CAN-CBX-DIO8/2

CANopen® Module
with 8 Digital I/O Ports

Hardware Manual

For Product C.3010.04

esd electronics gmbh
Vahrenwalder Str. 207 • 30165 Hannover • Germany
http://www.esd.eu
Phone: +49 (0) 511 3 72 98-0 • Fax: +49 (0) 511 3 72 98-68

Notes

The information in this document has been checked carefully and is considered to be entirely reliable. esd electronics makes no warranty of any kind regarding the material in this document and assumes no responsibility for any errors that may appear in this document. In particular, the descriptions and technical data specified in this document may not be constituted to be guaranteed product features in any legal sense.

esd electronics reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance, or design.

esd electronics gmbh
Vahrenwalder Str. 207
30165 Hannover
Germany

Tel.: +49-511-37298-0
Fax: +49-511-37298-68
E-Mail: info@esd.eu
Internet: www.esd.eu

This manual contains important information and instructions on safe and efficient handling of the CAN-CBX-DIO8/2. Carefully read this manual before commencing any work and follow the instructions.
The manual is a product component, please retain it for future use.

The software used in the CAN-CBX-DIO8/2 is subject to the license terms of the respective authors or rights holders. CAN-CBX-DIO8/2 may only be used in accordance with these license terms!

Links
esd electronics gmbh assumes no liability or guarantee for the content of Internet pages to which this document refers directly or indirectly. Visitors follow links to websites at their own risk and use them in accordance with the applicable terms of use of the respective websites.

Trademark Notices
CANopen® and CiA® are registered EU trademarks of CAN in Automation e.V.
CAN FD® is a registered trademark of the Robert Bosch GmbH.
FreeRTOS™ and FreeRTOS.org™ are trademarks of Amazon Web Services, Inc.
All other trademarks, product names, company names or company logos used in this manual are reserved by their respective owners.

Document Information

Document file:	I:\Texte\Doku\MANUALS\CAN\CBX\CAN-CBX-DIO8-2\CAN-CBX-DIO8-2_Manual_en_40.docx
Date of print:	2025-05-02
Document-type number:	DOC0800

Document History

The changes in the document listed below affect changes in the hardware as well as changes in the description of the facts, only.

Rev.	Chapter	Changes versus previous version	Date
1.0	-	First English version of the CAN-CBX-DIO8/2 manual	2025-05-02

Technical details are subject to change without further notice.

Classification of Warning Messages and Safety Instructions

This manual contains noticeable descriptions, warning messages and safety instructions, which you must follow to avoid personal injuries or death and property damage.

This is the safety alert symbol.
It is used to alert you to potential personal injury hazards. Obey all safety messages and instructions that follow this symbol to avoid possible injury or death.

DANGER, WARNING, CAUTION

Depending on the hazard level the signal words DANGER, WARNING or CAUTION are used to highlight safety instructions and warning messages. These messages may also include a warning relating to property damage.

DANGER
Danger statements indicate a hazardous situation which, if not avoided, will result in death or serious injury.

WARNING.
Warning statements indicate a hazardous situation that, if not avoided, could result in death or serious injury.

CAUTION
Caution statements indicate a hazardous situation that, if not avoided, could result in minor or moderate injury.

NOTICE

Notice statements are used to notify people on hazards that could result in things other than personal injury, like property damage.

NOTICE
This NOTICE statement indicates that the device contains components sensitive to electrostatic discharge.

NOTICE
This NOTICE statement contains the general mandatory sign and gives information that must be heeded and complied with for a safe use.

INFORMATION

INFORMATION
Notes to point out something important or useful.

Safety Instructions

- When working with the CAN-CBX-DIO8/2 follow the instructions below and read the manual carefully to protect yourself from injury and the CAN-CBX-DIO8/2 from damage.
- The assembly is classified as open equipment and must therefore be installed in a control cabinet that is designed for the specific environmental conditions. The control cabinet should be made of metal to improve the electromagnetic immunity of the device. It should be equipped with a key locking mechanism to prevent any unauthorized access.
- Do not use damaged or defective cables to connect the CAN-CBX-DIO8/2 and follow the CAN wiring hints in chapter: "Correct Wiring of Electrically Isolated CAN Networks".
- In case of damages to the device, which might affect safety, appropriate and immediate measures must be taken, that exclude an endangerment of persons and domestic animals and property.
- The galvanic isolation of the CAN-CBX-DIO8/2 has only functional tasks and is not a protection against hazardous electrical voltage.
- The CAN-CBX-DIO8/2 is a device of protection class III according to DIN EN IEC 61140 and may only be operated on supply circuits that offer sufficient protection against dangerous voltages.
- External circuits connected to the CAN-CBX-DIO8/2 must be sufficiently protected against dangerous voltage.
- The user is responsible for compliance with the applicable national safety regulations.
- Do not open the housing of the CAN-CBX-DIO8/2 .
- The CAN-CBX-DIO8/2 must be securely installed before commissioning.
- The permitted operating position is specified as shown (Figure: X). Other operating positions are not allowed.
- Never let liquids get inside CAN-CBX-DIO8/2. Otherwise, electric shocks or short circuits may result.
- Protect the CAN-CBX-DIO8/2 from dust, moisture, and steam.
- Protect the CAN-CBX-DIO8/2 from shocks and vibrations.
- The CAN-CBX-DIO8/2 may become warm during normal use. Always allow adequate ventilation around the CAN-CBX-DIO8/2 and use care when handling
- Do not operate the CAN-CBX-DIO8/2 adjacent to heat sources and do not expose it to unnecessary thermal radiation. Ensure an ambient temperature as specified in the technical data.

DANGER

Hazardous Voltage - Risk of electric shock due to unintentional contact with uninsulated live parts with high voltages inside of the system into which the CAN-CBX-DIO8/2 is to be integrated.

→ All current circuits which are connected to the device must be sufficiently protected against hazardous voltage, before you start with the installation.

→ Before you switch on the supply voltage, check that all plug connectors are correctly seated.

→ All current circuits which are connected to the device must be sufficiently protected against hazardous voltage, before you start with the installation.

→ Ensure the absence of voltage before starting any electrical work.

Qualified Personnel

This documentation is directed exclusively towards personnel qualified in control and automation engineering. The installation and commissioning of the product may only be carried out by qualified personnel, which is authorized to put devices, systems, and electric circuits into operation according to the applicable national standards of safety engineering.

Conformity

The CAN-CBX-DIO8/2 is an industrial product and meets the demands of the EU regulations and EMC standards printed in the conformity declaration at the end of this manual.

Warning: In a residential, commercial, or light industrial environment the CAN-CBX-DIO8/2 may cause radio interferences in which case the user may be required to take adequate measures.

Intended Use

The intended use of the CAN-CBX-DIO8/2 is the operation as a CANopen module with eight digital inputs/outputs.
The guarantee given by esd does not cover damages which result from improper use, usage not in accordance with regulations or disregard of safety instructions and warnings.

- The CAN-CBX-DIO8/2 is intended for installation in a control cabinet.
- The operation of the CAN-CBX-DIO8/2 in hazardous areas, or areas exposed to potentially explosive materials is not permitted.
- The operation of the CAN-CBX-DIO8/2 for medical purposes is prohibited.

Service Note

The CAN-CBX-DIO8/2 does not contain any parts that require maintenance by the user. The CAN-CBX-DIO8/2 does not require any manual configuration of the hardware. Unauthorized intervention in the device voids warranty claims

Disposal

Products marked with a crossed-out dustbin must not be disposed of with household waste. Devices which have become defective in the long run must be disposed in an appropriate way or must be returned to the manufacturer for proper disposal. Please, contribute to environmental protection.

