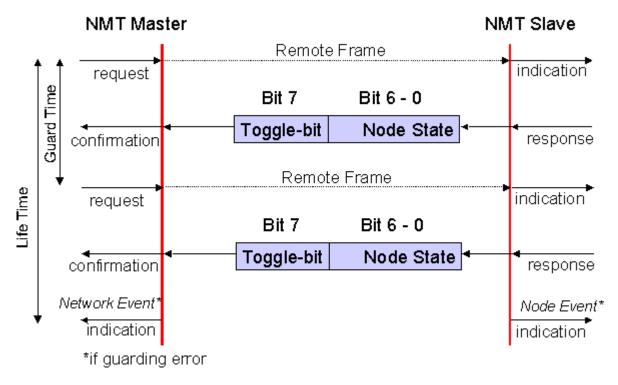


Fig. 7: Schematic diagram: "Guarding procedure"

#### **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<sub>dec</sub>). 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

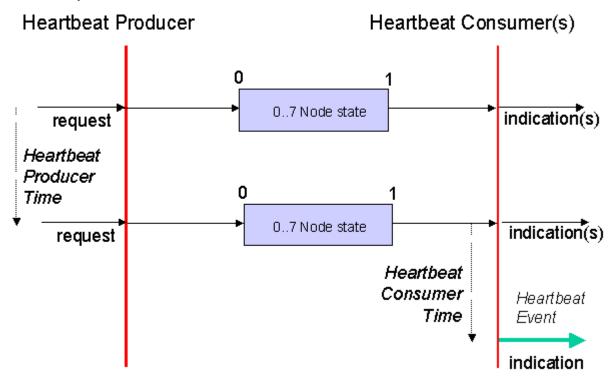


Fig. 8: Schematic diagram: "Heartbeat procedure"

#### **Protocol**

The toggle bit is not used in the heart beat procedure. The nodes send their status cyclically (s). See Guarding [▶ 25].

# 4.7.2 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 [ 193] 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.



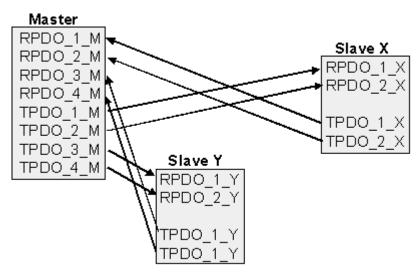


Fig. 9: Default identifier allocation: Master/Slave

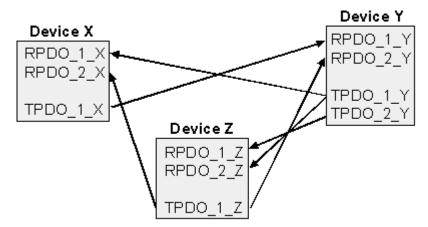


Fig. 10: PDO linking: Peer to Peer

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: <u>Notes on PDO Parameterization [▶ 33]</u>).



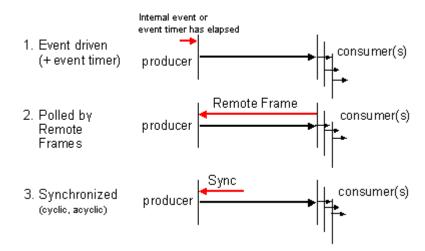


Fig. 11: Diagram: CAN process data transmission

#### **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.



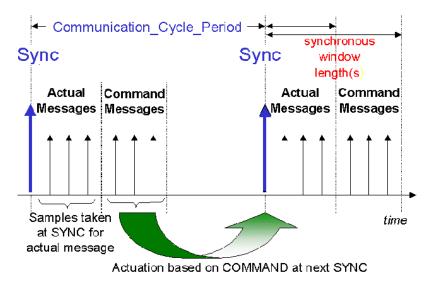


Fig. 12: Diagram: CAN "SYNC" telegram

#### **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.

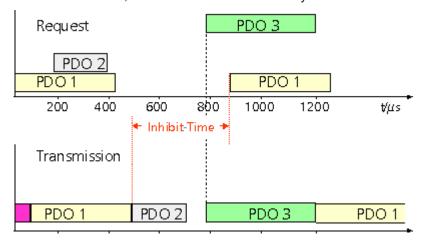


Fig. 13: Timing diagram: "Inhibit time"

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.



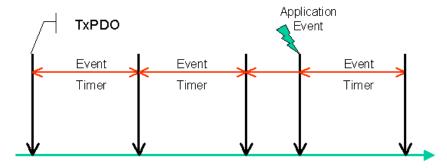


Fig. 14: Time representation of the event timer

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 [▶ 33]

# **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.

#### Object Directory Mapping Object Index Sub **Object Content** 1800h 01h 8 6TTTh TTh 1800h 02h 6000h 00h 8 03h 1800h 6WWWWwh WWh 16 PDO Length: 32 Bit TxPDO\_1 Object A Object B Object D TTh 6TTTh Object A Application Object 6UUUh UUh Object B √Vh Object C 6WWWhlWWr Object D 6XXXh XXh Object E 6YYYh YYh Object F 6ZZZh ZZh Object G

Fig. 15: 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.

# 4.7.3 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:
with the SYNC-Rate parameter the bus load can be set 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 programs (PLC or NC).

The guaranteed response time for cyclically synchronous communication is always at least as long as the cycle time, and the bus bandwidth is not used optimally, since old, unchanging data is also constantly 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 [\*\) 26], 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. The expiry timer can be used to set time windows for the transmit PDOs: The telegram is sent at the earliest after the inhibit time [\*\) 26] has elapsed and is sent again at the latest after the expiry timer has elapsed.
- The communication type is parameterized by means of the <u>Transmission Type [> 26]</u>.

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 both principles: Synchronous axis communication and short response time 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 the example mentioned, it can also be useful to specifically influence the identifier distribution in order to optimize the bus access through the priority distribution: the highest priority is given to the PDO with the limit switch data, the lowest to the one with the temperature values.

Optimization of bus access latency time through modification of the identifier allocation is not, however, normally required. In contrast, the identifiers must be changed to enable masterless communication(<u>PDO Linking [\* 26]</u>). In this example 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. [▶ 32]

# 4.7.4 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.

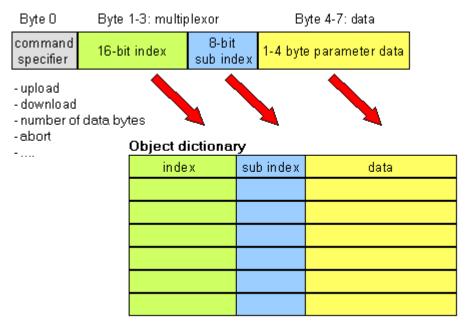


Fig. 16: SDO protocol: access to the object directory



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 (=1536 <sub>dec</sub> ) + node ID | 0x40             | Index0 | Index1 | Subldx | 0x00 | 0x00 | 0x00 | 0x00 |

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

# Client -> Server, Upload Response

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

| Parameter | Explanation                       |
|-----------|-----------------------------------|
| Index0    | Index low byte (Unsigned16, LSB)  |
| Index1    | Index high byte (Unsigned16, MSB) |
| Subldx    | 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 (=1536 <sub>dec</sub> ) + node ID | 0x22             | Index0 | Index1 | Subldx | Data0 | Data1 | Data2 | Data3 |

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



| Parameter | Explanation                |
|-----------|----------------------------|
| 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 us | er data |        |        |      |      |      |      |
|---|-----------|---------|--------|--------|------|------|------|------|
| 0x580 (=1408 <sub>dec</sub> ) + node ID | 0x60      | Index0  | Index1 | Subldx | 0x00 | 0x00 | 0x00 | 0x00 |

| Parameter | Explanation                       |
|-----------|-----------------------------------|
| Index0    | Index low byte (Unsigned16, LSB)  |
| Index1    | Index high byte (Unsigned16, MSB) |
| Subldx    | 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 us | er data |        |        |        |        |        |        |
|---|-----------|---------|--------|--------|--------|--------|--------|--------|
| 0x580 (client) or<br>0x600 (server) + node ID | 0x80      | Index0  | Index1 | Subldx | Error0 | Error1 | Error2 | Error3 |

| Parameter | Explanation                          |
|-----------|--------------------------------------|
| Index0    | Index low byte (Unsigned16, LSB)     |
| Index1    | Index high byte (Unsigned16, MSB)    |
| Subldx    | 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                   |



| SDO error code | Explanation                                      |
|----------------|--|
| 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 00  | General error                                    |
| 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        |

# 4.7.5 Objekt dictionary

# 4.7.5.1 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:

- 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 sub-index. 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 sub-index). The mapping parameters of the receive PDOs are in the region from 0x1600 to 0x160F (plus sub-index). 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

# 4.7.5.2 Object List



The objects in the object directory can be reached by SDO access, but not generally through the  $\underline{\text{KS2000}}$  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/<br>BK515x | BK5110 | LC5100 | BX5100/<br>BC5150 | CX705x/<br>CX8051/B510 |
|--|-----------------|-------------------|--------|--------|-------------------|------------------------|
| Device type [▶ 41]                                       | 0x1000          | x                 | x      | x      |                   | x                      |
| Error register [▶ 41]                                    | 0x1001          | x                 | x      | x      | x                 | x *                    |
| Error memory [▶ 41]                                      | 0x1003          | x                 | x      | x      |                   |                        |
| Sync Identifier [ > 41]                                  | 0x1005          | x                 | x      | x      | x                 | x                      |
| Sync Interval [> 41]                                     | 0x1006          | х                 | х      | х      | х                 | x                      |
| Device name [▶ 41]                                       | 0x1008          | х                 | х      | x      | х                 | x *                    |
| Hardware version [▶ 41]                                  | 0x1009          | x                 | x      | x      |                   |                        |
| Software version [> 41]                                  | 0x100A          | x                 | х      | x      | х                 | x                      |
| Node number [▶ 41]                                       | 0x100B          | х                 | х      | х      |                   |                        |
| Guard Time [▶ 41]  | 0x100C          | х                 | х      | х      | х                 | x                      |
| Life Time Factor [▶ 41]                                  | 0x100D          | х                 | х      | х      | х                 | x                      |
| Guarding Identifier [▶ 41]                               | 0x100E          | х                 | х      | х      |                   |                        |
| Save parameters [> 41]                                   | 0x1010          | х                 | х      | х      |                   |                        |
| Load default values [▶ 41]                               | 0x1011          | х                 | х      | х      |                   |                        |
| Emergency Identifier [ > 41]                             | 0x1014          | х                 | х      | х      |                   |                        |
| Consumer Heartbeat Time [ • 41]                          | 0x1016          | х                 | х      | х      | х                 | x                      |
| Producer Heartbeat Time [▶ 41]                           | 0x1017          | х                 | х      | х      | х                 | x                      |
| Device identifier (identity object)  [ \( \bullet 41 \)] | 0x1018          | х                 | Х      | х      | х                 | x *                    |
| Server SDO parameters [▶ 41]                             | 0x1200          | х                 | х      | х      |                   |                        |
| Communication parameters for the 1st - 5th RxPDO [ > 41] | 0x1400 - 0x1404 | х                 | х      | х      | х                 | х                      |
| Communication parameters for the 6th - 16th RxPDO [▶ 41] | 0x1405 - 0x140F | х                 |        |        | х                 | Х                      |



| Parameter  | Index           | BK5120/<br>BK515x | BK5110 | LC5100 | BX5100/<br>BC5150 | CX705x/<br>CX8051/B510 |
|--|-----------------|-------------------|--------|--------|-------------------|------------------------|
| Communication parameters for the 17th - 32nd RxPDO [ \( \bullet 41 \)] | 0x1410 - 0x141F |                   |        |        | x only<br>BX5100  | x                      |
| Mapping 1st -5th RxPDO [▶ 41]  | 0x1600 - 0x1604 | х                 | х      | х      | х                 | х                      |
| Mapping 6th -16th RxPDO [▶ 41]   | 0x1605 - 0x160F | х                 |        |        | х                 | х                      |
| Mapping 17th -32nd RxPDO [▶ 41]  | 0x1610 - 0x161F |                   |        |        | x only<br>BX5100  | х                      |
| Communication parameters for the 1st - 5th TxPDO [ • 41]               | 0x1800 - 0x1804 | х                 | Х      | Х      | х                 | х                      |
| Communication parameters for the 6th - 16th TxPDO [ • 41]              | 0x1805 - 0x180F | х                 |        |        | х                 | х                      |
| Communication parameters for the 17th - 32nd TxPDO [ 41]               | 0x1810 - 0x181F |                   |        |        | x only<br>BX5100  | х                      |
| Mapping 1st -5th TxPDO [ > 41]   | 0x1A00 - 0x1A04 | x                 | x      | x      | x                 | x                      |
| Mapping 6th -16th TxPDO [▶ 41]   | 0x1A05 - 0x1A0F | х                 |        |        | х                 | х                      |
| Mapping 17th -32nd TxPDO  [▶ 41]                                       | 0x1A10 - 0x1A1F |                   |        |        | x only<br>BX5100  | x                      |
| Flag area %MB0-511   | 0x2F00          |                   |        |        | х                 |                        |
| Flag area %MB511-1023  | 0x2F01          |                   |        |        | х                 |                        |
| Flag area %MB1024-1535   | 0x2F02          |                   |        |        | х                 |                        |
| Flag area %MB1536-2047   | 0x2F03          |                   |        |        | х                 |                        |
| Flag area %MB2048-2559   | 0x2F04          |                   |        |        | х                 |                        |
| Flag area %MB2560-3071   | 0x2F05          |                   |        |        | х                 |                        |
| Flag area %MB3072-3584   | 0x2F06          |                   |        |        | х                 |                        |
| Flag area %MB3585-4095   | 0x2F07          |                   |        |        | х                 |                        |
| 3-byte special terminals, input data [▶ 41]                            | 0x2600          | x                 |        |        |                   |                        |
| 3-byte special terminals, output data [▶ 41]                           | 0x2700          | х                 |        |        |                   |                        |
| 4-byte special terminals, input data [▶ 41]                            | 0x2800          | х                 |        |        |                   |                        |
| 4-byte special terminals, output data [▶ 41]                           | 0x2900          | х                 |        |        |                   |                        |
| 5-byte special terminals, input data [ • 41]                           | 0x2A00          | х                 |        |        |                   |                        |
| 5-byte special terminals, output data [ • 41]                          | 0x2B00          | x                 |        |        |                   |                        |
| 6-byte special terminals, input data [▶ 41]                            | 0x2C00          | х                 |        |        |                   |                        |
| 6-byte special terminals, output data [▶ 41]                           | 0x2D00          | х                 |        |        |                   |                        |
| 8-byte special terminals, input data [ • 41]                           | 0x3000          | х                 |        |        |                   |                        |
| 8-byte special terminals, output data [▶ 41]                           | 0x3100          | х                 |        |        |                   |                        |
| Register communication, bus node [▶ 41]                                | 0x4500          | х                 | х      | х      |                   |                        |



| Parameter  | Index  | BK5120/<br>BK515x | BK5110 | LC5100 | BX5100/<br>BC5150 | CX705x/<br>CX8051/B510 |
|--|--------|-------------------|--------|--------|-------------------|------------------------|
| Register communication, bus terminal/extension box [ • 41] | 0x4501 | х                 | X      | x      |                   |                        |
| Enable PDOs [> 41]   | 0x5500 | х                 | х      | x      |                   |                        |
| NetId  | 0x5FFE |                   |        |        | х                 |                        |
| Digital inputs [▶ 41]                                      | 0x6000 | X                 | Х      | x      |                   |                        |
| Interrupt mask [> 41]                                      | 0x6126 | х                 | х      | x      |                   |                        |
| Digital outputs [▶ 41]                                     | 0x6200 | х                 | х      | х      |                   |                        |
| Analog inputs [▶ 41]                                       | 0x6401 | х                 |        |        |                   |                        |
| Analog outputs [▶ 41]                                      | 0x6411 | х                 |        |        |                   |                        |
| Event control, analog inputs  [ • 41]                      | 0x6423 | х                 |        |        |                   |                        |
| Upper limit value, analog inputs  [▶ 41]                   | 0x6424 | х                 |        |        |                   |                        |
| Lower limit value, analog inputs  [ • 41]                  | 0x6425 | х                 |        |        |                   |                        |
| Delta function, analog inputs  [ \( \bullet 41 \)]         | 0x6426 | х                 |        |        |                   |                        |

<sup>\*</sup> When an ADS server is registered, these objects are relayed to the PLC via ADS notification and have to be answered there.

# 4.7.5.3 Objects and Data

### **Device type**

| Index  | Sub-index | Name        | Туре       | Attribute | 111 3 | Default value | Meaning                  |
|--------|-----------|-------------|------------|-----------|-------|---------------|--------------------------|
| 0x1000 | 0         | Device type | Unsigned32 | ro        | N     |               | 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 <sub>dez</sub> ) |

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, for example, 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  | Sub-index | Name              | Туре      | Attribute | 111 3 | Default value | Meaning           |
|--------|-----------|-------------------|-----------|-----------|-------|---------------|-------------------|
| 0x1001 | 0         | Error<br>register | Unsigned8 | ro        | N     | 0x00          | Error<br>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 1003.

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 store**

| Index  | Sub-index | Name  | Туре       | Attribute | Mapping | Default value | Meaning   |
|--------|-----------|---|------------|-----------|---------|---------------|---|
| 0x1003 | 0x00      | Predefined<br>error field<br>(Error<br>store) | Unsigned8  | rw        | N       | 0x00          | Object 1003h contains a description of the error that has occurred in the device - sub-index 0 has the number of error states stored. |
|        | 1         | Actual error                                  | Unsigned32 | ro        | N       | None          | Last error<br>state to<br>have<br>occurred  |
|        |           |   |            |           |         |               |   |
|        | 10        | Standard<br>error field                       | Unsigned32 | ro        | N       | None          | A maximum of 10 error states are stored.  |

The 32 bit value in the error store 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 sub-index 1, all the other sub-indices being appropriately incremented. The whole error store is cleared by writing a 0 to sub-index 0.

If there has not been an error since power up, then object 0x1003 only consists of sub-index 0 with a 0 entered into it. The error store 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  | Sub-index | Name    | Туре       | Attribute | 111 3 | Default value | Meaning       |
|--------|-----------|---------|------------|-----------|-------|---------------|---------------|
| 0x1005 | 0         | COB-ID  | Unsigned32 | rw        | N     | 8000008x0     | Identifier of |
|        |           | Sync    | _          |           |       | 0             | the SYNC      |
|        |           | Message |            |           |       |               | 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  | Sub-index | Name                              | Туре       | Attribute | Mapping | Default value | Meaning                                     |
|--------|-----------|-----------------------------------|------------|-----------|---------|---------------|---|
| 0x1006 |           | Communica<br>tion cycle<br>period | Unsigned32 | rw        | N       | 0             | Length of<br>the SYNC<br>interval in<br>µs. |

If a value other than zero is entered here, the bus node will go into the fault 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).
- The sync interval has been entered in object 0x1006 and (sync interval x lowest PDO transmission type) is less than 90ms.

The modules are then synchronised throughout.

### **Device name**

| Index  | Sub-index | Name                            | Туре              | Attribute | Mapping | Default value | Meaning                           |
|--------|-----------|---------------------------------|-------------------|-----------|---------|---------------|-----------------------------------|
| 0x1008 | 0         | Manufactur<br>er Device<br>Name | Visible<br>String | ro        | N       | ,             | Device<br>name of the<br>bus node |

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

### Hardware version

| Index  | Sub-index | Name                                     | Туре              | Attribute | Mapping | Default value | Meaning                                 |
|--------|-----------|--|-------------------|-----------|---------|---------------|---|
| 0x1009 | 0         | Manufactur<br>er<br>hardware-<br>version | Visible<br>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  | Sub-index | Name                                  | Туре | Attribute | Mapping | Default value | Meaning   |
|--------|-----------|---------------------------------------|------|-----------|---------|---------------|---|
| 0x100A | 0         | Manufactur<br>er software-<br>version |      | ro        | N       |               | Software<br>version<br>number of<br>the bus<br>node |

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

### Node number

| Index  | Sub-index | Name    | Туре       | Attribute | 111 3 | Default value | Meaning         |
|--------|-----------|---------|------------|-----------|-------|---------------|-----------------|
| 0x100B | 0         | Node-ID | Unsigned32 | ro        | N     |               | Set node number |

The node number is supported for reasons of compatibility.

### **Guard time**

| Index  | Sub-index | Name               | Туре       | Attribute | Mapping | Default value | Meaning  |
|--------|-----------|--------------------|------------|-----------|---------|---------------|--|
| 0x100C | 0         | Guard time<br>[ms] | Unsigned16 | rw        | N       |               | Interval<br>between<br>two guard<br>telegrams.<br>Is set by<br>the NMT<br>master or<br>configuratio<br>n tool. |

## Life time factor

| Index  | Sub-index | Name                | Туре      | Attribute | Mapping | Default value | Meaning  |
|--------|-----------|---------------------|-----------|-----------|---------|---------------|--|
| 0x100D | 0         | Life time<br>factor | Unsigned8 | rw        | N       | 0             | Life time<br>factor x<br>guard time<br>= life time<br>(watchdog<br>for life<br>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   | Sub-index | Name                           | Туре       | Attribute | Mapping | Default value                  | Meaning                                      |
|---------|-----------|--------------------------------|------------|-----------|---------|--------------------------------|--|
| 0x100 E | 0         | COB-ID<br>guarding<br>protocol | Unsigned32 | ro        | N       | 0x000007x<br>y, xy =<br>NodeID | Identifier of<br>the<br>guarding<br>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  | Sub-index | Name                    | Туре       | Attribute | Mapping | Default value | Meaning                                |
|--------|-----------|-------------------------|------------|-----------|---------|---------------|--|
| 0x1010 | 0         | Store<br>Parameter      | Unsigned8  | ro        | N       | 1             | Number of store options                |
|        | 1         | store all<br>parameters | Unsigned32 | rw        | N       | 1             | Stores all<br>(storable)<br>parameters |

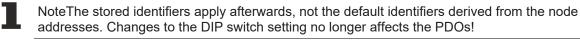
By writing the string *save* in ASCII code (hexadecimal 0x65766173) to sub-index 1, the current parameters are placed into non-volatile storage. (The byte sequence on the bus including the SDO protocol: 0x23 0x10 0x10 0x01 0x73 0x61 0x76 0x65).

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).

# [Gefahrinformation hier einfügen!]



- 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  | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value | Meaning   |
|--------|-----------|----------------------------------|------------|-----------|---------|---------------|---|
| 0x1011 | 0         | Restore<br>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<br>manufactur<br>er Defaults | Unsigned32 | rw        | N       | 1             | Resets all coupler parameters to manufactur er's settings (including registers) |



Writing the string *load* in ASCII code (hexadecimal 0x64616F6C) into sub-index 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  | Sub-index | Name                | Туре       | Attribute | Mapping | Default value  | Meaning                                       |
|--------|-----------|---------------------|------------|-----------|---------|----------------|---|
| 0x1014 | 0         | COB-ID<br>Emergency | Unsigned32 | rw        | N       | 0, +<br>NodelD | Identifier of<br>the<br>emergency<br>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  | Sub-index | Name                          | Туре       | Attribute | Mapping | Default value | Meaning   |
|--------|-----------|-------------------------------|------------|-----------|---------|---------------|---|
| 0x1016 | 0         | Number of elements            | Unsigned8  | ro        | N       | 2             | The consumer heartbeat time describes the expected heartbeat cycle time and the node ID of the monitored node |
|        | 1         | Consumer<br>heartbeat<br>time | Unsigned32 | rw        | N       | 0             | Watchdog<br>time in ms<br>and node<br>ID of the<br>monitored<br>node  |

The 32-bit value is used as follows:

| MSB          |                     | LSB                               |
|--------------|---------------------|-----------------------------------|
| Bit 3124     | Bit 2316            | Bit 150                           |
| 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  | Sub-index | Name                          | Туре       | Attribute | Mapping | Default value | Meaning  |
|--------|-----------|-------------------------------|------------|-----------|---------|---------------|--|
| 0x1017 | 0         | Producer<br>heartbeat<br>time | Unsigned16 | rw        | N       | 0             | Interval in ms between two transmitted heartbeat telegrams |

# **Device identifier (identity object)**

| Index            | Sub-index | Name   | Туре       | Attribute | Mapping | Default value          | Meaning  |
|------------------|-----------|--|------------|-----------|---------|------------------------|--|
| 0 <b>x1018</b> 0 | 0         | Identity<br>Object:<br>Number of<br>elements | Unsigned8  | ro        | N       | 4                      | The identity object contains general information about the type and version of the device.                             |
|                  | 1         | Vendor ID                                    | Unsigned32 | го        | N       | 0x0000000<br>2         | Manufactur<br>er identifier.<br>Beckhoff<br>has vendor<br>ID 2   |
|                  | 2         | Product<br>Code                              | Unsigned32 | ro        | N       | Depends on the product | Device<br>identifier   |
|                  | 3         | Revision<br>Number                           | Unsigned32 | ro        | N       | -                      | Version number   |
|                  | 4         | Serial<br>Number                             | Unsigned32 | ro        | N       | -                      | Production<br>date<br>low word,<br>high byte:<br>calendar<br>week (dec),<br>low word,<br>low byte:<br>calendar<br>year |

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



# **Server SDO parameters**

| Index  | Sub-index                | Name                         | Туре       | Attribute | Mapping | Default value                 | Meaning  |
|--------|--------------------------|------------------------------|------------|-----------|---------|-------------------------------|--|
| 0x1200 | 0                        | Number of elements           | Unsigned8  | ro        | N       |                               | Communica<br>tion<br>parameters<br>of the<br>server<br>SDO. Sub-<br>index 0:<br>number of<br>following<br>parameters |
|        | 1 COB-ID Client ->Server | Client                       | Unsigned32 | ro        | N       | 0x000006x<br>y,<br>xy=Node-ID | COB-ID<br>RxSDO<br>(Client -><br>Server)   |
|        | 2                        | COB-ID<br>Server<br>->Client | Unsigned32 | ro        | N       | 0x0000058<br>0 + Node-<br>ID  | COB-ID<br>TxSDO<br>(Client -><br>Server)   |

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



## Communication parameters for the 1st RxPDO

| Index  | Sub-index | Name                     | Туре       | Attribute | Mapping | Default value                 | Meaning  |
|--------|-----------|--------------------------|------------|-----------|---------|-------------------------------|--|
| 0x1400 | 0         | Number of elements       | Unsigned8  | го        | N       | 5                             | Communica<br>tion<br>parameters<br>for the first<br>receive<br>PDO. Sub-<br>index 0:<br>number of<br>following<br>parameters |
|        | 1         | COB-ID                   | Unsigned32 | rw        | N       | 0x000002x<br>y,<br>xy=Node-ID | COB-ID<br>(Communic<br>ation Object<br>Identifier)<br>RxPDO1   |
|        | 2         | Transmissi<br>on Type    | Unsigned8  | rw        | N       | 255                           | Transmissi<br>on type of<br>the PDO  |
|        | 3         | Inhibit Time             | Unsigned16 | rw        | N       | 0                             | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used in the<br>RxPDO.                                |
|        | 4         | CMS<br>Priority<br>Group | Unsigned8  | rw        | N       | -                             | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used.  |
|        | 5         | Event<br>Timer           | Unsigned16 | rw        | N       | 0                             | Event-<br>Timer.<br>Watchdog<br>time<br>defined for<br>monitoring<br>reception of<br>the PDO.                                |

Sub-index 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). Sub-index 2 contains the type of the transmission (see introduction to PDOs).



# Communication parameters for the 2nd RxPDO

| Index  | Sub-index | Name                     | Туре       | Attribute | Mapping | Default value                 | Meaning   |
|--------|-----------|--------------------------|------------|-----------|---------|-------------------------------|---|
| 0x1401 | 0         | Number of elements       | Unsigned8  | ro        | N       | 5                             | Communica<br>tion<br>parameter<br>for the<br>second<br>receive<br>PDO.                        |
|        | 1         | COB-ID                   | Unsigned32 | rw        | N       | 0x000003x<br>y,<br>xy=Node-ID | COB-ID<br>(Communic<br>ation Object<br>Identifier)<br>RxPDO2                                  |
|        | 2         | Transmissi<br>on Type    | Unsigned8  | rw        | N       | 255                           | Transmissi<br>on type of<br>the PDO   |
|        | 3         | Inhibit Time             | Unsigned16 | rw        | N       | 0                             | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used in the<br>RxPDO. |
|        | 4         | CMS<br>Priority<br>Group | Unsigned8  | rw        | N       | -                             | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used.                 |
|        | 5         | Event<br>Timer           | Unsigned16 | rw        | N       | 0                             | Event-<br>Timer.<br>Watchdog<br>time<br>defined for<br>monitoring<br>reception of<br>the PDO. |



# **Communication parameters for the 3rd RxPDO**

| Index  | Sub-index | Name                     | Туре       | Attribute | Mapping | Default value                 | Meaning   |
|--------|-----------|--------------------------|------------|-----------|---------|-------------------------------|---|
| 0x1402 | 0         | Number of elements       | Unsigned8  | ro        | N       | 5                             | Communica<br>tion<br>parameter<br>for the third<br>receive<br>PDO.                            |
|        | 1         | COB-ID                   | Unsigned32 | rw        | N       | 0x000004x<br>y,<br>xy=Node-ID | COB-ID<br>(Communic<br>ation Object<br>Identifier)<br>RxPDO3                                  |
|        | 2         | Transmissi<br>on Type    | Unsigned8  | rw        | N       | 255                           | Transmissi<br>on type of<br>the PDO   |
|        | 3         | Inhibit Time             | Unsigned16 | rw        | N       | 0                             | Present for reasons of backwards compatibilit y, but not used in the RxPDO.                   |
|        | 4         | CMS<br>Priority<br>Group | Unsigned8  | rw        | N       | -                             | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used.                 |
|        | 5         | Event<br>Timer           | Unsigned16 | rw        | N       | 0                             | Event-<br>Timer.<br>Watchdog<br>time<br>defined for<br>monitoring<br>reception of<br>the PDO. |



# Communication parameters for the 4th RxPDO

| Index  | Sub-index | Name                     | Туре       | Attribute | Mapping | Default value                 | Meaning   |
|--------|-----------|--------------------------|------------|-----------|---------|-------------------------------|---|
| 0x1403 | 0         | Number of elements       | Unsigned8  | ro        | N       | 5                             | Communica<br>tion<br>parameters<br>for the<br>fourth<br>receive<br>PDO.                       |
|        | 1         | COB-ID                   | Unsigned32 | rw        | N       | 0x000005x<br>y,<br>xy=Node-ID | COB-ID<br>(Communic<br>ation Object<br>Identifier)<br>RxPDO4                                  |
|        | 2         | Transmissi<br>on Type    | Unsigned8  | rw        | N       | 255                           | Transmissi<br>on type of<br>the PDO   |
|        | 3         | Inhibit Time             | Unsigned16 | rw        | N       | 0                             | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used in the<br>RxPDO. |
|        | 4         | CMS<br>Priority<br>Group | Unsigned8  | rw        | N       | -                             | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used.                 |
|        | 5         | Event<br>Timer           | Unsigned16 | rw        | N       | 0                             | Event-<br>Timer.<br>Watchdog<br>time<br>defined for<br>monitoring<br>reception of<br>the PDO. |



# Communication parameters for the 5th-16th RxPDOs

| Index  | Sub-index       | Name                     | Туре       | Attribute | Mapping | Default value | Meaning   |
|--|-----------------|--------------------------|------------|-----------|---------|---------------|---|
| 0x1404 -<br>0x140F<br>(dependin<br>g on the<br>device<br>type) | 0               | Number of elements       | Unsigned8  | го        | N       | 5             | Communica<br>tion<br>parameter<br>for the 5 <sup>th</sup> to<br>16 <sup>th</sup> receive<br>PDOs. |
|  | 1dth="5%"><br>1 | COB-ID                   | Unsigned32 | rw        | N       | 0x8000000     | COB-ID<br>(Communic<br>ation Object<br>Identifier)<br>RxPDO51                                     |
|  | 2               | Transmissi<br>on Type    | Unsigned8  | rw        | N       | 255           | Transmissi<br>on type of<br>the PDO   |
|  | 3               | Inhibit Time             | Unsigned16 | rw        | N       | 0             | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used in the<br>RxPDO.     |
|  | 4               | CMS<br>Priority<br>Group | Unsigned8  | rw        | N       | -             | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used.                     |
|  | 5               | Event<br>Timer           | Unsigned16 | rw        | N       | 0             | Event-<br>Timer.<br>Watchdog<br>time<br>defined for<br>monitoring<br>reception of<br>the PDO.     |

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



## Mapping parameters for the 1st RxPDO

| Index  | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value                        | Meaning   |
|--------|-----------|----------------------------------|------------|-----------|---------|--------------------------------------|---|
| 0x1600 | 0         | Number of elements               | Unsigned8  | rw        | N       | Depending<br>on type and<br>fittings | Mapping parameter of the first receive PDO; subindex 0: number of mapped objects.                                   |
|        | 1         | 1 <sup>st</sup> mapped object    | Unsigned32 | rw        | N       | 0x6200010<br>8                       | 1 <sup>st</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width) |
|        | 2         | 2 <sup>nd</sup> mapped object    | Unsigned32 | rw        | N       | 0x6200020<br>8                       | 2 <sup>nd</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width) |
|        |           |                                  |            |           |         |                                      |   |
|        | 8         | 8 <sup>th</sup> mapped<br>object | Unsigned32 | rw        | N       | 0x6200080<br>8                       | 8 <sup>th</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>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 organised in bytes, the length of the PDO in bytes can be found directly at sub-index 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 (sub-index 1) of the communication parameters to 1)
- 2. Deactivate mapping (set sub-index 0 of the mapping entry to 0)
- 3. Change mapping entries (sub-indices 1...8)
- 4. Activate mapping (set sub-index 0 of the mapping entry to the correct number of mapped objects)
- 5. Create PDO (set bit 31 in the identifier entry (sub-index 1) of the communication parameters to 0)



## Mapping parameters for the 2nd RxPDO

| Index | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value                        | Meaning   |
|-------|-----------|----------------------------------|------------|-----------|---------|--------------------------------------|---|
|       | 0         | Number of elements               | Unsigned8  | rw        | N       | Depending<br>on type and<br>fittings | Mapping parameter of the second receive PDO; sub-index 0: number of mapped objects.                                 |
|       | 1         | 1 <sup>st</sup> mapped object    | Unsigned32 | rw        | N       | 0x6411011<br>0                       | 1st mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width)             |
|       | 2         | 2 <sup>nd</sup> mapped<br>object | Unsigned32 | rw        | N       | 0x6411021<br>0                       | 2 <sup>nd</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width) |
|       |           |                                  |            |           |         |                                      |   |
|       | 8         | 8 <sup>th</sup> mapped<br>object | Unsigned32 | rw        | N       | 0x0000000<br>0                       | 8 <sup>th</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>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 organised in words, the length of the PDO in bytes can be found directly at sub-index 0.

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



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

| Index   | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value                        | Meaning   |
|---|-----------|----------------------------------|------------|-----------|---------|--------------------------------------|---|
| 0x1602-<br>0x160F<br>(dependin<br>g on the<br>device<br>type) | 0         | Number of elements               | Unsigned8  | rw        | N       | Depending<br>on type and<br>fittings | Mapping parameters for the third to sixteenth receive PDOs; subindex 0: number of mapped objects.                   |
|   | 1         | 1 <sup>st</sup> mapped object    | Unsigned32 | rw        | N       | 0x0000000<br>0 (see text)            | 1st mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width)             |
|   | 2         | 2 <sup>nd</sup> mapped<br>object | Unsigned32 | rw        | N       | 0x0000000<br>0 (see text)            | 2 <sup>nd</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width) |
|   |           |                                  |            |           |         |                                      |   |
|   | 8         | 8 <sup>th</sup> mapped<br>object | Unsigned32 | rw        | N       | 0x0000000<br>0 (see text)            | 8 <sup>th</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width) |

The 3<sup>rd</sup> to 16<sup>th</sup> 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.

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



## [Gefahrinformation hier einfügen!]



NoteDS401 V2 specifies analog input and/or output data as the default mapping for PDOs 3+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 behaviour of the devices therefore corresponds to DS401 V1, where in all other respects they accord with DS401 V2.



## Communication parameters for the 1st TxPDO

| Index  | Sub-index | Name                     | Туре       | Attribute | Mapping | Default value                | Meaning   |
|--------|-----------|--------------------------|------------|-----------|---------|------------------------------|---|
| 0x1800 | 0         | Number of elements       | Unsigned8  | го        | N       | 5                            | Communica<br>tion<br>parameters<br>for the first<br>transmit<br>PDO. Sub-<br>index 0:<br>number of<br>following<br>parameters |
|        | 1         | COB-ID                   | Unsigned32 | rw        | N       | 0x0000018<br>0 + Node-<br>ID | COB-ID<br>(Communic<br>ation Object<br>Identifier)<br>TxPDO1  |
|        | 2         | Transmissi<br>on Type    | Unsigned8  | rw        | N       | 255                          | Transmissi<br>on type of<br>the PDO   |
|        | 3         | Inhibit Time             | Unsigned16 | rw        | N       | 0                            | Repetition<br>delay [value<br>x 100 µs]   |
|        | 4         | CMS<br>Priority<br>Group | Unsigned8  | rw        | N       | -                            | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used.   |
|        | 5         | Event<br>Timer           | Unsigned16 | rw        | N       | 0                            | Event-<br>Timer   |

Sub-index 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). Sub-index 2 contains the type of transmission, sub-index 3 the repetition delay between two PDOs of the same type, while sub-index 5 contains the event timer. Sub-index 4 is retained for reasons of compatibility, but is not used. (See also the introduction to PDOs.)



## Communication parameters for the 2nd TxPDO

| Index  | Sub-index | Name                     | Туре       | Attribute | Mapping | Default value                | Meaning   |
|--------|-----------|--------------------------|------------|-----------|---------|------------------------------|---|
| 0x1801 | 0         | Number of elements       | Unsigned8  | го        | N       | 5                            | Communica<br>tion<br>parameters<br>for the<br>second<br>transmit<br>PDO. Sub-<br>index 0:<br>number of<br>following<br>parameters |
|        | 1         | COB-ID                   | Unsigned32 | rw        | N       | 0x0000028<br>0 + Node-<br>ID | COB-ID<br>(Communic<br>ation Object<br>Identifier)<br>TxPDO1  |
|        | 2         | Transmissi<br>on Type    | Unsigned8  | rw        | N       | 255                          | Transmissi<br>on type of<br>the PDO   |
|        | 3         | Inhibit Time             | Unsigned16 | rw        | N       | 0                            | Repetition<br>delay [value<br>x 100 µs]   |
|        | 4         | CMS<br>Priority<br>Group | Unsigned8  | rw        | N       | -                            | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used.   |
|        | 5         | Event<br>Timer           | Unsigned16 | rw        | N       | 0                            | Event-<br>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 parameters for the 3rd TxPDO

| Index  | Sub-index | Name                     | Туре       | Attribute | Mapping | Default value                | Meaning   |
|--------|-----------|--------------------------|------------|-----------|---------|------------------------------|---|
| 0x1802 | 0         | Number of elements       | Unsigned8  | ro        | N       | 5                            | Communica<br>tion<br>parameters<br>for the third<br>transmit<br>PDO. Sub-<br>index 0:<br>number of<br>following<br>parameters |
|        | 1         | COB-ID                   | Unsigned32 | rw        | N       | 0x0000038<br>0 + Node-<br>ID | COB-ID<br>(Communic<br>ation Object<br>Identifier)<br>TxPDO1  |
|        | 2         | Transmissi<br>on Type    | Unsigned8  | rw        | N       | 255                          | Transmissi<br>on type of<br>the PDO   |
|        | 3         | Inhibit Time             | Unsigned16 | rw        | N       | 0                            | Repetition<br>delay [value<br>x 100 µs]   |
|        | 4         | CMS<br>Priority<br>Group | Unsigned8  | rw        | N       | -                            | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used.   |
|        | 5         | Event<br>Timer           | Unsigned16 | rw        | N       | 0                            | Event-<br>Timer   |

The third transmit PDO contains analog input data as a rule (see Mapping). 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 for the 4th TxPDO

| Index | Sub-index | Name                     | Туре       | Attribute | Mapping | Default value                | Meaning   |
|-------|-----------|--------------------------|------------|-----------|---------|------------------------------|---|
|       | 0         | Number of elements       | Unsigned8  | го        | N       | 5                            | Communica<br>tion<br>parameters<br>for the<br>fourth<br>transmit<br>PDO. Sub-<br>index 0:<br>number of<br>following<br>parameters |
|       | 1         | COB-ID                   | Unsigned32 | rw        | N       | 0x0000048<br>0 + Node-<br>ID | COB-ID<br>(Communic<br>ation Object<br>Identifier)<br>TxPDO1  |
|       | 2         | Transmissi<br>on Type    | Unsigned8  | rw        | N       | 255                          | Transmissi<br>on type of<br>the PDO   |
|       | 3         | Inhibit Time             | Unsigned16 | rw        | N       | 0                            | Repetition<br>delay [value<br>x 100 µs]   |
|       | 4         | CMS<br>Priority<br>Group | Unsigned8  | rw        | N       | -                            | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used.   |
|       | 5         | Event<br>Timer           | Unsigned16 | rw        | N       | 0                            | Event-<br>Timer   |

The fourth transmit PDO contains analog input data as a rule (see Mapping). 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 for the 5th-16th TxPDOs

| Index   | Sub-index | Name                     | Туре       | Attribute | Mapping | Default value | Meaning   |
|---|-----------|--------------------------|------------|-----------|---------|---------------|---|
| 0x1804-<br>0x180F<br>(dependin<br>g on the<br>device<br>type) | 0         | Number of elements       | Unsigned8  | го        | N       | 5             | Communica<br>tion<br>parameters<br>for the 5 <sup>th</sup> to<br>16 <sup>th</sup><br>transmit<br>PDOs. Sub-<br>index 0:<br>number of<br>following<br>parameters |
|   | 1         | COB-ID                   | Unsigned32 | rw        | N       | 0x0000000     | COB-ID<br>(Communic<br>ation Object<br>Identifier)<br>TxPDO1  |
|   | 2         | Transmissi<br>on Type    | Unsigned8  | rw        | N       | 255           | Transmissi<br>on type of<br>the PDO   |
|   | 3         | Inhibit Time             | Unsigned16 | rw        | N       | 0             | Repetition<br>delay [value<br>x 100 µs]   |
|   | 4         | CMS<br>Priority<br>Group | Unsigned8  | rw        | N       | -             | Present for<br>reasons of<br>backwards<br>compatibilit<br>y, but not<br>used.   |
|   | 5         | Event<br>Timer           | Unsigned16 | rw        | N       | 0             | Event-<br>Timer   |



## **Mapping 1st TxPDO**

| Index  | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value                        | Meaning   |
|--------|-----------|----------------------------------|------------|-----------|---------|--------------------------------------|---|
| 0x1A00 | 0         | Number of elements               | Unsigned8  | rw        | N       | Depending<br>on type and<br>fittings | Mapping<br>parameter<br>of the first<br>transmit<br>PDO; sub-<br>index 0:<br>number of<br>mapped<br>objects.        |
|        | 1         | 1 <sup>st</sup> mapped<br>object | Unsigned32 | rw        | N       | 0x6000010<br>8                       | 1st mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width)             |
|        | 2         | 2 <sup>nd</sup> mapped<br>object | Unsigned32 | rw        | N       | 0x6000020<br>8                       | 2 <sup>nd</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width) |
|        |           |                                  |            |           |         |                                      |   |
|        | 8         | 8 <sup>th</sup> mapped<br>object | Unsigned32 | rw        | N       | 0x6000080<br>8                       | 8 <sup>th</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>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 organised in bytes, the length of the PDO in bytes can be found directly at sub-index 0.

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



## **Mapping 2nd TxPDO**

| Index  | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value                        | Meaning   |
|--------|-----------|----------------------------------|------------|-----------|---------|--------------------------------------|---|
| 0x1A01 | 0         | Number of elements               | Unsigned8  | rw        | N       | Depending<br>on type and<br>fittings | Mapping parameter of the second transmit PDO; sub-index 0: number of mapped objects.                                |
|        | 1         | 1 <sup>st</sup> mapped object    | Unsigned32 | rw        | N       | 0x6401011<br>0                       | 1st mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width)             |
|        | 2         | 2 <sup>nd</sup> mapped<br>object | Unsigned32 | rw        | N       | 0x6401021<br>0                       | 2 <sup>nd</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width) |
|        |           |                                  |            |           |         |                                      |   |
|        | 8         | 8 <sup>th</sup> mapped<br>object | Unsigned32 | rw        | N       |                                      | 8 <sup>th</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>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 organised in words, the length of the PDO in bytes can be found directly at sub-index 0.

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



## Mapping 3rd-16th TxPDO

| Index   | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value                        | Meaning   |
|---|-----------|----------------------------------|------------|-----------|---------|--------------------------------------|---|
| 0x1A02-<br>0x1A0F<br>(dependin<br>g on the<br>device<br>type) | 0         | Number of elements               | Unsigned8  | rw        | N       | Depending<br>on type and<br>fittings | Mapping<br>parameters<br>for the third<br>to sixteenth<br>transmit<br>PDOs; sub-<br>index 0:<br>number of<br>mapped<br>objects. |
|   | 1         | 1 <sup>st</sup> mapped object    | Unsigned32 | rw        | N       | 0x0000000<br>0 (see text)            | 1st mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width)                         |
|   | 2         | 2 <sup>nd</sup> mapped<br>object | Unsigned32 | rw        | N       | 0x0000000<br>0 (see text)            | 2 <sup>nd</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width)             |
|   |           |                                  |            |           |         |                                      |   |
|   | 8         | 8 <sup>th</sup> mapped<br>object | Unsigned32 | rw        | N       | 0x0000000<br>0 (see text)            | 8 <sup>th</sup> mapped<br>application<br>object (2<br>bytes index,<br>1 byte sub-<br>index, 1<br>byte bit<br>width)             |

The 3<sup>rd</sup> to 16<sup>th</sup> 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.

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



## [Gefahrinformation hier einfügen!]



NoteDS401 V2 specifies analog input and/or output data as the default mapping for PDOs 3+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 to object 0x6100) |



| Index  | Meaning  |
|--------|--|
| 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  | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value                        | Meaning  |
|--------|-----------|----------------------------------|------------|-----------|---------|--------------------------------------|--|
| 0x2600 | 0         | Number of elements               | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>available 3-<br>byte special<br>channels,<br>inputs |
|        | 1         | 1 <sup>st</sup> input<br>block   | Unsigned24 | ro        | Υ       | 0x000000                             | 1 <sup>st</sup> input<br>channel                                 |
|        |           |                                  |            |           |         |                                      |  |
|        | 0X80      | 128 <sup>th</sup> input<br>block | Unsigned24 | ro        | Υ       | 0x000000                             | 128 <sup>th</sup> input<br>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              | Sub-index | Name                              | Туре       | Attribute | Mapping | Default value                        | Meaning   |
|--------------------|-----------|-----------------------------------|------------|-----------|---------|--------------------------------------|---|
| 0 <b>x2700</b> 0 1 | 0         | Number of elements                | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>available 3-<br>byte special<br>channels,<br>outputs |
|                    | 1         | 1 <sup>st</sup> output<br>block   | Unsigned24 | rww       | Y       | 0x000000                             | 1 <sup>st</sup> output<br>channel                                 |
|                    |           |                                   |            |           |         |                                      |   |
|                    | 0X80      | 128 <sup>th</sup> output<br>block | Unsigned24 | rww       | Y       | 0x000000                             | 128 <sup>th</sup> 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  | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value                        | Meaning  |
|--------|-----------|----------------------------------|------------|-----------|---------|--------------------------------------|--|
| 0x2800 | 0         | Number of elements               | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>available 4-<br>byte special<br>channels,<br>inputs |
|        | 1         | 1 <sup>st</sup> input<br>block   | Unsigned32 | ro        | Υ       | 0x0000000<br>0                       | 1 <sup>st</sup> input<br>channel                                 |
|        |           |                                  |            |           |         |                                      |  |
|        | 0X80      | 128 <sup>th</sup> input<br>block | Unsigned32 | ro        | Υ       | 0x0000000<br>0                       | 128 <sup>th</sup> input<br>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  | Sub-index | Name                              | Туре       | Attribute | Mapping | Default value                        | Meaning   |
|--------|-----------|-----------------------------------|------------|-----------|---------|--------------------------------------|---|
| 0x2900 | 0         | Number of elements                | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>available 4-<br>byte special<br>channels,<br>outputs |
|        | 1         | 1 <sup>st</sup> output<br>block   | Unsigned32 | rww       | Y       | 0x0000000<br>0                       | 1 <sup>st</sup> output<br>channel                                 |
|        |           |                                   |            |           |         |                                      |   |
|        | 0X80      | 128 <sup>th</sup> output<br>block | Unsigned32 | rww       | Y       | 0x0000000<br>0                       | 128 <sup>th</sup> output<br>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  | Sub-index | Name                            | Туре       | Attribute | Mapping | Default value                        | Meaning  |
|--------|-----------|---------------------------------|------------|-----------|---------|--------------------------------------|--|
| 0x2A00 | 0         | Number of elements              | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>available 5-<br>byte special<br>channels,<br>inputs |
|        | 1         | 1 <sup>st</sup> input<br>block  | Unsigned40 | ro        | Υ       | 0x0000000<br>000                     | 1 <sup>st</sup> input<br>channel                                 |
|        |           |                                 |            |           |         |                                      |  |
|        | 0X40      | 64 <sup>th</sup> input<br>block | Unsigned40 | ro        | Y       | 0x0000000<br>000                     | 64 <sup>th</sup> input<br>channel                                |

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



## 5-byte special terminals, output data

| Index  | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value                        | Meaning   |
|--------|-----------|----------------------------------|------------|-----------|---------|--------------------------------------|---|
| 0x2B00 | 0         | Number of elements               | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>available 5-<br>byte special<br>channels,<br>outputs |
|        | 1         | 1 <sup>st</sup> output<br>block  | Unsigned40 | rww       | Υ       | 0x0000000<br>000                     | 1 <sup>st</sup> output<br>channel                                 |
|        |           |                                  |            |           |         |                                      |   |
|        | 0X40      | 64 <sup>th</sup> output<br>block | Unsigned40 | rww       | Υ       | 0x0000000<br>000                     | 64 <sup>th</sup> output<br>channel                                |

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

## 6-byte special terminals, input data

| Index  | Sub-index | Name                            | Туре       | Attribute | Mapping | Default value                        | Meaning  |
|--------|-----------|---------------------------------|------------|-----------|---------|--------------------------------------|--|
| 0x2C00 | 0         | Number of elements              | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>available 6-<br>byte special<br>channels,<br>inputs |
|        | 1         | 1 <sup>st</sup> input<br>block  | Unsigned48 | ro        | Υ       | 0x0000000<br>000                     | 1 <sup>st</sup> input<br>channel                                 |
|        |           |                                 |            |           |         |                                      |  |
|        | 0X40      | 64 <sup>th</sup> input<br>block | Unsigned48 | ro        | Υ       | 0x0000000<br>000                     | 64 <sup>th</sup> input<br>channel                                |

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

# 6-byte special terminals, output data

| Index  | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value                        | Meaning   |
|--------|-----------|----------------------------------|------------|-----------|---------|--------------------------------------|---|
| 0x2D00 | 0         | Number of elements               | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>available 6-<br>byte special<br>channels,<br>outputs |
|        | 1         | 1 <sup>st</sup> output<br>block  | Unsigned48 | rww       | Υ       | 0x0000000<br>000                     | 1 <sup>st</sup> output<br>channel                                 |
|        |           |                                  |            |           |         |                                      |   |
|        | 0X40      | 64 <sup>th</sup> output<br>block | Unsigned48 | rww       | Υ       | 0x0000000<br>000                     | 64 <sup>th</sup> output<br>channel                                |

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



## 8-byte special terminals, input data

| Index  | Sub-index | Name                            | Туре       | Attribute | Mapping | Default value                        | Meaning  |
|--------|-----------|---------------------------------|------------|-----------|---------|--------------------------------------|--|
| 0x3000 | 0         | Number of elements              | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>available 6-<br>byte special<br>channels,<br>inputs |
|        | 1         | 1 <sup>st</sup> input<br>block  | Unsigned64 | ro        | Υ       | 0x0000000<br>000                     | 1 <sup>st</sup> input<br>channel                                 |
|        |           |                                 |            |           |         |                                      |  |
|        | 0x40      | 64 <sup>th</sup> input<br>block | Unsigned64 | ro        | Υ       | 0x0000000<br>000                     | 64 <sup>th</sup> input<br>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  | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value                        | Meaning   |
|--------|-----------|----------------------------------|------------|-----------|---------|--------------------------------------|---|
| 0x3100 | 0         | Number of elements               | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>available 6-<br>byte special<br>channels,<br>outputs |
|        | 1         | 1 <sup>st</sup> output<br>block  | Unsigned64 | rww       | Υ       | 0x0000000<br>000                     | 1 <sup>st</sup> output<br>channel                                 |
|        |           |                                  |            |           |         |                                      |   |
|        | 0X40      | 64 <sup>th</sup> output<br>block | Unsigned64 | rww       | Υ       | 0x0000000<br>000                     | 64 <sup>th</sup> output<br>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  | Sub-index | Name               | Туре       | Attribute | Mapping | Default value | Meaning                               |
|--------|-----------|--------------------|------------|-----------|---------|---------------|---------------------------------------|
| 0x4500 | 0         | Register<br>Access | Unsigned32 | rw        | N       |               | Access to internal bus node registers |

## The 32 bit value is composed as follows:

| MSB                                     |                 |                          | LSB                     |
|---|-----------------|--------------------------|-------------------------|
| Access (bit 7) + table number (bits 60) | Register number | High byte register value | Low byte register value |
| [01] + [00x7F]                          | [00xFF]         | [00xFF]                  | [00xFF]                 |

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.

69



### 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/sub-index combination, with:

- table number (access bit = 0) in byte 3
- register address in byte 2 of the 32-bit data value.

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 sub-index.

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 4500, sub-index 0, with the 32 bit data value 0x64 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/sub-index. 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

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 sub-index 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 |                | Corresponding<br>SDO download<br>value (0x4500/0) |
|------|-------|----------|----------------|---|
| 1.   | 99    | 2        | 45054 (0xAFFE) | 0xE3 02 AF FE<br>(0xE3=0x63(=99)+<br>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, sub-index 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 n<sup>th</sup> terminal:

| Table number | Register number | Description                                     | Value range |
|--------------|-----------------|---|-------------|
| 9            | 0               | Bus station identifier                          | 0 - 65535   |
| 9            |                 | 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. The Fieldbus Box modules contain the identifier 510 dec = 0x1FE in register 0.

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<sub>dec</sub> (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 | а | е |

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

This identifier scheme results in the terminal descriptions listed below:

| Extension module identifier | 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 input/output terminal, e.g. KL2212                  |

### 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 | MSB |   |   |   |   |   | LSB |   |   |   |   |   |   |   |   |
|-----|-----|---|---|---|---|---|-----|---|---|---|---|---|---|---|---|
| 0   | 0   | 0 | 0 | 0 | 0 | 0 | 0   | 0 | 0 | 0 | 0 | 0 | D | G | Α |

#### 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

Reporting (by means of emergency telegram), that, for example

- a current input is open circuit (with diagnostics)
- 10 V exceeded at a 1-10V input terminal
- 0: Device diagnostics switched off
- 1: Device diagnostics active (default)

### D: Diagnostic data

from digital terminals is included in 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 | а | 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: User data only (default)

1: Complete evaluation (note: 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  | Sub-index | Name                              | Туре       | Attribute | Mapping | Default value | Meaning   |
|--------|-----------|-----------------------------------|------------|-----------|---------|---------------|---|
| 0x4501 | 0         | Access<br>Terminal<br>Register    | Unsigned8  | ro        | N       | none          | Index 0x4501 allows access to all the registers in the bus terminal or extension module. Sub-index 0 contains the number of attached bus terminals. |
|        | 1         | Access<br>Reg.<br>Terminal 1      | Unsigned32 | rw        | N       | none          | Access to<br>bus<br>terminal or<br>extension<br>module<br>register 1  |
|        | 0XFE      | Access<br>Reg.<br>Terminal<br>254 | Unsigned32 |           | 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 (bits 60) | Register number | High byte register value | Low byte register value |
| [01] + [00x7F]                            | [00xFF]         | [00xFF]                  | [00xFF]                 |

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 sub-index; 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/sub-index combination, with:

- channel number (access bit = 0) in byte 3
- register address in byte 2 of the 32-bit data value.

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 sub-index.

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 4501, sub-index 5 with 32 bit data value 01 20 00 00 (0x01 = 2nd channel, 0x20 = 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/sub-index. 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 sub-index 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.

# **NOTICE**

# [Gefahrinformation hier einfügen!]

WarningIf the write protection is not removed (as a result, for instance, of a faulty codeword), 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 codeword is written for this purpose into register 31 of the channel concerned:

| Write protection | Channel     | Register  |     | Corresponding<br>SDO download<br>value (0x4500/0) |
|------------------|-------------|-----------|-----|---|
|                  | 1,2, 3 or 4 | 31 (0x1F) | , , | 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, sub-index 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  | Sub-index | Name                        | Туре       | Attribute | Mapping | Default value | Meaning  |
|--------|-----------|-----------------------------|------------|-----------|---------|---------------|--|
| 0x5500 | 0         | Activate<br>PDO<br>Defaults | Unsigned32 | rw        | N       | 0             | sets PDO<br>communica<br>tion<br>parameters<br>for PDOs<br>211 |

CANopen defines default identifiers for 4 transmit (Tx) and 2 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 PDO2...11. PDOs that do not have process data (and which are thus superfluous in the present configuration) are not activated.



### [Gefahrinformation hier einfügen!]

NoteThis 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<br>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 10ms (= $100 \times 100\mu s$ ), 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            |

# **NOTICE**

# [Gefahrinformation hier einfügen!]

WarningEnsure 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/<br>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  | Sub-index | Name                             | Туре      | Attribute | Mapping | Default value                        | Meaning   |
|--------|-----------|----------------------------------|-----------|-----------|---------|--------------------------------------|---|
| 0x6000 | 0         | Number of elements               | Unsigned8 | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>available<br>digital 8-bit<br>input data<br>blocks |
|        | 1         | 1 <sup>st</sup> input<br>block   | Unsigned8 | ro        | Y       | 0x00                                 | 1 <sup>st</sup> input<br>channel                                |
|        |           |                                  |           |           |         |                                      |   |
|        | 0XFE      | 254 <sup>th</sup> input<br>block | Unsigned8 | ro        | Y       | 0x00                                 | 254 <sup>th</sup> input channel                                 |

#### Interrupt mask

| Index  | Sub-index | Name                | Туре       | Attribute | Mapping | Default value     | Meaning   |
|--------|-----------|---------------------|------------|-----------|---------|-------------------|---|
| 0x6126 | 0         | Number of elements  | Unsigned8  | ro        | N       | Depending on type | The number of 32-bit interrupt masks = 2 x the number of TxDPOs |
|        | 1         | IR-Mask0<br>TxPDO1  | Unsigned32 | rw        | N       | 0xFFFFFF<br>F     | IR-mask<br>bytes 03<br>TxPDO1                                   |
|        | 2         | IR-Mask1<br>TxPDO1  | Unsigned32 | rw        | N       | 0xFFFFFF<br>F     | IR-mask<br>bytes 47<br>TxPDO1                                   |
|        | 3         | IR-Mask0<br>TxPDO2  | Unsigned32 | rw        | N       | 0xFFFFFF<br>F     | IR-mask<br>bytes 03<br>TxPDO2                                   |
|        |           |                     |            |           |         |                   |   |
|        | 0x20      | IR-Mask1<br>TxPDO16 | Unsigned32 | rw        | N       | 0xFFFFFF<br>F     | IR-mask<br>bytes 47<br>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.

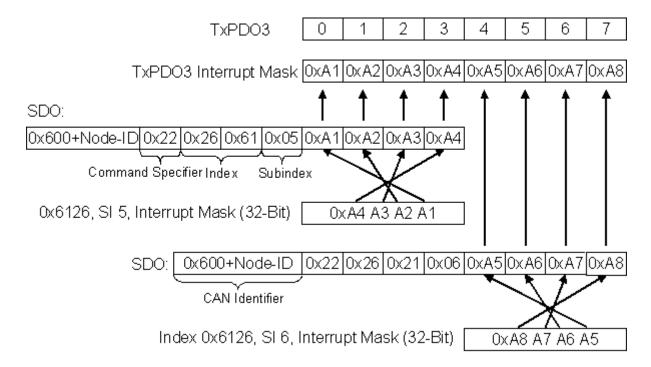
The interrupt mask only has an effect on TxPDOs with transmission types 254 and 255. It is not stored in the device (not even through object 0x1010). Changes to the mask at runtime (when the status is operational) are possible, and are evaluated starting from the next change of input data.

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.



### **Example of 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, sub-index5 must receive the value 0x0000 00FF and that sub-index6 must have 0xFFFF FF00 written into it.

The corresponding SDOs therefore appear as follows:

| 11 bit identifier | 8 bytes of | user data |      |      |      |      |      |      |
|-------------------|------------|-----------|------|------|------|------|------|------|
| 0x600+<br>node ID | 0x22       | 0x26      | 0x61 | 0x05 | 0xFF | 0x00 | 0x00 | 0x00 |

| 11 bit identifier | 8 bytes of | user data |      |      |      |      |      |      |
|-------------------|------------|-----------|------|------|------|------|------|------|
| 0x600+<br>node ID | 0x22       | 0x26      | 0x61 | 0x06 | 0x00 | 0xFF | 0xFF | 0xFF |

# **Digital outputs**

| Index  | Sub-index | Name                             | Туре      | Attribute | Mapping | Default value                        | Meaning  |
|--------|-----------|----------------------------------|-----------|-----------|---------|--------------------------------------|--|
| 0x6200 | 0         | Number of elements               | Unsigned8 | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>available<br>digital 8-bit<br>output data<br>blocks |
|        | 1         | 1 <sup>st</sup> input<br>block   | Unsigned8 | rw        | Υ       | 0x00                                 | 1 <sup>st</sup> output<br>channel                                |
|        |           |                                  |           |           |         |                                      |  |
|        | 0XFE      | 254 <sup>th</sup> input<br>block | Unsigned8 | rw        | Υ       | 0x00                                 | 254 <sup>th</sup> output<br>channel                              |



### **Analog inputs**

| Index  | Sub-index | Name                    | Туре       | Attribute | Mapping | Default value                        | Meaning   |
|--------|-----------|-------------------------|------------|-----------|---------|--------------------------------------|---|
| 0x6401 | 0         | Number of elements      | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>analog<br>input<br>channels<br>available |
|        | 1         | 1 <sup>st</sup> input   | Unsigned16 | ro        | Υ       | 0x0000                               | 1 <sup>st</sup> input<br>channel                      |
|        |           |                         |            |           |         |                                      |   |
|        | 0XFE      | 254 <sup>th</sup> input | Unsigned16 | ro        | Υ       | 0x0000                               | 254 <sup>th</sup> 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  | Sub-index | Name                             | Туре       | Attribute | Mapping | Default value                  | Meaning  |
|--------|-----------|----------------------------------|------------|-----------|---------|--------------------------------|--|
| 0x6411 | 0         | Number of elements               | Unsigned8  | ro        | N       | Depending on type and fittings | Number of<br>analog<br>output<br>channels<br>available |
|        | 1         | 1 <sup>st</sup> input<br>block   | Unsigned16 | rw        | Υ       | 0x0000                         | 1 <sup>st</sup> output<br>channel                      |
|        |           |                                  |            |           |         |                                |  |
|        | 0XFE      | 254 <sup>th</sup> input<br>block | Unsigned16 | rw        | Υ       | 0x0000                         | 254 <sup>th</sup> 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  | Sub-index | Name                          | Туре    | Attribute | Mapping | Default value | Meaning   |
|--------|-----------|-------------------------------|---------|-----------|---------|---------------|---|
| 0x6423 | 0         | Global<br>Interrupt<br>Enable | Boolean | rw        | N       |               | Activates<br>the event-<br>driven<br>transmissio<br>n of PDOs<br>with analog<br>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, sub-index 3) and/or limit value monitoring (objects 0x6424 + 0x6425) and/or delta function (object 0x6426).



# Upper limit value analog inputs

| Index  | Sub-index | Name                                   | Туре       | Attribute | Mapping | Default value                        | Meaning  |
|--------|-----------|--|------------|-----------|---------|--------------------------------------|--|
| 0x6424 | 0         | Number of elements                     | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of analog input channels available                      |
|        | 1         | upper limit<br>1 <sup>st</sup> input   | Unsigned16 | rw        | Υ       | 0x0000                               | Upper limit value for 1st input channel                        |
|        |           |  |            |           |         |                                      |  |
|        | 0XFE      | upper limit<br>254 <sup>th</sup> input | Unsigned16 | rw        | Υ       | 0x0000                               | Upper limit<br>value for<br>254 <sup>th</sup> input<br>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  | Sub-index | Name                                   | Туре       | Attribute | Mapping | Default value                        | Meaning  |
|--------|-----------|--|------------|-----------|---------|--------------------------------------|--|
| 0x6425 | 0         | Number of elements                     | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of<br>analog<br>input<br>channels<br>available          |
|        | 1         | lower limit<br>1st input               | Unsigned16 | rw        | Y       | 0x0000                               | Lower limit<br>value for 1st<br>input<br>channel               |
|        |           |  |            |           |         |                                      |  |
|        | 0XFE      | lower limit<br>254 <sup>th</sup> input | Unsigned16 | rw        | Y       | 0x0000                               | Lower limit<br>value for<br>254 <sup>th</sup> input<br>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  | Sub-index | Name                                   | Туре       | Attribute | Mapping | Default value                        | Meaning  |
|--------|-----------|--|------------|-----------|---------|--------------------------------------|--|
| 0x6426 | 0         | Number of elements                     | Unsigned8  | ro        | N       | Depending<br>on type and<br>fittings | Number of analog input channels available                    |
|        | 1         | delta value<br>1 <sup>st</sup> input   | Unsigned16 | rw        | Y       | 0x0000                               | Delta value<br>for the 1 <sup>st</sup><br>input<br>channel   |
|        |           |  |            |           |         |                                      |  |
|        | 0XFE      | delta value<br>254 <sup>th</sup> input | Unsigned16 | rw        | Y       | 0x0000                               | Delta value<br>for the 254 <sup>th</sup><br>input<br>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).

# 5 Commissioning

# 5.1 Mounting

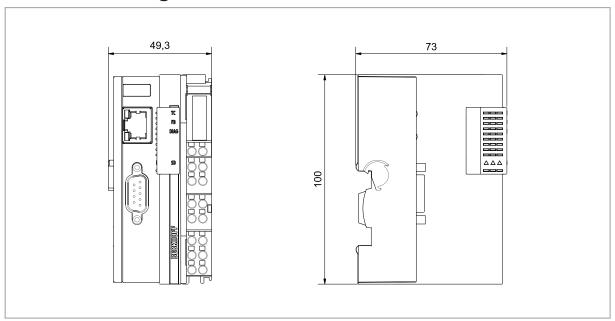


Fig. 17: CX70xx Embedded PC, dimensions.

# 5.1.1 Note the permissible installation positions

# **NOTICE**

### Overheating

The Embedded PC may overheat if the installation position is incorrect or the minimum distances are not adhered to. Adhere to the maximum ambient temperature of 60°C and the mounting instructions.

Install the Embedded PC horizontally in the control cabinet on a DIN rail, in order to ensure optimum heat dissipation.

Note the following specifications for the control cabinet:

- The Embedded PC should only be operated at ambient temperatures between -25 °C and 60 °C. Measure the temperature below the Embedded PC at a distance of 30 mm to the cooling fins, in order to determine the ambient temperature correctly.
- Adhere to the minimum distances of 30 mm above and below the Embedded PC.
- Additional electrical equipment affects the heat generation in the control cabinet. Select a suitable
  control cabinet enclosure depending on the application, or ensure that excess heat is dissipated from
  the control cabinet.

The Embedded PC must be mounted horizontally on the DIN rail. Ventilation openings are located at the top and bottom of the housing. This ensures an optimum airflow through the Embedded PC in vertical direction. In addition, a minimum clearance of 30 mm above and below the Embedded PC is required, in order to ensure adequate ventilation.



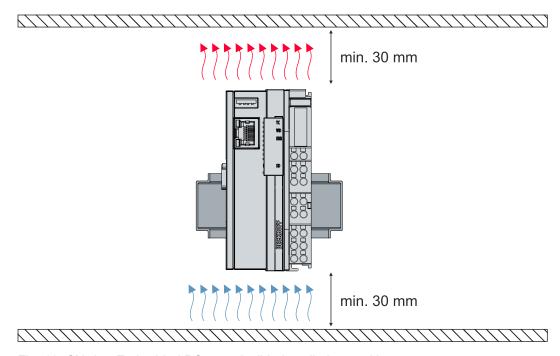


Fig. 18: CX70xx Embedded PC, permissible installation position.

If vibrations and impact occur in the same direction as the DIN rail, the Embedded PC must be secured with an additional bracket, in order to prevent it slipping.

# Installation positions with reduced temperature range up to 45 °C

You can also mount the Embedded PC vertically or horizontally on the mounting rail. Note that you can then only operate the Embedded PC up to an ambient temperature of 45 °C.

Ensure that Bus Terminals that are connected to the Embedded PC are designed for operation in vertical or horizontal position.

# Restrictions for E-bus/K-bus current

The maximum E-bus/K-bus current varies depending on the selected installation position and the ambient temperature.

Table 5: Maximum E-bus/K-bus current depending on the selected installation position and the ambient temperature.

| E-bus/K-bus current | Installation position | Ambient temperature |
|---------------------|-----------------------|---------------------|
| max. 1.5 A          | variable              | -2545 °C            |
| max. 1.3 A          | horizontal            | -2555 °C            |
| max. 1 A            | variable              | -2555 °C            |
| max. 1 A            | horizontal            | -2560 °C            |



# 5.1.2 Fastening to the DIN rail

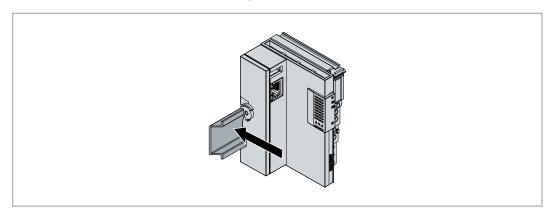
The housing is designed such that the Embedded PC can be pushed against the DIN rail and latched onto it.

### Requirements:

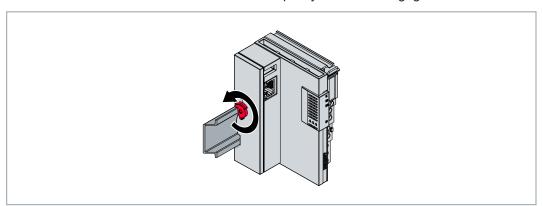
• DIN rail of the type TS35/7.5 or TS35/15 according to EN 60715.

# Fasten the Embedded PC to the DIN rail as follows:

1. Place the Embedded PC on the DIN rail. Slightly press the Embedded PC onto the DIN rail until a soft click can be heard and the Embedded PC has latched.



- 2. Subsequently, lock the catch on the left side of the Embedded PC.
- 3. Turn the latch counter clockwise until the latch quietly clicks and engages.



⇒ You have installed the Embedded PC successfully. Check again that the mounting is correct and that the Embedded PC is engaged on the DIN rail.



# 5.1.3 Changing the MicroSD card



#### Loss of data



MicroSD cards are subjected to heavy load during operation and have to withstand many write cycles and extreme ambient conditions. MicroSD cards from other manufacturer may fail, resulting in data loss.

Only use industrial MicroSD cards provided by Beckhoff.

The MicroSD card slot is intended for an industrially compatible MicroSD card. The firmware of the Embedded PC is stored on the MicroSD card. If necessary, the MicroSD card can be written to from TwinCAT 3, allowing user-defined data to be stored.

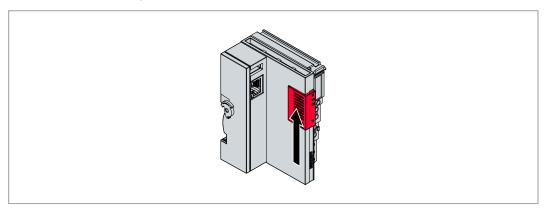
The eject mechanism is based on the push/push principle. Below, we show you how to change the MicroSD card.

#### Requirements:

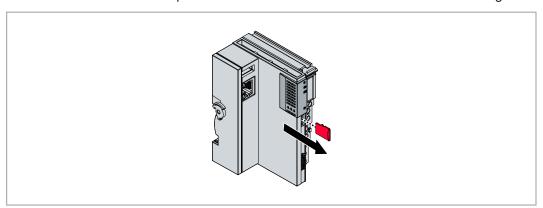
• The Embedded PC must be switched off. The MicroSD card may only be installed or removed in switched-off state.

#### Changing the MicroSD card

1. Push the black cover upwards.



- 2. Gently push the MicroSD card.
- 3. The card is unlatched with a quiet click and raised about 2 3 mm out of the housing.



- 4. Push the new MicroSD card into the card slot with the contacts at the front. The contacts face to the right.
- 5. A soft click can be heard when the MicroSD card engages.
- ⇒ The card is seated correctly when it is about 1 mm deeper than the front side of the housing.