Typographical Conventions

Throughout this manual the following typographical conventions are used to distinguish technical terms.

Convention	Example
File and path names	**`/dev/null`** or **`<stdio.h>`**
Function names	***open()***
Programming constants	`NULL`
Programming data types	`uint32_t`
Variable names	*Count*

Number Representation

All numbers in this document are base 10 unless designated otherwise. Hexadecimal numbers have a prefix of 0x. For example, 42 is represented as 0x2A in hexadecimal.

Table of Contents

List of Tables

List of Figures

1 Overview

1.1 About this Manual

This manual describes the hardware and the software of the CAN-CBX-DIO8/2 module.

1.2 Description of CAN-CBX-DIO8/2

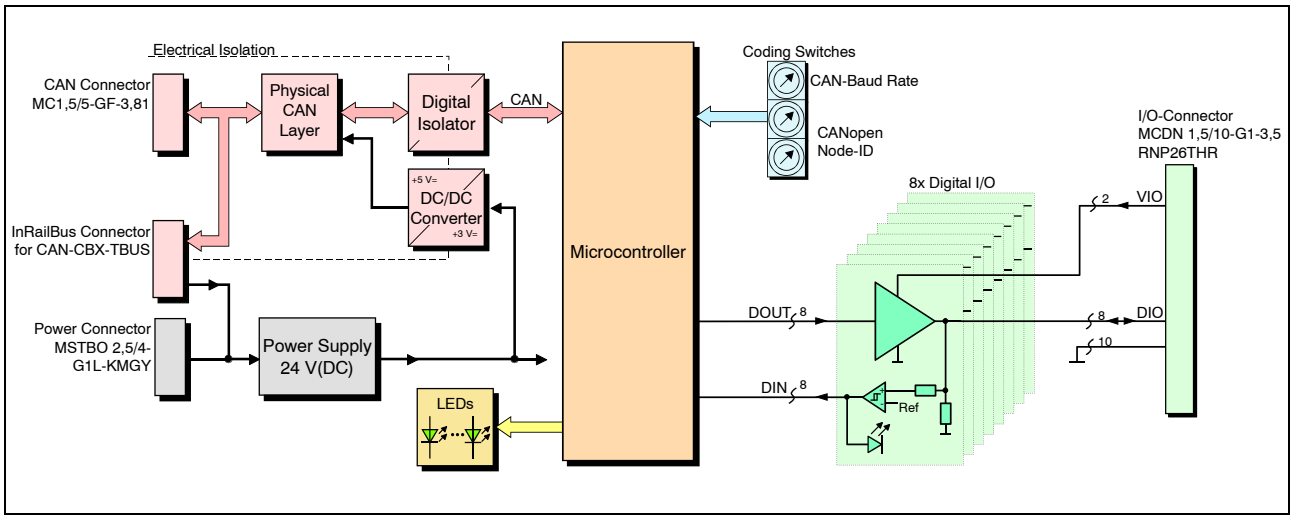


Figure 1: Block circuit diagram

The CAN-CBX-DIO8/2 is a CANopen module that provides 8 flexible digital IO ports that can be independently assigned as inputs or outputs. It operates at a nominal voltage of 24 V and supports a maximum output current of 0.5 A at 24 V, making it suitable for various control scenarios. The inputs can be configured with various functions, including edge-triggered events and counters, allowing customization for specific requirements
The module is connection-compatible with the predecessor module CAN-CBX-DIO8 (C.3010.02) and the functionality is largely identical.
Each port has a dedicated LED that indicates the current I/O state, while four additional LEDs clearly visualize the status of the CANopen node and I/O errors.

The module features a high-speed CAN port compliant with ISO11898 standards. It offers electrical isolation and supports bit rates up to 1 Mbit/s. The CANopen® node number and the CAN bit rate can be configured via three coding switches
CANopen® integration is ensured in accordance with the CiA® specifications:

- CiA® 301 “CANopen application layer and communication profile” (1)
- CiA® 401 “CANopen profile for I/O devices” (2)

The CAN-CBX-DIO8/2 is easy to combine with other modules of esd’s CBX IO series.
The CBX module series offers compact industrial CAN input/output modules with InRailBus.
These modules feature an efficient wiring concept for CAN and supply voltage and are housed in a slim design that focuses on usability.

The InRailBus simplifies the supply of power and CAN bus signals. For user-friendly installation, the CAN-CBX-DIO8/2. It can be seamlessly integrated into the DIN rail via an optional connector (TBUS connector, see page 141). At the same time, individual modules can be removed from the InRailBus without interrupting the bus signals, which enables efficient maintenance.
Alternatively, power and signals can also be connected separately via the terminal connections.

We offer customization options to meet your specific needs. For detailed information, kindly reach out to our sales team.

1.3 Glossary

Abbreviations

Abbreviation	Term	Description
API	Application Programming Interface	
CAN	Controller Area Network	In this manual the term CAN only includes CAN CC and CAN FD. CAN XL is not supported
CAN CC	CAN classic	
CAN FD	CAN flexible data rate	
CPU	Central Processing Unit	
CiA	CAN in Automation	
EDS	Electronic Data Sheet	Description file for CANopen devices
HW	Hardware	
I/O	Input/Output	
LSB	Least Significant Bit	
MSB	Most Significant Bit	
n.a.	not applicable	
OS	Operating System	
PDO	Process Data Object	
RTR	Remote Transmission Request	
SDK	Software Development Kit	
SDO	Service Data Object	

INFORMATION
The CANopen terms and abbreviations are described in the chapter “Definition of Terms” on page 28.

2 Hardware

2.1 Connecting Diagram

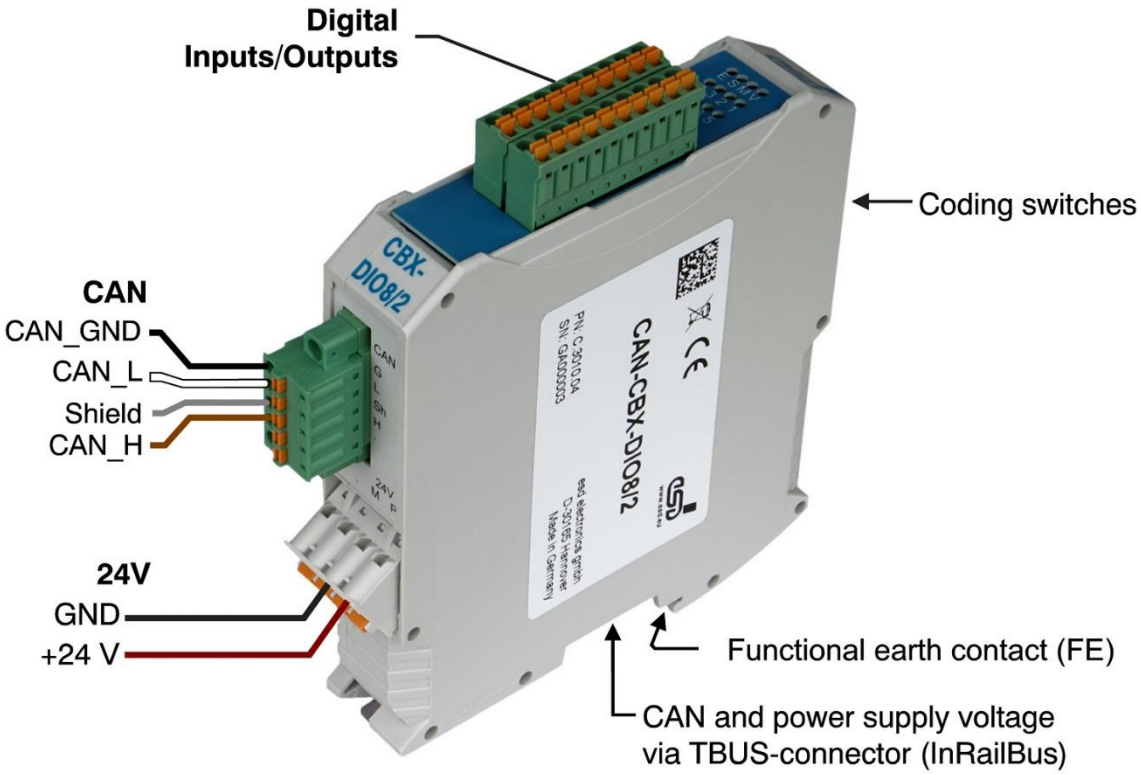


Figure 2: Connecting diagram of CAN-CBX-DIO8/2

See also page 121 for signal assignment of the CAN connectors.
For conductor connection and conductor cross section see page 127

NOTICE
Read chapter “Installing and Uninstalling Hardware” from page 21, before you start with the installation of the hardware!

2.2 LEDs

2.2.1 Position of the LEDs

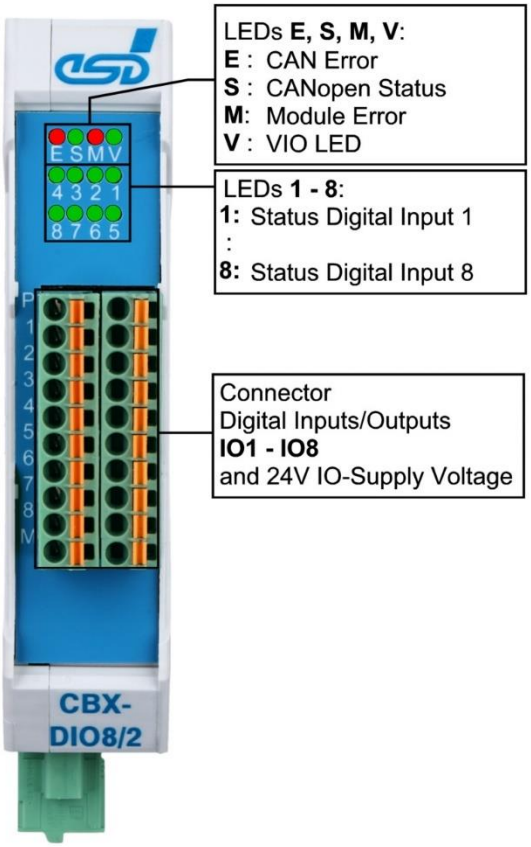


Figure 3: Connectors and LEDs

The CAN-CBX-DIO8/2 module is equipped with 4 status LEDs and 8 LEDs for the input and output ports.

2.2.2 Indicator States

The terms of the indicator states of the LEDs are chosen in accordance with the terms recommended by the CiA (3). The indicator states are described in the following chapters.
In principle there are 8 indicator states distinguished:

Indicator state	Display
On	LED constantly on
Off	LED constantly off
Blinking	LED blinking with a frequency of approx. 2.5 Hz
Flickering	LED flickering with a frequency of approx. 10 Hz
Single flash	One single flash (1x (LED 200 ms on, 1400 ms off))
Double flash	Two short flashes (1x (LED 200 ms on, 200 ms off) + 1x (LED 200 ms on 1000 ms off))
Triple flash	Three short flashes (2x (LED 200 ms on, 200 ms off) + 1x (LED 200 ms on, 1000 ms off))

Table 1: Indicator states

NOTICE
Red and green LEDs are strictly switched in phase opposition according to the CANopen Specification (3)
For certain indicator states viewing all LEDs together might lead to a misinterpretation of the indicator states of adjacent LEDs. It is therefore recommended to look at the indicator state of an LED individually, in covering the adjacent LEDs.

2.2.3 Operation of the CAN-Error LED E

Label	Name	Colour	Indicator state	Description
E	CAN Error	red	Off	No error
			Single flash	CAN controller is in *Error Active* state
			On	CAN controller state is *Bus Off* (or coding switch position ID-node > 0x7F when switching on; see Special Indicator States' on page 18)
			Double flash	Heartbeat or Nodeguard error occurred. The LED automatically turns off if Nodeguard/Heartbeat-messages are received again.

Table 2: Indicator states of the red CAN Error-LED

2.2.4 Operation of the CANopen-Status LED S

Label	Name	Colour	Indicator state	Description
S	CANopen Status	green	blinking	*Pre-operational*
			on	*Operational*
			Single flash	*Stopped*

Table 3: Indicator states of the CANopen Status-LED

2.2.5 Operation of the Error-LED M

Label	Name	Colour	Indicator state	Description
M	Error	red	Off	No error
			On	Error on an error-controlled output - If the module has switched to the *stopped* state due to an error, the LED remains on even if the error is no longer existing - Errors that occur after changing to *stopped* state are not indicated
			Double flash	Internal software error, e.g.: - Stored data have an invalid checksum, therefore default values are loaded - Internal watchdog has triggered - Indicator state is continued until the module resets or an error occurs at the outputs.

Table 4: Indicator state of the Error-LED

2.2.6 Operation of the Module Status LED V

The display functions described in the following table are only valid if the module is in the *Operational* state and Output Port and Output Port Config are set.
If these conditions do not apply, the LED is on.

Label	Name	Colour	Indicator state	Description
V	VIO_ Sense	green	Off	Read value of the operating voltage of the digital outputs (object 0x2300, sub-index 1, see page 110) is lower than 5 V
			Blinking	Read value of the operating voltage of the digital outputs is lower than the value set in object 0x2300 subindex 2 (see page 110)
			Flickering	Read value of the operating voltage of the digital outputs is higher than the value set in object 0x2300 subindex 3 (see page 110)
			On	Read value of the operating voltage of the digital outputs is within the limits (object 0x2300, sub-index 2 and 3, see page 110)

Table 5: Indicator state of the Power-LED

2.2.7 Special Indicator States

The special indicator state described in the following table is indicated by the CANopen-Status-LED and the CAN-Error-LED together:

LED indication	Description
CANopen-Status LED: Triple flash and CAN-Error LED: On	Invalid node ID: The coding switches for the node-ID are set to an invalid ID-value, when switching on. The firmware application will be stopped.

Table 6: Special Indicator States

LED	Indication function = I/O port state
1	IO1
2	IO2
3	IO3
4	IO4
5	IO5
6	IO6
7	IO7
8	IO8

Indicator state of port x	State of IO cannel x
OFF	Input voltage level of IO port x is below the lower switching threshold, or output state is ‘off’
ON	Input voltage level of IO port x is higher than the upper switching threshold, or output state is ‘on’

x = 1 – 8

Table 7: Indication of LEDs 1-8 and state of the ports

2.3 Coding Switch

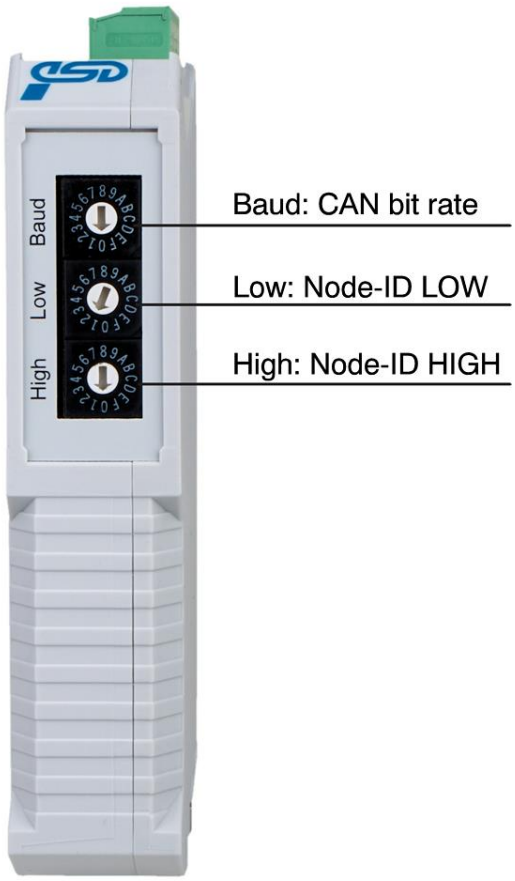


Figure 4: Position of the coding switches

NOTICE
The states of the coding switches are determined the moment the module is switched on. Changes of the settings must therefore be made **before the module is switched on**, as changes of the settings are not detected during operation.
After a reset (e.g. NMT reset), the settings are read in again.