# 5.1.4 Installing passive EtherCAT Terminals

# **Incorrectly installed passive EtherCAT Terminals**

1

The E-bus signal between an Embedded PC and the EtherCAT Terminals can be impaired due to incorrectly installed passive EtherCAT Terminals.

Passive EtherCAT Terminals should not be installed directly on the power supply unit.

EtherCAT Terminals that do not take part in active data exchange are referred to as passive terminals. Passive EtherCAT Terminals have no process image and do not require current from the terminal bus (E-bus).

Passive EtherCAT Terminals (e.g. EL9195) can be detected in TwinCAT. In the tree structure the EtherCAT Terminal is displayed without process image, and the value in column "E-bus (mA)" does not change, compared to the preceding EtherCAT Terminal.

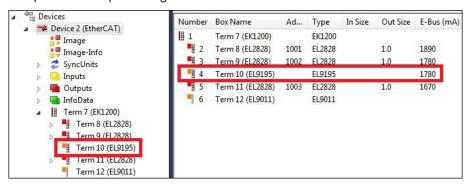


Fig. 19: Identifying a passive EtherCAT Terminal in TwinCAT.

The entry "Current consumption via E-Bus" in the technical data of an EtherCAT Terminal indicates whether a particular EtherCAT Terminal requires power from the terminal bus (E-bus).

The following diagram shows the permissible installation of a passive EtherCAT Terminal. The passive EtherCAT Terminal was not directly attached to the power supply unit.

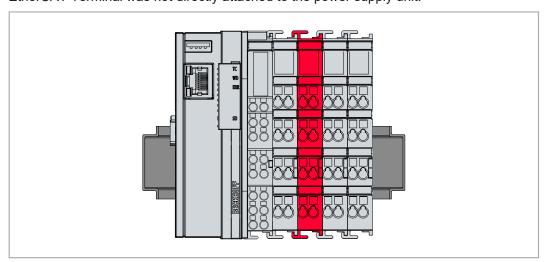


Fig. 20: Passive EtherCAT Terminals, permissible installation.



# 5.2 Power supply

### **NOTICE**

# Damage to the Embedded PCs

The Embedded PCs may be damaged during wiring. The cables for the power supply should only be connected in de-energized state.

The power supply terminals require an external voltage source, which provides 24 V DC (-15% / +20%).

The cabling of the Embedded PC in the control cabinet must be done in accordance with the standard EN 60204-1:2006 (PELV = Protective Extra Low Voltage):

- The "PE" and "0 V" conductors of the voltage source for a basic CPU module must be on the same potential (connected in the control cabinet).
- Standard EN 60204-1:2006, section 6.4.1:b stipulates that one side of the circuit, or one point of the energy source for this circuit must be connected to the protective earth conductor system.

#### Connections

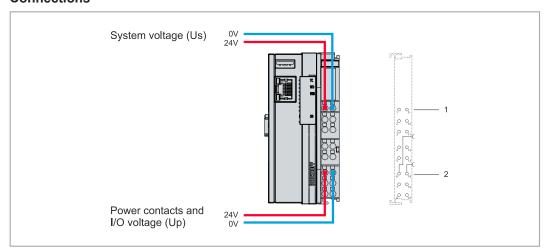


Fig. 21: Connections for system voltage (Us) and power contacts (Up).

Table 6: Legend for the connection example.

| No. | Description  |
|-----|--|
| 1   | The upper spring-loaded terminals with the designations "+24 V Us" and "0 V Us" supply the basic CPU module and the terminal bus (data transmission via K-bus or E-bus) with voltage.        |
| 2   | The upper spring-loaded terminals with the designations "+24 V Up" and "0 V Up" supply the multifunction I/Os, the Bus Terminals and EtherCAT terminals with voltage via the power contacts. |

#### **Fuse**

- When dimensioning the fuse for the system voltage (Us), observe the maximum power consumption of the Embedded PC (see: <u>Technical data</u> [▶ 191])
- Protect the power contacts (Up) with a fuse with a max. rating of 10 A (slow-blow).

### Interrupting / switching off the power supply

To switch off the Embedded PC, do not disconnect the ground (0 V), because otherwise current may continue to flow via the shielding, depending on the device, and damage the Embedded PC or peripheral devices.

Always disconnect the 24 V line. Devices connected to the Embedded PC, which have their own power supply (e.g. a Panel) must have the same potential for "PE" and "0 V" as the Embedded PC have (no potential difference).



# 5.2.1 Connect Embedded PC

The cables of an external voltage source are connected to spring-loaded terminals on the power supply terminal. Observe the required conductor cross-sections and strip lengths.

Table 7: Required wire cross-sections and strip lengths.

| Conductor cross-section | e*: 0.08 1.5 mm <sup>2</sup> | e*: AWG 28 16 |
|-------------------------|------------------------------|---------------|
|                         | f*: 0.25 1.5 mm <sup>2</sup> | f*: AWG 22 16 |
|                         | a*: 0.14 0.75 mm²            | a*: AWG 26 19 |
| Strip length            | 8 9 mm                       | 0.33 inch     |

<sup>\*</sup>e: single-wire, solid wire; f: stranded wire; a: with wire end sleeve

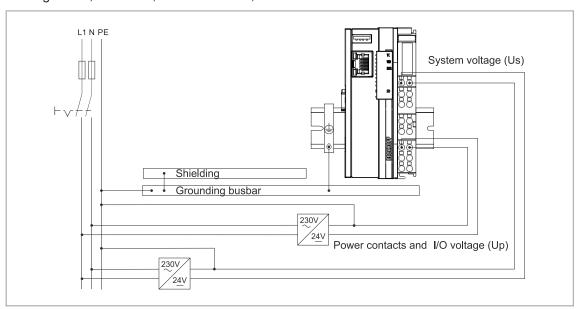
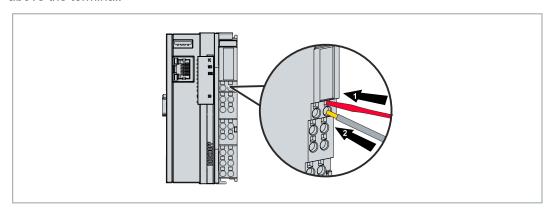


Fig. 22: Connection example with a CX7000.

### Connect the Embedded PC as follows:

1. Open a spring-loaded terminal by slightly pushing with a screwdriver or a rod into the square opening above the terminal.



- 2. The wire can now be inserted into the round terminal opening without any force.
- 3. The terminal closes automatically when the pressure is released, holding the wire safely and permanently.
- ⇒ You have successfully connected the voltage source to the power supply terminal when the two upper LEDs of the power supply terminal light up green.

The left LED (Us 24V) indicates the supply of the basic CPU module and terminal bus. The red LED (Up 24V) indicates the Bus Terminal supply via the power contacts.



# 5.2.2 UL requirements

The CX7051 Embedded PCs are UL-certified. The corresponding UL label can be found on the name plate.

The CX7051 Embedded PCs can thus be used in areas where special UL requirements have to be met. These requirements apply to the system voltage (Us) and the power contacts (Up). Applications without special UL requirements are not affected by UL regulations.

### UL requirements:

- The Embedded PCs must not be connected to unlimited voltage sources.
- Embedded PCs may only be supplied from a 24 V DC voltage source. The voltage source must be insulated and protected with a fuse of maximum 4 A (corresponding to UL248).
- Or the power supply must originate from a voltage source that corresponds to NEC class 2. An NEC class 2 voltage source must not be connected in series or parallel with another NEC class 2 voltage source.

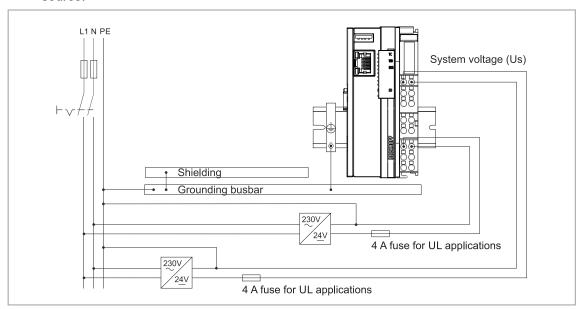


Fig. 23: Connection example for areas with special UL requirements.



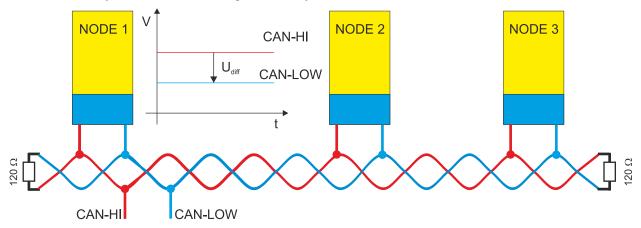
# 5.3 CANopen: Connection and wiring

# **NOTICE**

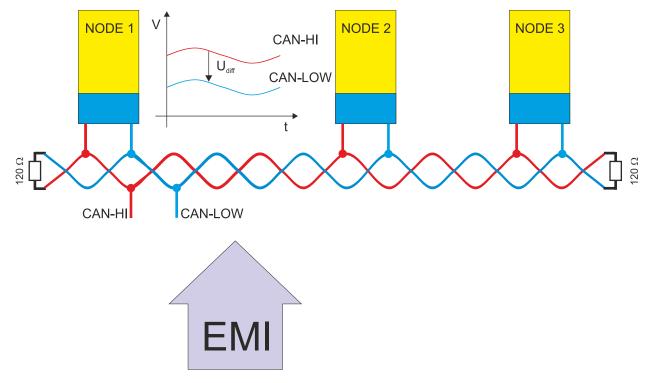
### Improper wiring

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.

CAN is a 2-wire bus system, to which all 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 termination resistor to prevent reflections. This is also necessary even if the cable lengths are very short!

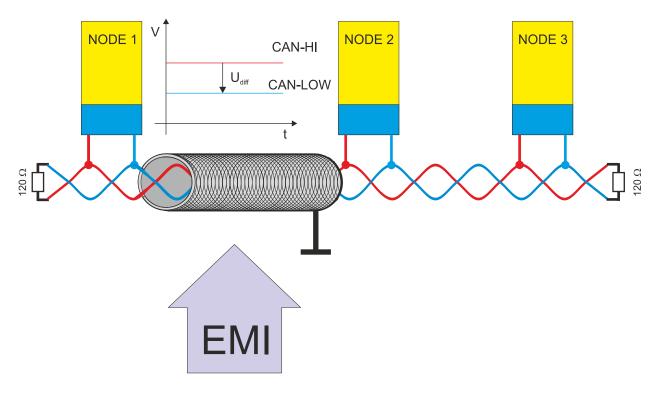


Since the CAN signals are represented as differential levels on the bus, the CAN line is comparatively insensitive to interference (EMI). Both lines are affected, so the interference hardly changes the differential level.



Additional shielding of the twisted wires can be used to further reduce EMI interference.





### **Bus length**

The maximum length of a CAN bus is primarily limited by the signal propagation time. The multi-master bus access method (arbitration) requires that the signals are present quasi-simultaneously (before sampling within a bit time) at all nodes. Since the signal propagation time in the CAN connections (transceiver, optocoupler, CAN controller) are almost constant, the cable length must be adapted to 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   |

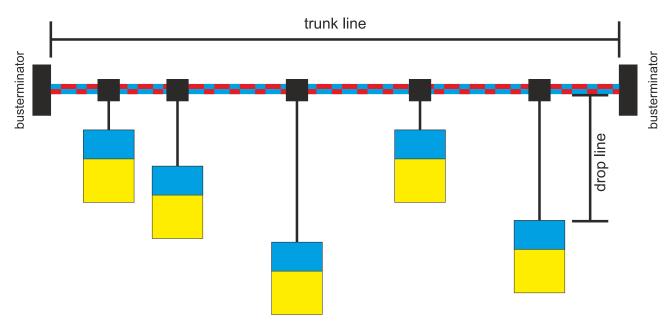
<sup>\*)</sup> Often you can find the specification 40 m at 1 Mbit/s in the literature for CAN. However, this does not apply to networks with opto-decoupled CAN controllers. The worst case calculation for opto-couplers yields a figure 5 m at 1 Mbit/s - in practice, however, 20 m can be reached without difficulty.

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

# **Drop lines**

Drop lines must always be avoided as far as possible, since they inevitably cause signal reflections. The reflections caused by drop lines are not however usually critical, provided they have decayed fully before the sampling time.





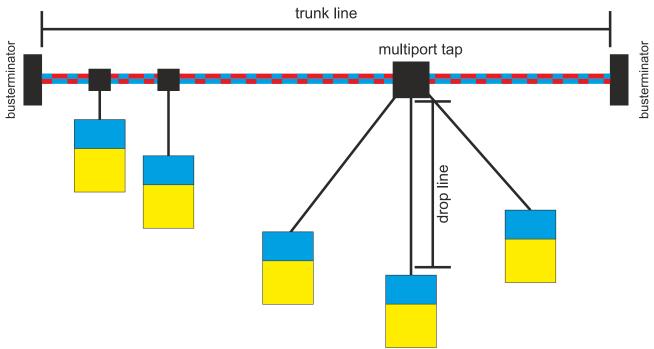
In the case of the bit timing settings selected in the bus couplers it can be assumed that this is the case, provided the following drop line lengths are not exceeded:

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

Drop lines must not have termination resistors.

# **Star Hub (Multiport Tap)**

When using passive distributors ("Multiport Taps"), e.g. the BECKHOFF distribution box ZS5052-4500, shorter drop line lengths must be maintained.



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                                |

# 5.3.1 D-sub connector (X003)

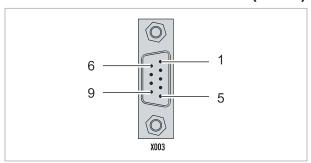


Fig. 24: CANopen interface X003.

The CAN bus line is connected via a 9-pin D-sub connector with the following pin assignment:

| Pin | Connection                                 |
|-----|--|
| 1   | not used                                   |
| 2   | CAN low (CAN-)                             |
| 3   | CAN Ground (internally connected to pin 6) |
| 4   | not used                                   |
| 5   | Shield                                     |
| 6   | CAN Ground (internally connected to pin 3) |
| 7   | CAN high (CAN+)                            |
| 8   | not used                                   |
| 9   | not used                                   |

The DIN rail contact spring and the connector shield are connected together. An auxiliary voltage of up to  $30~V_{DC}$  may be connected to pin 9, which is used by some CAN devices to supply the transceivers.



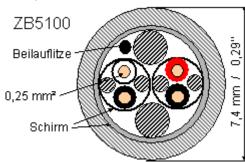
# 5.3.2 Cable and shielding

Shielded 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 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 mm<sup>2</sup> (AWG 24) twisted pairs, cable colors: red/black + white/black
- · double shielded
- shield braid 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 MHz): 120 ohm
- conductor resistance < 80 ohm/km</li>
- sheath: gray PVC, outer diameter 7.3 +/- 0.4 mm
- · weight: 64 kg/km.
- printed with "Beckhoff ZB5100 CAN-BUS 2x2x0.25" and meter marking (length data every 20 cm)

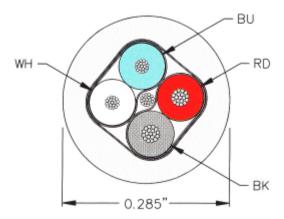


### **ZB5200 CAN/DeviceNet Cable**

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 mm<sup>2</sup> (AWG 22) twisted pairs
- · double shielded · shield braid with filler strand
- · characteristic impedance (1 MHz): 126 ohm
- · conductor resistance 54 ohm/km
- sheath: gray PVC, outer 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





# **Shielding**

The shield 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 shielding, in which HF interference is diverted through R/C elements to the mounting rail assumes that the rail is appropriately grounded 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 shield of the bus cable. In that case the shield should not be attached to the couplers - it should nevertheless still be fully connected through.

# Cable colors

Recommended application of Beckhoff CAN cables:

| Function   | ZB5100 cable color | ZB5200 cable color |
|------------|--------------------|--------------------|
| CAN Ground | black /(red)       | black              |
| CAN Low    | black              | blue               |
| Shield     | Filler strand      | Filler strand      |
| CAN high   | white              | white              |
| not used   | (red)              | (red)              |



# 6 Multifunction I/Os

A total of four adjustable slots are available for configuring the operation modes. A slot is a certain number of inputs and outputs. For each slot a maximum of one module (DI, DIO, ENC, CNT or PWM) can be assigned, which in turn determines the operation mode for the respective slot. A module is therefore a function that these inputs and outputs can assume. The current module configuration is listed in TwinCAT under the CX7028 interface. Note that the CX7028 interface for controlling the multifunction I/Os has its own CPU and the CX7028 interface is not displayed or does not work under TwinCAT if the power supply (Up) is not connected.

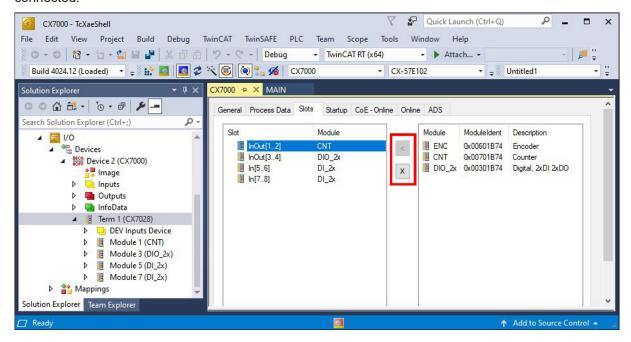


Fig. 25: CX7028 interface, slot and module configuration under TwinCAT.

Modules can be assigned to a specific slot with the button < or removed again with  $\mathbf{x}$ . There is a choice of different modules depending on the slot used. The module used by each slot is listed in the following.

#### Cycle time for multifunction I/Os

Communication to the multifunction I/Os is monitored with a fixed watchdog of 100 ms. This means that the cycle time for the multifunction I/O must be faster than 100 ms.

#### Slot 1:

When using slot 1, inputs 1, 2 and (\*3) as well as outputs 1 and 2 are configured.

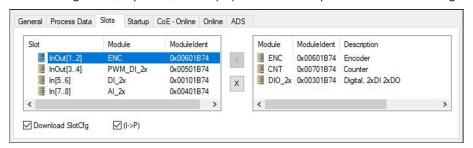


Fig. 26: Supported modules when using slot 1.

- ENC (incremental encoder mode). 2 x digital input for 250 kHz encoder signal, 2 x encoder digital output.
- CNT (counter mode). 1 x counter digital input 100 kHz, 1 x digital input as up/down counter 20 kHz, 2 x counter digital output.
- DIO\_2x (digital inputs and outputs). 2 x digital input, 24 V DC, filter 3 ms, type 3, 2 x digital output, 24 V DC, 0.5 A, 1-wire technique.



\*) Input 3 is only available in incremental encoder mode. If the level is high, the value of the incremental encoder can be latched or the counter reset.

#### Slot 2:

When using slot 2, inputs 3 and 4 as well as outputs 3 and 4 are configured.

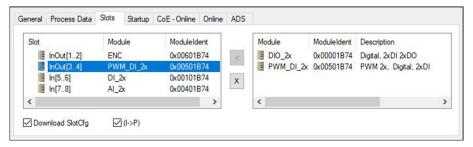


Fig. 27: Supported modules when using slot 2.

- DIO\_2x (digital inputs and outputs). 2 x digital input, 24 V DC, filter 3 ms, type 3, 2 x digital output, 24 V DC, 0.5 A, 1-wire technique.
- PWM\_DI\_2x (PWM signal mode). 2 x digital input, 24 V DC, filter 3 ms, 2 x digital output configured for PWM signal.

#### Slot 3:

When using slot 3, inputs 5 and 6 are configured.

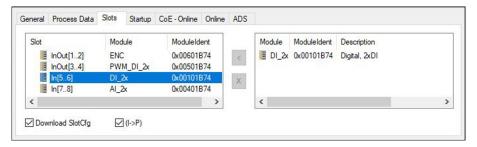


Fig. 28: Supported modules when using slot 3.

Slot 3 contains only one module and therefore cannot be configured differently. The module supports 2 x digital input, 24 V DC, filter 3 ms, type 3.

#### Slot 4:

When using slot 4, inputs 7 and 8 are configured.

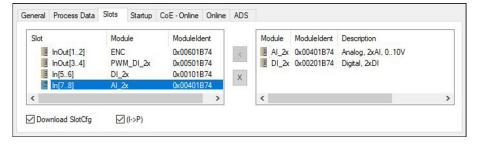


Fig. 29: Supported modules when using slot 4.

- Al 2x (analog signal mode). 2 x digital input configured as analog input 0 to 10 V, 12 bits
- DI\_2x (digital input). 2 x digital input, 24 V DC, filter 3 ms, type 3



# 6.1 Digital inputs

The digital inputs acquire binary control signals from the process level. Typically, these are mechanical contacts such as normally closed contacts or normally open contacts, electronic sensors such as inductive proximity switches, optical sensors or other methods in order to generate a low/high signal in the sense of control technology. Thanks to integrated multi-function I/Os, the CX70xx has a total of 8 digital inputs, 24 V DC, filter 3 ms, type 3.

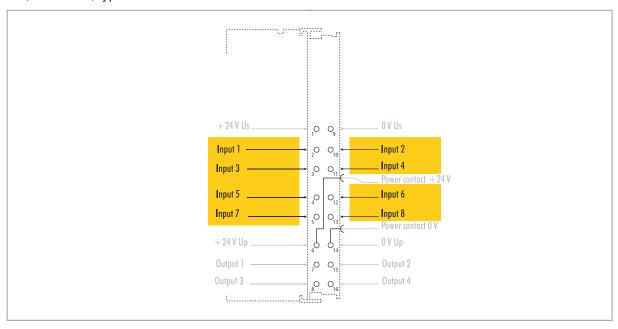


Fig. 30: Configurable digital inputs.

The digital inputs have a 3 ms input filter. The signal status of each individual input is displayed by an LED. For digital inputs 3, 4, 5 and 6, additional filter settings can be made in the appropriate CoE objects and, for example, the resolution and filter time can be set.

Table 8: Technical data, multi-function I/Os as digital inputs.

| Technical data               | CX7051   |
|------------------------------|--|
| Connection technology        | 1-wire   |
| Number of inputs             | 8  |
| Nominal voltage              | 24 V DC (-15 %/+20 %)                                    |
| Specification                | EN 61131-2, type 3                                       |
| Signal voltage "0"           | -3+5 V   |
| Signal voltage "1"           | 1130 V   |
| Input filter                 | Configurable, default: 3 ms, min.: 10 µs                 |
| Connection cross-section     | e*: 0.081.5 mm²,<br>f*: 0.251.5 mm²,<br>a*: 0.140.75 mm² |
| Connection cross section AWG | e*: AWG 2816,<br>f*: AWG 2216,<br>a*: AWG 2619           |
| Strip length                 | 8 9 mm   |

<sup>\*</sup>e: single-wire, solid wire; f: stranded wire; a: with ferrule



# 6.2 Digital outputs

### **NOTICE**

# Feedback at the 24 V outputs

A voltage of 24 V at the outputs can destroy the device if the power supply (Up) is not connected (feedback). Connect the power supply (Up) so that 24 V can be applied to the outputs.

The digital outputs forward binary 24 V DC control signals, electrically isolated, to actuators at the process level. The high level of the positive switching logic corresponds to the supply voltage.

Outputs 3 and 4 have a PWM output stage. If the two digital outputs are used as normal digital outputs, the internal wiring will cause a leakage current of less than 100  $\mu$ A, which will cause a voltage of about 5 V. If you want to reach nearly 0 V at the low level of the output, you have to connect a 47 k $\Omega$  resistance to ground.

Another possibility is to operate the two outputs in PWM mode and to write the variable PWM output of the PWM signal for FALSE with 0x0000 and for TRUE with 0xFFFF. This activates the PWM output stage, which does not generate any leakage current.

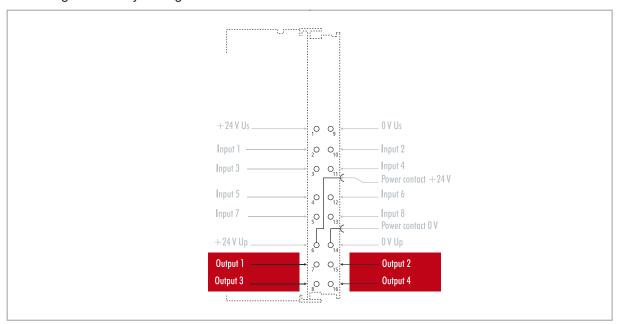


Fig. 31: Configurable digital outputs.

The CX7051 contains a total of four outputs, which indicate their signal state by means of light emitting diodes. The outputs can be used to switch standard actuators such as contactors and valves.

Table 9: Technical data, multi-function I/Os as digital outputs.

| Technical data              | CX7051  |
|-----------------------------|---|
| Connection technology       | 1-wire  |
| Number of outputs           | 4   |
| Nominal voltage             | 24 V DC (-15 %/+20 %)                                       |
| Load type                   | ohmic, inductive, lamp load                                 |
| Max. output current         | 24 V/0.5 A (short-circuit proof)                            |
| Changeover times            | T <sub>ON</sub> : 20 μs typ., T <sub>OFF</sub> : 10 μs typ. |
| Short circuit current       | < 2 A typ.  |
| Max. breaking energy (ind.) | < 150 mJ/channel  |
| Connection cross-section    | e*: 0.081.5 mm²,<br>f*: 0.251.5 mm²,<br>a*: 0.140.75 mm²    |



| Technical data | CX7051   |
|----------------|--|
|                | e*: AWG 2816,<br>f*: AWG 2216,<br>a*: AWG 2619 |
| Strip length   | 8 9 mm   |

<sup>\*</sup>e: single-wire, solid wire; f: stranded wire; a: with ferrule



# 6.3 Counter mode

The CX7051 Embedded PC can be configured as an up/down counter that enables the counting of a pulse. The embedded PC is suitable for fast counting tasks with a cut-off frequency of up to 100 kHz, whereby the CX7051 can be operated in 1-counter mode.

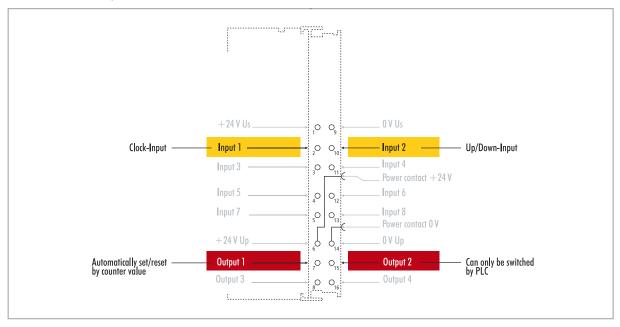


Fig. 32: Configurable inputs and outputs in counter mode.

The CX7051 supports three operation modes in counter mode:

- · Up/down counter
- · Up counter
- · Down counter

In addition, output 1 can be switched depending on the counter value. Output 2 can be switched from the PLC. This allows fast control signals for field devices to be used and switched.

The operation modes are set in TwinCAT via CoE objects.

# Up/down counter

In the up/down counter operation mode, the pulse to be counted is detected by digital input 1. The counting direction is specified by digital input 2.

If there is a high level at input 1 and at the same time at input 2, the counter counts upwards. If there is a high level at input 1 and a low level at input 2, the counter counts downwards.

#### Up counter

In this operation mode, the signal is detected at digital input 1.

### **Down counter**

In this operation mode, the signal is detected at digital input 1.



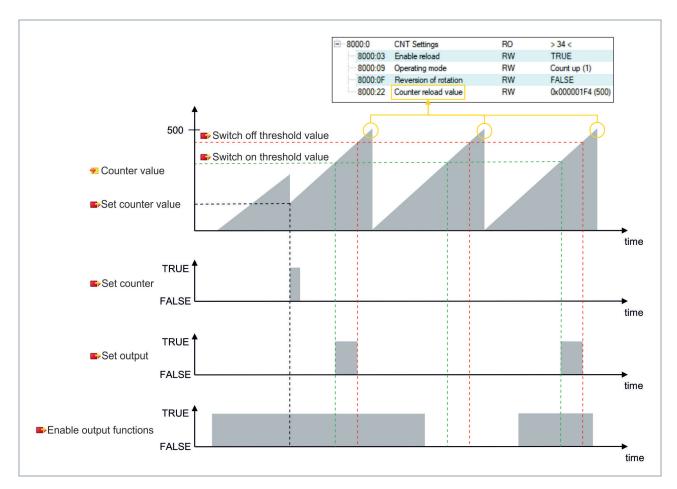


Table 10: Technical data, multi-function I/Os in counter mode.

| Technical data               | CX7051  |
|------------------------------|---|
| Number of counters           | 1 x up/down counter, 1 x up or down counter                                     |
| Nominal voltage              | 24 V DC (-15 %/+20 %)   |
| Specification                | EN 61131-2, type 3  |
| Signal voltage "0"           | -3+5 V  |
| Signal voltage "1"           | 1130 V  |
| Cut-off frequency            | Up/down counter: 20 kHz <sup>1)</sup> , counting in one direction only: 100 kHz |
| Counter depth                | 32-bit  |
| Max. output current          | 24 V/0.5 A (short-circuit proof)  |
| Special features             | Set counter, switch outputs, reset counter                                      |
| Connection cross-section     | e*: 0.081.5 mm²,<br>f*: 0.251.5 mm²,<br>a*: 0.140.75 mm²                        |
| Connection cross section AWG | e*: AWG 2816,<br>f*: AWG 2216,<br>a*: AWG 2619                                  |
| Strip length                 | 8 9 mm  |

<sup>&</sup>lt;sup>1)</sup>The up/down counter can also count up to 100 kHz, only with a direction reversal the counting frequency must be <= 20 kHz, otherwise pulses will be lost.

<sup>\*</sup>e: single-wire, solid wire; f: stranded wire; a: with ferrule

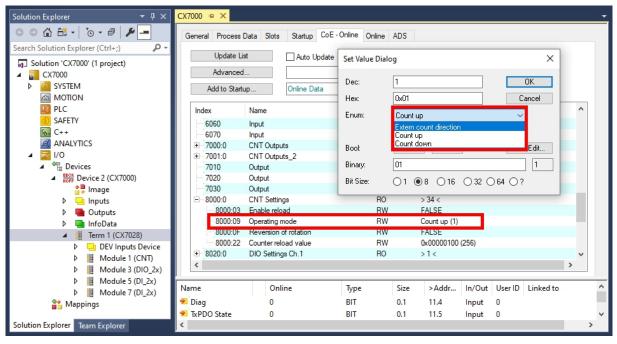


# 6.3.1 Select operation mode

The CX7051 supports three operation modes in counter mode: The operation mode is set in TwinCAT via CoE objects. You can choose between the three operating modes up/down counter, up counter and down counter.

#### Proceed as follows:

- 1. Click the CX7028 device on the left in the structure tree.
- 2. Click the CoE-Online tab.



- 3. Double-click the CoE object 8000:09 Operating mode.
- 4. Under the **Enum** option, select the required operation mode.
- ⇒ The operation mode is applied. Note that you can only use one operation mode at a time with the CX7051 and mode mixing is not possible.



# 6.3.2 Switching outputs

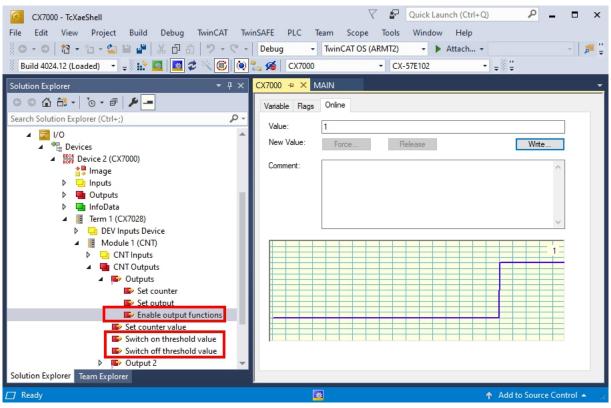
With the CX7051, it is possible to switch output 1 automatically as soon as a certain counter value is reached. This enables fast processing without the PLC. A second output, output 2, can be switched via the PLC irrespective of the counter value.

Output 1 is switched or switched off respectively by the variables **Switch on threshold value** and **Switch off threshold value**:

- If the value set under Switch on threshold value is reached, the output is switched.
- If the value set under Switch off threshold value is reached, the output is switched off.

When counting downwards, the corresponding switching instruction is executed in reverse. If the value falls below the value set in **Switch on threshold value**, output 1 is switched off.

#### Proceed as follows:



- Use the variable Switch on threshold value to specify a counter value at which the output should be switched.
- Use the variable Switch off threshold value to specify a counter value at which the output should be switched off.
- 3. Then set the variable **Enable output functions** to **True** so that the settings are applied.
- Only when the variable Enable output functions is set to True the function is enabled and the output is switched.

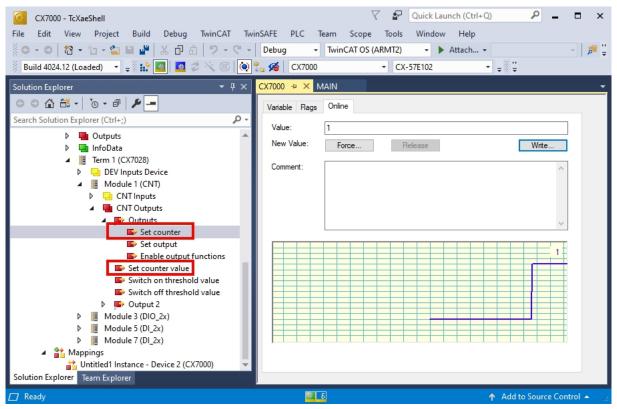
If the parameterized counter value from **Switch on/off threshold** is reached or exceeded, but the variable **Enable output functions** is not set, the switching order is not executed. The output is switched as soon as **Enable output functions** is set. Likewise, a subsequently activated counter value **Switch on/off threshold** affects the output immediately when the switching condition is fulfilled.



# 6.3.3 Set counter value

This step shows you how to set the counter value to a specific value. The variable **Set counter value is** used to specify a value and the variable **Set counter** is used to set the counter value. Both variables can be controlled from the PLC.

#### Proceed as follows:



- 1. Use the variable **Set counter value** to specify a value to set as a counter value.
- 2. Then set the variable **Set counter** to **True** to apply the settings.
- ⇒ Only when the variable **Set counter** is set to **True**, the value set under **Set counter value** is applied for the counter value.



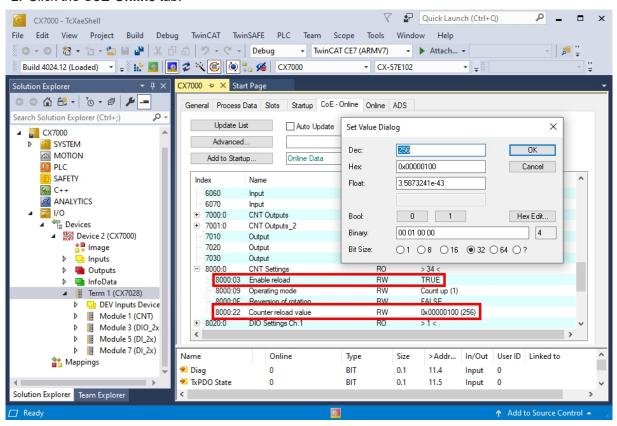
# 6.3.4 Setting the limit value for counters

This step shows you how to set a limit value in TwinCAT from which the counter value is automatically reset to zero. When counting upwards, the counter value is reset to zero when the limit value is reached. When counting downwards, the counter value is reset to the set limit value when zero is reached.

The counter value is a UDINT variable. The counter counts only in the positive range from 0 to 0xFFFF\_FFFF (4294967295). If the value falls below zero, the counter is set to the maximum positive value. If it exceeds 4294967295, the counter is set to zero. The two variables **Counter underflow** or **Counter overflow** respectively indicate the overflow and are reset either on reaching 0x4000 in the positive direction or on reaching 0xFFFFC000 in a negative direction or if the corresponding other overflow has been reached.

#### Proceed as follows:

- 1. Click the CX7028 device on the left in the structure tree.
- 2. Click the CoE-Online tab.



- 3. Double-click the CoE object 8000:22 Counter reload value and set the limit value.
- 4. Then double-click the CoE object 8000:03 Enable reload and set the value to True.
- ⇒ Only when the CoE object **8000:03 Enable reload** is set to **True** are the function and the defined limit value active.



# 6.4 Incremental encoder mode

In incremental encoder mode, the CX7051 can be configured as an interface for direct connection of 24 V incremental encoders. A quadruple evaluation is used and both high level and low level are detected at input 1 and input 2.

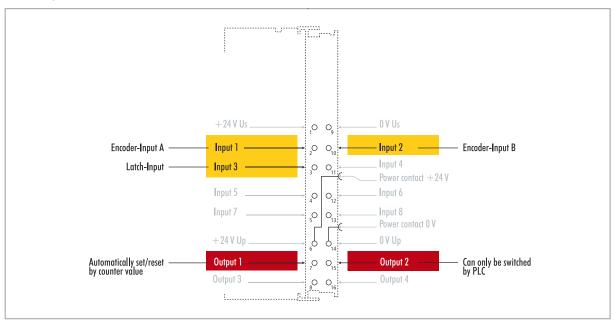


Fig. 33: Configurable inputs and outputs in incremental encoder mode.