2.3.1 Setting the Node-ID via Coding Switch

The address range of the CAN-CBX-module can be set decimal from 1 to 127 or hexadecimal up to 0x7F.
The three higher-order bits (higher-order nibble) can be set with coding switch **HIGH**.
The four lower-order bits (lower-order nibble) can be set with coding switch **LOW**.

NOTICE
Avoid the following settings:

- Setting the address range of the coding switches to 0x00 or values higher than 0x7F causes error messages, the red CAN-Error LED is on.

2.3.2 Setting the Bit Rate

The bit rate can be set with the coding switch **Baud**.

Values from 0x0 to 0xF can be set via the coding switch. The values of the baud rate can be taken from the following table:

Setting	Bit rate [Kbit/s]
0	1000
1	$666.\overline{6}$
2	500
3	$333.\overline{3}$
4	250
5	$166.\overline{6}$
6	125
7	100
8	$66.\overline{6}$
9	50
A	$33.\overline{3}$
B	20
C	12.5
D	10
E	800
F	$83.\overline{3}$

Table 8: Index of the bit rate

3 Installing and Uninstalling Hardware

To install or uninstall the CAN-CBX-DIO8/2, please follow the installation notes.

Step	Procedure	See Page
	**NOTICE** **Read the safety instructions at the beginning of this document carefully before you start with the hardware installation/!**	5
	**DANGER** Hazardous voltage - Risk of electric shock due to unintentional contact with uninsulated live parts with high voltages inside of the system into which the CAN-CBX-DIO8/2 is to be integrated. → The CAN-CBX-DIO8/2 is a device of protection class III according to DIN EN IEC 61140 and may only be operated on supply circuits that offer sufficient protection against dangerous voltages. → External circuits connected to the CAN-CBX-DIO8/2 must be sufficiently protected against dangerous voltages. → Compliance with the applicable national safety regulations is the responsibility of the user. → Ensure the absence of voltage before starting any electrical work. → The plug connectors must not be plugged in or unplugged under voltage or load!	
To install, continue as described in chapter 3.1 ‘Installing the Hardware’. To uninstall, continue as described in chapter 3.2 ‘Uninstalling the Hardware’		

3.1 Installing the Hardware

Step	Procedure	See Page
1.	Follow the safety instructions at the beginning of chapter 3	21
2.	Mount the CAN-CBX-DIO8/2 module in the control cabinet. Connect the ports (power supply voltage, CAN, digital IOs).	14
	See also chapter 6 for Connector Assignments	121
	If the InRailBus is used, read and follow chapter Using InRailBus	
	**NOTICE** Incorrect wiring of the 24V power supply voltage can cause damage to the module! → Make sure to connect the cables correctly to the 24V cable connector! → Only use suitable cables for the plug.	123, 127
3.	Please note that the CAN bus must be terminated at both ends! esd offers special T-connectors and termination connectors for external termination. Additionally, the CAN_GND signal must be connected to earth at exactly one point in the CAN network. For details, please read chapter “Correct Wiring of Electrically Isolated CAN Networks”.	128

<table>
<tr><td>4.</td><td>Set the configuration switches according to your needs or program the CAN-CBX-DIO8/2 as needed.
Set the bit rate (only if another bit rate than the default bit rate is requested.)
The default bit rate is 1 Mbit/s. It can be configured via the
coding switch BAUD, as described in chapter 2.3.2.</td><td>20</td></tr>
<tr><td>5.</td><td>Set the module number (Node-ID), see chapter 2.3.1.
The node-ID can be configured via the coding switches LOW und HIGH.
It can be set to values between 1 and 127 (0x01-0x7F).
Example: For node- ID 1 the coding switch LOW has to be set to ‘1' and
the coding switch HIGH has to be set to '0'.</td><td>19</td></tr>
<tr><td>6.</td><td>Before you switch on the supply voltage, check that all plug connectors are correctly seated.
Switch on the 24 V-power supply voltage of the CAN-CBX-DIO8/2.</td><td>-</td></tr>
<tr><td>7.</td><td>Start the module with the NMT Start Command
<table>
<tr><th>CAN Identifier</th><th>Len</th><th colspan="2">Data</th></tr>
<tr><td>0</td><td>2 Byte</td><td>1</td><td>0
(Node-ID,
0 = start all modules)</td></tr>
</table></td><td></td></tr>
<tr><td>8</td><td>Transmit data CAN -> Digital Output
<table>
<tr><th>CAN Identifier</th><th>Len</th><th>Data</th></tr>
<tr><td>0x200 + Node-ID</td><td>8 Byte</td><td>0 ... 8 byte user data
Set digital outputs</td></tr>
</table></td><td></td></tr>
<tr><td>9.</td><td>Receive data Digital Input -> CAN
<table>
<tr><th>CAN Identifier</th><th>Len</th><th>Data</th></tr>
<tr><td>0x180 + Node-ID</td><td>8 Byte</td><td>0 ... 8 byte user data
Set digital inputs</td></tr>
</table></td><td></td></tr>
</table>

3.2 Uninstalling the Hardware

Step	Procedure	See Page
1.	Follow the safety instructions at the beginning of chapter 3	21
2.	Make sure that all connected interfaces and power supply are switched off.	
3.	Disconnect the CAN-CBX-DIO8/2 from the connected interfaces.	
4.	Use a screwdriver to pull the fastening spring downwards while swivelling the CAN-CBX-DIO8/2 module upwards until it comes loose.	
5.	Carefully take the CAN-CBX-DIO8/2 out.	

3.3 Using InRailBus

This chapter describes the installation of the module using InRailBus as an example for CAN-CBX-modules. The InRailBus connectors are not included in delivery

3.3.1 Installation of the Module when using the InRailBus Connector

If the CAN bus signals and the power supply voltage shall be fed via the InRailBus, please proceed as follows:

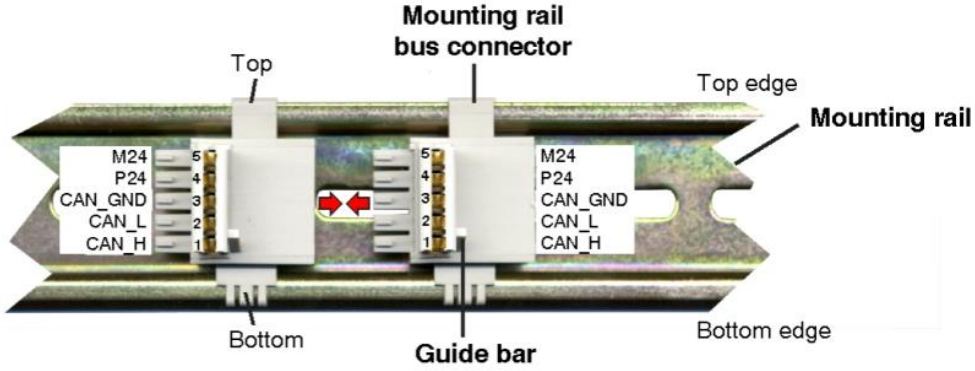


Figure 5: Mounting rail with bus connector

1. Position the InRailBus connector on the mounting rail and snap it onto the mounting rail using slight pressure. Plug the bus connectors together to contact the communication and power signals (in parallel with one). The bus connectors can be plugged together before or after the CAN-CBX module is plugged on.