The range of functions in encoder mode corresponds to the range of functions in counter mode. In addition, the counter value at input 3 can be latched, i.e. the value is entered in the process data on a high level at input 3. Alternatively, the counter can be reset on a high level at input 3.

In addition, output 1 can be switched depending on the counter value. Output 2 can be switched from the PLC. This allows fast control signals for field devices to be used and switched.



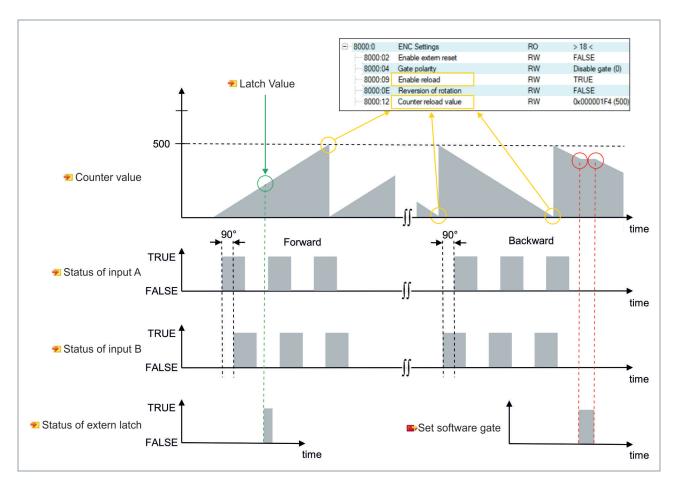


Table 11: Technical data, multi-function I/Os in encoder mode.

| Technical data               | CX7051   |
|------------------------------|--|
| Technology                   | Incremental encoder interface  |
| Nominal voltage              | 24 V DC (-15 %/+20 %)  |
| Specification                | EN 61131-2, type 3   |
| Encoder connection           | 1 x A, B: 24 V, single-ended   |
| Additional inputs            | Latch input, 24 V DC   |
| Cut-off frequency            | 250,000 increments/s (with 4-fold evaluation), corresponds to 62.5 kHz     |
| Counter depth                | 32-bit   |
| Quadrature decoder           | 4-fold evaluation  |
| Max. output current          | 24 V/0.5 A (short-circuit proof)   |
| Special features             | Latch function, software gate, set counter, switch outputs, reset counters |
| Connection cross-section     | e*: 0.081.5 mm²,<br>f*: 0.251.5 mm²,<br>a*: 0.140.75 mm²                   |
| Connection cross section AWG | e*: AWG 2816,<br>f*: AWG 2216,<br>a*: AWG 2619                             |
| Strip length                 | 8 9 mm   |

<sup>\*</sup>e: single-wire, solid wire; f: stranded wire; a: with ferrule



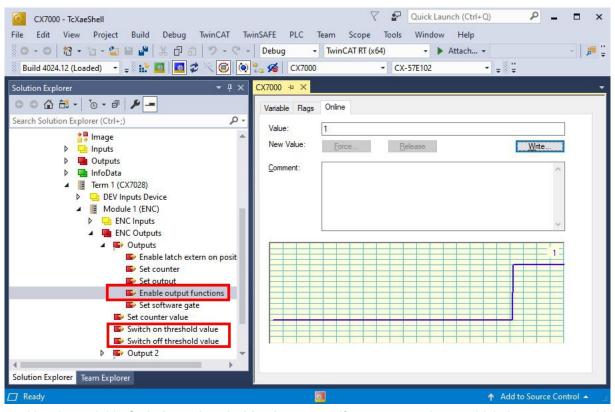
# 6.4.1 Switching outputs

With the CX7051, it is possible to switch output 1 automatically as soon as a certain counter value is reached. This enables fast processing without the PLC. A second output, output 2, can be switched via the PLC irrespective of the counter value.

Output 1 is switched or switched off respectively by the variables **Switch on threshold value** and **Switch off threshold value**:

- If the value set under Switch on threshold value is reached, the output is switched.
- · If the value set under Switch off threshold value is reached, the output is switched off.

#### Proceed as follows:



- Use the variable Switch on threshold value to specify a counter value at which the output should be switched.
- 2. Use the variable **Switch off threshold value** to specify a counter value at which the output should be switched off.
- 3. Then set the variable **Enable output functions** so that the settings are applied.
- ⇒ Only when the variable **Enable output functions** is set to **True** is the function enabled and the settings applied.

If the parameterized counter value from **Switch on/off threshold** is reached or exceeded, but the variable **Enable output functions** is not set, the switching order is not executed. The output is switched as soon as **Enable output functions** is set. Likewise, a subsequently activated counter value **Switch on/off threshold** affects the output immediately when the switching condition is fulfilled.

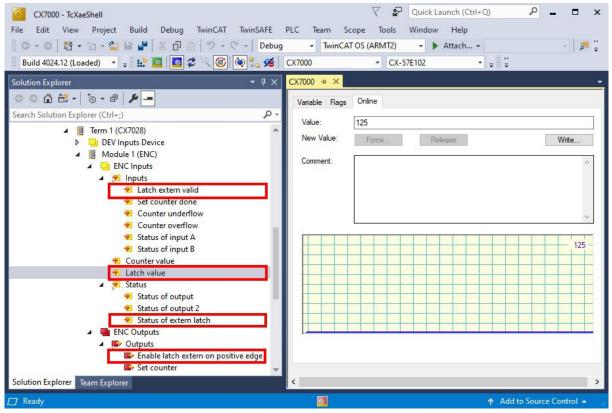


# 6.4.2 Latching the counter value

In incremental encoder mode, the counter value can be latched and thus the current value can be entered in the process data. Input 3 is used as a latch input.

To enable the function, the variable **Enable latch extern on positive edge** must be set to **True**. On a high level at input 3, the current counter value is entered into the variable **Latch Value**. You can monitor the validity of the variable. As soon as the latch value is entered, the variable **Latch extern valid** is also set to **True**.

### Proceed as follows:



- 1. Set the variable Enable latch extern on positive edge to True to enable the latch function.
- 2. Monitor the status of the latch input with the variable **Status of extern latch**.
- 3. On a high level at input 3, the current counter value is entered into the variable Latch Value.
- 4. Monitor the validity of the latch value via the variable **Latch extern valid**. Once the latch value has been written, the variable is also set to **True**.
- ⇒ To execute a latch again, the variable Enable latch extern on positive edge must receive a high level again.

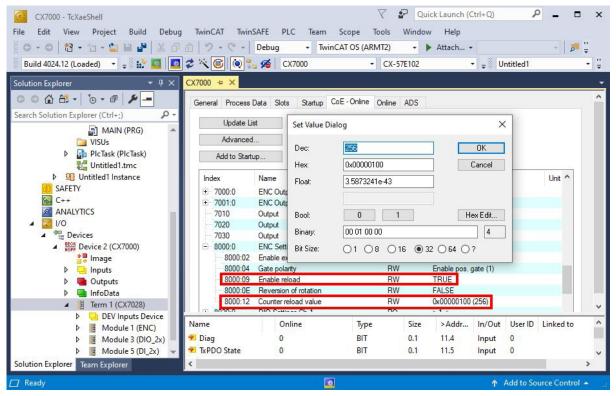


# 6.4.3 Setting the limit value for counters

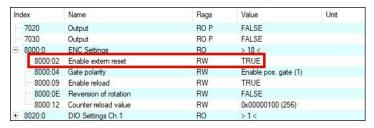
This step shows how you can set a limit value in TwinCAT from which the counter value is automatically reset to zero. When counting upwards, the counter value is reset to zero when the limit value is reached. When counting downwards, the counter value is reset to the set limit value when zero is reached.

#### Proceed as follows:

- 1. Click the CX7028 device on the left in the structure tree.
- 2. Click the CoE-Online tab.



- 3. Double-click the CoE object 8000:12 Counter reload value and set the limit value.
- 4. Then double-click the CoE object 8000:09 Enable reload and set the value to True.
- ⇒ The function is only active when Enable reload is set. Alternatively, the latch input can be used and the counter value can thus be reset externally. To do this, the latch function must be disabled and the CoE object Enable extern reset set to True. With this setting, the current counter value is set to zero on a high level at input 3.





# 6.5 Analog signal mode

The single-ended inputs 7 and 8 acquire signals in the range of 0 to 10 V.

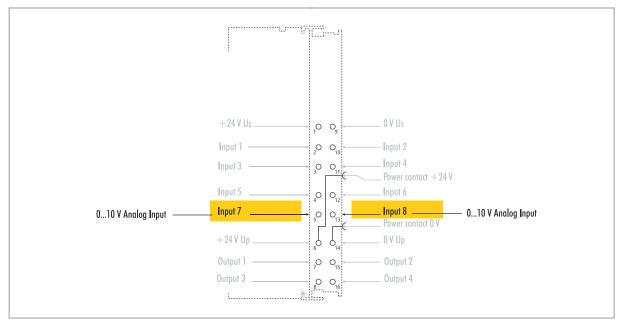


Fig. 34: Configurable analog inputs.

The voltage is digitized with a resolution of 12 bits. LEDs are used to indicate the signal state.

Table 12: Technical data, multi-function I/Os in analog mode.

| Technical data                 | CX7051   |
|--------------------------------|--|
| Technology                     | single ended   |
| Number of inputs               | 2  |
| Signal voltage                 | 010 V  |
| Internal resistance            | 500 kΩ   |
| Input filter cut-off frequency | 2 kHz  |
| Resolution                     | 12-bit (16-bit representation)                           |
| Measuring error                | < ±0.3 % (relative to full scale value)                  |
| Connection cross-section       | e*: 0.081.5 mm²,<br>f*: 0.251.5 mm²,<br>a*: 0.140.75 mm² |
| Connection cross section AWG   | e*: AWG 2816,<br>f*: AWG 2216,<br>a*: AWG 2619           |
| Strip length                   | 8 9 mm   |

<sup>\*</sup>e: single-wire, solid wire; f: stranded wire; a: with ferrule

CX7051 Version: 1.0 111



# 6.6 PWM signal mode

# **NOTICE**

## Feedback at the 24 V outputs

A voltage of 24 V at outputs 3 and 4 can destroy the device (feedback). No voltage may be applied to the outputs in PWM mode.

The PWM signal mode enables a pulse width modulated binary signal to be output at outputs 3 and 4.

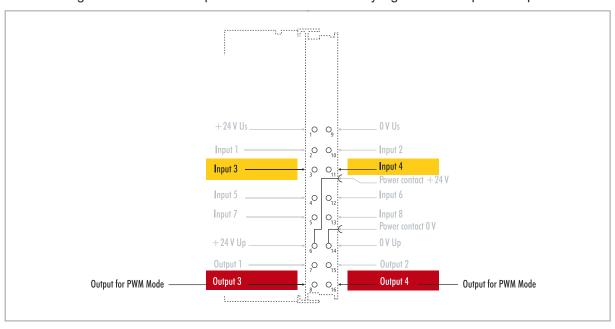


Fig. 35: Configurable inputs and outputs in PWM signal mode

This signal is separated into duty cycle (0... 100 %) and PWM clock frequency (15 Hz... 100 kHz). The LEDs are clocked with the outputs, and show the duty cycle by their brightness. The signal values are transferred in 16-bit values.

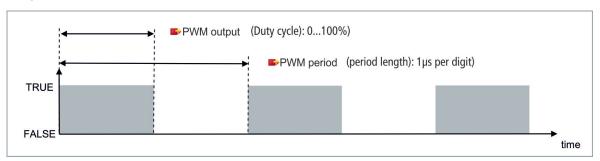


Table 13: Technical data, multi-function I/Os in PWM mode.

| Technical data        | Digital inputs  |
|-----------------------|---|
| Connection technology | PWM output  |
| Number of outputs     | 2   |
| Nominal voltage       | 24 V DC (-15 %/+20 %)                                       |
| Load type             | ohmic, inductive, lamp load                                 |
| Max. output current   | 24 V/0.5 A (short-circuit proof)                            |
| PWM clock frequency   | 15 Hz100 kHz  |
| Duty cycle            | 0100 % (T <sub>ON</sub> > 20 ns, T <sub>OFF</sub> > 200 ns) |
| Short circuit current | < 2 A typ.  |
| Special features      | separate frequency can be set for each channel              |



| Technical data               | Digital inputs   |
|------------------------------|--|
| Connection cross-section     | e*: 0.081.5 mm²,<br>f*: 0.251.5 mm²,<br>a*: 0.140.75 mm² |
| Connection cross section AWG | e*: AWG 2816,<br>f*: AWG 2216,<br>a*: AWG 2619           |
| Strip length                 | 8 9 mm   |

<sup>\*</sup>e: single-wire, solid wire; f: stranded wire; a: with ferrule

CX7051 Version: 1.0 113



# 6.6.1 Setting the PWM clock frequency and duty cycle

The signals at outputs 3 and 4 are output with pulse width modulation, the signals being separated into duty cycle and PWM clock frequency. Separate values for duty cycle and PWM clock frequency can be defined for both outputs.

Table 14: PWM output (duty cycle), representation of the PWM signal in the delivery state.

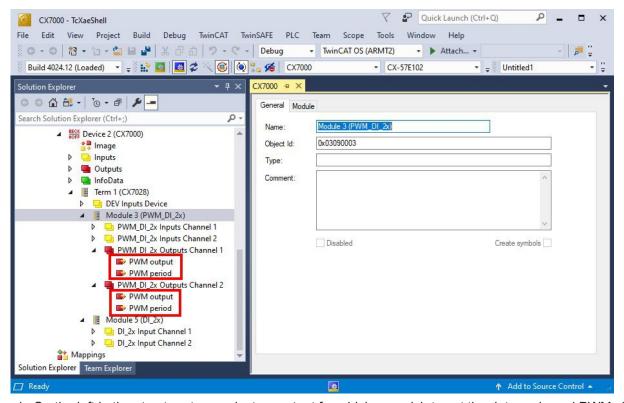
| Value | Decimal | Hexadecimal |
|-------|---------|-------------|
| 0 %   | 0       | 0x0000      |
| 25 %  | 16383   | 0x3FFF      |
| 50 %  | 32767   | 0x7FFF      |
| 100 % | 65.535  | 0xFFFF      |

Table 15: PWM period (PWM clock frequency), representation of the PWM signal in the delivery state.

| Value    | Decimal | Hexadecimal   | Frequency  |
|----------|---------|---------------|------------|
| 0.010 ms | 010     | 0x0000-0x000A | 100 kHz    |
| 0.011 ms | 11      | 0x000B        | 90.909 kHz |
| 0.100 ms | 100     | 0x0064        | 10 kHz     |
| 1.000 ms | 1000    | 0x03E8        | 1 kHz      |
| 16.38 ms | 16383   | 0x3FFF        | 61.04 Hz   |
| 65.53 ms | 65535   | 0xFFFF        | 15.26 Hz   |

The variable **PWM output** correspond to the duty cycle and **PWM period** to the PWM clock frequency at which the signal is output.

#### Proceed as follows:



- 1. On the left in the structure tree, select an output for which you wish to set the duty cycle and PWM clock frequency.
- 2. Link the variables PWM output and PWM period with the appropriate variables from your PLC project.
- 3. In the variables, set the values for duty cycle and PWM clock frequency according to the above tables.



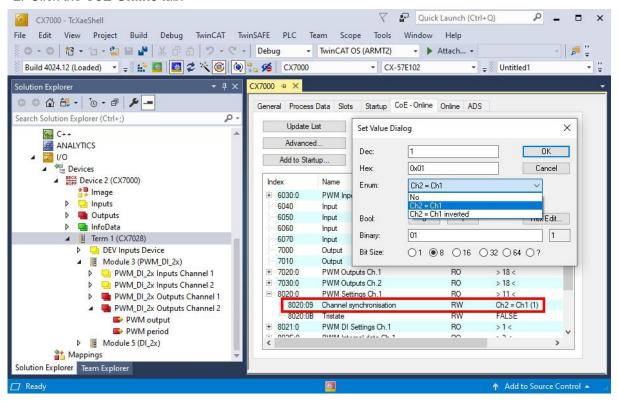
# 6.6.2 Setting the channel synchronization

The channel synchronization option makes the output of output 2 dependent on output 1. The following values are available in the CoE objects:

- · No: no dependency
- Ch2 = Ch1: Duty cycle and PWM clock frequency of output 1 are also applied to output 2. The phase position is 0, i.e. the rising and falling edges of output 1 and output 2 are synchronized.
- Ch2 = Ch1 inverted: Duty cycle and PWM clock frequency of output 1 are also applied to output 2. However, the PWM clock frequency is inverted. The phase position is 0, i.e. a rising edge at output 1 triggers a falling edge at output 2 at the same time.

#### Proceed as follows:

- 1. Click the CX7028 device on the left in the structure tree.
- 2. Click the CoE-Online tab.



- 3. Double-click the CoEobject 8020:09 Channel synchronization.
- 4. Under the option **Enum**, select the type of synchronization required.



# 7 Configuration

# 7.1 Starting the Beckhoff Device Manager

Using the Beckhoff Device Manager, an Industrial PC can be configured by remote access with the aid of a web browser. The access takes place via the HTTP protocol and Port 80 (TCP).

### Requirements:

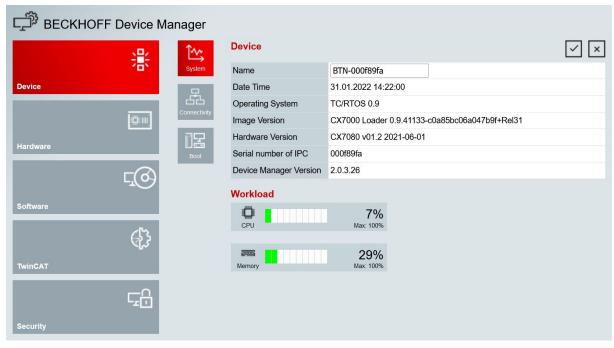
- Host PC and Embedded PC must be located in the same network. The network firewall must allow access via port 80 (HTTP).
- · IP address or host name of the Embedded PC.

Table 16: Access data for the Beckhoff Device Manager on delivery.

| User name     | Password |
|---------------|----------|
| Administrator | 1        |

## Start the Beckhoff Device Manager as follows:

- 1. Open a web browser on the host PC.
- 2. Enter the IP address or the host name of the Industrial PC in the web browser to start the Beckhoff Device Manager.
  - Example with IP address: http://169.254.136.237/config
  - Example with host name: http://BTN-000f89fa/config
- 3. Enter the user name and password. The start page appears:



⇒ Navigate forward in the menu and configure the Industrial PC. Note that modifications only become active once they have been confirmed. It may be necessary to restart the Industrial PC.



# 7.2 Persistent data

## **NOTICE**

## Application example

In the following example, changes to the loads, the power supply or even just aging components can lead to the application no longer fulfilling the desired function. Beckhoff takes no responsibility for the implementation of the example in an application.

Normally, persistent data are only stored during the TwinCAT stop or by a function block. This chapter shows you how to store persistent data on a CX7051 without a UPS.

In the case of an Embedded PC with UPS, the function block is usually linked to the UPS. The function block becomes active as soon as a power failure is detected, writes the persistent data and then shuts down the Embedded PC. With a 1-second UPS, the Embedded PC is not shut down because there is too little time left for this.

In the case of a small controller such as the CX7051 which is delivered without a 1-second UPS, you can still use this function. All that is needed is to use a power supply unit that has enough residual energy to supply power to the CX7051 with this residual energy for a certain period of time. A small test can show you if this is possible with your power supply unit:

## Testing a power supply unit

When the CX7051 is running, turn off the AC voltage of your power supply unit and measure how long the CX7051 continues to run. If it is more than three seconds, you may be able to use the power supply unit as a replacement for a 1-second UPS. Note that power supply units also age and lose capacity. You should therefore include a safety factor, such as a factor of three, so that you have enough reserve to be able to operate the power supply unit for a longer period of time as a replacement for a 1-second UPS.

Now determine how long the power supply unit maintains the supply of power. You need an EL1722 for this, which you connect to the AC side of the power supply unit. Then write a small program:

```
VAR

bPower230V AT %I*: BOOL; (*link to the EL1722*)

END_VAR

VAR RETAIN

Counter: INT;

END_VAR

Program:

IF NOT bPower230V THEN (*bPower230V is linked to the EL1722*)

Counter:=counter+1; (*the counter is a retain value*)

END_IF
```

Create a boot project and turn off the AC voltage of the power supply unit. As soon as the EL1722 no longer displays a value, the counter is incremented and the data are copied to the internal NOVRAM. Turn the AC voltage back on and log in. You must now multiply the counter value by the task time. Repeat this a few times to be sure that the power supply unit always behaves in the same way. Next, you have to insert the function block FB\_WritePersistentData. This is contained in the Tc2\_Utilities library (in the "TwinCAT PLC" folder).

Then determine how long it takes to store the persistent data. Repeat this process a few times too, so that you obtain a constant value and can determine a maximum value in case of fluctuations. You can determine the time required via the Busy flag. The function block is being processed as long as the Busy flag is set. Multiply the value determined by two to incorporate a further safety factor.

#### Example:

Your measurement shows that the power supply unit maintains the supply of power for three seconds and that the persistent data is written in about 400 ms. With the recommended safety factors, the power supply is maintained for one second and the persistent data is written in about 800 ms.

The power supply is therefore maintained for a longer period of time than is needed to store the persistent data. Therefore you can use the example power supply unit as a replacement for the 1-second UPS.

CX7051 Version: 1.0 117



# 7.3 NOVRAM

The NOVRAM can be used to reliably save important variable values, such as production data or counter values, in the event of a power failure. The memory size of the NOVRAM is limited and only suitable for smaller data quantities up to 4 kB.

In this chapter we show you how the NOVRAM is used in TwinCAT 3.

## **Functioning**

The NOVRAM (Non-Volatile Random Access Memory) is a special memory component that is used to reliably save important data. The NOVRAM consists of two sections, a volatile memory and a non-volatile memory.

TwinCAT only writes to the volatile section of the NOVRAM. In the event of a power failure, the data are automatically copied from the volatile memory into the non-volatile memory. The energy required for this process is supplied by a capacitor. As soon as the power supply is restored, the data are automatically copied back into the volatile memory, so that TwinCAT can continue to use them.

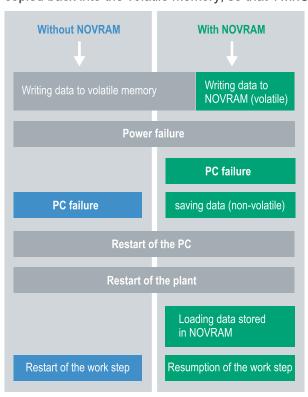


Fig. 36: Controller behavior with and without NOVRAM.

### **Memory size**

The NOVRAM has a capacity of 4 kB. The data are saved cyclically and alternately based on the dual buffer principle, in order to avoid the risk of data inconsistency.

## Requirements

| Development environ-<br>ment | Target platforms         |                                    | PLC libraries to include |
|------------------------------|--------------------------|------------------------------------|--------------------------|
| TwinCAT 3.1 Build: 4020      | PC or CX (x86, x64, ARM) | CX70xx, CX9020,<br>CX20x0, CX20x2, | Tc2_loFunctions          |
|                              |                          | CX20x3                             |                          |



# 7.3.1 Creating a Retain Handler

Under TwinCAT 3 (from Build 4020) a delta algorithm is used to save data in the NOVRAM. The algorithm does not save all the variables in the NOVRAM. Instead, it searches for changes (delta function) compared to the previous cycle and only saves variables that have changed.

To use the delta algorithm, a Retain Handler must be created in TwinCAT 3, and the relevant variables must be declared in the PLC with the keyword VAR RETAIN.

A new feature of this method is that no function blocks have to be used. The Retain Handler saves data in the NOVRAM in the event of a power failure and makes them available again once the power has been restored.

This chapter describes how to create a Retain Handler in TwinCAT 3. The Retain Handler saves data in the NOVRAM and makes them available again. In other words, important variable values such as production data or counter values are retained during a restart or power failure.

Requirements for this step:

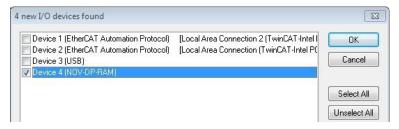
- TwinCAT 3.1 Build: 4020.
- · A target device selected in TwinCAT.

## Create the Retain Handler as follows:

- 1. Right-click on **Devices** in the tree view on the left-hand side.
- 2. In the context menu click on Scan.



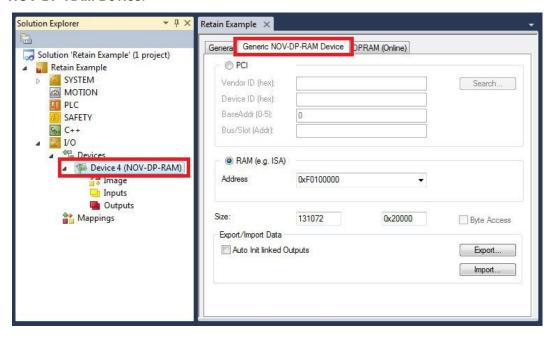
3. Select Device (NOV-DP-RAM) and confirm with OK.



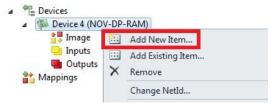
4. Click on Yes to search for boxes.



5. Click on **Device (NOV-DP-RAM)** in the tree view on the left-hand side and then on the tab **Generic NOV-DP-RAM Device**.



- 6. Click the option RAM.
- 7. Right-click on Device (NOV-DP-RAM) in the tree view and then on Add New Item.



8. Select the Retain Handler and click on OK.



⇒ You have successfully created a Retain Handler in TwinCAT.



In the next step you can create retain variables in the PLC and link them with the Retain Handler.



# 7.3.2 Creating and linking variables

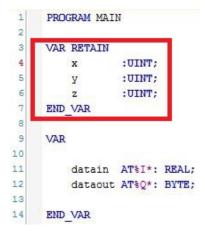
Once you have created a Retain Handler in TwinCAT, you can declare variables in the PLC and link them to the Retain Handler. The variables have to be identified in the PLC with the keyword VAR\_RETAIN.

Prerequisite for this step:

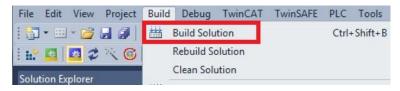
· A PLC project created in TwinCAT.

### Create variables as follows:

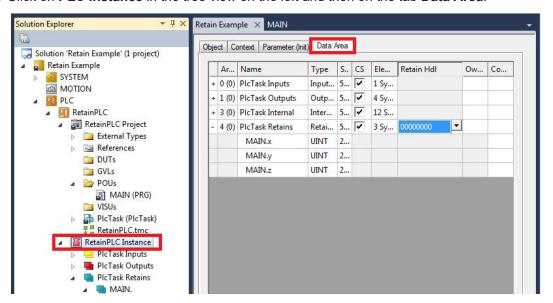
1. Create the variables in your PLC project in a VAR RETAIN area.



2. Click on **Build** in the toolbar at the top, then on **Build Solution**.

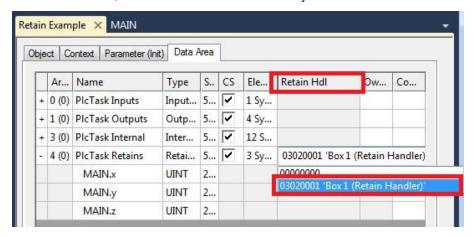


3. Click on PLC Instance in the tree view on the left and then on the tab Data Area.



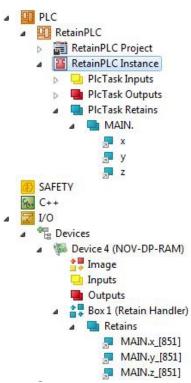


4. Under Retain HdI, select the Retain Handler that you have created.



⇒ After selecting a Retain Handler as a target, the symbols in the tree view are linked and a mapping is created.

In the tree view the variables are created from the PLC under the Retain Handler and linked to the variables from the PLC instance.



An existing link is displayed with an arrow symbol.



# 7.3.3 Deleting variables under the Retain Handler

If variables are deleted from the PLC, the link with the Retain Handler is cancelled. However, the variables continue to be shown under the Retain Handler and are not deleted automatically.

Under TwinCAT 3 the variables have to be deleted manually.

Prerequisites for this step:

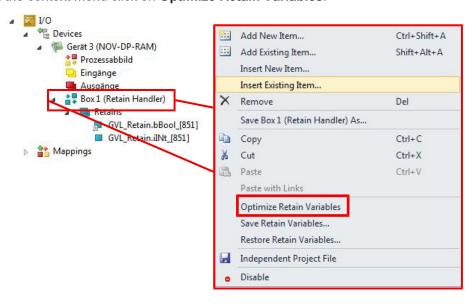
· Variables declared with VAR RATAIN were deleted from the PLC.

### Delete the variables under the Retain Handler as follows:

1. The variable GVL Retain.iNt under the Retain Handler is to be deleted.



- 2. Right-click on the Retain Handler in the tree view on the left.
- 3. In the context menu click on Optimize Retain Variables.



⇒ The variable under the Retain Handler is deleted.



# 7.4 Software configuration

# 7.4.1 User name and password

In the delivery state, the CX7051 has a preset user name with password, which is necessary for logging in to TwinCAT or the Beckhoff Device Manager.

• User name: Administrator

• Password: 1

The user name is fixed and cannot be changed. It is also not possible to add another user name. The preset password can be changed via the Beckhoff Device Manager (see: Starting the Beckhoff Device Manager). The password can contain a maximum of 32 characters. Numbers, letters and special characters are allowed and a distinction is also made between upper and lower case letters.

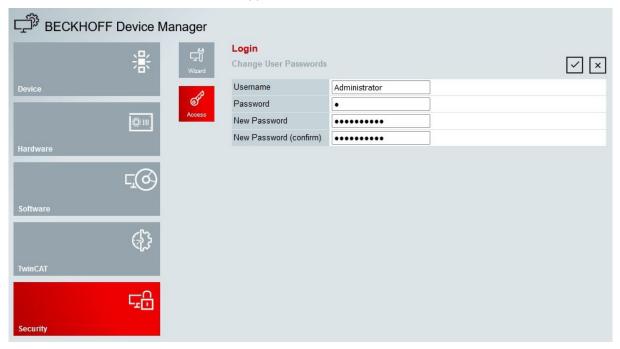


Fig. 37: Changing the password in the Beckhoff Device Manager.

You can restore the delivery state and preset password by removing the MicroSD card, accessing the MicroSD card with a card reader and deleting the <code>device.conf</code> file in the <code>/etc</code> folder. The password cannot be reset without physical access to the CX7051 and thus to the MicroSD card.



# 7.4.2 Setting the IP address

DHCP is enabled by default for the CX7051. Without a DHCP server, the CX7051 uses a local IP address in the address range 169.254.x.x

In the case of the CX7051 Embedded PC, there are several ways to set the IP address. One way is to call the Beckhoff Device Manager and set the IP address for the CX7051 in the browser (see: Starting the Beckhoff Device Manager).

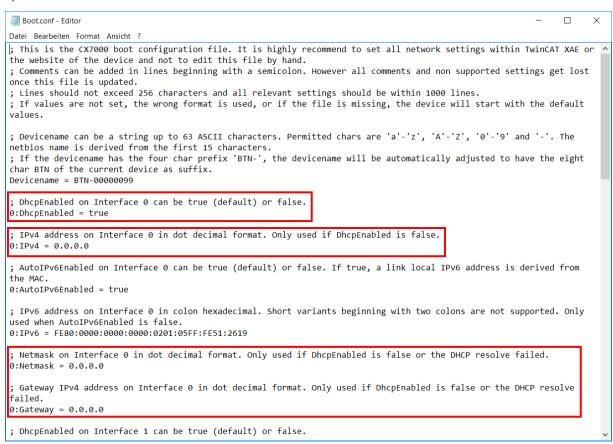
Another way to set the IP address is offered by the boot.conf file, which is created on the MicroSD card after the first start. This step shows you how to set the IP address in the boot.conf file.

#### Requirements:

· MicroSD card reader

#### Proceed as follows:

- 1. Switch the Embedded PC off and remove the MicroSD card from the Embedded PC.
- 2. Open the Boot.conf file under \etc



- Set the **DhcpEnabled** entry to **false**.
- 4. Assign an IP address under IPv4.
- 5. Make the settings for subnet mask, gateway and DNS server.
- ⇒ Save the changes and install the MicroSD card in the Embedded PC again. The settings are effective after startup.



# 7.4.3 Update image

## **NOTICE**

## Failure of the power supply

The bootloader may be corrupted if the update is interrupted. The CX70x0 thus becomes unusable and must be sent in for repair. Ensure a stable power supply during initial start-up and do not interrupt the update.

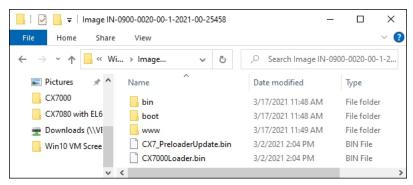
The new image will be copied directly to the MicroSD card in order to update the image of the Embedded PC. The new image is made available by Beckhoff Service. Perform the update only after consulting with Beckhoff Service.

#### Requirements:

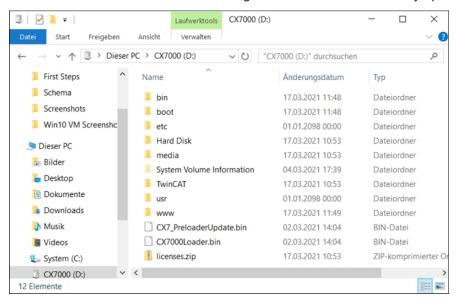
· Card reader for MicroSD cards.

## Update the image as follows:

- 1. Switch the Embedded PC off and remove the MicroSD card from the Embedded PC.
- 2. Insert the MicroSD card into an external card reader and open the MicroSD card's folder tree.
- 3. Delete all files and folders on the MicroSD card.
- 4. Copy all files and folders of the new image to the empty MicroSD card.



- 5. Re-install the MicroSD card in the Embedded PC and start the Embedded PC.
- ⇒ The Embedded PC is started and saves the current hardware configuration. New folders are created, such as Hard Disk or TwinCAT. The image has now been successfully updated.





# 7.4.4 Updating the firmware for multifunction I/Os

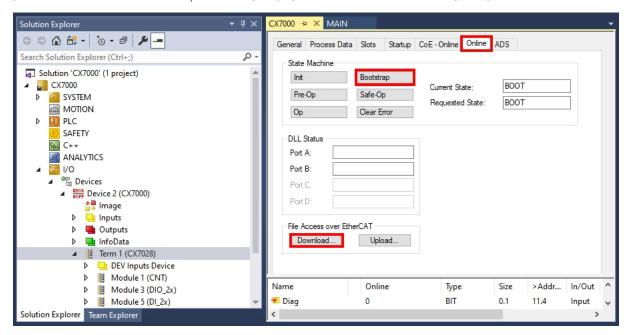
This step shows you how to update the firmware of the multifunction I/Os. The firmware is provided by Beckhoff Service and the update is carried out in TwinCAT.

### Requirements:

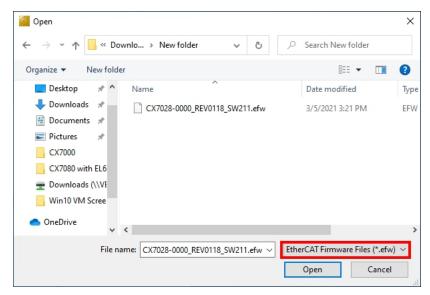
EtherCAT firmware file (\*.efw)

#### Proceed as follows:

- 1. Start TwinCAT in configuration mode (config mode).
- 2. On the left in the structure tree, click the CX7028 device and then click the Online tab.



- 3. Click the Bootstrap button to switch the multifunction I/Os to the bootstrap state.
- 4. Click the **Download** button and select a current efw file.



⇒ The update takes about 3 to 4 minutes. A progress bar indicates the progress of the update. Do not switch the CX7051 off during this time.

When the update is complete, return to the Operational (Op) state by clicking the **Op** button.



# 7.4.5 Updating the ESI device description

The TwinCAT System Manager and the TwinCAT EtherCAT Master require the device description files of all EtherCAT devices for configuration in online and offline mode. These device descriptions are the so-called ESI files (EtherCAT Slave Information) in XML format. These files can be requested from the respective vendor and are made available for download. An \*.xml file may contain several device descriptions.