2. Hold the CAN-CBX module tilted backwards at a slight angle and place it on the bus connector so that the DIN rail guideway is placed on the top edge of the mounting rail.

Figure 6: Mounting CAN-CBX modules

3. Now swivel the CAN-CBX module onto the mounting rail by moving the module downwards according to the direction of the arrow in Figure 6. The housing is mechanically guided by the guide bar of the bus connector.

4. When mounting the CAN-CBX module the moveable snap-on foot snaps onto the bottom edge of the mounting rail.
   The module is now firmly seated on the mounting rail and is connected to the InRailBus via the bus connector. If necessary, connect the bus connectors to each other and connect the +24 V supply voltage and the CAN interface to the InRailBus as described below.

Figure 7: Mounted CAN-CBX module

3.3.2 Connecting via the InRailBus

To connect the power supply and the CAN signals via the InRailBus, a terminal plug is needed. The terminal plug is not included in the scope of delivery and must be ordered separately (order no.: C.3000.02, see order information for InRailBus Accessories).

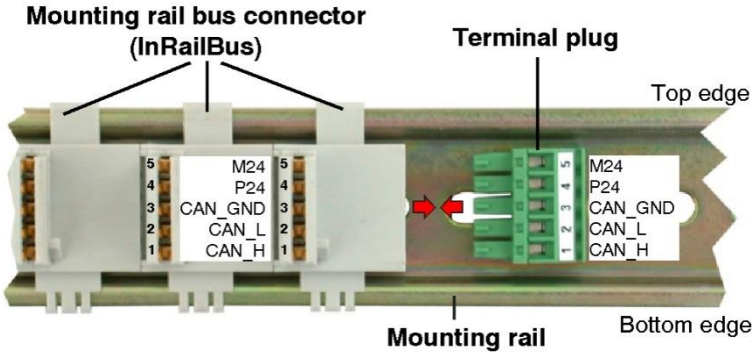


Figure 8: Mounting rail with InRailBus and terminal plug

Insert the terminal plug from the right into the socket side of the outer mounting rail bus connector of the InRailBus, as shown in Figure 8. Then connect the CAN interface and the supply voltage via the terminal plug.

3.3.3 Connection of the Supply Voltage

DANGER
Hazardous Voltage - **Risk of electric shock** due to unintentional contact with uninsulated live parts with high voltages inside of the system into which the CAN-CBX-module is to be integrated.

→ Read the safety instructions at the beginning of this document (from page 5) carefully before you start with the hardware installation!
→ Ensure the absence of voltage before starting any electrical work.
→ Switch off the power supply, before you connect it to the system.

DANGER
The CAN-CBX-DIO8/2 is a device of protection class III according to DIN EN IEC 61140 and may only be operated on supply circuits that offer sufficient protection against dangerous voltages.

There are two ways to feed the 24 V power supply voltage into the CBX station:
- via the terminal plug of the InRailBus, see 3.3.3.1
- via the 24 V connector of the first module in the CBX station, see 3.3.3.2

NOTICE
The two connections for the 24 V power supply (via InRailBus or 24 V connector) are connected internally and must not be supplied by two independent power sources at the same time!
Connecting 24 V at both connectors will cause damage to the CAN-CBX module.
Also read the chapter on the assignment of the 24 V connector for further information.

3.3.3.1 Connection of the Power Supply Voltage via InRailBus

NOTICE
If you feed the 24V power supply via the terminal plug of the InRailBus (see Figure below), the maximum load current must not exceed $I_{MAX\_LOAD\_InRailBus}$ = 8 A.

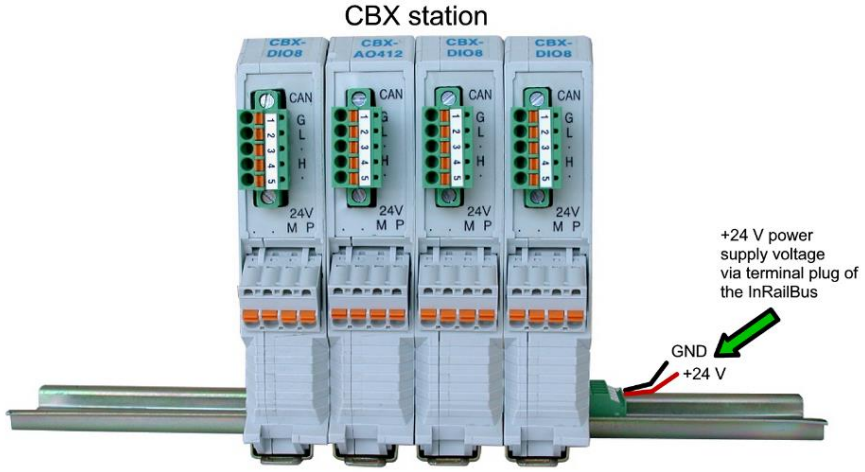


Figure 9: Connection via terminal plug

3.3.3.2 Connection of the Power Supply Voltage via 24 V Connector

NOTICE
Note that the connection between the 24V plug and the InRailBus is not designed to feed the 24V supply voltage via the plug to the InRailBus!
If the modules are mounted on the DIN rail without InRailBus connectors, it is allowed to bridge the power supply from one module to another (see Figure below), but the maximum load current must not exceed $I_{MAX\_LOAD\_24V\_plug}$ = 2 A.

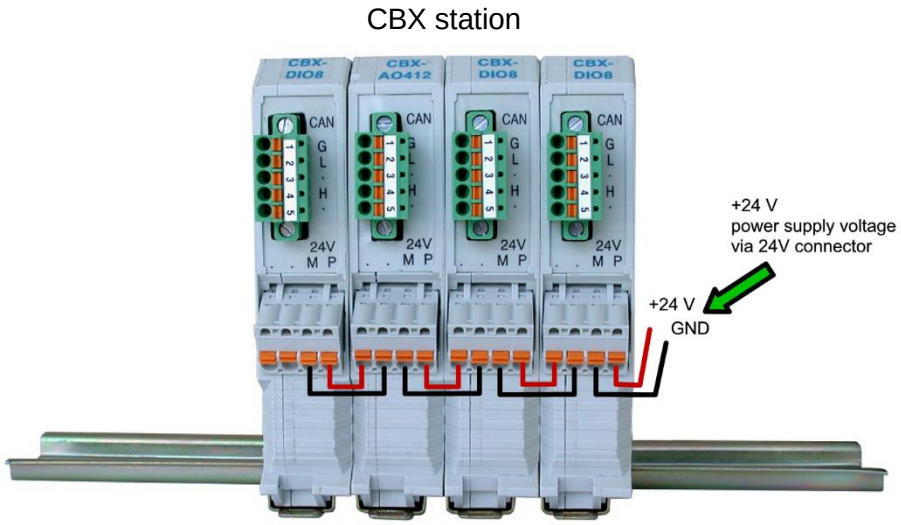


Figure 10: Connection via 24V Connector

3.3.3.3 Earthing of the Mounting Rail

NOTICE
The module is connected to the mounting rail via its functional earth contact. This improves the stability against electromagnetic disturbances. The mounting rail must therefore be connected to a suitable functional earth contact in the environment or in the installation. It must be ensured that the impedance of the connection is kept low.
The functional earth contact of the module does not ensure electrical safety.

3.3.4 Connection of CAN

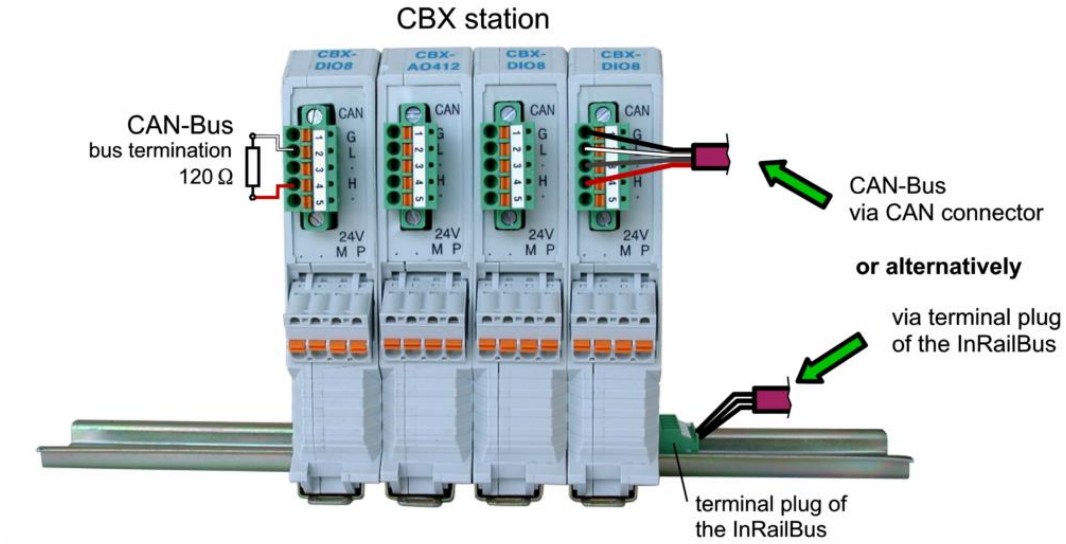


Figure 11: Connecting the CAN signals to the CAN-CBX station

In general, the CAN signals can be fed in via the InRailBus or via the CAN connector of the first CAN-CBX module in the CAN-CBX station. The signals are then connected through the CAN-CBX station via the InRailBus. The CAN signals may be lead through via the CAN connector of the CAN-CBX module mounted at the other end of the CBX station. However, the CAN signals must not be connected via the CAN connectors of the middle CAN-CBX modules of the CBX station, as this would lead to impermissible branching.
Please note that a bus terminating resistor must be connected to the CAN-CBX module located at the end of the InRailBus if the CAN bus ends there (see Figure 11).

3.3.5 Remove the CAN-CBX Module from InRailBus

If the CAN-CBX module is only connected via the InRailBus, proceed as follows when removing it:
Release the module from the mounting rail by moving the snap-on foot (see Figure 7) downwards (e.g., with a screwdriver). This releases the module from the bottom edge of the mounting rail, and it can be removed.

INFORMATION
It is possible to remove individual devices from the CBX station without interrupting the InRailBus connection, because the contact chain will not be disrupted.

4 CANopen Firmware

The CAN-CBX-DIO8/2 presents itself as a CANopen responder device. The "Digital In" and "Digital Out" functions conform to the CiA® profile DS-401.

The EDS file can be downloaded under *Software Downloads* from the CAN-CBX-DIO8/2 product page on esd website: https://esd.eu/en/products/can-cbx-dio8-2

For the purpose of backwards compatibility, the CAN-CBX-DIO8/2 can also be operated with the EDS file of the predecessor module CAN-CBX-DIO8 (within the scope of the functions of the predecessor module). The properties and functions described in the manual (v.1.1) of the CAN-CBX-DIO8 predecessor module in section 'CANopen Firmware' are supported.

It is possible to update the firmware via CANopen in the field.

Apart from basic descriptions of CANopen, this chapter contains the most significant information about the implemented functions.
A complete CANopen description is too extensive for the purpose of this manual.
Further information can therefore be taken from the CiA® CANopen documentation (1) and (2).

4.1 Definition of Terms

Name	Description
COB ...	Communication Object
Emergency-Id...	Emergency Data Object
NMT...	Network Management (Manager)
PDOs	Process Data Object (see description below)
Rx …	Receive
SDO	Service Data Object (see description below)
Sync...	Sync(frame) Telegram
Tx …	Transmit

Description of SDO and PDO

SDO (Service Data Object)

SDOs are used to transmit module internal configuration- and parameter data. In opposition to the PDOs, SDO-messages are confirmed.
A write or read request on a data object is always answered by a response telegram with an error index.

PDOs (Process Data Objects)

PDOs are used to transmit process data.
In the 'Transmit'-PDO (TPDO) the CAN-CBX-module transmits data to the CANopen network.
In the 'Receive'-PDO (RPDO) the CAN-CBX-module receives data from the CANopen network.

4.2 NMT-Boot-up

The CAN-CBX module can be initialized with the 'Minimum Capability Device' boot-up as described in (1).
Usually, a telegram to switch from *Pre-operational* status to the *Operational* status after boot-up is sufficient. For this the 2-byte telegram '0x01', '0x00', for example, must be transmitted with CAN identifier '0x0000' (= Start Remote Node all Devices).

4.3 The CANopen-Object Directory

The object directory is basically a (sorted) group of objects which can be accessed via the CAN network. Each object in this directory is addressed with a 16-bit index. The index in the object directories is represented in hexadecimal format.
The index can be a 16-bit parameter in accordance with the CANopen specification (1) or a manufacturer-specific code. By means of the MSBs of the index the object class of the parameter is defined.
Part of the object directory are among others:

Index	Object
0x0001 ... 0x009F	Definition of data types
0x1000 ... 0x1FFF	Communication Profile Area
0x2000 ... 0x5FFF	Manufacturer Specific Profile Area
0x6000 ... 0x9FFF	Standardized Device Profile Area
0xA000 ... 0xFFFF	Reserved

4.4 Communication Parameters of the PDOs

The communication parameters of the PDOs (according to (1) are transmitted as SDO (Service Data Objects) on ID '**0x600 + Node-ID**' (Request).
The receiver acknowledges the parameters on ID '**0x580 + Node-ID**' (Response).
The Node-ID (module No.) is configured via coding switches Low and High. Please refer to chapter "Coding Switches" for a detailed description of possible configurations.

4.4.1 Access on the Object Directory via SDOs

The SDOs (Service Data Objects) are used to access the object directory of a device.
An SDO is therefore a 'channel' to access the parameters of the device. Access via this channel is possible in *Operational* and *Pre-operational* status.
The SDOs (Service Data Objects) are transmitted on ID '**0x600 + Node-ID**' (request).
The server acknowledges the parameters on ID '**0x580 + Node-ID**' (response).

An SDO is structured as follows:

Identifier	Command code	Index		Sub-index	LSB	Data field		MSB
		(low)	(high)					
Example:								
0x600+ Node-ID	0x23	0x00	0x14	0x01	0x7F	0x04	0x00	0x00
	(write)	(Index=0x1400) (Receive-PDO-Comm-Para)		(COB-def.)	COB Node ID = 0x0000 047F			

Identifier
The parameters are transmitted with ID '0x600 + NodeID' (request).
The receiver acknowledges the parameters with ID '0x580 + NodeID' (response).

Command code
The command code transmitted consists among other things of the Command Specifier and the length.
Frequently required combinations are, for instance:

0x40 (= 64 decimal):	Read Request, i.e. a parameter is to be read.
0x23 (= 35 (decimal):	Write Request with 32-bit data, i.e. a parameter is to be set.