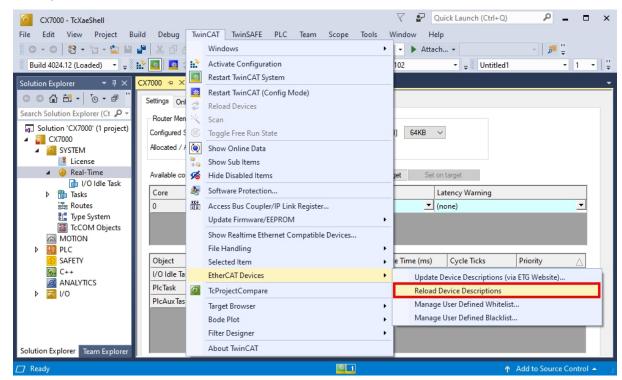
ESI files for Beckhoff EtherCAT devices are provided at <a href="https://www.beckhoff.com">https://www.beckhoff.com</a>.

## Requirements:

- · ESI file for the CX7051 in XML format.
- If necessary, the associated \*.xsd file, which describes the structure of the XML file.

#### Proceed as follows:

- 1. Copy the ESI file into the TwinCAT installation directory: \TwinCAT\3.1\Config\lo\Onboardlo.
- 2. Create the folder manually if it doesn't exist.
- 3. Open TwinCAT and click in the menu under **TwinCAT > EtherCAT Devices** on **Reload Device Description**.



⇒ The ESI file is re-read into TwinCAT. An error is returned if there is a faulty ESI file. Check whether the structure of the \*.xml corresponds to the associated \*.xsd file or whether the files match the CX7051.



# 8 TwinCAT

# 8.1 First Steps

# 8.1.1 Connect to the CX70x0

Before you can configure the CX7051 in TwinCAT, you must establish a connection between your engineering computer and the CX7051 (target system). The engineering computer and the Embedded PC must be in the same network and subnet or alternatively connected directly via an Ethernet cable (peer-to-peer).

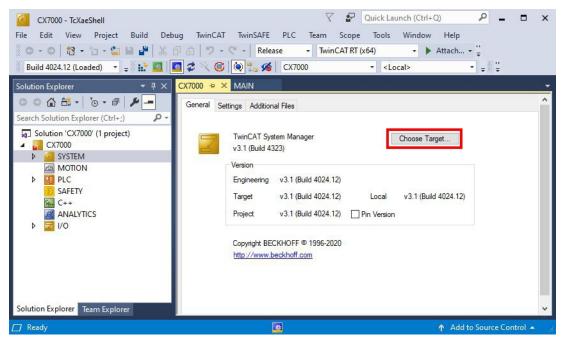
The IP address or host name of the CX7051 is required for the connection.

## Requirements:

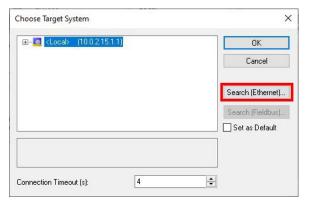
- · TwinCAT must be in Config mode.
- · IP address or host name of the Embedded PC.

### Establish a connection as follows:

- 1. In the menu at the top click on File > New > Project and create a new TwinCAT XAE project.
- 2. In the tree view on the left click on SYSTEM, and then Choose Target.

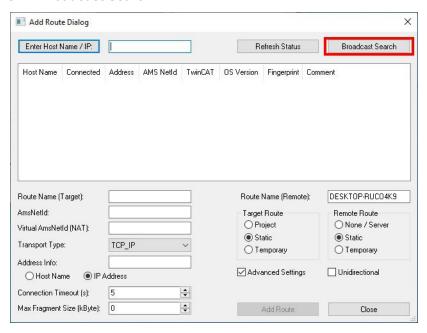


3. Click on Search (Ethernet).

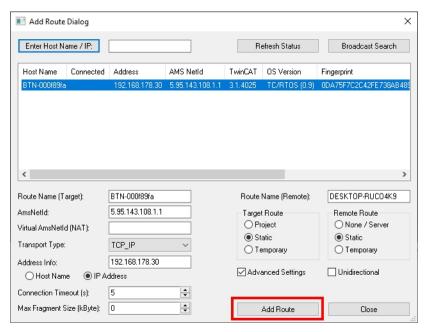




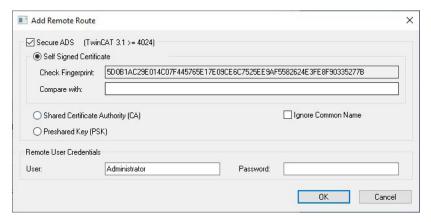
4. Click **Broadcast Search** and search for available devices on the network.



5. Mark the appropriate CX7051 and click **Add Route**. The host name and IP address facilitate identification.



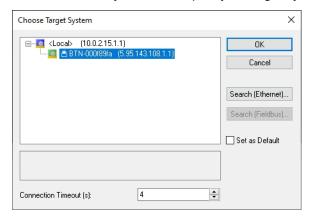
6. Enter the user name and password in the **User** and **Password** fields respectively and click **OK**. User name: Administrator **Password**: 1



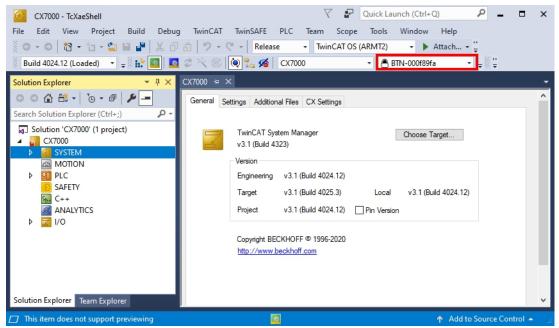
7. The new device is displayed in the **Choose Target System** window.



8. Select the device you want to specify as target system and click **OK**.



⇒ You have successfully established a connection between your engineering computer and the CX7051 (target system) in TwinCAT. The new target system and the host name are displayed in the menu bar.



Using this procedure you can search for all available devices and also switch between the target systems at any time.

## 8.1.2 Scan multifunction I/Os

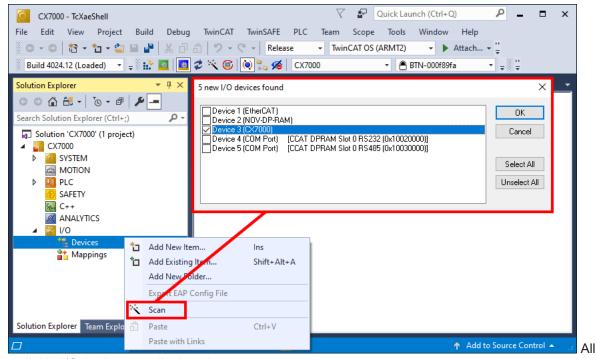
Special features of the CX7000 series are the eight integrated multifunction inputs and four integrated multifunction outputs. This chapter shows how to scan and create the multifunction I/Os in TwinCAT.

Note that the CX7028 interface for controlling the multifunction I/Os has its own CPU and the CX7028 interface is not displayed or does not work under TwinCAT if the power supply(Up) is not connected.

### Proceed as follows:

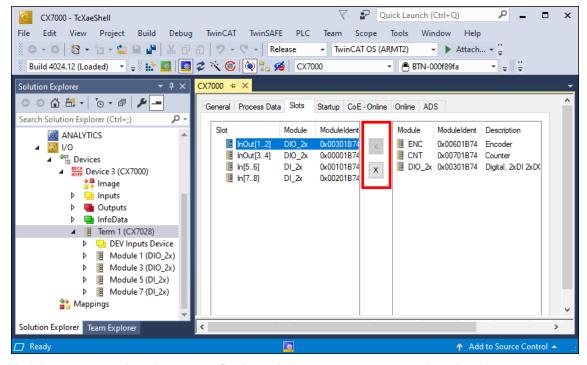


1. On the left side of the tree view, right-click **Devices** and then click **Scan**.



available I/O devices are displayed.

- 2. Select the appropriate I/O devices. For this example, at least the CX7028 interface, i.e. the CX7000 device, must be selected. If you still want to operate Bus or EtherCAT Terminals on the CX7000, then you must also select EtherCAT as a device.
- 3. A total of four slots are created. For each slot a maximum of one module (DI, DIO, ENC, CNT or PWM) can be assigned, which in turn determines the operation mode for the respective slot.



- 4. Modules can be assigned to a specific slot with the button < or removed again with x.
- ⇒ Define the required modules according to their requirements. There is a choice of different modules depending on the slot used. Which modules are supported by which slot is listed in the chapter Multifunction I/Os [▶ 95].



# 8.1.3 Establishing ADS communication

This chapter shows you how to connect a CX7051 to another CX70x0 or any TwinCAT controller. The ADS protocol provides the simplest way to connect two TwinCAT systems to each other. With the ADS protocol, data can be both read and written. ADS function blocks are normally used for communication; these are included in the Tc2\_System library. In the following example, data are to be written to and read from a memory area.

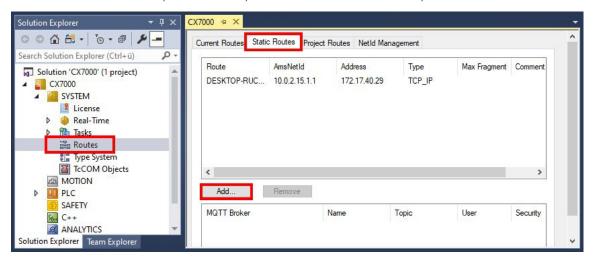
In order to set up an ADS connection, an ADS route is created first. Communication then takes place via Ethernet and data exchange via the TCP/IP protocol. The ADS route is then the interface between the ADS and TCP/IP connection. The ADS route indicates which AmsNetId is assigned to which TCP/IP address. As a result, the ADS function blocks no longer use the TCP/IP address, but the AmsNetId.

## Requirements:

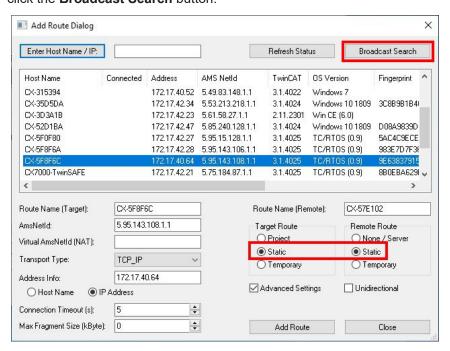
- Two CX70x0 Embedded PCs.
- Both CX70x0s are in the same network and accessible via ADS.

## Proceed as follows:

- 1. Start TwinCAT and connect to the first CX70x0 (see: Connect to the CX70x0 [▶ 129]).
- 2. On the left in the tree view, click Routes, select the Static Routes tab, and click the Add button.

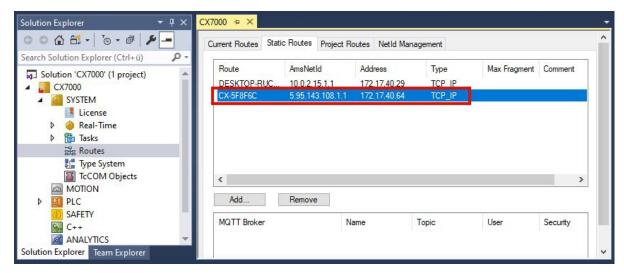


3. Under **Remote Route**, select the **Static** option so that the ADS route remains in the project, and then click the **Broadcast Search** button.





4. Select the second CX70x0 as the destination of the ADS route. The ADS route is entered for both Embedded PCs. The AmsNetId of the second CX70x0 is displayed and can be used in the program for ADS function blocks.



5. Now connect to the second CX70x0, which has been set as the destination of the ADS route, and write a small program. Define an array and increment a value of the array.

```
VAR
MarksTest AT %MB0 : ARRAY[0..9] of INT;
END_VAR

Program:
MarksTest[0]:=MarksTest[0]+1;
```

- 6. Activate the configuration and switch the CX70x0 to Run mode.
- 7. For the first CX70x0, write a program that reads the incremented value of the array.

```
VAR
    ADSREAD : ADSREAD:
    NetID : STRING:='5.81.38.23.1.1'; (* AMSNetId of the target*)
    Value : INT; (* value of target MarksTest[0]*)
    Error : INT;
    NoError : INT;
END VAR
Program:
    ADSREAD (
        NETID:=NetID ,
PORT:=851 , (* plc port of the target*)
        IDXGRP:=16#4020 , (* Marks %MB*)
        IDXOFFS:=0 , (* Marks offset in byte*)
LEN:=2 , (* length of data in byte*)
        DESTADDR:=ADR(Value) , (* pointer to the data in which the value is to be stored *)
         READ:=TRUE ,
        TMOUT:= ,
        BUSY=> ,
        ERR=> ,
        ERRID=> );
    IF NOT ADSREAD.BUSY THEN
        IF NOT ADSREAD.ERR THEN
             NoError:=NoError+1;
        ELSE
            Error:=Error+1;
        END IF
    ADSREAD (Read:=FALSE);
    END IF
```

- 8. The incremented value is read out and transmitted to the first CX70x0.
- ⇒ You should see on the first CX70x0 how the value of the Value variable is incremented. The writing of the data works in the same way. Data can be written with the ADSWRITE function block. Make sure that you set the offset (IDXOFFSET) to 10 in this sample setup so that the array [4... 9] is written. Limit the length to 10 bytes, as an array of 0... 9 of type INT was created and the memory thus uses %MB0... MB19 (10 \* 2 bytes) (The elements 0...4 for reading the array and the elements 5...9 for writing it).

Use one ADS command at a time. Wait until the ADS service is finished, i.e. the BUSY output of the



function block is switched to FALSE, and only then use the next ADS function block. To optimize the access timing, you can also use an ADSREADWRITE function block that reads and writes the data at the same time.

# 8.1.4 Creating a PLC project

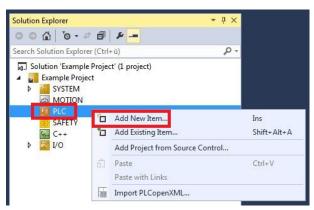
The next steps describe how to create a PLC project in TwinCAT and add it in the tree view.

Prerequisites for this step:

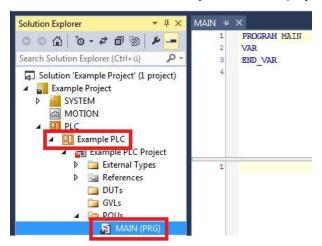
· A newly created TwinCAT XAE project.

## Create a PLC project as follows:

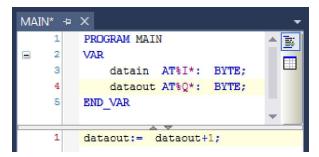
- 1. Right-click on PLC in the tree view.
- 2. In the context menu click on Add New Item and select the Standard PLC Project.



3. In the tree view click on the newly created PLC project, then double-click on MAIN (PRG) under POUs.

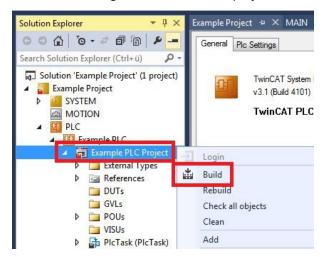


4. Write a small program, as shown in the diagram below.

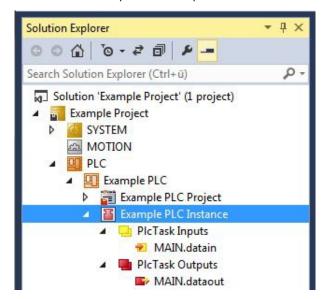




5. In the tree view right-click on the PLC project, then click on **Build** in the context menu.



⇒ You have successfully created a PLC project and added the project in TwinCAT. A PLC instance with the variables for the inputs and outputs is created from the PLC project.



In the next step you can link the variables with the hardware.



# 8.1.5 Linking variables

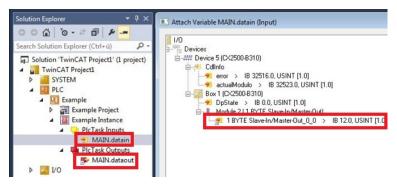
Once the PLC project was successfully added in the System Manager, you can link the newly created input and output variables from the PLC project with the inputs and outputs of your hardware.

Prerequisites for this step:

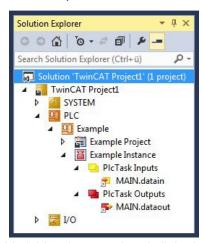
· A PLC program attached in TwinCAT.

#### Link the variables as follows:

Double-click on the input or output variables in the tree view under PLC.
 The Attach Variable window appears and shows which inputs or outputs can be linked with the variables from the PLC project.

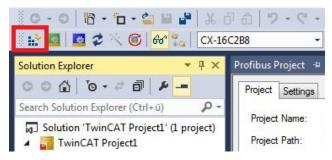


2. Double-click on the inputs or outputs of the hardware in the **Attach Variable** window. Link the input variables with the inputs and the output variables with the outputs of the hardware.



Variables that are already linked are indicated with a small arrow icon in TwinCAT.

3. In the toolbar click on Activate Configuration.



- 4. Confirm the request whether TwinCAT is to start in Free Run mode with Yes.
- ⇒ You have successfully linked variables with the hardware. Use Activate Configuration to save and activate the current configuration.

The configuration can now be loaded on the CX, in order to automatically start TwinCAT in Run mode, followed by the PLC project.



# 8.1.6 Load configuration to CX

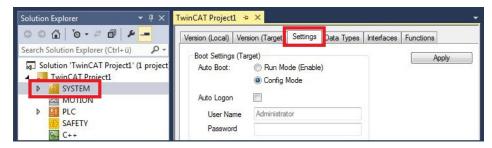
Once variables are linked, the configuration can be saved and loaded on the CX. This has the advantage that the PLC project is loaded and started automatically when the CX is switched on. The start of the previously created PLC project can thus be automated.

Prerequisites for this step:

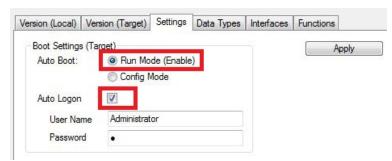
- · A completed PLC project, added in the System Manager.
- · Variables from the PLC project, linked with the hardware in the System Manager.
- · A CX selected as target system.

## Load the configuration from the System Manager to the CX as follows:

- 1. In the tree view on the left click on SYSTEM.
- 2. Click on the **Settings** tab.



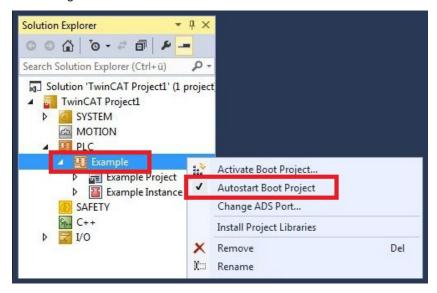
3. Under Boot Settings select the option Run Mode (Enable) and tick the Auto Logon checkbox.



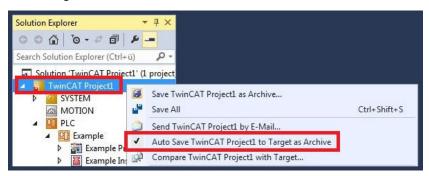
- 4. Enter the user name and password for the CX in the User Name and Password fields.
- 5. Click on Apply.
- 6. In the tree view on the left right-click on the PLC project under PLC.



7. In the context menu click on **Autostart Boot Project**. The setting is selected



- 8. Right-click on the project folder in the tree view.
- 9. In the context menu click on **Auto Save to Target as Archive**. The setting is selected.



⇒ You have successfully loaded the CX configuration. From now on, TwinCAT will start in Run mode and the PLC project will start automatically.

Next, the master can be added in a new project in the System Manager and can then be used to find slaves that have already been set up.

CX7051 Version: 1.0 139



# 8.2 TwinCAT tabs

In TwinCAT, information and settings for the CANopen interface are added under tabs. The main TwinCAT tabs are described in this chapter. In addition, the section illustrates how the CANopen interface is displayed in the tree view under TwinCAT.

# 8.2.1 Tree view

A CANopen master and a CANopen slave connected to it are displayed in the tree view as follows:

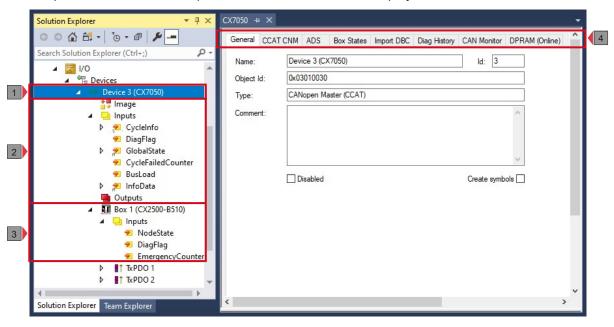


Fig. 38: CANopen master and CANopen slave in the TwinCAT tree view with tabs.

In this sample the slave was linked to the master. TwinCAT was then scanned for the master, and the master was created in TwinCAT together with the slave.

| No. | Description  |
|-----|--|
| 1   | The device name of the master is shown in brackets. All CANopen slaves are added under the master.   |
| 2   | Under the CANopen master, status messages are listed as input variables. The variables can be linked with the PLC and used for diagnostic purposes (e.g. error codes, counters, etc.). |
| 3   | CANopen slaves are added under the master, labeled as box and numbered consecutively. The device name appears in brackets after it.  |
|     | Each CANopen slave has its own input variables for diagnostic purposes, which indicate the state of the communication.   |
| 4   | Further settings for the CANopen master or slave can be implemented under the tabs.  |
|     | Other tabs are displayed, depending on whether the master or slave is selected in the tree view.   |



A CANopen slave and the corresponding tabs are shown as follows in the tree view:

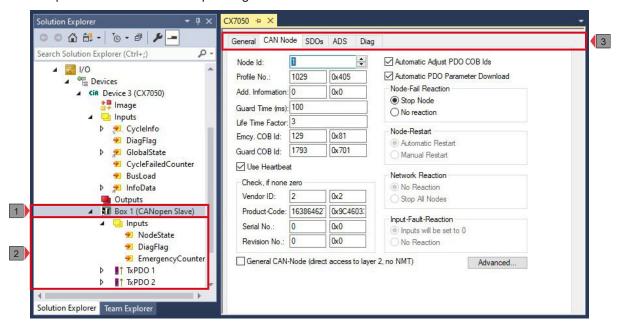


Fig. 39: CANopen slave in the TwinCAT tree view with associated tabs.

| No. | Description   |
|-----|---|
| 1   | Under the CANopen slave, status messages are listed as input variables. The variables can be linked with the PLC and used for diagnostic purposes.                                      |
| 2   | The process data objects (PDO) are displayed under the CANopen slave. At this point the variables for the data transmission are also created. The variables can be linked with the PLC. |
|     | The data transfer direction is described from the perspective of the slave:   |
|     | RxPDOs are received by the device.  |
|     | TxPDOs are sent by the device.  |
| 3   | Further settings for the CANopen slave can be implemented under the tabs.   |
|     | Other tabs are displayed, depending on whether slave or other entries are selected in the tree view.  |

When the PLC process image is read, the variables for status messages and the variables under the process data objects can be linked with the variables from the PLC program. Double-click on a variable name in the tree view to open the link dialog. The link variables are identified with a small arrow icon.

Further information about TwinCAT can be found in the TwinCAT documentation on the Beckhoff website: www.beckhoff.de

CX7051 Version: 1.0 141



# 8.2.2 CANopen slave

## 8.2.2.1 CAN node

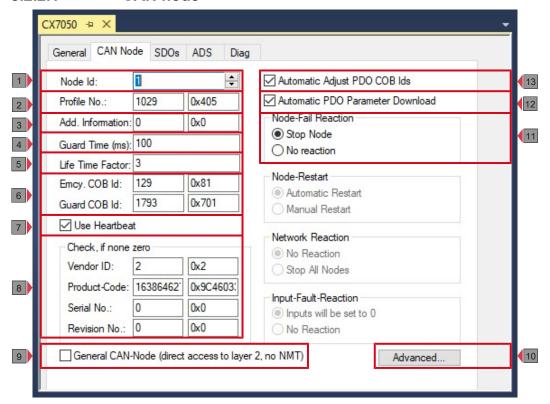


Fig. 40: CAN Node tab of a CANopen slave in TwinCAT.

| No. | Description   |
|-----|---|
| 1   | The address is set here.  |
| 2   | According to CANopen the parameter 0x1000 "Device Type" contains in the two least significant bytes the number of the device profile supported by the device. This number is entered here and compared with the parameter in the device on system startup. If no device profile is supported, the parameter will contain the value 0.   |
| 3   | Add. Information:  The Add. Information is in the two most significant bytes of the object dictionary entry 0x1000 (Device Type). The set/actual configuration comparison only takes place if Profile No. or Add. Information (i.e. object directory entry 0x1000) is set to a value that is not null. If the expected values do not match the actual values on system startup, the node start is aborted and a corresponding error message is displayed on the Diag tab. |
| 4   | Guard Time: The Guard Time determines the interval in which the node is monitored (Node Guarding). The value entered is rounded up to the next multiple of 10 ms. 0 signifies no monitoring.  |
| 5   | Life Time Factor: Guard Time x Life Time Factor determines the watchdog length for the mutual monitoring of master and slave. The entry 0 means that the slave does not monitor the master. If 0 is entered, the master directly takes the guard time as watchdog length.   |
|     | The heartbeat protocol is also supported, and the system initially tries to initiate this form of node monitoring on the CANopen node. If this attempt fails, guarding is activated.  |
| 6   | Emcy COB ID / Guard COB ID are identifiers for emergency messages or the guarding protocol. They result from the node address.  |



| No. | Description   |
|-----|---|
| 7   | Heartbeat is used for monitoring of the node. If heartbeat is disabled, guarding is used for monitoring.  |
|     | The guard time as producer heartbeat time and (guard time x lifetime factor) as consumer heartbeat time are entered. In this case a heartbeat telegram with the smallest configured guard time sent. The guard time can be set individually for each node.  |
| 8   | If values other than zero are entered here, these identity object inputs (0x1018 in the object directory) are read off at the system StartUp and compared with the configured values. The corresponding node will be started only if the values coincide. It is also possible to compare only some of the values (e.g. the vendor ID and the product code). In this case, parameters that are not used must be set to zero. |
| 9   | If this option is selected, the entire CANopen network management is disabled for this device. It is not started, monitored, etc. The PDO entries are regarded as pure CAN telegrams (layer 2) and are made available to the controller on an event-driven basis.   |
| 10  | Opens a window with further settings, which can be enabled:   |
|     | Switch off upload object 0x1000.  |
|     | Switch off download object 0x1006.  |
|     | <ul> <li>Switch off automatic sending of start node (then has to be sent<br/>manually).</li> </ul>  |
|     | Continue to send start SDOs, in the event of a termination.   |
| 11  | The option StopNode is used to set the node to "stopped" state after a fault. It can be used to set nodes to a safe state, although they can no longer be addressed via SDO.  |
| 12  | If the option is selected, entries are created automatically in TwinCAT, which are transferred via SDO on system startup (see: <u>SDOs [▶ 144]</u> tab).  |
| 13  | If the option is selected, the default identifiers of the process data objects are automatically adjusted if the node ID changes (see: no. 6).  |

CX7051 Version: 1.0 143



# 8.2.2.2 SDOs

The SDO tab is used to display and manage entries, which are sent to the node on startup.

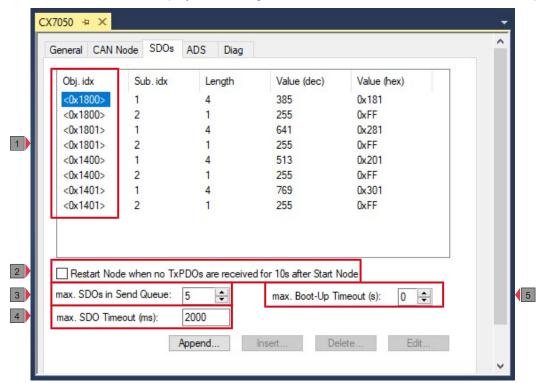


Fig. 41: SDO tab of a CANopen slave in TwinCAT.

| No. | Description   |
|-----|---|
| 1   | Object index entries in angle brackets were created automatically based on the current configuration. |
|     | Further entries can be created and managed via "Append", "Insert", Delete" and "Edit".                |
| 2   | If this option is selected, the slave is restarted if no TxPDO was received after 10 seconds.         |
| 3   | This option can be used to set the maximum number of SDOs in the send queue.                          |
| 4   | The maximum timeout (ms) for the SDO is set here.   |
| 5   | The boot-up timeout (s) is set here.  |



#### 8.2.2.3 PDO

This tab appears if you click on a process data object (PDO) in the tree view.

Process Data Objects (PDOs) are CAN telegrams which transport process data without a protocol overhead.

- · RxPDOs are received by the device.
- · TxPDOs are sent by the device.

A device sends its input data with TxPDOs and receives the output data in the RxPDOs. This designation is retained in TwinCAT.

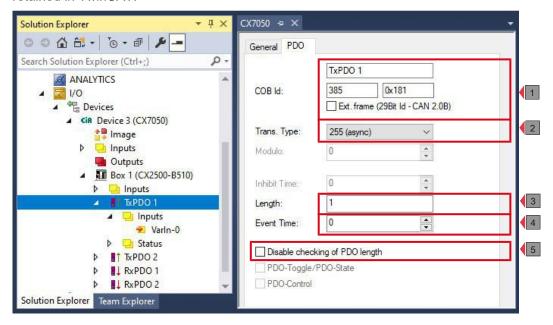


Fig. 42: PDO tab of a CANopen slave in TwinCAT.

| No. | Description   |
|-----|---|
| 1   | CAN identifier of the PDO. For two send and receive PDOs per node, CANopen provides Default Identifiers. These can then be changed.   |
| 2   | The Transmission Type determines the send behavior of the PDO. 255 corresponds to the event-driven sending (see: Setting the transmission type).  |
| 3   | The length of the PDO depends on the created variables and can therefore not be edited here.  |
| 4   | Enter the value for the Event Timer in ms. For send PDOs (RxPDOs), PDOs are sent again after a timer has elapsed. For receive PDOs (TxPDOs), the arrived PDOs are monitored, and the box state of the node may be modified.                                       |
|     | TwinCAT creates corresponding inputs in the node object directory on the basis of the parameters entered here. These are transferred via SDO at the system start. The entries can be viewed in the SDO tab (see: <a href="SDOs [&gt; 144]">SDOs [&gt; 144]</a> ). |
|     | This function can be disabled via the checkbox <b>Automatic PDO Parameter Download</b> on the <b>CAN Node</b> tab (see: <u>CAN node</u> [ <u>&gt; 142</u> ]).   |
| 5   | The PDO length check can be disabled here.  |

## 8.3 Creating CX705x as slave

This section shows how to create a CX7051 as CANopen slave. In order for the CANopen slave to be recognized later by a CANopen master with all inputs and outputs, the CANopen slave must first be created in TwinCAT and configured with all associated PDOs and variables.

Requirements for this step:

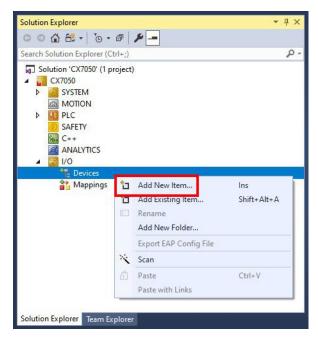


CX7051

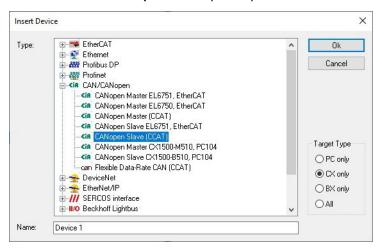
• CX7051 selected as target device.

#### Create the CANopen slave as follows:

- 1. In the tree view on the left, right-click on **Devices**.
- 2. In the context menu click Add New Item.



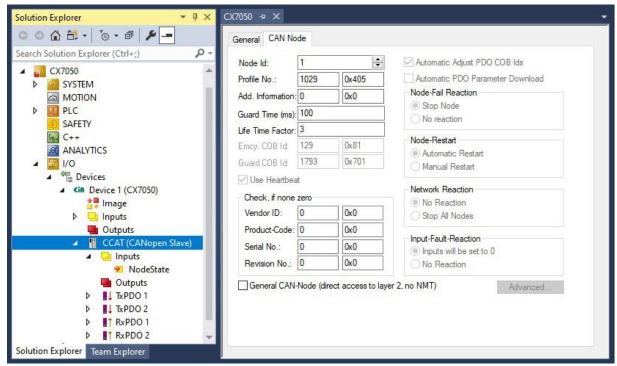
3. Select as device CANopen Slave (CCAT) and confirm the selection with OK.



4. Confirm the request with Yes, in order to look for boxes.



⇒ The CANopen slave has been successfully added in TwinCAT and is displayed in the tree view with the inputs and outputs.



In the next step you can extend the process image by creating additional virtual slaves. Or you can set the address, once the slave configuration is complete.



### 8.3.1 Creating a virtual slave

Additional virtual slaves can be created on the same hardware interface. This enables more data to be exchanged with a CANopen master, or a connection with a second CANopen master can be established. Up to three virtual slaves can be created on the same hardware interface of a slave.

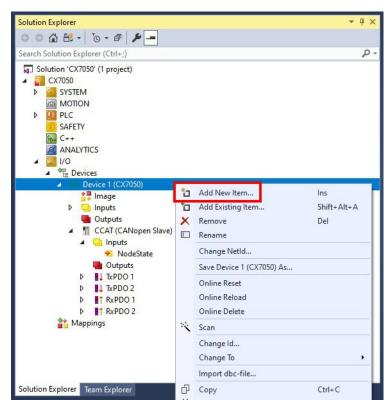
Because a maximum of 16 PDOs can be configured for each slave, the additional three virtual slaves increase the maximum possible number of PDOs to 4 x 16 PDOs in each send direction. Each virtual slave is assigned a dedicated address via TwinCAT and is configured like an independent device for the CANopen master.

Requirements for this step:

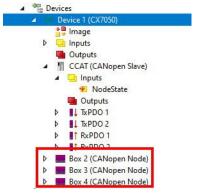
· A CANopen slave, created in TwinCAT.

#### Create a virtual slave as follows:

- 1. Right-click on a CANopen slave in the tree view on the left.
- 2. In the context menu click Add New Item.



⇒ Another box (virtual slave) is created.



Own variables can now be created for the virtual slave. In the next step you can set the address for the slave.



### 8.3.2 Setting the address

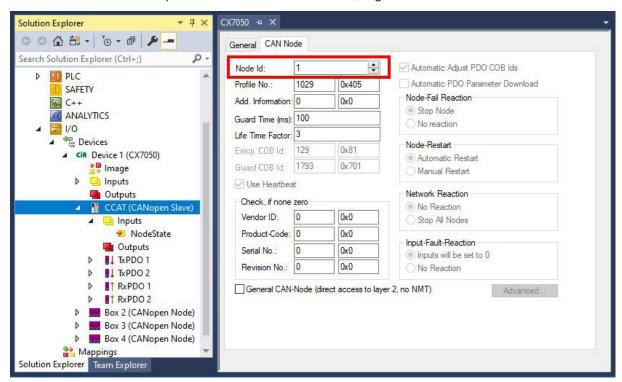
Once the CANopen slave was successfully added in TwinCAT, the address of the CANopen slave can be set. This step shows how to set the address in TwinCAT so that the CANopen slave can be reached by the CANopen master via this address.

Requirements for this step:

· An added CANopen slave in TwinCAT.

#### Parameterize the CANopen slave as follows:

- 1. Click on a slave box.
- 2. Click the CAN Node tab.
- 3. Enter a value for the CANopen address in the Node Id field, e.g. "1".



⇒ You have set the address successfully. The CANopen master can reach the CANopen slave with the set address. You can now create further PDOs.

CX7051 Version: 1.0 149



## 8.3.3 Creating further PDOs

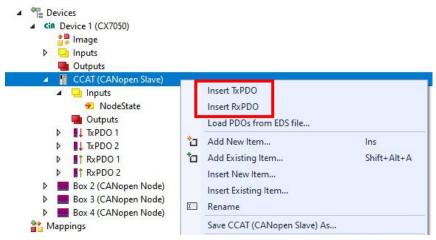
The CANopen slave can exchange up to 16 PDOs (each with 8 bytes of process data) with the CANopen master in input and output direction. By default 2 PDOs are created in Tx and Rx direction. Here we shown how to create further PDOs for a CANopen slave.

Requirements for this step:

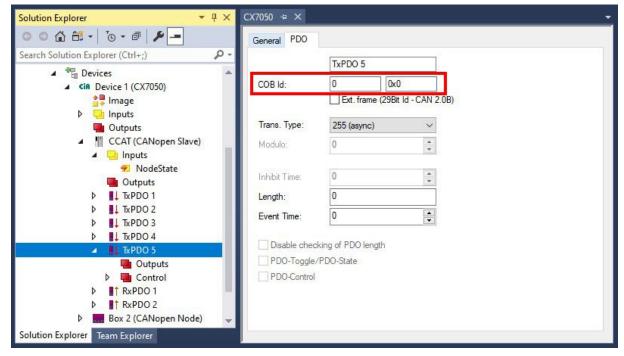
· A CANopen slave added in the tree view.

#### Create the PDOs as follows:

- 1. Right-click on a CANopen slave in the tree view.
- 2. Click in the context menu on **Insert TxPDO** or **Insert RxPDO** to create PDOs in Tx or Rx direction.



The new TxPDOs or RxPDOs are inserted under the already created PDOs and numbered consecutively in the tree view. **Notice** From the fifth PDO in Tx or Rx direction the COB Id is no longer entered automatically (see the following figure).



- 3. From the fifth PDO in Tx or Rx direction click on the PDO tab.
- 4. Enter the desired value in the COB Id field.
- ⇒ You have successfully created further PDOs; in the next step you can create variables for the data exchange under the PDOs.



### 8.3.4 Creating variables

In TwinCAT the PDOs are filled with variables, which can later be linked with the PLC program. A maximum of 8 bytes of data can be created under the corresponding PDOs. It is also allowed to use different variable types, only the limit of 8 bytes per PDO must be observed.

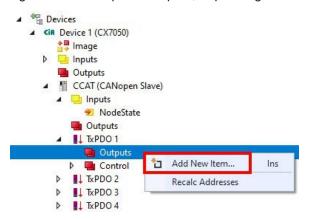
If it is not configured differently in the master, the data is sent automatically with every change. At the planning stage please ensure that the data in a PDO "only" change at a moderate rate (e.g. not with ms frequency). Failure to adhere to this can lead to CAN overload. If this is not observed, the CAN can be overloaded, which can happen quickly, especially at low baud rates.

Requirements for this step:

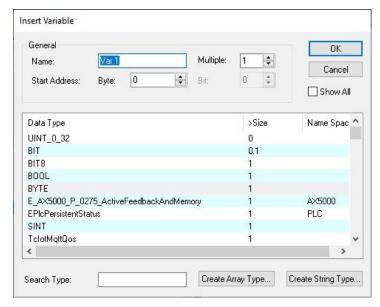
· Newly created PDOs, which are to be filled with variables.

#### Create the variables as follows:

- 1. In the tree view click on a TxPDO or RxPDO to show more information.
- 2. Right-click on Outputs or Inputs, depending on whether a TxPDO or RxPDO is selected.



- 3. Click **Add New Item** in the context menu. The **Insert Variable** window appears.
- 4. Click on the appropriate variable and click **OK**.



⇒ You have successfully created a variable. The new variable is shown in the tree view on the left. In this way you can add further variables for the CANopen slave. In the next step you can specify the transmission type, thereby specifying how the process data objects are transferred.



## 8.3.5 Setting the transmission type

The transmission type determines how the process data objects are transferred. The transmission type for the RxPDOs and TxPDOs is set on the PDO tab.

The available transmission types are: acyclic synchronous, cyclic synchronous and asynchronous.

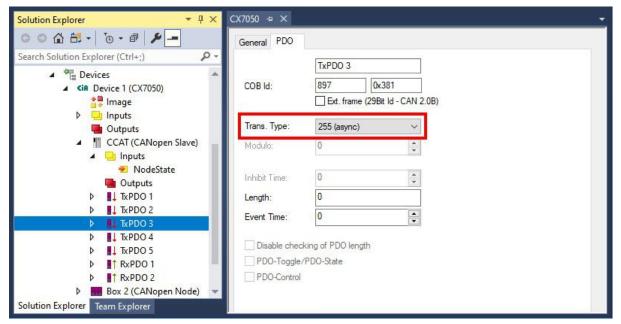
| Transmission type: | Acyclic Synchronous | Cyclic Synchronous | Asynchronous |
|--------------------|---------------------|--------------------|--------------|
| Name in TwinCAT:   | (acyc, sync)        | (cyc, sync)        | (async)      |

Requirements for this step:

· A CANopen slave with process data objects (PDO) added in TwinCAT

#### Specify the transmission type as follows:

- 1. In the tree view, left-click on a process data object (PDO).
- 2. Click the PDO tab.
- 3. Select the required transmission type under **Trans. Type**.



⇒ You have successfully specified a transmission type for a process data object. The transmission types for the remaining process data objects are specified in the same way. Next, you can create a PLC project for the CANopen slave.



## 8.3.6 Receiving SDO data in the PLC

SDO data that are unknown to the CANopen part of the software and cannot be processed automatically are transferred to the PLC, where they are evaluated and answered via ADS notification.

To this end the ADS port must be enabled in the System Manager under CAN device.

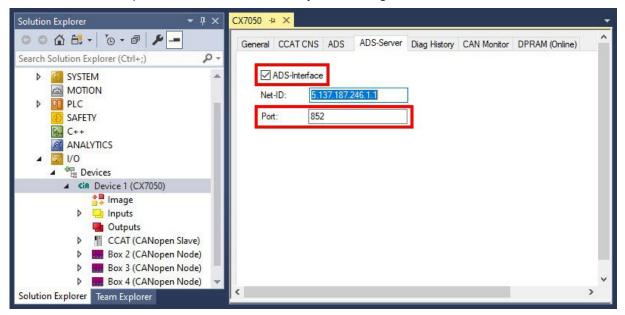


Fig. 43: Enabling of an ADS port for a CANopen slave.

#### **SDO Read request**

Data to be read must be received with ADSREADIND and answered with ADSREADRES.

| Input parameter ADSREADIND | Description   |
|----------------------------|---|
| NETID                      | NetID of the CAN interface                              |
| Port number                | 0x1000 <sub>hex</sub> + Node number                     |
| IDXGRP                     | 16#8000_0000 + SDO Index (IDXGRP.31 = ADS-Notification) |
| IDXOFFS                    | SDO Subindex  |
| LEN                        | not required for reading                                |

You now have to respond to the ADS indication with an ADS Read response.

| Input parameter ADSREADRES | Description                               |
|----------------------------|---|
| NETID                      | NetID of the CAN interface                |
| Port number                | 0x1000 <sub>hex</sub> + Node number       |
| INVOKEID                   | INVOKEID of the ADSREADIND function block |
| RESULT                     | Error <> 0, error-free = 0                |
| LEN                        | Length of the data                        |

#### **SDO Write request**

Data to be written must be received with ADSWRITEIND and answered with ADSWRITERES.

| Output parameter ADSWRITEIND | Description   |
|------------------------------|---|
| NETID                        | NetID of the CAN interface                              |
| Port number                  | 0x1000 <sub>hex</sub> + Node number                     |
| IDXGRP                       | 16#8000_0000 + SDO Index (IDXGRP.31 = ADS Notification) |
| IDXOFFS                      | SDO Subindex  |

CX7051 Version: 1.0 153



| Output parameter ADSWRITEIND | Description                      |
|------------------------------|----------------------------------|
| LEN                          | Number of received data in bytes |

You now have to respond to the ADS indication with an ADS Write Response.

| Input parameter ADSWRITERES | Description                                |
|-----------------------------|--|
| NETID                       | NetID of the CAN interface                 |
| Port number                 | 0x1000 <sub>hex</sub> + Node number        |
| INVOKEID                    | INVOKEID of the ADSWRITEIND function block |
| RESULT                      | Error <> 0, error-free = 0                 |

# 8.3.7 Switching slave node to PreOp from the PLC

The ADSWRTCTL block can be used to set individual CANopen nodes to pre-operational or operational state. A fixed baud rate is required for this purpose.

| Input parameters | Description                                   |
|------------------|---|
| NETID            | NetId of the CAN interface                    |
| Port number      | 0x1000 <sub>hex</sub> + Nodeld (slave number) |
| ADSSTATE         | ADSSTATE_RUN                                  |
| DEVSTATE         | 0 - Pre / 1 - Operational                     |
| LEN              | 0   |
| SRCADDR          | 0   |



# 8.4 Reading the CAN baud rate

The baud rate can be displayed and evaluated via the variable InfoData[1]. This can be helpful for slaves with AutoBaud, if for example the communication is not running. This can be used to check whether the correct baud rate has been set with AutoBaud.

| NodeState value | Description |
|-----------------|-------------|
| 0x01040400      | 1 Mbaud     |
| 0x01040600      | 800 kbaud   |
| 0x01040C00      | 500 kbaud   |
| 0x010A0C00      | 250 kbaud   |
| 0x01160C00      | 125 kbaud   |
| 0x011C0C00      | 100 kbaud   |
| 0x013A0C00      | 50 kbaud    |
| 0x01940C00      | 20 kbaud    |
| 0x01941A10      | 10 kbaud    |

# 8.5 Sending arbitrary CAN telegrams

The ADSWRITE command can be used to send any CAN message.

| Input parameters | Description                 |
|------------------|-----------------------------|
| NETID            | NetId of the CAN interface  |
| Port number      | 200                         |
| IDXGRP           | 16#0000F921                 |
| IDXOFFS          | 0                           |
| LEN              | 11 bytes                    |
| SRCADDR          | Pointer to an 11 byte ARRAY |

Table 17: Structure of the 11 byte CAN data

| Byte | Description     | Example Node 7 SDO 0x607<br>Len 8 Download Request 0x2100<br>(Index)<br>Sub Index 1 - Value "1" |
|------|-----------------|---|
| 1    | COB-ID LowByte  | 0x06 (SDO Low Byte)   |
| 2    | COB-ID HighByte | 0x07 (SDO High Byte)  |
| 3    | LEN (length)    | 0x08 (LEN, may be 5 in this case)   |
| 4    | Data[1]         | 0x22 (Download Request)   |
| 5    | Data[2]         | 0x00 (Index Low Byte)   |
| 6    | Data[3]         | 0x21 (Index High Byte)  |
| 7    | Data[4]         | 0x01 (Sub Index)  |
| 8    | Data[5]         | 0x01 (Value "1")  |
| 9    | Data[6]         | 0x00  |
| 10   | Data[7]         | 0x00  |
| 11   | Data[8]         | 0x00  |

CX7051 Version: 1.0 155



## 8.6 Reading the IP and MAC addresses

This sample shows you how to read the IP and MAC addresses. The function block FB\_MDP\_NIC\_Read can be used to retrieve information from the network adapter.

#### Sample

```
Var
    FB MDP NIC Read
                        : FB MDP NIC Read;
END_VAR
PROGRAM:
FB MDP NIC Read(
   bExecute:=TRUE ,
    tTimeout:= ,
    iModIdx:= ,
    sAmsNetId:= ,
    bBusy=> ,
    bError=> ,
    nErrID=> ,
    iErrPos=>
    stMDP ModuleHeader=> ,
    stMDP ModuleContent=> );
```

The output stMDP\_ModuleHeader displays the header information. The output stMDP\_ModuleContent displays, among other things, the information about the IP and MAC addresses.

| stMDP_ModuleHeader | ST_MDP_ModuleHea |                     |
|--------------------|------------------|---------------------|
| iLen               | UINT             | 4                   |
| nAddr              | DWORD            | 131072              |
| sType              | T_MaxString      | 'Nic'               |
| sName              | T_MaxString      | 'st'                |
| nDevType           | DWORD            | 141072              |
|                    | ST_MDP_NIC_Prope |                     |
| iLen               | UINT             | 8                   |
| sMACAddress        | T_MaxString      | '00:01:05:5f:0f:7a' |
| sIPAddress         | T_MaxString      | '169.254.123.15'    |
| sSubnetMask        | T_MaxString      | '255.255.0.0'       |
| ▶ bDHCP            | BOOL             | TRUE                |
|                    | BYTE             | 0                   |

Fig. 44: Content of the MDP module with IP and MAC address.

## 8.7 Virtual Ethernet interface

The virtual Ethernet interface integrates network adapters into the TwinCAT system. This makes it possible to establish a virtual Ethernet communication via ADS, TCP or UDP to a BK9xx0. Do not use more than two BK9xx0 and a cycle time > 50 ms.

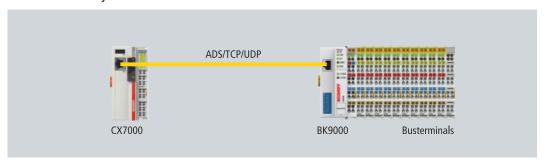
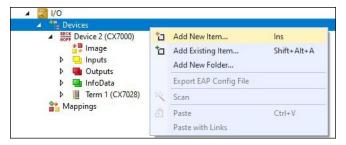


Fig. 45: Virtual Ethernet communication via ADS, TCP or UDP.

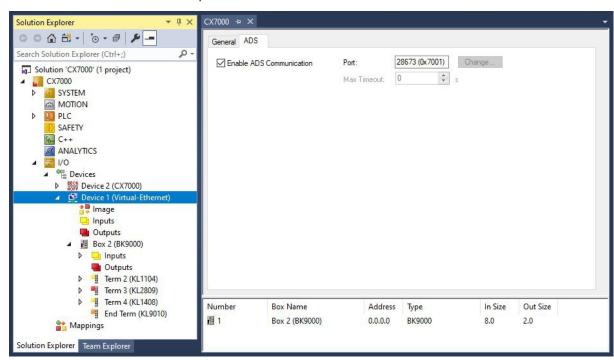
#### Proceed as follows:



1. In the tree view on the left, right-click on **Devices**.



- 2. Click on Add New Item and select the Virtual Ethernet Interface.
- ⇒ The Virtual Ethernet Interface is created in the tree view on the left. The ADS port number can be read out under the ADS tab. The Enable ADS Communication option must be active so that ADS communication to the BK9xx0 is possible.



## 8.8 CoE access to multi-function I/Os

The FB\_EcCoeSdoReadEx function block allows data to be read from an object directory of an EtherCAT slave via SDO data (Service Data Object). The nSubIndex and nIndex parameters allow the object that is to be read to be selected. Via bCompleteAccess := TRUE the parameter can be read with subelements.

Sample: Read the firmware version of the multi-function I/Os.

```
VAR
AMSNetID AT %I*:T_AmsNetIdArr;
Port AT %I*:T_AmsPort;
FB_EcCoESdoReadEx: FB_EcCoESdoReadEx;
FirmwareVersion: STRING;
END_VAR
```

The AmsNetId and port number are required for communication with the CX7028 interface. The inputs of the function block FB\_EcCoeSdoReadEx can be linked with the input variables netId and port under TwinCAT, so that the function block is permanently connected to the CX7028 interface.

CX7051 Version: 1.0 157





Fig. 46: CoE access to multi-function I/Os, input variables "netId" and "port" under TwinCAT.

The input sNetId of the function block corresponds to the input netId under TwinCAT. The function block requests a string and the link returns a byte array. You can convert the byte array to a string using the F CreateAmsNetId function. The input nSlaveAddr corresponds to the input port under TwinCAT.

```
FB_EcCoESdoReadEx(
sNetId:=F_CreateAmsNetId(nIds:=AMSNetID) , (* AmsNetId of the CX7028 Interface *)
nSlaveAddr:=Port , (* Port Number(nSlaveAddr): 0x1000 *)
nSubIndex:= ,
nIndex:=16#100A , (* Index Number *)
pDstBuf:=ADR(FirmwareVersion) ,
cbBufLen:=SIZEOF(FirmwareVersion) ,
bExecute:=TRUE ,
tTimeout:= ,
bCompleteAccess:= ,
bBusy=> ,
bError=> ,
nErrId=> );
```

The index number for the CoE object Software version is located under the CoE Online tab.

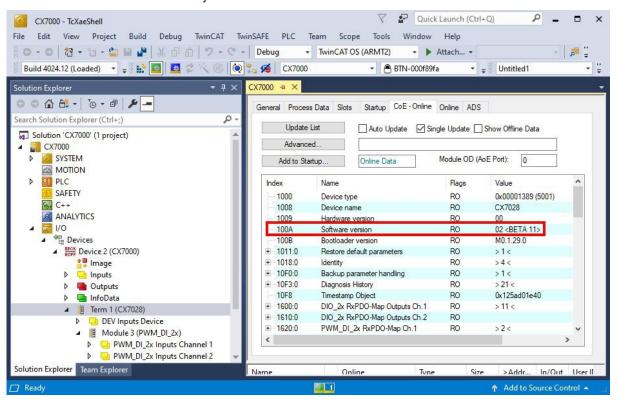


Fig. 47: CoE communication, listing of CoE objects with matching index number.

With the FB\_EcCoeSdoWriteEx function block an object from the object directory of an EtherCAT slave can be written by SDO-Download. Pay attention to whether the object can be accessed for reading; this is displayed in the Flags column. The nSubIndex and nIndex parameters allow the object that is to be written to be selected. Via bCompleteAccess := TRUE the parameter can be written with subelements.



## 8.9 Power supply terminal

EtherCAT Terminals (E-bus) or Bus Terminals (K-bus) can optionally be connected directly on the right-hand side; the CX7051 automatically recognizes which system is connected during the start-up phase.

#### K-bus interface

The CX7051 reads out the terminal types during scanning and creates them in the System Manager under a Bus Coupler.

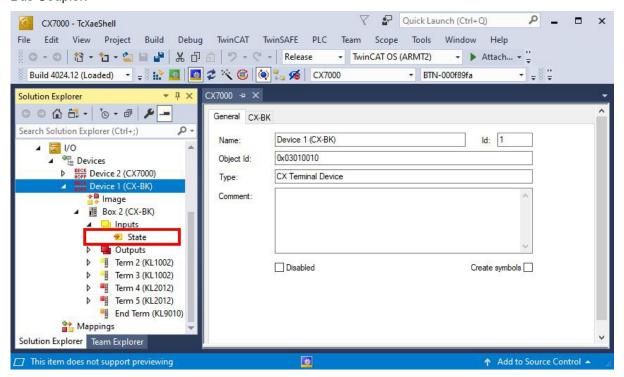


Fig. 48: K-bus interface of a CX7051 in the TwinCAT System Manager.

For K-bus diagnostics there is a status variable in TwinCAT under the Bus Coupler, which can be used for diagnostic purposes and indicates the status of the K-bus communication. For more information, refer to the chapter "Error handling and diagnostics" at K-bus [\dagger 175].

#### E-bus interface



#### **Distributed clocks**



The Embedded PCs of the CX7000 series are not suitable for the use of EtherCAT slaves that use distributed clocks or require them.

The operation of EtherCAT Terminals and EtherCAT devices is also possible at CX7051. The CX7051 also recognizes these terminals automatically during scanning, reads out the terminal types and creates them in the System Manager under an EtherCAT Coupler.



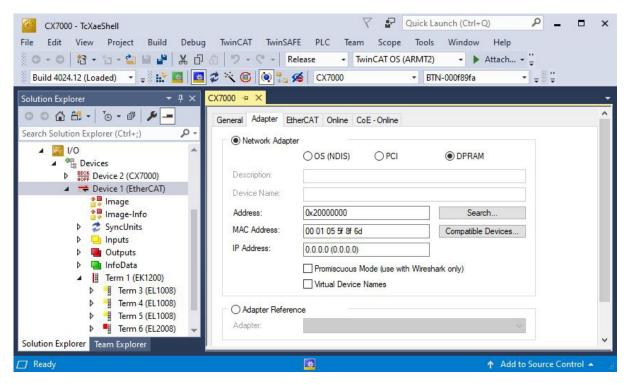


Fig. 49: E-bus interface of a CX7051 in the TwinCAT System Manager.

For more information on diagnostics, refer to the chapter "Error handling and diagnostics" at <u>E-bus [▶ 178]</u>.



# 8.10 Cycle and processing times

## 8.10.1 Measuring processing time in the PLC program

This sample shows you how to determine the processing time of a program code with the help of a small PLC program. This allows you to measure, for example, how long the PLC needs for a mathematical function, a loop or a specific program part. The resolution is 1 ns per digit.

#### Sample

```
VAR

MeasureStart : T_DCTIME64;
MeasureResult : T_DCTIME64;
END_VAR

PROGRAM:
MeasureStart:=F_GetActualDcTime64(); (*Insert your program code to measure the processing time*)
MeasureResult:=F GetActualDcTime64()-MeasureStart;
```

#### Requirements

| Development environment | Target platform          | PLC libraries to include |
|-------------------------|--------------------------|--------------------------|
| TwinCAT v3.1.0          | PC or CX (x86, x64, ARM) | Tc2_EtherCAT             |

## 8.10.2 Real-Time Clock (RTC)

The CX7051 has an internal, capacitor-buffered real-time clock (RTC) for time and date, which continues to run in the switched-off state. The capacitance of the capacitor is sufficient for at least 30 days and, unlike a battery-backed solution, is maintenance-free. The time is lost and must be reset if the CX7051 is turned off for more than 30 days

The following settings are possible in the boot.conf file:

- SNTP Server
- Update time (default = 1 hour)
- · Change UTC Offset
- DHCP server

#### Sample

The sample below shows you how to read the time. In the sample, the time is output as UTC time and one hour is added to get the CET time.

```
VAR
                        : FB_LocalSystemTime;
: DATE_AND_TIME;
    FB LocalSvstemTime
    DATEANDTIME
    DATEANDTIME_Add1h : DATE_AND_TIME;
END VAR
PROGRAM:
FB LocalSystemTime(
       sNetID:= ,
       bEnable:=TRUE ,
       dwCycle:= ,
        dwOpt:= ,
        tTimeout:= ,
        bValid=> ,
        systemTime=> ,
        tzID=> );
DATEANDTIME:=SYSTEMTIME TO DT(TIMESTR:=FB LocalSystemTime.systemTime); (*UTC Time*)
DATEANDTIME_Add1h:=DATEANDTIME+T#1H; (*UTC Time + 1h*)
```



#### Requirements

| Development environment |   | PLC libraries to be integrated (category group) |
|-------------------------|---|---|
|                         | PC or CX (WES7/Win7/Win10: TC<br>RT x86/x64, WEC6/7: TC RT x86,<br>WEC7: TC CE7 ARMV7, TC/BSD:<br>TC RT x64, TC OS ARMT2) | Tc2_Utilities (System)                          |

### 8.10.3 Cycle time of 250 μs

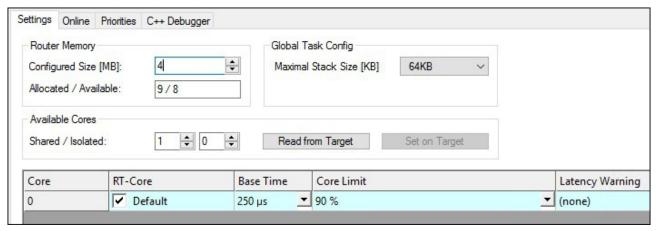
Note that a cycle time of 250  $\mu$ s on a CX7051 represents an extreme optimum and all boundary conditions must be right. Furthermore, a cycle time of 250  $\mu$ s only makes sense if the inputs and outputs are correspondingly fast.

The CX7051 has different interfaces, including, for example, the K-bus. The K-bus can achieve perhaps 1 ms under optimal conditions and is therefore unsuitable for cycle times of 250 µs. The E-bus (EtherCAT) is much faster, but the structure of an EtherCAT frame and the merging of the data into an EtherCAT frame is much more complex, so that only 1 ms is possible here as well.

Of course, EtherCAT can be operated with other Industrial PCs under 100 µs. However, these are usually equipped with more powerful CPUs and may use a DMA controller for EtherCAT processing. That is not the case with the CX7051, however, so the CPU power and the interfaces to EtherCAT are the limiting factors. Of course, the CX7051 as a small controller was not developed for high-speed applications and, due to its cost-efficiency, should not be compared to more powerful Industrial PCs.

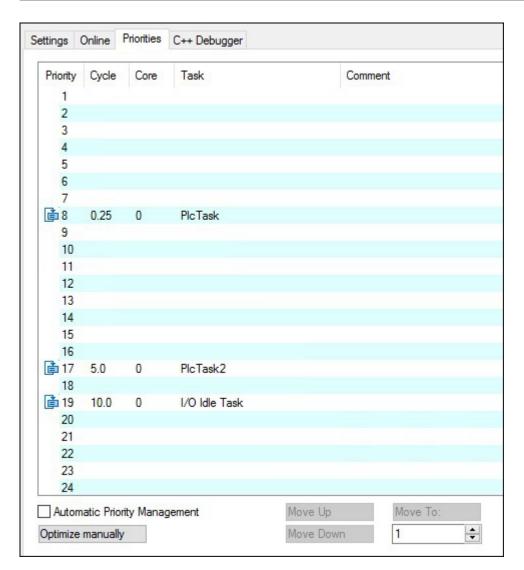
#### Setting a cycle time of 250 µs

A cycle time of 250 µs is possible on a CX7051 if the boundary conditions are right. The CX7051 is helped by the multifunction I/Os, which are connected to the CPU via a fast IO connection. The connection is kept very lean and has a correspondingly good data throughput. It is possible to reach the 250 µs with the help of the multifunction I/Os. Of course, the PLC program may contain only very little code and the core limit must be set to 90%, which in turn results in the described disadvantages (see: Real-time and CPU load [▶ 189]).



In addition, you should set the priority of the task so that the 250  $\mu$ s task has the highest priority in the system.





If you now allow a digital output of the CX7028 interface to toggle, for example with  $Out_01:=not_0ut_01$  in the 250 µs task, the task is output at a frequency of 2 kHz. In order for the output to be optimally fast, this output should have a load. Only wire the output with a digital input; as a result, the load is very small and the switch-off behavior of the driver is relatively slow. Slow here means in relation to the 250 µs task time. It makes a difference whether the output requires 50 µs or 100 µs to switch off. If you now wish to measure the response time, i.e. the time it takes for the CX7051 to react to an input, the following background is important:

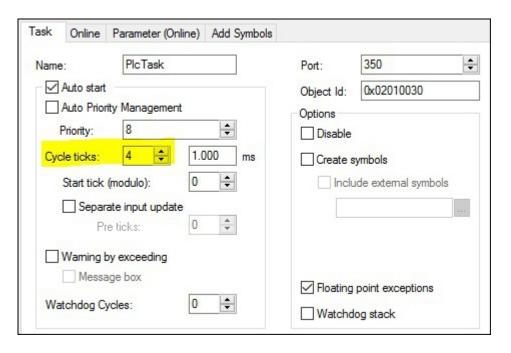
From a cycle time of 1 ms or greater, an optimal cycle is operated, i.e. the inputs of the CX7028 interface are read by the processor of the CX7028 interface about 20% before the new task cycle. If the task time is faster than 1 ms, the time is not sufficient for the optimized response time. In this case the inputs are read with the task cycle. As a result, a task time of 500 µs achieves the same response time as a task time of 1 ms. With a task time of less than 1 ms, the update needs four task cycles for a cycle. With 1 ms or slower it needs two task cycles. This should make you aware that it is not always the shortening of the cycle time that shortens the reaction time, but also the internal process, which plays a decisive role in the reading of the data.

Here is a sample, so that you can reproduce this behavior yourself and see and measure the differences:

- 1. Connect the +24 V Up and 0 V Up power supply to power the multifunction I/Os.
- 2. Connect output 1 to input 1 to toggle the output as described.
- 3. Connect output 2 to input 2.
- 4. Set the core limit to 90%, the base time to 250  $\mu$ s, the priority of the fast task to the highest priority, and the idle task to 10 ms.

The inputs have only a minimal filter time and are therefore well suited for the measurement. A load on the output is not necessary in this case. For the following samples, we always leave the base time at 250  $\mu$ s and only increase the number of cycle ticks in order to set the corresponding task time.





#### Sample program

```
PROGRAM MAIN
VAR
    bOut 1 AT %Q*:BOOL; (*toggle Output link to digital Output pin 7*)
    bOut_2 AT %Q*:BOOL; (*reaction time link to digital Output pin 14*)
    bIn 1 AT %I*: BOOL; (*toggle Output link to digital Input pin 2*)
    bIn 2 AT %I*: BOOL; (*reaction time link to digital Input pin 10*)
    fbTimer : TON;
    fbflanke1 : R TRIG;
    fbflanke2 : R TRIG;
    cnt1: INT; (*toggle Output*)
    cnt1 M: INT; (*toggle Output*)
    cnt2: INT; (*reaction time*)
    cnt2_M: INT; (*reaction time*)
END VAR
PROGRAM MAIN
bOut_1:= NOT bOut_1; (*toggle Output*)
bOut 2:= NOT bIn \overline{2}; (*reaction time*)
fbflanke1(CLK:=bIn_1);
IF fbflanke1.Q THEN
   cnt1:=cnt1+1; (*toggle Output*)
END IF
fbflanke2(CLK:=bIn 2);
IF fbflanke2.Q THEN
   cnt2:=cnt2+1; (*reaction time*)
fbTimer(PT:=T#1S,in:=NOT fbTimer.Q);
IF fbTimer.Q THEN
    cnt2_M:=cnt2; (*reaction time*)
    cnt1_M:=cnt1; (*toggle Output*)
    cnt1:=0;
    cnt2:=0;
END IF
```

The toggling of the output results in a frequency of 2 kHz - 250  $\mu$ s On, 250  $\mu$ s Off - i.e. a period duration of 500  $\mu$ s. When measuring the positive edge, this is 2000 edge changes in one second.



|               | BOOL   | TRUE  |
|---------------|--------|-------|
|               | BOOL   | TRUE  |
|               | BOOL   | FALSE |
| bIn_2         | BOOL   | FALSE |
| ⊌ 🎓 fbTimer   | TON    |       |
| ⊌ 📦 fbflanke1 | R_TRIG |       |
| ⊌ 🎓 fbflanke2 | R_TRIG |       |
|               | INT    | 1014  |
| ont1_M        | INT    | 2000  |
|               | INT    | 253   |
|               | INT    | 500   |

Fig. 50: Measurement at a task time of 250 µs.

In the case of the response time, it is 500 changes in one second, as the optimized access to the inputs does not apply here.

|   | • | bOut_1    | BOOL   | TRUE  |
|---|---|-----------|--------|-------|
|   | • | bOut_2    | BOOL   | TRUE  |
|   | • | bIn_1     | BOOL   | TRUE  |
|   | • | bIn_2     | BOOL   | FALSE |
| Ŧ | • | fbTimer   | TON    |       |
| H | • | fbflanke1 | R_TRIG |       |
| Ŧ | • | fbflanke2 | R_TRIG |       |
|   | • | cnt1      | INT    | 68    |
|   | • | cnt1_M    | INT    | 1001  |
|   | • | cnt2      | INT    | 17    |
|   |   | cnt2_M    | INT    | 250   |

Fig. 51: Measurement at a task time of 500 µs.

As expected, the values are only half as large with a task time that is twice as long.

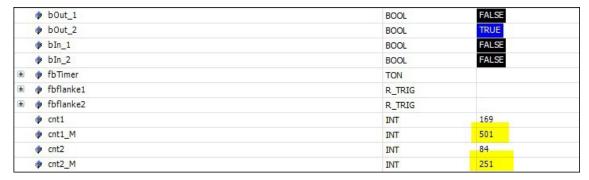


Fig. 52: Measurement at a task time of 1 ms.

With a task time of 1 ms, you can clearly see that the optimized mode actually helps to reduce the response time. While the toggle change has halved again, i.e. it is now still 500 Hz with a task time of 1 ms, the value for the response time has remained the same.

CX7051 Version: 1.0 165



### 8.10.3.1 Cycle time ≥1 ms



Fig. 53: CX7051 CPU and PLC.

Yellow and red: Mapping and update of the IOs.

Light grey: Time remaining until the task begins again (OS).

Dark grey: PLC cycle.

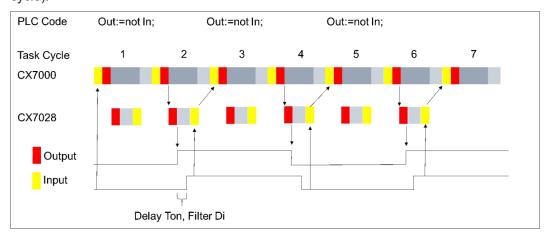


Fig. 54: CPU of the CX7028 interface.

Red: Output update.

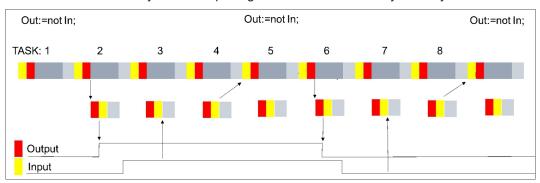
Grey: CPU processing of the multifunction IOs.

**Yellow:** Input update (from a cycle time of 1 ms there is a waiting period of up to approx. 80% of the cycle time before the update of the input signals so that the inputs are read as late as possible, i.e. before the next cycle).



### 8.10.3.2 Cycle time < 1 ms

From a cycle time of < 1 ms, the update of the input signals is carried out immediately and is therefore only available with the next cycle. The input signals are therefore always one cycle old.



With this background knowledge, you should be able to make the right settings on the CX7051 for your application.



### 8.11 Function Blocks

### 8.11.1 FB\_CX70xx\_RW\_EEPROM

```
FB_CX70xx_RW_EEPROM

bExecute BOOL

eMode E_CX70xx_EEPROM_Mode

pSrcBuf PVOID

cbSrcBufSize UINT

pDstBuf PVOID

cbDstBufSize UINT

UINT nWritesCycles

cbDstBufSize UINT
```

The function block allows a maximum of 120 bytes to be written to the EEPROM (hardware) of the CX70xx. The EEPROM may be written to a maximum of 200 times. The memory is intended for one-time writing.

This function block can be used to personalize the CX70xx. That means, in the simplest case you write your company ID into the EEPROM. When starting the CX70xx program, read the contents of the memory. For example, if it is empty, you cannot continue to run the program because it is no longer your original CX70xx that you programmed.

If you want to exchange a CX70xx for a new device, the EEPROM must be written again by you.

### Inputs

```
VAR_INPUT

bExecute : BOOL; // rising edge triggers process with selected mode

eMode : E_CX70xx_EEPROM_Mode; // select RW mode

pSrcBuf : PVOID; // pointer to WRITE EEPROM data buffer

cbSrcBufSize : UINT; // size of WRITE EEPROM data buffer (max.120 Bytes)

pDstBuf : PVOID; // pointer to READ EEPROM data buffer

cbDstBufSize : UINT; // max.size of READ EEPROM data buffer (max.120 Bytes)

END VAR
```

| Name        | Туре                         | Description  |
|-------------|------------------------------|--|
| bExecute    | BOOL                         | A positive edge starts the function block.   |
| eMode       | E_CX70xx_<br>EEPROM_M<br>ode | ReadOnly: EEPROM read<br>WriteOnly: EEPROM write<br>WriteAndRead: EEPROM write and read  |
| pSrcBuf     | PVOID                        | Pointer to the data buffer to be written.  |
| cbSrcBufLen | UINT                         | Length of data to be written (max. 120 bytes)  |
| pDstBuf     | PVOID                        | Pointer to the data buffer into which the contents of the EEPROM are to be copied.   |
| cbDstBufLen | UINT                         | Length of data to be read. (maximum 120 bytes) When reading, the length information must be greater than or equal to the data contained in the EEPROM. |

## Outputs

```
VAR OUTPUT
   bBusy
                    : BOOL;
                                    // FB is working
                  : BOOL;
                                    // FB has an Error
   nErrorID
                     : UDINT;
                                     (* Error Code
   If nErrorID=DEVICE INVALIDACCESS the EEPROM write cycles reached max. value.
   If nErrorID=DEVICE_INVALIDPARM the given pointer parameter is invalid/null.
   If nErrorID=DEVICE INVALIDSIZE the given buffer size is too small or too big.
   If nErrorID=DEVICE_SRVNOTSUPP probably the image version need to be updated to support this feat
ure. *)
   nDataSizeEEPROM : UINT;
                                     // current size of (read) EEPROM data in bytes (max.120 Bytes)
   nWritesCycles
                   : UINT;
                                     // already performed EEPROM write cycles (maximum possible = 20
END VAR
```

| Name  | Туре | Description                               |
|-------|------|---|
| bBusy | BOOL | The function block is active and working. |



| Name            | Туре  | Description   |
|-----------------|-------|---|
| bError          | BOOL  | The function block has an error.  |
| nErrorID        | UDINT | ADS Error Code<br>Examples:   |
|                 |       | DEVICE_INVALIDACCESS: the EEPROM write cycles have reached the maximum value. The EEPROM cannot be rewritten.       |
|                 |       | DEVICE_INVALIDPARM: the allocated pointers are invalid/NULL.  |
|                 |       | DEVICE_INVALIDSIZE: the allocated buffer size is too small or too large.  |
|                 |       | DEVICE_SRVNOTSUPP: the image version of the CX70xx does not support this feature. An update (>=35695) is necessary. |
| nDataSizeEEPROM | UINT  | Current size in bytes of the read EEPROM data   |
| nWritesCycles   | UINT  | Number of write operations still available  |

## 8.11.2 FB\_CX70xx\_ResetOnBoardIO



The function block allows to execute a reset from the OnBoard I/O of the CX70xx Embedded PC.

Typical use case is after an error in the communication to the OnBoard I/Os (CX7028). Such an error occurs when the power supply (Up) of the OnBoard I/Os is interrupted.

#### **NOTICE**

#### State of the I/Os

Outputs that are still set in the process image are switched on again immediately after a reset.

Further details on the OnBoard I/O can be found in the documentation of the CX70xx Embedded PC.