Response telegram
The CAN-CBX-module responds to every received telegram with a response telegram. This can contain the following command codes:

0x43 (= 67 decimal):	Read Response with 32-bit data, this telegram contains the parameter requested
0x 60 (= 96 decimal):	Write Response, i.e. a parameter has been set successfully
0x 80 (= 128 decimal):	Error Response, i.e. the CAN-CBX-module reports a communication error

Frequently Used Command Codes
The following table summarizes frequently used command codes. The command frames must always contain 8 data bytes. Notes on the syntax and further command codes can be found in (1).

Command	Number of data bytes	Command code
Write Request (Initiate Domain Download)	1 2 3 4	0x2F 0x2B 0x27 0x23
Write Response (Initiate Domain Download)	-	0x60
Read Request (Initiate Domain Upload)	-	0x40
Read Response (Initiate Domain Upload)	1 2 3 4	0x4F 0x4B 0x47 0x43
Error Response (Abort Domain Transfer)	-	0x80

Index, Sub-Index
Index and sub-index will be described in the chapters "Device Profile Area" and "Manufacturer Specific Objects" of this manual.

Data Field
The Data Field has got a size of a maximum of 4 bytes and is always structured 'LSB first, MSB last'. The least significant byte is always in 'Data 1'. With 16-bit values the most significant byte (bits 8...15) is always in 'Data 2', and with 32-bit values the MSB (bits 24...31) is always in 'Data 4'.

Error Codes of the SDO Domain Transfer
The following error codes might occur (according to (1)):

Abort code	Description
0x05040001	Wrong command specifier
0x06010002	Wrong write access
0x06020000	Wrong index
0x06040041	Object cannot be mapped to PDO
0x06060000	Access failed due to a hardware error
0x06070010	Wrong number of data bytes
0x06070012	Service parameter too long
0x06070013	Service parameter too small
0x06090011	Wrong sub-index
0x06090030	Transmitted parameter is outside the accepted value range
0x08000000	Undefined cause of error
0x08000020	Data cannot be transferred or stored in the application
0x08000022	Data cannot be transferred or stored in the application because of the present device state
0x08000024	Access to flash failed

4.4.2 Non-volatile Storage of Parameters to EEPROM

After the transfer the parameters are immediately active.
The non-volatile storage of the parameters however is not carried out automatically. It must be initiated with a write access to object 0x1010 and should only be carried out if the module is in the state *pre-operational*.
The storage mode is shown in the content of the object 0x1010:
Bit 1 of object 0x1010, sub-index 1 is not set, i.e the CAN-CBX-module does not save the configuration automatically. The storage must be initiated by writing the character string ‘save’ (0x73 61 76 65, order from CAN telegram) to object 0x1010, sub-index 1.

4.5 Overview of used CANopen-Identifiers

Function	Identifier	Description
Network management	0	NMT
SYNC	0x80	Sync to all, (configurable via object 0x1005)
Emergency Message	0x80 + *NodeID*	Configurable via object 0x1014
TPDO1	0x180 + *NodeID*	PDO from CAN-CBX-DIO8/2 (Tx) (object 0x1800)
RPDO1	0x200 + *NodeID*	PDO to CAN-CBX-DIO8/2 (Rx) (object 0x1400)
Client-SDO	0x580 + *Node-ID*	SDO from CAN-CBX-DIO8/2 (Tx)
Server-SDO	0x600 + *Node-ID*	SDO to CAN-CBX-DIO8/2 (Rx)
Node Guarding	0x700 + *NodeID*	See object 0x100E

NodeID: CANopen address [0x01...0x7F]

4.5.1 Setting the COB-ID

The COB-IDs which can be set (except the one of SYNC), are deduced initially from the setting of the Node-ID via the coding switches (see 2.3). If the COB-IDs are set via SDO, this setting is valid even if the coding switches are set to another Node-ID after that.
To accept the Node-ID from the coding switches again, the *Comm defaults* or all defaults must be restored (object 0x1011).

4.6 Default PDO-Assignment

PDOs (Process Data Objects) are used to transmit process data. The PDO mapping can be changed. The following tables show the default mapping at delivery of the module:

PDO	CAN identifier	Length	Transmission direction	Assignment
RPDO1	0x200 + Node-ID	1 byte	To CAN-CBX-DIO8/2 (Receive-PDO)	Setting of the outputs
TPDO1	0x180 + Node-ID	1 byte	From CAN-CBX-DIO8/2 (Transmit PDO)	Request the state of the inputs

RPDO1 (-> CAN-CBX-DIO8/2)

CAN Identifier: 0x200 + Node-ID

Byte	0	1	2	3	4	5	6	7
Parameter	*Write_ Output_8-Bit*	-	-	-	-	-	-	-

Parameter description:

Parameter	Description	Data type	See page
Write_ Output_ 8-Bit	Setting of the digital outputs	Byte	92

TPDO1 (CAN-CBX-DIO8/2 ->)

CAN Identifier: 0x180 + Node-ID

Byte	0	1	2	3	4	5	6	7
Parameter	*Read_Input_8-Bit*	-	-	-	-	-	-	-

Parameter description:

Parameter	Description	Data type	See page
Read_Input_ 8-Bit	State of the the digital inputs	Byte	92

4.7 Setting and Reading the Outputs/Inputs

4.7.1 Status Message of the Digital Inputs

The following transmission types are available at the CAN-CBX-DIO8/2:

- acyclic, synchronous: The transmission is initiated if a SYNC-message has been received (PDO transmission type 0) and data has changed.
- cyclic, synchronous: The transmission is initiated if a defined number of SYNC-messages have been received (PDO-transmission type 1...240).
- synchronous, remote request: The state of the inputs is latched with each SYNC-message and is transmitted after the reception of a RTR-frame (PDO-transmission type 252).
- asynchronous, remote request: After the reception of a RTR-frame the last latched state of the inputs is transmitted (PDO-transmission type 253).
- event controlled, asynchronous: The transmission is initiated if the state of selected inputs has changed (PDO-transmission type 254, 255).

4.7.2 Digital Outputs

The digital outputs are set, as soon as an output setting object is received by the CAN-CBX-DIO8/2 (e.g. object 0x6200 via RPDO).

4.7.3 Supported Transmission Types Based on CiA 301

Trans-mission Type	PDO-Transmission					supported by CAN-CBX-DIO8/2
	cyclic	acyclic	synchro-nous	asynchro-nous	RTR	
0		X	X			x
1...240	X		X			x
241...251	reserved					-
252			X		X	x
253				X	X	x
254				X	X	x
255				X	X	x

4.8 Communication Profile Area

4.8.1 Used Names and Abbreviations

The following names are used in the tables for the description of the communication parameters:

PDO-Mappable	PDO-Mapping is possible for this sub-index of the PDO
Save to EEPROM	The value of this parameter is stored in the local EEPROM, if the command ‘save’ is called (see object 0x1010)
Data type	Data type (e.g. unsigned 8, unsigned 32)
Access mode	Allowed access modes to this parameter ro... read_only This parameter can only be read. Write accesses will cause an error message. rw... read&write This parameter can be read or written.
Value range	Value range of the parameter
Default value	Default setting of the parameter
Name/Description	Name and short description of the parameter

4.9 Implemented CANopen Objects

A detailed description of the Implemented CANopen Objects can be taken from CiA 301.