### Inputs

| Name     | Туре       | Description  |
|----------|------------|--|
| bExecute | BOOL       | A positive edge starts the function block.   |
| sNetId   | T_AmsNetID | AMS Net ID of the OnBoard I/Os   |
| tTimeout |            | States the length of the timeout that may not be exceeded by execution of the ADS command. |

## Outputs

```
VAR_OUTPUT

bBusy : BOOL;  // FB is working

bError : BOOL;  // FB has an Error

nErrorID : UDINT;  (* Error Code. If nErrorID=DEVICE_SRVNOTSUPP probably the image version need to be updated to support this feature. *)

END VAR
```

| Name   | Туре | Description                               |
|--------|------|---|
| bBusy  | BOOL | The function block is active and working. |
| bError | BOOL | The function block has an error.          |



| Name     | Туре  | Description   |
|----------|-------|---|
| nErrorID | UDINT | ADS Error Code Examples:  |
|          |       | DEVICE_SRVNOTSUPP: the image version of the CX70xx does not support this feature. An update (>=47912) is necessary. |

#### Sample:

```
FUNCTION BLOCK FB Test ResetOnboardIO
   AMSNetID : T AmsNetIdArr;
                                 // link to the AMS Net ID of the OnBoard IOs
   // link to the State of the OnBoard IOs
                                 // if Ready to Reset you can reset the OnBoard IOs
END VAR
IF State<>8 AND NOT State.8 AND State.4 THEN // if OnBoard IO device signals an error and is not OP
but present
   bReset := TRUE;
   bReset := FALSE;
END IF
IF NOT fbReset.bBusy AND bReset THEN
   fbReset(bExecute:=TRUE, sNetId:=F CreateAmsNetId(AMSNetID));
   fbReset (bExecute:=FALSE);
END IF
```

## 8.12 Important attribute pragmas

Attribute pragmas are used to influence compilation and pre-compilation. TwinCAT supports a number of predefined attribute pragmas. Attributes are defined in the declaration part.

### 8.12.1 Attribute 'Tc2GvIVarNames'

The pragma has the effect that symbols, which are declared in a GVL, are addressed via ADS just like in TwinCAT 2 (without the use of the GVL name as namespace).

Syntax: {attribute 'Tc2GvIVarNames'}

#### Sample:

```
{attribute 'Tc2GvlVarNames'}
VAR_GLOBAL
    Test : INT;
END_VAR

GVL.Test:=GVL.Test+1; (*without attribute*)
Test:=Test+1; (*with attribute*)
```

## 8.12.2 Attribute 'pack\_mode'

This attribute pragma specifies how a data structure is packaged during allocation. The attribute must be inserted above the data structure and affects the packing of the whole structure.

Syntax: {attribute 'pack\_mode' := '<Value>'}

#### Sample

```
{attribute 'pack_mode' := '0'}
TYPE str_Test :
STRUCT
    byTest1 : BYTE;
    iTest : DINT;
    byTest2 : BYTE;
    nValue : INT;
END_STRUCT
END_TYPE
```



In this sample, the pack mode has been set to 0. If you determine the size of the structure in the sample with SIZEOF, you get the value 8.

1 byte + 4 bytes (DINT) + 1 byte + 2 bytes (INT) = 8 bytes

If you set the pack mode to 2 (WordAlignment), you get the value 10 because a padding byte is inserted after each byte. If you set the pack mode to 4 (DWordAlignment), then you get the value 12, because this time three padding bytes are inserted after each byte. A pack mode of 8 (LWordAlignment) does not change anything, because the sample does not use variables that require 8 bytes.

The CX7051 works with the DWordAlignment (pack mode 4) if you do not use the attribute.

For more information about the pack mode attribute, see: Attribute 'pack mode'

## 8.12.3 Attribute 'TcCallAfterOutputUpdate'

The attribute pragma TcCallAfterOutputUpdate causes the IO update to take place before the PLC cycle and not after the PLC program as is set by default.

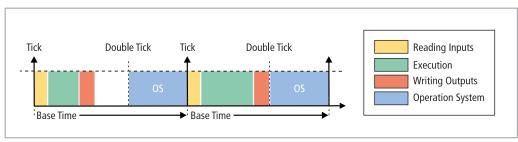


Fig. 55: Default calling of a PLC task.

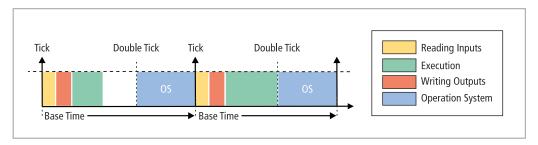


Fig. 56: Calling a PLC task with the attribute tcCallAfterOutputUpdate.

This function can be used for projects with strongly fluctuating cycle times. In projects with strongly fluctuating cycle times, the outputs, since they are written after the PLC cycle, are sometimes written earlier (short PLC cycle time) and sometimes later (long PLC cycle time). These fluctuations cause jitter in the outputs. The disadvantage is that the attribute cannot react quite as quickly and a cycle is always lost. You have to decide whether you want to react quickly to an input (default setting) or whether you prefer to have a deterministic behavior of the outputs (setting of the attribute).

Syntax: {attribute 'TcCallAfterOutputUpdate'}

Insertion location: This attribute must be added to all program POUs, which are to be called after the output update.

#### Sample:

To illustrate the behavior, you need a digital output terminal such as an EL2008 and an oscilloscope.

Write a small PLC program and link the variable bout with a digital output:

bOut:=not bOut;

The PLC program is very simple and does not cause any fluctuations. The pulse is displayed on the oscilloscope as follows:



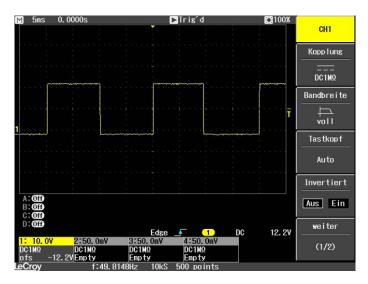


Fig. 57: Pulse of a digital output without load.

Now extend the PLC program with a For loop to create a program load. The mathematical function used does not matter and is intended only to generate a load:

Whenever the output is set to TRUE, the loop is run through and a load is generated. As a result, more time is needed to run the PLC and the output is written later than usual. During the next cycle, the output is set back to FALSE, the loop is not run through and the output is set to FALSE faster, because the PLC program is finished faster without a For loop. The result is that the pulse is very much shorter.

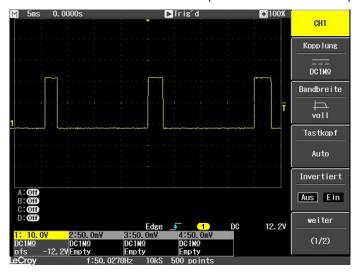


Fig. 58: Shortened pulse of a digital output with load.

If the For loop is called upon FALSE instead of TRUE, the result is inverted.



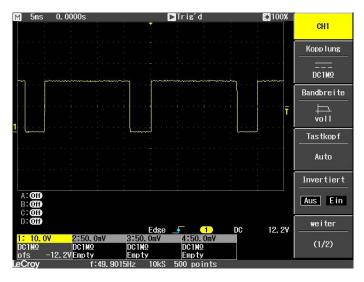


Fig. 59: Inverted representation of a digital output.

With the attribute pragma TcCallAfterOutputUpdate, the pulse is constant and is independent of how long the For loop takes or whether it is called. The whole thing only works if the PLC task is not exceeded. Therefore, when reproducing the sample, pay attention to the exceed counters of the task.

#### Detecting a PLC program with different runtimes

The PLC program must be supplemented in order to detect PLC programs with different runtimes. Different runtimes are not recognizable in the online view, since an average value is always formed over several cycles. Therefore, outliers can only be detected if they lie above the task time. If the outliers are still within the task time, they are not easily visible.

For this we then use the system variable: PlcTaskSystemInfo

```
VAR
    bOut : BOOL;
    PlcTaskSystemInfo : PlcTaskSystemInfo;
    udiValue : ARRAY[0..19] of UDINT;
    Cnt : INT;
END VAR
Program:
bOut:=not bOut;
IF bOut THEN
   For loop:=1 to 2000 do
        lrTest:=SIN(INT TO LREAL(loop)*3.14);
    END FOR
END IF
PlcTaskSystemInfo:= TaskInfo[1];
udiValue[Cnt]:= PlcTaskSystemInfo.LastExecTime;
cnt:=cnt+1;
IF Cnt >19 THEN
        Cnt := 0;
END IF
```

With this program extension you can see that the PLC program with a For loop requires 7.7 ms and without a For loop 1.1 ms. The specification is 100 ns per digit.



| ■ 🐞 udiValue      | ARRAY [0 19] OF U |       |
|-------------------|-------------------|-------|
| udiValue[0]       | UDINT             | 77728 |
| udiValue[1]       | UDINT             | 10713 |
| udiValue[2]       | UDINT             | 71049 |
| udiValue[3]       | UDINT             | 11065 |
| udiValue[4]       | UDINT             | 69882 |
| udiValue[5]       | UDINT             | 11027 |
| udiValue[6]       | UDINT             | 77084 |
| udiValue[7]       | UDINT             | 11939 |
| udiValue[8]       | UDINT             | 77494 |
| udiValue[9]       | UDINT             | 18527 |
| udiValue[10]      | UDINT             | 76724 |
| udiValue[11]      | UDINT             | 11043 |
| udiValue[12]      | UDINT             | 71519 |
| udiValue[13]      | UDINT             | 11406 |
| udiValue[14]      | UDINT             | 79004 |
| udiValue[15]      | UDINT             | 11118 |
| udiValue[16]      | UDINT             | 70745 |
| udiValue[17]      | UDINT             | 12007 |
| www.udi\/alug[19] | LIDINE            | 77261 |

Fig. 60: Determination of different running times in the PLC program.

The measurement coincides with the displays on the oscilloscope, on which it can be seen that a pulse is sometimes 6.5 ms longer or 6.5 ms shorter. You can measure the processing time of the For loop (Measuring processing time in the PLC program [ $\triangleright$  161]). The result of this measurement will coincide with the observed values through the program extension, with a certain inaccuracy and jitter.

CX7051 Version: 1.0 173



# 9 Error handling and diagnostics

# 9.1 Diagnostic LEDs

| Display    | LED  | Color                          | Description   |
|------------|------|--------------------------------|---|
|            | TC   | Green                          | TwinCAT is in Run mode.   |
| TC         |      | Red                            | TwinCAT is in Stop mode.  |
| FB<br>DIAG |      |                                | Additionally indicates errors during system startup by error code and error argument (see table: TC-LED, error description and remedy). The red LED flashes with two different frequencies. |
|            |      | Blue                           | TwinCAT is in Config mode.  |
| SD         |      | Yellow                         | Error or crash of the PLC.  |
| חכ         | FB   | Green                          | CAN is OK.  |
|            |      | Red                            | CAN in bus-off.   |
|            |      | Green and red flashing, 200 ms | CAN warning.  |
|            |      | Off                            | CAN not configured.   |
|            | DIAG | Green                          | All nodes have NodeState = 0  |
|            |      | Green and red flashing, 200 ms | All boxes in OP mode, but the task has not yet started.   |
|            |      | Red 200 ms                     | Not all nodes in OP mode.   |
|            |      | Off                            | No boxes configured.  |
|            |      | Red                            | If only the DIAG LED lights up when starting the CX70xx, then the bootloader is damaged and the device must be sent in for repair.  |

The TC-LED flashes at a specified frequency and in a specified order, thus indicating the error code and argument.

Table 18: TC LED, order and meaning.

| Sequence             | Meaning               |
|----------------------|-----------------------|
| Fast flashing        | Starting the sequence |
| First slow sequence  | Error code            |
| No display           | Pause, the LED is off |
| Second slow sequence | Error argument        |

Count how many times the red TC LED flashes in order to determine the error code and argument.

Table 19: TC LED, error description and remedy.

| Error code | Error argument | Description                                      | Remedy                                     |
|------------|----------------|--|--|
| 1          | 1              | microSD card not recognized                      | Check the microSD card.                    |
|            | 2              | Card init failed - preloader                     | Image is defective. Install a new image on |
|            | 3              | No partition found - preloader                   | the microSD card.                          |
|            | 4              | Filesystem mount failed - preloader              |  |
|            | 5              | Card init failed - loader                        |  |
|            | 6              | No partition found - loader                      |  |
|            | 7              | Filesystem mount failed - loader                 |  |
| 2          | 1              | Loader not found                                 |  |
|            | 2              | Loader file invalid (checksum, size, read error) |  |
|            | 3              | TC dll not found                                 |  |



| Error code | Error argument | Description                                    | Remedy                             |
|------------|----------------|--|------------------------------------|
|            | 4              | TC dll checksum error                          |                                    |
|            | 5              | EEPROM file missing or invalid                 |                                    |
|            | 6              | TcOsSys.dll version not compatible with loader |                                    |
| 3          | 1              | Rbf not found                                  |                                    |
|            | 2              | CCAT 1 init failed                             |                                    |
|            | 3              | CCAT 2 init failed                             |                                    |
|            | 4              | CCAT EEPROM writing failed                     |                                    |
|            | 5              | CCAT 1 EEPROM reloaded failed                  |                                    |
|            | 6              | CCAT 2 EEPROM reloaded failed                  |                                    |
| 4          | 1              | Peripheral not working                         | Hardware defective, replace the CX |
|            | 2              | Voltage Vo not reached                         |                                    |
|            | 3              | Low speed external oscillator not running      |                                    |
|            | 4              | High speed external oscillator not running     |                                    |
|            | 5              | Flash failed                                   |                                    |
|            | 6              | Device overclocked (old Hardware)              |                                    |
| 5          | 5              | RAM error detected                             |                                    |

### 9.1.1 K-bus

The power supply unit checks the connected Bus Terminals for errors. The red LED "K-bus ERR" is off if no error is present. The red LED "K-bus ERR" flashes if Bus Terminal errors are present.

Table 20: Diagnostic LEDs in K-Bus mode.

| Display                                    | LED       | Meaning  |
|--|-----------|--|
| Us Up                                      | Us 24 V   | Power supply for basic CPU module. The LED lights green if the power supply is correct.  |
| Link / Act IO                              | Up 24V    | Power supply for terminal bus. The LED lights green if the power supply is correct.  |
| DI5 DI6 DI8 / AI2 DO2 DWM / DO3 DO4 / PWM2 | K-BUS RUN | Diagnostic K-bus. The green LED lights up in order to indicate error-free operation. "Error-free" means that the communication with the fieldbus system is also running. |
| THE POTENTIAL                              | K-BUS ERR | Diagnostic K-bus. The red LED flashes to indicate an error. The red LED flashes with two different frequencies.  |

The frequency and number of the flashes can be used to determine the error code and the error argument. An error is indicated by the "K-bus ERR" LED in a particular order.

Table 21: K-bus ERR LED, fault indication sequence through the LED.

| Order                | Meaning               |
|----------------------|-----------------------|
| Fast flashing        | Starting the sequence |
| First slow sequence  | Error code            |
| No display           | Pause, the LED is off |
| Second slow sequence | Error code argument   |

CX7051 Version: 1.0 175



Count how often the red LED K-bus ERR flashes, in order to determine the error code and the error argument. In the error argument the number of pulses shows the position of the last Bus Terminal before the error. Passive Bus Terminals, such as a power feed terminal, are not included in the count.

Table 22: K-BUS ERR LED, fault description and troubleshooting.

| Error code             | Error code argu-<br>ment | Description  | Remedy   |
|------------------------|--------------------------|--|--|
| Persistent, continuous |                          | EMC problems.  | Check power supply for undervoltage or overvoltage peaks.  |
| flashing               |                          |  | Implement EMC measures.  |
|                        |                          |  | If a K-bus error is present, it can be localized by a restart of the power supply unit (by switching it off and then on again).  |
| 3 pulses               | 0                        | K-bus command error.   | No Bus Terminal inserted.  |
|                        |                          |  | 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 this procedure until the faulty Bus Terminal has been found. |
| 4 pulses               | 0                        | K-bus data error, break behind the power supply unit.                        | Check whether the Bus End Terminal 9010 is connected.  |
|                        | n                        | Break behind Bus<br>Terminal n.  | Check whether Bus Terminal n+1 after the power supply unit is connected correctly; replace if necessary.   |
| 5 pulses               | n                        | K-bus error in register communication with Bus Terminal n.                   | Replace Bus Terminal at location n.  |
| 6 pulses               | 0                        | Error at initialization.   | Replace Embedded PC.   |
|                        | 1                        | Internal data error.   | Hardware reset of the Embedded PC (switch off and back on again).  |
|                        | 8                        | Internal data error.   | Hardware reset of the Embedded PC (switch off and back on again).  |
| 7 pulses               | 0                        | Process data lengths of the set and actual configurations do not correspond. | Check the configuration and the Bus<br>Terminals for consistency.  |

For some error the LED "K-BUS ERR" does not go out, even if the error was rectified. Switch the power supply for the power supply unit off and back on again to switch off the LED after the error has been rectified.



#### State variable

In TwinCAT there is a State variable under the Bus Coupler for K-bus diagnostics.

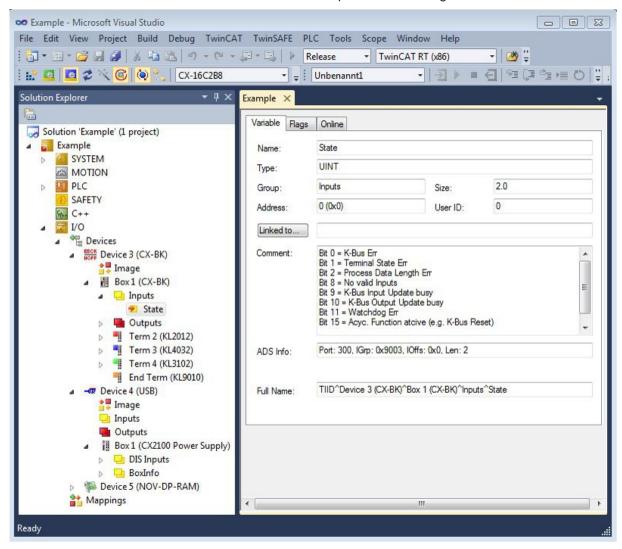


Fig. 61: Status variable for error handling and diagnostics under TwinCAT.

If the value is "0", the K-bus operates synchronous and without error. If the value is <> "0" there may be a fault, or it may only be an indication that the K-bus cycle is longer than the task. In which case it would no longer be synchronous with the task. The task time should be faster than 100 ms. We recommend a task time of less than 50 ms. The K-bus update time typically lies between one and five ms.

Table 23: Description of the State variable values.

| Bit    | Description   |
|--------|---|
| Bit 0  | K-bus error.  |
| Bit 1  | Terminal configuration has changed since the start. |
| Bit 2  | Process image lengths do not match.                 |
| Bit 8  | (still) no valid inputs.                            |
| Bit 9  | K-bus input update not yet complete.                |
| Bit 10 | K-bus output update not yet complete.               |
| Bit 11 | Watchdog.   |
| Bit 15 | Acyclic K-bus function active (e.g. K-bus reset).   |

If there is a K-bus error, this can be reset via the IOF\_DeviceReset function block (in the TcIoFunctions.lib).



## 9.1.2 E-bus

The power supply unit checks the connected EtherCAT Terminals. In E-bus mode the "Link/Act IO" LED is lit. When data are transferred, the "Link/Act IO" LED flashes.

Table 24: Diagnostic LEDs in K-Bus mode.

| Display               | LED      |         | Meaning   |
|-----------------------|----------|---------|---|
| Us Up                 | Us       |         | Power supply for basic CPU module. The LED lights green if the power supply is correct. |
| Link / Act 10         | Up       |         | Power supply for terminal bus. The LED lights green if the power supply is correct.     |
| DI5 DI6               | Link/Act | off     | E-bus not connected.  |
| Al1 / Dl7             | IO       | on      | E-bus connected / no data traffic.  |
| PWM1 / D03 D04 / PWM2 |          | flashes | E-bus connected / data traffic on the E-bus.  |



# 9.2 CANopen diagnostics

## 9.2.1 Status messages

The CANopen status messages provide additional information and can be used for diagnostic purposes.



The following table shows which values the variables can assume:

| Inputs             | Meaning   |
|--------------------|---|
| CycleInfo          | Cycle Counter: This counter is incremented by one after each cycle.   |
|                    | Error: Displays the number of boxes with a non-null BoxState.   |
|                    | ActualCycle Time: Reserved for future use   |
| DiagFlag           | This variable provides information on changes to the diagnostic data.   |
|                    | 0: Data unchanged.  |
|                    | 1: Data changed. Use ADS Read to read the data.   |
| GlobalState        | This variable provides information on the status of the master.   |
|                    | GlobalState[0]: 0: Device is in RUN state. 1: Device is in RESET state. 2: Device is in OFFLINE state. 3: Device is in STOP state.  |
|                    | GlobalState[1] (FW V02.14 and higher): Bit 0-7: RxError counter of the CAN controller. Bit 8-15: TxError counter of the CAN controller.   |
|                    | GlobalState[2]: Bit 0: CAN controller is in BUS-OFF. Bit 1: CAN controller warning limit reached.   |
|                    | Bit 2: Rx-Queue exceeded. Bit 3: Hi-Prio Tx-Queue exceeded. Bit 4: Lo-Prio Tx-Queue exceeded. Bit 5: CAN-Send Error (FW V02.14 and higher). Bit 6-14: reserved for future use. Bit 15: switches on every SYNC message sent. |
|                    | GlobalState[3]: Bus utilization in %.   |
| CycleFailedCounter | This counter is incremented by one whenever the last bus cycle is incomplete at the start of a TwinCAT cycle.   |
| BusLoad            | Bus load in %.  |
| InfoData           |   |



### 9.2.2 Communication

In the tree view, input variables are listed under the **Inputs** menu item, which provide information about a CANopen device.

The NodeState variable can be used to show the state of the CANopen communication, to indicate whether the slave is in data exchange or an error is present.

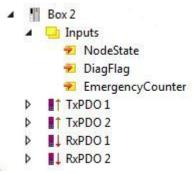


Fig. 62: Diagnosis of the CANopen communication with the variables NodeState, DiagFlag and EmergencyCounter.

#### **NodeState**

The following table shows which values the variable NodeState can assume:

| Value | Meaning  |
|-------|--|
| 0     | No error   |
| 1     | Node deactivated   |
| 2     | Node not found   |
| 4     | SDO syntax error at StartUp  |
| 5     | SDO data mismatch at StartUp                                       |
| 8     | Node StartUp in progress   |
| 11    | FC510x Bus-OFF   |
| 12    | Pre-Operational  |
| 13    | Severe bus fault   |
| 14    | Guarding: toggle error   |
| 20    | TxPDO too short  |
| 22    | Expected TxPDO is missing  |
| 23    | Node is Operational but not all TxPDOs were received               |
| 31    | only for EtherCAT gateways: WC-State of cyclic EtherCAT frame is 1 |
| 128   | Node is Operational but not all RxPDOs were received               |
| 129   | Node is Pre-Operational  |
| 130   | Node is Stopped  |

#### **DiagFlag**

The following table shows which values the variable DiagFlag can assume. This variable provides information on changes to the diagnostic data.

| Value | Meaning                                      |
|-------|--|
| 0     | Data unchanged.                              |
| 1     | Data changed. Use ADS Read to read the data. |

#### **EmergencyCounter**

The EmergencyCounter variable is incremented by one if an emergency telegram was received.



Table 25: Reading the emergency telegrams with the ADSREAD function block.

| Input parameters | Description  |
|------------------|--|
| NETID            | NetId of the CAN interface   |
| Port number      | 200  |
| IDXGRP           | 16#xxxxF180 (xxxx) Node-Id, the Diag flag is only reset when reading at least 106 bytes 16#xxxxF181 (xxxx) Node-Id, the Diag flag is reset immediately |
| IDXOFFS          | Byte Offset  |

Table 26: Description of the array

| Offset  | Bit        | Value / description                              |  |
|---------|------------|--|--|
| 0 - 1   | Bit 0      | reserved   |  |
|         | Bit 1      | Boot up message not received or incorrect        |  |
|         | Bit 2      | Emergency-Overflow                               |  |
|         | Bit 3 - 15 | reserved   |  |
| 2 - 3   | Bit 0 - 14 | TX-PDO (i+1) received                            |  |
|         | Bit 15     | All TX PDOs 16-n received                        |  |
| 4 - 5   | Bit 0 - 4  | 1: Incorrect TX PDO length                       |  |
|         |            | 2: synchronous TX PDO missing                    |  |
|         |            | 3: Node signaling PRE-OPERATIONAL                |  |
|         |            | 4: Event timer expired for a TX PDO              |  |
|         |            | 5: no response during guarding                   |  |
|         |            | 6: toggling missed several times during guarding |  |
|         | Bit 5 - 15 | Associated COB ID                                |  |
| 6       | Bit 0 - 7  | 1: incorrect value during SDO upload             |  |
|         |            | 2: incorrect length during SDO upload            |  |
|         |            | 3: Abort during SDO up/download                  |  |
|         |            | 4: incorrect date during a boot-up message       |  |
|         |            | 5: timeout while waiting for a boot-up message   |  |
| 7       | Bit 0 - 7  | 2: incorrect SDO command specifier               |  |
|         |            | 3: SDO toggle bit has not changed                |  |
|         |            | 4: SDO length too great                          |  |
|         |            | 5: SDO-Abort                                     |  |
|         |            | 6: SDO-Timeout                                   |  |
| 8 - 9   | Bit 0 - 7  | SDO up/download index                            |  |
| 10      | Bit 0 - 7  | SDO up/download subindex                         |  |
| 11      | Bit 0 - 7  | reserved   |  |
| 12      | Bit 0 - 7  | Abort errorClass                                 |  |
| 13      | Bit 0 - 7  | Abort errorCode                                  |  |
| 14 - 15 | Bit 0 - 15 | Abort additionalCode                             |  |
| 16 - 19 |            | Read value (if offset 6 = 1)                     |  |
| 20 - 23 |            | Expected value (if offset 6 = 1)                 |  |
| 24 - 25 |            | Number of consecutive emergencies                |  |
| 26 - n  |            | Emergencies (8 bytes each)                       |  |



## 9.2.3 PDOs

### **SendCounter**

TxPDOs feature an additional SendCounter variable under the **Control** menu item.



Fig. 63: Diagnostic variable SendCounter of a CANopen slave.

By default, PDOs are sent automatically when a change is made. This variable can be used if the data is to be sent not only when a change is made, but also when nothing has changed in the data in the PDO.

The variable must therefore be incremented by one if the data in the PDO is to be sent even without a change. If the variable is incremented in the same cycle and a change of data is made in parallel, only one telegram is sent.

Apart from this scenario, this variable can be used to monitor whether the corresponding PDO was sent when the data changed.

#### ReceiveCounter

RxPDOs feature an additional ReceiveCounter variable under the **Status** menu item.

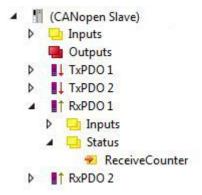


Fig. 64: Diagnostic variable ReceiveCounter of a CANopen slave.

The input variable is incremented by one whenever a PDO is received. In this way newly arrived PDO are always logged, even if the data in the PDO are unchanged. The variable also indicates whether a device is still sending data on a regular basis. It is useful to link variable with the PLC and monitor it.



## 9.2.4 Troubleshooting

### **Error Frames**

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



#### **Error Frames**



Warning limit exceeded, passive error or bus-off state are indicated first of all at those nodes that have detected the most errors. These nodes are not necessarily the cause for the occurrence of error frames!

If, for instance, one node contributes unusually heavily to the bus traffic (e.g. because it is the only one with analog inputs, the data for which triggers event-driven PDOs at a high rate), then the probability of its telegrams being damaged increases. Its error counter will, correspondingly, be the first to reach a critical level.

## Node ID / Setting the Baud Rate

Care must be taken to ensure that node addresses are not assigned twice: there may only be one sender for each CAN data telegram.

#### Test 1

Check node addresses. If the CAN communication works at least temporarily and all devices support the boot-up message, the address assignment can also be checked by recording the boot-up messages after switching on the devices - but this does not detect any swapping of node addresses.

#### Test 2

Check that the same baud rate has been set everywhere. For special devices, if the bit timing parameters are accessible, do they agree with the CANopen definitions (sampling time, SJW, oscillator).

## **Testing the CAN wiring**

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

#### **Network terminator and signal leads**

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

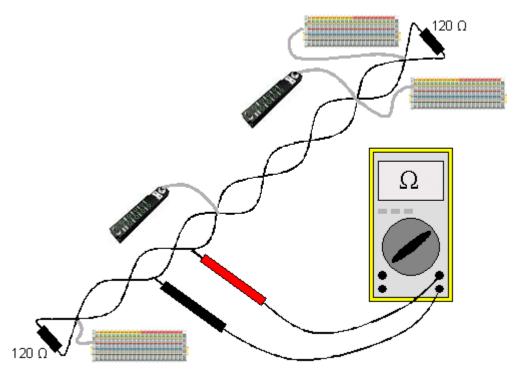


Fig. 65: Wiring diagram for test setup

### Test 3

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

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

## Test 4

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

### Test 5

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

## **Topology**

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

#### Test 6

Measure the lengths of the drop lines and the total bus lengths (do not just make rough estimates!) and compare them with the topology rules for the relevant baud rate.

## Screening and earthing

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



#### Test 7

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

#### Test 8

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

#### **Potential differences**

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

## Test 9

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

#### **Detect and localize faults**

The "low-tech approach" usually works best: disconnect parts of the network and observe when the error disappears.

However, this does not work well for problems such as excessive potential differences, ground loops, EMC or signal distortion, since the reduction in the size of the network often solves the problem without the "missing" piece being the cause. The bus load also changes as the network is reduced in size, which can mean that external interference "hits" CAN telegrams less often.

Diagnosis by means of an oscilloscope usually does not lead to success: CAN signals sometimes look quite confused even in an undisturbed state. It may be possible to trigger on error frames using a storage oscilloscope - this type of diagnosis, however, is only possible for expert technicians.

## **Protocol problems**

In rare cases, protocol problems (e.g. faulty or incomplete CANopen implementation, unfavorable timing at boot up, etc.) can be the cause of faults. In this case, a trace of the bus traffic with subsequent evaluation by CANopen experts is required - the Beckhoff support team can help here.

A free channel of a Beckhoff FC5102 CANopen PCI card is suitable for such a trace - Beckhoff provides the necessary trace software on the Internet. Alternatively, it is of course possible to use a normal commercial CAN analysis tool.

Protocol problems can be avoided if devices that have not been conformance tested are not used. The official CANopen Conformance Test (and the appropriate certificate) can be obtained from the CAN in Automation Association (<a href="https://www.can-cia.org">https://www.can-cia.org</a>).



# 9.3 Diagnosis of the multi-function I/Os

This chapter describes the diagnostic options for multi-function I/O communication. This is important, for example, if the 24 V power supply for the multi-function I/Os fails or the circuit breaker has triggered.

### Status variable

The status variable state can be used for diagnostic purposes. In the normal state, the status variable takes the value 0x = 8 (OP, Operational) and thus indicates that everything is error-free.

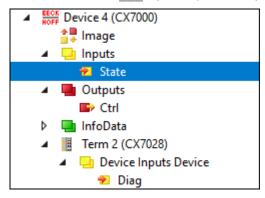


Fig. 66: Multi-function I/O status variable.

The following table shows which values the variables can assume:

| Value  | Meaning                            |
|--------|------------------------------------|
| 0x1    | Slave in 'INIT' state              |
| 0x2    | Slave in 'PREOP' state             |
| 0x3    | Slave in 'BOOT' state              |
| 0x4    | Slave in 'SAFEOP' state            |
| 0x8    | Slave in 'OP' state                |
| 0x001_ | Slave signals error                |
| 0x002_ | Invalid vendorld, productCode read |
| 0x004_ | Initialization error occurred      |
| 0x010_ | Slave not present                  |

If there is a power supply failure, the multi-function I/Os do not automatically go back into data exchange. To do this, the multi-function I/Os must be reset. A function block that can be used to reset the multi-function I/Os is the FB CX70xx ResetOnBoardIO [ > 168] function block.

**Notice**: If outputs are still set in the PLC, the outputs of the multi-function I/Os are immediately reactivated as soon as the multi-function I/Os are reset with the function block.

## Other diagnostic variables

The diagnostic variables <code>Diag</code> and <code>TxPDO</code> State are currently not in use and are reserved for future use. The variable <code>Input cycle counter</code>, on the other hand, increments with each cycle and indicates the number of I/O cycles exchanged with the multi-function I/Os. As soon as the variable is no longer incremented, no more I/O cycles are exchanged with the multi-function I/Os.

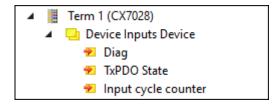


Fig. 67: Further diagnostic variables for multi-function I/Os



| Variable            | Meaning  |
|---------------------|--|
| Diag                | Reserved, currently not used.  |
| TxPDO State         | Reserved, currently not used.  |
| Input cycle counter | Incremented by 1 with each cycle. If this counter stops, then no more I/O cycles are exchanged with the multi-function I/Os. |

# 9.4 Memory usage

The CX7051 has 32 MB of RAM that is used by the firmware (TC/RTOS) and TwinCAT (TwinCAT memory). The TwinCAT memory is further divided into the router memory and the PLC memory. The router memory is used for ADS communication and the PLC memory for the actual PLC program including TcConfiguration, mapping and data.

19.1 MB of TwinCAT memory are available to the CX7051. Because the size of the memory is limited, it is important to control the memory usage and to adapt your PLC project if it is exceeded.

## **Router memory**

On the one hand, you can adjust the size of the router memory in TwinCAT and set a smaller router memory depending on the ADS communication actually used.

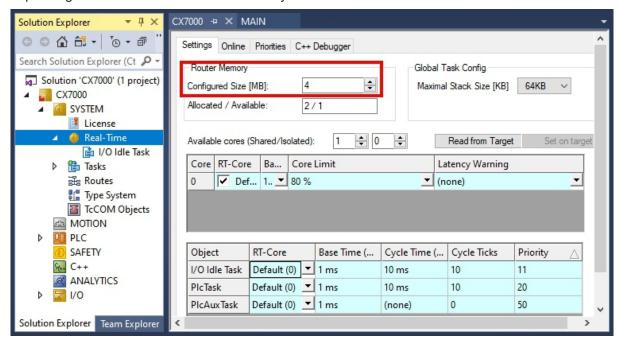


Fig. 68: Settings for router memory in the TwinCAT System Manager.

By default, a value of 32 MB is entered in TwinCAT, which in turn is limited to 9 MB for the CX7051 because of the small RAM in the CX7051. A router memory of 9 MB is usually much too large for a small controller. A router memory of 4 MB is recommended for the CX7051 and can be even smaller if little to no ADS communication is used. However, a router memory of at least 1 MB should be adhered to and should not be any smaller. You can determine how much router memory is used with the function block FB GetRouterStatusInfo or alternatively with the Beckhoff Device Manager.

Note that the router memory is only re-created with a power off/on of the CX7051. A TwinCAT restart is not sufficient. The rule of thumb is: The smaller the router memory for ADS communication is set, the larger the application can be, i.e. the PLC program, TcConfiguration, mapping and data.

### **Determining the memory usage**

With the function block FB\_GetRouterStatusInfo, or alternatively with the Beckhoff Device Manager, it is possible to determine how large the memory requirement of the router memory is.



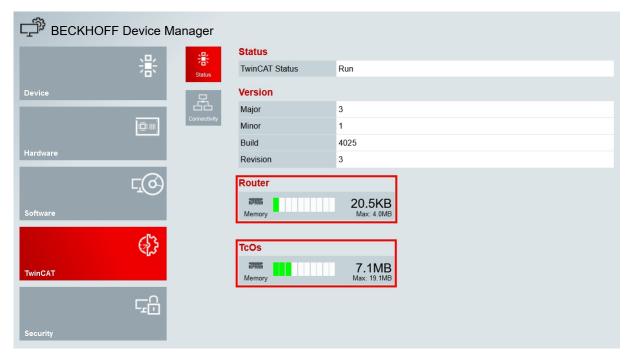


Fig. 69: Utilization of the router and TwinCAT memory.

The **Router** display can be used to determine the memory requirements of the router memory. In this example, 20.5 kB of a maximum of 4 MB are occupied. The **TcOs** display shows the total memory consumption of the TwinCAT memory including router memory and PLC program. In this example, 7.1 MB are occupied in total.

With the help of this display, the size of the PLC program can also be calculated, since the router memory is fixed at 4 MB and is part of the TwinCAT memory. If you subtract the 4 MB from 7.1 MB, therefore, the PLC program occupies 3.1 MB.

## **Memory reserve**

Since in this example the TwinCAT memory occupies 7.1 MB of 19.1 MB, a reserve of 12 MB remains for the PLC program. Note that more memory is needed for a short time for an Online Change in TwinCAT. If you want to use the Online Change function, it is advisable to always have a certain reserve. In the most extreme case, twice the currently consumed PLC program may be required to perform an Online Change. An error message is displayed in TwinCAT if there is not enough memory available for the Online Change.



## 9.5 Real-time and CPU load

For the proper functioning of the CX7051, it is important to keep an eye on CPU load and real-time compliance. Otherwise, the CX7051 will no longer work reliably in the event of an overload. Note that in the event of an overload, the load indicator is also affected and no longer provides current values. For example, a load of 40% can be incorrectly displayed, but the PLCs are no longer working in real time and the system is overloaded. You should therefore gradually approach the load limit with a small controller.

What is meant by real-time in this context? By default, the PLC works in synchronization with the cycle, which means that a task time is always defined and called at a fixed time. The PLC works in synchronization with the cycle if the task time is not exceeded. For example, if you define a task time of 10 ms and the PLC only needs 2 ms for processing, the selected task time is fine and the PLCs work in synchronization with the cycle.

Even if you do not need the real-time, it is recommended to adhere to the real-time, because otherwise negative effects can occur. These could be connection problems or problems with subsystems such as K-bus or EtherCAT. You can perform the following steps to check whether the CX7051 is optimally set or rather overloaded:

- · Observe the exceed counter.
- · Check the CPU load.

#### Observe the exceed counter

The exceed counter is incremented as soon as the PLC no longer works in synchronization with the cycle and the defined task time is exceeded. Ideally, the counter value should be zero.

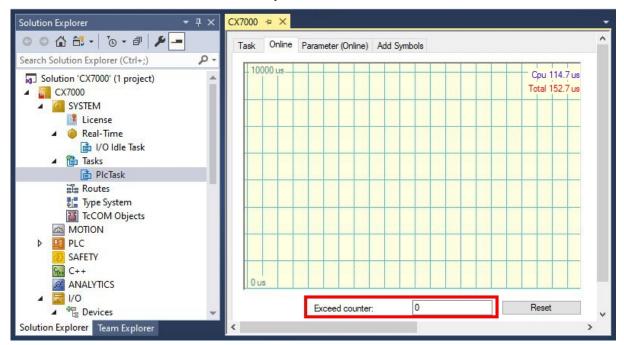


Fig. 70: Display of the exceed counter in TwinCAT.

It is possible for the exceed counter to be incremented at the start of the PLC, for example, because the PLC is called for the first time or certain components are initialized. Observe the exceed counter over a period of several hours. One can only speak of a stable state when the exceed counter is no longer incremented over a longer period of time.

## **Check the CPU load**

In TwinCAT, the CPU load is displayed under Realtime and on the Online tab. Check the value to determine whether you can run additional program code or reduce the task time.



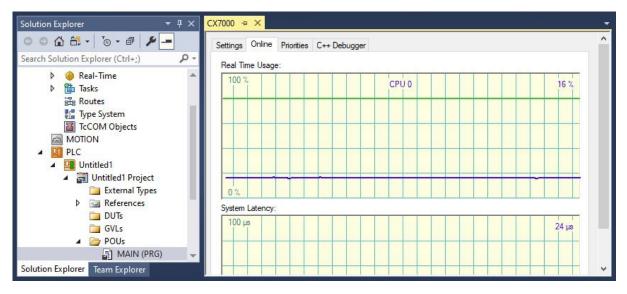


Fig. 71: Display of the CPU load in TwinCAT.

The light green line indicates the preset CPU limit. If the load is ≥ 65%, the CX7051 is already very busy and no more code should be executed or the task time shortened. You should not go to the limit and use the CX7051 to full capacity.

### Measures in the event of overload

If an overload is detected with the help of the steps shown, the load can be reduced by improving the programming or increasing the task time. To find places in the program code with long processing times, the sample in Measuring processing time in the PLC program [ • 161] can be used.

The selected terminal system also has an influence on the real-time. Depending on the number of terminals, the K-bus, for example, can also take several milliseconds and must be taken into account when choosing the task time. It may well be that, with a set task time of 10 ms, the PLC program only needs 5 ms, but the exceed counter still increments. This is due to the fact that the K-bus requires more than 5 ms for processing and the task time of 10 ms including PLC program and K-bus is exceeded. This problem can be solved by reducing the number of terminals or increasing the task time.

By default, the real-time is set to 80%. This is already the maximum value and an increase to 90% is equivalent to an increase to 100%.

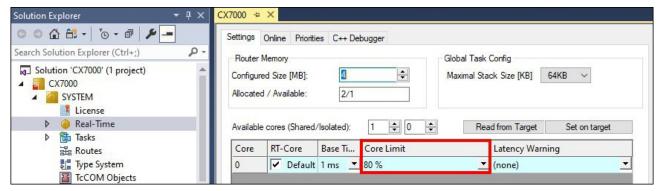


Fig. 72: Setting the real-time load in TwinCAT.

TwinCAT would then consume all the CPU power, and services that the operating system serves would no longer work or would not work adequately. If you increase the real-time load to 90%, you should be aware of the potential consequences for the operating system.



# 10 Technical data

Table 27: Technical data, dimensions and weights.

|                        | CX7051                 |
|------------------------|------------------------|
| Dimensions (W x H x D) | 49 mm x 100 mm x 73 mm |
| Weight                 | 142 g                  |

Table 28: Technical data, general data.

| Technical data               | CX7051   |
|------------------------------|--|
| Processor                    | ARM Cortex™ M7, 480 MHz  |
| Number of cores              | 1  |
| Flash memory                 | 512 MB microSD (optionally 1 GB, 2 GB, 4 GB or 8 GB)   |
| Main memory                  | 32 MB SDR (internal, non-extendable)   |
| Number of inputs             | 8 multifunction inputs (24 V DC)   |
| Number of outputs            | 4 multifunction outputs (24 V DC, 0.5 A, 1-wire technique)   |
| NOVRAM                       | 4 kB   |
| Interfaces                   | 1 x RJ45 10/100 Mbit/s, 1x USB (max 12 Mbit/s, max 100 mA)   |
| Bus interface                | D-sub connector, 9-pin, 1 x CANopen responder (slave)  |
| Data transfer rate           | 10, 20, 50, 100, 125, 250, 500, 800, 1000 kbaud  |
| Diagnostic LED               | 1 x TC Status, 1 x WD LED, 1 x ERR LED   |
| Clock                        | internal, capacitor-buffered real-time clock for time and date (memory > 21 days)  |
| Operating system             | TC/RTOS  |
| Control software             | TwinCAT 3 Runtime (XAR)  |
| Power supply                 | 24 V <sub>DC</sub> (-15 %/+20 %)   |
| Max. power consumption       | < 2 W (max. 12 W with E-bus/K-bus)   |
| TwinCAT 3 functions included | TC1000 TwinCAT 3 ADS, TC1100 TwinCAT 3 I/O, TC1200 TwinCAT 3 PLC, TF4100 TwinCAT 3 Controller Toolbox, TF4110 TwinCAT 3 Temperature Controller, TF6255 TwinCAT 3 Modbus RTU, TF6340 TwinCAT 3 Serial Communication, TF6701 TwinCAT 3 IoT Communication (MQTT), TF6730 TwinCAT 3 IoT Communicator |
| Approvals                    | CE, UL   |

Table 29: Technical data, I/O terminals.

| Technical data                   | CX7051  |
|----------------------------------|---|
| I/O connection                   | via power supply terminal (E-bus or K-bus, automatic recognition)   |
| Power supply for I/O terminals   | max. 1.5 A (installation position any, temp2545 °C) max. 1.3 A (installation position horizontal, temp2555 °C) max. 1 A (installation position any, temp2555 °C) max. 1 A (installation position horizontal, temp2560 °C) |
| Power contacts current load      | max. 10 A   |
| Process data on the K-bus        | max. 512 bytes In and 512 bytes Out   |
| max. number of terminals (K-bus) | 64 (255 with K-bus extension)   |
| E-bus process data               | max. 4 kB In and 4 kB Out   |
| max. number of terminals (E-bus) | up to 65534 terminals.  |

Table 30: Technical data, environmental conditions.

| Technical data             | CX7051        |
|----------------------------|---------------|
| Ambient temperature during | -25° C +60° C |
| operation                  |               |



| Technical data             | CX7051  |
|----------------------------|---|
| Ambient temperature during | -40° C +85° C   |
| storage                    | see notes under: <u>Transport and storage</u> [ <u>12</u> ] |
| Relative humidity          | 95 % no condensation  |
| Vibration resistance       | conforms to EN 60068-2-6                                    |
| Shock resistance           | conforms to EN 60068-2-27                                   |
| EMC immunity               | conforms to EN 61000-6-2                                    |
| EMC emission               | conforms to EN 61000-6-4                                    |
| Protection rating          | IP20  |

Table 31: Technical data, Ethernet interface X001.

| Technical data       | CX7051  |
|----------------------|---|
| Data transfer medium | 4 x 2 twisted pair copper cables category 5 (100 Mbit/s)  |
| Cable length         | 100 m from switch to CX7051   |
| Data transfer rate   | 10/100 Mbit/s   |
| Topology             | star wiring   |
| Protocols            | all non-real-time-capable protocols that are based on TCP or UDP and do not require a real-time extension |

Table 32: Technical data, CANopen interface X003.

| Technical data           | CX7051   |
|--------------------------|--|
| Fieldbus                 | CANopen  |
| Data transfer rate       | 10, 20, 50, 100, 125, 250, 500, 800, 1,000 kbaud       |
| Bus interface            | 1 x D sub-socket, 9-pin                                |
| Extendable process image | Up to 3 virtual slaves in addition                     |
| max. process image       | 4 slaves x (16 Tx PDOs / 16 Rx PDOs (8 bytes per PDO)) |
| Autobaud                 | Yes  |
| Electrical isolation     | Yes  |
| Protocol                 |  |
| CANopen slave            | Yes  |
| CAN (virtual slave)      | 4 (3 virtual CANopen nodes)                            |
| ADS Interface            | yes (only via Ethernet)                                |
| Services                 |  |
| CAN Layer 2              | No   |
| CAN 2.0A                 | acc. to CANopen  |
| CAN 2.0B                 | No   |
| Diagnosis/Status/Alarm   |  |
| TC LED                   | Yes, green/red   |
| BF LED                   | Yes, green/red   |
| DIA LED                  | Yes, green/red   |
| Diagnostic messages      | Yes  |

# 11 Appendix

## 11.1 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 [\( \) 194], 500 [\( \) 199], 600 [\( \) 199], 700 [\( \) 195], 800 [\( \) 195], 900 [\( \) 196], 1000 [\( \) 200], 1100 [\( \) 201], 1200 [\( \) 197], 1300 [\( \) 197], 1400 [\( \) 201], 1500 [\( \) 202], 1600 [\( \) 202], 1700 [\( \) 198], 1800 [\( \) 204], 1900 [\( \) 198]

Hexadecimal: 0x181 [ $\triangleright$  194], 0x1C1 [ $\triangleright$  199], 0x201 [ $\triangleright$  194], 0x301 [ $\triangleright$  195], 0x401 [ $\triangleright$  197], 0x501 [ $\triangleright$  197], 0x601 [ $\triangleright$  204], 0x701 [ $\triangleright$  204]

The identifier distribution via object 0x5500 follows this pattern:

| Object                               | Resulting COB ID (dec) | Resulting COB ID (hex) |
|--------------------------------------|------------------------|------------------------|
| Emergency [▶ 194]                    | 129 to 191 [255]       | 0x81 to 0xBF [0xFF]    |
| <u>TxPDO1 [▶ 194]</u>                | 385 to 447 [511]       | 0x181 to 0x1BF [0x1FF] |
| RxPDO1 [▶ 194]                       | 513 to 575 [639]       | 0x201 to 0x23F [0x27F] |
| <u>TxPDO2 [▶ 195]</u>                | 641 to 676 [767]       | 0x281 to 0x2BF [0x2FF] |
| RxPDO2 [ • 195]                      | 769 to 831 [895]       | 0x301 to 0x33F [0x37F] |
| TxDPO3 [▶ 196]                       | 897 to 959 [1023]      | 0x381 to 0x3BF [0x3FF] |
| RxPDO3 [▶ 197]                       | 1025 to 1087 [1151]    | 0x401 to 0x43F [0x47F] |
| <u>TxPDO4 [▶ 197]</u>                | 1153 to 1215 [1279]    | 0x481 to 0x4BF [0x4FF] |
| RxPDO4 [▶ 197]                       | 1281 to 1343 [1407]    | 0x501 to 0x53F [0x57F] |
| TxPDO5 [▶ 198]                       | 1665 to 1727           | 0x681 to 0x6BF         |
| RxPDO5 [▶ 198]                       | 1921 to 1983           | 0x781 to 0x7BF         |
| <u>TxPDO6 [▶ 199]</u>                | 449 to 511             | 0x1C1 to 0x1FF         |
| RxPDO6 [▶ 199]                       | 577 to 639             | 0x241 to 0x27F         |
| <u>TxDPO7 [▶ 199]</u>                | 705 to 767             | 0x2C1 to 0x2FF         |
| RxPDO7 [ > 200]                      | 833 to 895             | 0x341 to 0x37F         |
| <u>TxPDO8 [▶ 200]</u>                | 961 to 1023            | 0x3C1 to 0x3FF         |
| RxPDO8 [ > 201]                      | 1089 to 1151           | 0x441 to 0x47F         |
| <u>TxPDO9 [▶ 201]</u>                | 1217 to 1279           | 0x4C1 to 0x4FF         |
| RxPDO9 [ > 201]                      | 1345 to 1407           | 0x541 to 0x57F         |
| TxDPO10 [▶ 202]                      | 1473 to 1535           | 0x5C1 to 0x5FF         |
| RxPDO10 [ 202]                       | 1601 to 1663           | 0x641 to 0x67F         |
| TxPDO11 [ > 203]                     | 1729 to 1791           | 0x6C1 to 0x6FF         |
| RxPDO11 [ 203]                       | 1857 to 1919           | 0x741 to 0x77F         |
| SDO (Tx) [▶ 203]                     | 1409 to 1471 [1535]    | 0x581 to 0x5BF [0x5FF] |
| SDO (Rx) [▶ 204]                     | 1537 to 1599 [1663]    | 0x601 to 0x63F [0x67F] |
| Guarding / Heartbeat/ Bootup [▶ 204] | 1793 to 1855 [1919]    | 0x701 to 0x73F [0x77F] |



**Identifier List** 

Identifiers marked with  $^{\star}$  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,<br>Nd.43 |
| 514 | 0x202 | RxPDO1, DO, Nd.2 | 535 | 0x217 | RxPDO1, DO, Nd.23 | 556 | 0x22C | RxPDO1, DO,<br>Nd.44 |
| 515 | 0x203 | RxPDO1, DO, Nd.3 | 536 | 0x218 | RxPDO1, DO, Nd.24 | 557 | 0x22D | RxPDO1, DO,<br>Nd.45 |
| 516 | 0x204 | RxPDO1, DO, Nd.4 | 537 | 0x219 | RxPDO1, DO, Nd.25 | 558 | 0x22E | RxPDO1, DO,<br>Nd.46 |

194 Version: 1.0 CX7051



| dec | hex   | Telegram type     | dec | hex   | Telegram type     | dec | hex   | Telegram type        |
|-----|-------|-------------------|-----|-------|-------------------|-----|-------|----------------------|
| 517 | 0x205 | RxPDO1, DO, Nd.5  | 538 | 0x21A | RxPDO1, DO, Nd.26 | 559 | 0x22F | RxPDO1, DO,<br>Nd.47 |
| 518 | 0x206 | RxPDO1, DO, Nd.6  | 539 | 0x21B | RxPDO1, DO, Nd.27 | 560 | 0x230 | RxPDO1, DO,<br>Nd.48 |
| 519 | 0x207 | RxPDO1, DO, Nd.7  | 540 | 0x21C | RxPDO1, DO, Nd.28 | 561 | 0x231 | RxPDO1, DO,<br>Nd.49 |
| 520 | 0x208 | RxPDO1, DO, Nd.8  | 541 | 0x21D | RxPDO1, DO, Nd.29 | 562 | 0x232 | RxPDO1, DO,<br>Nd.50 |
| 521 | 0x209 | RxPDO1, DO, Nd.9  | 542 | 0x21E | RxPDO1, DO, Nd.30 | 563 | 0x233 | RxPDO1, DO,<br>Nd.51 |
| 522 | 0x20A | RxPDO1, DO, Nd.10 | 543 | 0x21F | RxPDO1, DO, Nd.31 | 564 | 0x234 | RxPDO1, DO,<br>Nd.52 |
| 523 | 0x20B | RxPDO1, DO, Nd.11 | 544 | 0x220 | RxPDO1, DO, Nd.32 | 565 | 0x235 | RxPDO1, DO,<br>Nd.53 |
| 524 | 0x20C | RxPDO1, DO, Nd.12 | 545 | 0x221 | RxPDO1, DO, Nd.33 | 566 | 0x236 | RxPDO1, DO,<br>Nd.54 |
| 525 | 0x20D | RxPDO1, DO, Nd.13 | 546 | 0x222 | RxPDO1, DO, Nd.34 | 567 | 0x237 | RxPDO1, DO,<br>Nd.55 |
| 526 | 0x20E | RxPDO1, DO, Nd.14 | 547 | 0x223 | RxPDO1, DO, Nd.35 | 568 | 0x238 | RxPDO1, DO,<br>Nd.56 |
| 527 | 0x20F | RxPDO1, DO, Nd.15 | 548 | 0x224 | RxPDO1, DO, Nd.36 | 569 | 0x239 | RxPDO1, DO,<br>Nd.57 |
| 528 | 0x210 | RxPDO1, DO, Nd.16 | 549 | 0x225 | RxPDO1, DO, Nd.37 | 570 | 0x23A | RxPDO1, DO,<br>Nd.58 |
| 529 | 0x211 | RxPDO1, DO, Nd.17 | 550 | 0x226 | RxPDO1, DO, Nd.38 | 571 | 0x23B | RxPDO1, DO,<br>Nd.59 |
| 530 | 0x212 | RxPDO1, DO, Nd.18 | 551 | 0x227 | RxPDO1, DO, Nd.39 | 572 | 0x23C | RxPDO1, DO,<br>Nd.60 |
| 531 | 0x213 | RxPDO1, DO, Nd.19 | 552 | 0x228 | RxPDO1, DO, Nd.40 | 573 | 0x23D | RxPDO1, DO,<br>Nd.61 |
| 532 | 0x214 | RxPDO1, DO, Nd.20 | 553 | 0x229 | RxPDO1, DO, Nd.41 | 574 | 0x23E | RxPDO1, DO,<br>Nd.62 |
| 533 | 0x215 | RxPDO1, DO, Nd.21 | 554 | 0x22A | RxPDO1, DO, Nd.42 | 575 | 0x23F | RxPDO1, DO,<br>Nd.63 |

| dec | hex   | Telegram type     | dec | hex   | Telegram type     | dec | hex   | Telegram type     |
|-----|-------|-------------------|-----|-------|-------------------|-----|-------|-------------------|
| 641 | 0x281 | TxPDO2, Al, Nd.1  | 662 | 0x296 | TxPDO2, Al, Nd.22 | 683 | 0x2AB | TxPDO2, AI, Nd.43 |
| 642 | 0x282 | TxPDO2, Al, Nd.2  | 663 | 0x297 | TxPDO2, Al, Nd.23 | 684 | 0x2AC | TxPDO2, AI, Nd.44 |
| 643 | 0x283 | TxPDO2, Al, Nd.3  | 664 | 0x298 | TxPDO2, Al, Nd.24 | 685 | 0x2AD | TxPDO2, AI, Nd.45 |
| 644 | 0x284 | TxPDO2, Al, Nd.4  | 665 | 0x299 | TxPDO2, Al, 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, Al, Nd.6  | 667 | 0x29B | TxPDO2, Al, Nd.27 | 688 | 0x2B0 | TxPDO2, AI, Nd.48 |
| 647 | 0x287 | TxPDO2, Al, Nd.7  | 668 | 0x29C | TxPDO2, Al, Nd.28 | 689 | 0x2B1 | TxPDO2, AI, Nd.49 |
| 648 | 0x288 | TxPDO2, Al, Nd.8  | 669 | 0x29D | TxPDO2, Al, Nd.29 | 690 | 0x2B2 | TxPDO2, AI, Nd.50 |
| 649 | 0x289 | TxPDO2, Al, Nd.9  | 670 | 0x29E | TxPDO2, Al, Nd.30 | 691 | 0x2B3 | TxPDO2, AI, Nd.51 |
| 650 | 0x28A | TxPDO2, Al, Nd.10 | 671 | 0x29F | TxPDO2, Al, Nd.31 | 692 | 0x2B4 | TxPDO2, AI, Nd.52 |
| 651 | 0x28B | TxPDO2, Al, Nd.11 | 672 | 0x2A0 | TxPDO2, Al, Nd.32 | 693 | 0x2B5 | TxPDO2, AI, Nd.53 |
| 652 | 0x28C | TxPDO2, Al, Nd.12 | 673 | 0x2A1 | TxPDO2, Al, Nd.33 | 694 | 0x2B6 | TxPDO2, AI, Nd.54 |
| 653 | 0x28D | TxPDO2, Al, Nd.13 | 674 | 0x2A2 | TxPDO2, Al, Nd.34 | 695 | 0x2B7 | TxPDO2, AI, Nd.55 |
| 654 | 0x28E | TxPDO2, Al, Nd.14 | 675 | 0x2A3 | TxPDO2, Al, Nd.35 | 696 | 0x2B8 | TxPDO2, AI, Nd.56 |
| 655 | 0x28F | TxPDO2, Al, Nd.15 | 676 | 0x2A4 | TxPDO2, Al, Nd.36 | 697 | 0x2B9 | TxPDO2, AI, Nd.57 |
| 656 | 0x290 | TxPDO2, Al, Nd.16 | 677 | 0x2A5 | TxPDO2, Al, Nd.37 | 698 | 0x2BA | TxPDO2, AI, Nd.58 |
| 657 | 0x291 | TxPDO2, Al, Nd.17 | 678 | 0x2A6 | TxPDO2, Al, Nd.38 | 699 | 0x2BB | TxPDO2, AI, Nd.59 |
| 658 | 0x292 | TxPDO2, Al, Nd.18 | 679 | 0x2A7 | TxPDO2, Al, Nd.39 | 700 | 0x2BC | TxPDO2, AI, Nd.60 |
| 659 | 0x293 | TxPDO2, Al, Nd.19 | 680 | 0x2A8 | TxPDO2, AI, Nd.40 | 701 | 0x2BD | TxPDO2, AI, Nd.61 |
| 660 | 0x294 | TxPDO2, Al, Nd.20 | 681 | 0x2A9 | TxPDO2, Al, Nd.41 | 702 | 0x2BE | TxPDO2, AI, Nd.62 |
| 661 | 0x295 | TxPDO2, Al, Nd.21 | 682 | 0x2AA | TxPDO2, Al, 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 |     | RxPDO2, AO,   |
|     |       |                  |     |       |                   |     |     | Nd.43         |



| dec | hex   | Telegram type     | dec | hex   | Telegram type     | dec | hex   | Telegram type        |
|-----|-------|-------------------|-----|-------|-------------------|-----|-------|----------------------|
| 770 | 0x302 | RxPDO2, AO, Nd.2  | 791 | 0x317 | RxPDO2, AO, Nd.23 | 812 | 0x32C | RxPDO2, AO,<br>Nd.44 |
| 771 | 0x303 | RxPDO2, AO, Nd.3  | 792 | 0x318 | RxPDO2, AO, Nd.24 | 813 | 0x32D | RxPDO2, AO,<br>Nd.45 |
| 772 | 0x304 | RxPDO2, AO, Nd.4  | 793 | 0x319 | RxPDO2, AO, Nd.25 | 814 | 0x32E | RxPDO2, AO,<br>Nd.46 |
| 773 | 0x305 | RxPDO2, AO, Nd.5  | 794 | 0x31A | RxPDO2, AO, Nd.26 | 815 | 0x32F | RxPDO2, AO,<br>Nd.47 |
| 774 | 0x306 | RxPDO2, AO, Nd.6  | 795 | 0x31B | RxPDO2, AO, Nd.27 | 816 | 0x330 | RxPDO2, AO,<br>Nd.48 |
| 775 | 0x307 | RxPDO2, AO, Nd.7  | 796 | 0x31C | RxPDO2, AO, Nd.28 | 817 | 0x331 | RxPDO2, AO,<br>Nd.49 |
| 776 | 0x308 | RxPDO2, AO, Nd.8  | 797 | 0x31D | RxPDO2, AO, Nd.29 | 818 | 0x332 | RxPDO2, AO,<br>Nd.50 |
| 777 | 0x309 | RxPDO2, AO, Nd.9  | 798 | 0x31E | RxPDO2, AO, Nd.30 | 819 | 0x333 | RxPDO2, AO,<br>Nd.51 |
| 778 | 0x30A | RxPDO2, AO, Nd.10 | 799 | 0x31F | RxPDO2, AO, Nd.31 | 820 | 0x334 | RxPDO2, AO,<br>Nd.52 |
| 779 | 0x30B | RxPDO2, AO, Nd.11 | 800 | 0x320 | RxPDO2, AO, Nd.32 | 821 | 0x335 | RxPDO2, AO,<br>Nd.53 |
| 780 | 0x30C | RxPDO2, AO, Nd.12 | 801 | 0x321 | RxPDO2, AO, Nd.33 | 822 | 0x336 | RxPDO2, AO,<br>Nd.54 |
| 781 | 0x30D | RxPDO2, AO, Nd.13 | 802 | 0x322 | RxPDO2, AO, Nd.34 | 823 | 0x337 | RxPDO2, AO,<br>Nd.55 |
| 782 | 0x30E | RxPDO2, AO, Nd.14 | 803 | 0x323 | RxPDO2, AO, Nd.35 | 824 | 0x338 | RxPDO2, AO,<br>Nd.56 |
| 783 | 0x30F | RxPDO2, AO, Nd.15 | 804 | 0x324 | RxPDO2, AO, Nd.36 | 825 | 0x339 | RxPDO2, AO,<br>Nd.57 |
| 784 | 0x310 | RxPDO2, AO, Nd.16 | 805 | 0x325 | RxPDO2, AO, Nd.37 | 826 | 0x33A | RxPDO2, AO,<br>Nd.58 |
| 785 | 0x311 | RxPDO2, AO, Nd.17 | 806 | 0x326 | RxPDO2, AO, Nd.38 | 827 | 0x33B | RxPDO2, AO,<br>Nd.59 |
| 786 | 0x312 | RxPDO2, AO, Nd.18 | 807 | 0x327 | RxPDO2, AO, Nd.39 | 828 | 0x33C | RxPDO2, AO,<br>Nd.60 |
| 787 | 0x313 | RxPDO2, AO, Nd.19 | 808 | 0x328 | RxPDO2, AO, Nd.40 | 829 | 0x33D | RxPDO2, AO,<br>Nd.61 |
| 788 | 0x314 | RxPDO2, AO, Nd.20 | 809 | 0x329 | RxPDO2, AO, Nd.41 | 830 | 0x33E | RxPDO2, AO,<br>Nd.62 |
| 789 | 0x315 | RxPDO2, AO, Nd.21 | 810 | 0x32A | RxPDO2, AO, Nd.42 | 831 | 0x33F | RxPDO2, AO,<br>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 |



| dec  | hex   | Telegram type  | dec  | hex   | Telegram type  | dec  | hex   | Telegram type  |
|------|-------|----------------|------|-------|----------------|------|-------|----------------|
| 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 |



| dec | hex   | Telegram type  | dec | hex   | Telegram type  | dec | hex   | Telegram type  |
|-----|-------|----------------|-----|-------|----------------|-----|-------|----------------|
| 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 |



| dec  | hex   | Telegram type  | dec  | hex   | Telegram type  | dec  | hex   | Telegram type  |
|------|-------|----------------|------|-------|----------------|------|-------|----------------|
| 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  |



| dec  | hex   | Telegram type | dec  | hex   | Telegram type | dec  | hex   | Telegram type |
|------|-------|---------------|------|-------|---------------|------|-------|---------------|
| 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 |



# 11.2 Third-Party components

This device contains Beckhoff software and third-party software. Please refer to the license file on the storage medium.

## 11.3 Accessories

Table 33: microSD cards.

| Order number | Description         |
|--------------|---------------------|
| CX1900-0122  | 512 MB microSD card |
| CX1900-0132  | 16 GB microSD card  |

Table 34: Further spare parts.

| Order number | Description                                   |
|--------------|---|
| ZB8701       | Slotted screwdriver 2.0 x 40 mm, HD terminals |



## 11.4 Certifications

## **FCC Approvals for the United States of America**

## FCC: Federal Communications Commission Radio Frequency Interference Statement

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

## **FCC Approval for Canada**

### **FCC: Canadian Notice**

This equipment does not exceed the Class A limits for radiated emissions as described in the Radio Interference Regulations of the Canadian Department of Communications.



# List of tables

| Table 1  | Dimensions and weight.   | 12  |
|----------|--|-----|
| Table 2  | Legend for the configuration of the basic CPU module   | 14  |
| Table 3  | Information on the name plate.   | 15  |
| Table 4  | Ethernet interface X001, pin assignment.   | 16  |
| Table 5  | Maximum E-bus/K-bus current depending on the selected installation position and the ambient temperature. | 82  |
| Table 6  | Legend for the connection example.   | 86  |
| Table 7  | Required wire cross-sections and strip lengths   | 87  |
| Table 8  | Technical data, multi-function I/Os as digital inputs.   | 97  |
| Table 9  | Technical data, multi-function I/Os as digital outputs   | 98  |
| Table 10 | Technical data, multi-function I/Os in counter mode.   | 101 |
| Table 11 | Technical data, multi-function I/Os in encoder mode  | 107 |
| Table 12 | Technical data, multi-function I/Os in analog mode.  | 111 |
| Table 13 | Technical data, multi-function I/Os in PWM mode.   | 112 |
| Table 14 | PWM output (duty cycle), representation of the PWM signal in the delivery state                          | 114 |
| Table 15 | PWM period (PWM clock frequency), representation of the PWM signal in the delivery state                 | 114 |
| Table 16 | Access data for the Beckhoff Device Manager on delivery  | 116 |
| Table 17 | Structure of the 11 byte CAN data  | 155 |
| Table 18 | TC LED, order and meaning.   | 174 |
| Table 19 | TC LED, error description and remedy.  | 174 |
| Table 20 | Diagnostic LEDs in K-Bus mode.   | 175 |
| Table 21 | K-bus ERR LED, fault indication sequence through the LED.  | 175 |
| Table 22 | K-BUS ERR LED, fault description and troubleshooting.  | 176 |
| Table 23 | Description of the State variable values.  | 177 |
| Table 24 | Diagnostic LEDs in K-Bus mode.   | 178 |
| Table 25 | Reading the emergency telegrams with the ADSREAD function block.   | 181 |
| Table 26 | Description of the array   | 181 |
| Table 27 | Technical data, dimensions and weights.  | 191 |
| Table 28 | Technical data, general data   | 191 |
| Table 29 | Technical data, I/O terminals.   | 191 |
| Table 30 | Technical data, environmental conditions.  | 191 |
| Table 31 | Technical data, Ethernet interface X001.   | 192 |
| Table 32 | Technical data, CANopen interface X003.  | 192 |
| Table 33 | microSD cards.   | 205 |
| Table 24 | Further energ parts  | 205 |

Version: 1.0



# List of figures

| Fig. 1   | Sample configuration of a CX7051 Embedded PC.  | . 14 |
|----------|--|------|
| Fig. 2   | Name plate example.  | . 15 |
| Fig. 3   | Ethernet interface X001.   | . 16 |
| Fig. 4   | CANopen interface X003.  | . 18 |
| Fig. 5   | CANopen Device Model   | . 21 |
| Fig. 6   | CANopen bootup state diagram   | . 23 |
| Fig. 7   | Schematic diagram: "Guarding procedure"  | . 25 |
| Fig. 8   | Schematic diagram: "Heartbeat procedure"   | . 26 |
| Fig. 9   | Default identifier allocation: Master/Slave  | . 28 |
| Fig. 10  | PDO linking: Peer to Peer  | . 28 |
| Fig. 11  | Diagram: CAN process data transmission   | . 29 |
| Fig. 12  | Diagram: CAN "SYNC" telegram   | . 30 |
| Fig. 13  | Timing diagram: "Inhibit time"   | . 31 |
| Fig. 14  | Time representation of the event timer   | . 32 |
| Fig. 15  | Mapping representation   | . 32 |
| Fig. 16  | SDO protocol: access to the object directory   | . 35 |
| Fig. 17  | CX70xx Embedded PC, dimensions.  | . 81 |
| Fig. 18  | CX70xx Embedded PC, permissible installation position  | . 82 |
| Fig. 19  | Identifying a passive EtherCAT Terminal in TwinCAT.  |      |
| Fig. 20  | Passive EtherCAT Terminals, permissible installation   | . 85 |
| Fig. 21  | Connections for system voltage (Us) and power contacts (Up)  |      |
| Fig. 22  | Connection example with a CX7000.  |      |
| Fig. 23  | Connection example for areas with special UL requirements  |      |
| Fig. 24  | CANopen interface X003.  |      |
| Fig. 25  | CX7028 interface, slot and module configuration under TwinCAT  |      |
| Fig. 26  | Supported modules when using slot 1.   |      |
| Fig. 27  | Supported modules when using slot 2.   |      |
| Fig. 28  | Supported modules when using slot 3.   |      |
| Fig. 29  | Supported modules when using slot 4.   |      |
| Fig. 30  | Configurable digital inputs  |      |
| Fig. 31  | Configurable digital outputs   |      |
| Fig. 32  | Configurable inputs and outputs in counter mode  |      |
| Fig. 33  | Configurable inputs and outputs in incremental encoder mode.   |      |
| Fig. 34  | Configurable analog inputs   |      |
| Fig. 35  | Configurable inputs and outputs in PWM signal mode   |      |
| Fig. 36  | Controller behavior with and without NOVRAM.   |      |
| Fig. 37  | Changing the password in the Beckhoff Device Manager.  |      |
| Fig. 38  | CANopen master and CANopen slave in the TwinCAT tree view with tabs.   |      |
| Fig. 39  | CANopen slave in the TwinCAT tree view with associated tabs  |      |
| Fig. 40  | CAN Node tab of a CANopen slave in TwinCAT.  |      |
| Fig. 41  | SDO tab of a CANopen slave in TwinCAT  |      |
| Fig. 42  | PDO tab of a CANopen slave in TwinCAT  |      |
| Fig. 43  | Enabling of an ADS port for a CANopen slave.   |      |
| Fig. 44  | Content of the MDP module with IP and MAC address.   |      |
| . 19. 77 | Content of the Media Integral William Content of Content of the Media Integral Content of the Me | .00  |



| Fig. 45 | Virtual Ethernet communication via ADS, TCP or UDP.  | 156 |
|---------|--|-----|
| Fig. 46 | CoE access to multi-function I/Os, input variables "netId" and "port" under TwinCAT                | 158 |
| Fig. 47 | CoE communication, listing of CoE objects with matching index number                               | 158 |
| Fig. 48 | K-bus interface of a CX7051 in the TwinCAT System Manager.   | 159 |
| Fig. 49 | E-bus interface of a CX7051 in the TwinCAT System Manager.   | 160 |
| Fig. 50 | Measurement at a task time of 250 µs.  | 165 |
| Fig. 51 | Measurement at a task time of 500 μs.  | 165 |
| Fig. 52 | Measurement at a task time of 1 ms.  | 165 |
| Fig. 53 | CX7051 CPU and PLC   | 166 |
| Fig. 54 | CPU of the CX7028 interface.   | 166 |
| Fig. 55 | Default calling of a PLC task  | 170 |
| Fig. 56 | Calling a PLC task with the attribute tcCallAfterOutputUpdate.                                     | 170 |
| Fig. 57 | Pulse of a digital output without load   | 171 |
| Fig. 58 | Shortened pulse of a digital output with load.   | 171 |
| Fig. 59 | Inverted representation of a digital output  | 172 |
| Fig. 60 | Determination of different running times in the PLC program.                                       | 173 |
| Fig. 61 | Status variable for error handling and diagnostics under TwinCAT                                   | 177 |
| Fig. 62 | Diagnosis of the CANopen communication with the variables NodeState, DiagFlag and EmergencyCounter | 180 |
| Fig. 63 | Diagnostic variable SendCounter of a CANopen slave   | 182 |
| Fig. 64 | Diagnostic variable ReceiveCounter of a CANopen slave  | 182 |
| Fig. 65 | Wiring diagram for test setup  | 184 |
| Fig. 66 | Multi-function I/O status variable   | 186 |
| Fig. 67 | Further diagnostic variables for multi-function I/Os   | 186 |
| Fig. 68 | Settings for router memory in the TwinCAT System Manager   | 187 |
| Fig. 69 | Utilization of the router and TwinCAT memory.  | 188 |
| Fig. 70 | Display of the exceed counter in TwinCAT.  | 189 |
| Fig. 71 | Display of the CPU load in TwinCAT.  | 190 |
| Fig. 72 | Setting the real-time load in TwinCAT  | 190 |

Version: 1.0

More Information: www.beckhoff.com/CX7051

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