4.9.1 Overview Communication Profile Objects / Product-Specific Values

Index	Sub-index	Description	Data type	Access mode	Product-specific properties
0x1000	-	Device Type	unsigned 32	ro	Value of device type: 0x0003 0191
0x1001	-	Error Register	unsigned 8	ro	Supported error bits: 0: generic 1: current 2: voltage 4: communication error
0x1003	0xA	Pre-Defined-Error-Field	unsigned 32	rw	0x00
0x1005	-	COB-ID-Sync	unsigned 32	rw	0x80
0x1006	-	Communication Cycle Period	unsigned 32	rw	0x00
0x1008	-	Manufacturer Device Name	visible string	ro	“CAN-CBX-DIO8/2”
0x1009	-	Manufacturer Hardware Version	visible string	ro	x.yy (depending on version)
0x100A	-	Manufacturer Software Version	visible string	ro	x.yy (depending on version)
0x100B	-	Node-ID	unsigned 32	ro	-
0x100C	-	Guard Time	unsigned 16	rw	0x0000
0x100D	-	Life Time Factor	unsigned 8	rw	0x00
0x100E	-	Node Guarding Identifier	unsigned 32	ro	Node-ID + 0x700
0x1010	0x4	Store Parameter	unsigned 32	rw	n.a.
0x1011	0x4	Restore Parameter	unsigned 32	rw	n.a.
0x1014	-	COB-ID Emergency Object	unsigned 32	rw	0x80 + Node-ID
0x1015	-	Inhibit Time EMCY	unsigned 16	rw	0x00
0x1016	0x1	Consumer Heartbeat Time	unsigned 32	rw	0x00
0x1017	-	Producer Heartbeat Time	unsigned 16	rw	0x00
0x1018	0x4	Identity Object	unsigned 32	ro	Vendor Id: 0x00000017 Prod. code: 0x23010004, Rev. number: e.g. 0x10, Serial number depending on individual module
0x1019	-	Synchronous Counter Overflow	unsigned 8	rw	0x00
0x1020	0x2	Verify Configuration	unsigned 32	rw	Configuration date and time
0x1029	0x6	Error Behaviour	unsigned 8	rw	Supported error classes: 6

Index	Sub-index	Description	Data type	Access Mode
0x1400	2	1. Receive PDO-Parameter	PDO CommPar (0x20)	rw
0x1600	1	1. Receive PDO-Mapping	PDO Mappping (0x21)	ro
0x1800	5	1. Transmit PDO Parameter	PDO CommPar (0x20)	rw
0x1801	5	2. Transmit PDO Parameter	PDO CommPar (0x20)	rw
0x1A00	1	1. Transmit PDO Mapping	PDO Mapping (0x21)	ro
0x1A01	4	2. Transmit PDO Mapping	PDO Mapping (0x21)	ro

Index	Sub-index	Description	Data type	Access mode	Product-specific properties
0x1F80	-	NMT startup	unsigned 32	rw	default: 2 (autostart disabled)
0x1F91	1	Self-starting nodes timing parameters	unsigned 16	rw	default: 0x64 (= 100 ms)

4.9.2 DeviceTpe (0x1000)

INDEX	**0x1000**
Name	*device type*
Data type	unsigned 32
Access mode	ro
Default value	see chapter ‘Overview Communication Profile Objects/ Product-Specific Values’ (page 36)

Example: Reading the Device Type

The CANopen manager transmits the read request with identifier ‘0x603’ (0x600 + Node-ID) to the CAN-CBX module with the module no. 3 (Node-ID=0x3):

ID	RTR	LEN	DATA							
			1	2	3	4	5	6	7	8
0x603	0x0	0x8	0x40	0x00	0x10	0x00	0x00	0x00	0x00	0x00
			Read Request	Index=0x1000		Sub-Index				

The CAN-CBX module no. 3 responds to the client by means of read response with identifier ‘0x583’ (0x580 + Node-ID) with the value of the device type:

ID	RTR	LEN	DATA							
			1	2	3	4	5	6	7	8
0x583	0x0	0x8	0x43	0x00	0x10	0x00	0x94	0x01	0x02	0x00
		Read Response		Index=0x1000		Sub Index	Example here: Device Profile No. 0x0194		Example here: Input	

Value of the device type in the example above: 0x0002.0194.
The value of the device type of the CAN-CBX-DIO8/2 module is printed in chapter (page 36).

The data field is always structured following the rule: ‘LSB first, MSB last‘, see **Data Field** page 30.

4.9.3 Error Register (0x1001)

The CAN-CBX module uses the error register to indicate error messages.

INDEX	**0x1001**
Name	*error register*
Data type	unsigned 8
Access mode	ro
Default value	0

The following bits of the error register are being supported at present:

Bit	Meaning
0	*generic*
1	*current*
2	*voltage*
3	*temperature*
4	*communication error* (overrun, error state)
5	*device profile*
6	reserved
7	*manufacturer*

For a list of the error bits supported by the CAN-CBX-DIO8/2 module see in chapter (page 36).

Bits which are not supported are always returned as ‘0’.
If an error is active, the according bit is set to ‘1’.

4.9.4 Pre-defined Error Field (0x1003)

INDEX	**0x1003**
Name	*pre-defined error field*
Data type	unsigned 32
Access mode	ro
Default value	No

The *pre-defined error field* provides an error history of the errors that have occurred on the device and have been signalled via the Emergency Object.

Sub-index 0 contains the current number of errors stored in the list.
Under sub-index 1 the last error which occurred is stored.
If a new error occurs, the previous error is stored under sub-index 2 and the new error under sub-index 1, etc. In this way a list of the error history is created.
The error buffer is structured like a ring buffer. If it is full, the oldest entry is deleted for the latest entry.

This module supports a maximum of 10 error entries. When the 11th error occurs the oldest error entry is deleted. To delete the entire error list, sub-index 0 has to be set to '0'. This is the only permissible write access to the object.

With every new entry to the list the module transmits an **Emergency Frame** to report the error.

Index	Sub-index	Description	Value range	Default	Data type	Access mode
0x1003	0	*no_of_errors_in_list*	0, 1, ... 0xA	-	unsigned 8	rw
	1	*error-code n*	0 ... 0xFFFF FFFF	0	unsigned 32	ro
	2	*error-code (n-1)*	0 ... 0xFFFF FFFF	0	unsigned 32	ro
	:	*:*	:	:	:	:
	0xA	*error-code (n-9)*	0 ... 0xFFFF FFFF	0	unsigned 32	ro

Meaning of the variables:

no_of_errors_in_list - contains the number of error codes currently on the list.
n = number of the error, which occurred last.
- To delete the error list, this variable has to be set to '0'.
- If *no_of_errors_in_list* ≠ 0, the error register (Object 0x1001) is set

error-code x The 32-bit long error code consists of the CANopen-emergency error code described in (1) and the error code defined by esd (manufacturer-specific error field).

Bit:	31 16	15 0
Contents:	*manufacturer-specific error field*	*emergency-error-code*

manufacturer-specific

error field: Always ‘00’, unless *emergency-error-code* = 0x2300 (see below)

emergency-error-code: The following error-codes are supported:

0x8110 -	CAN overrun error: Sample rate is set too high. Thus, the firmware is not able to transmit all data to the CAN bus.
0x8120 -	CAN in error passive mode
0x8130 -	Lifeguard error / heartbeat error
0x8140 -	Recovered from “Bus Off
0x8240 -	Unexpected SYNC data length
0x6000 -	Software error: EEPROM checksum error (no transmission of this error message as emergency message)
0x6110 -	Internal Software error e.g.: Saved data had invalid checksum and default data is loaded
0xFF10 -	Data loss (A/D data overflow)
0x5000 -	Hardware error (e.g. A/D converter defective)
0x5030 -	Sensor error
0x3110	VIO too high, (configurable via object 0x2300)
0x3120	VIO too low, (configurable via object 0x2300)
0x2300	Output Error: open load (no load detected at least at one output)
0x2320	Output Error: short circuit at outputs (at least one output shorted to GND)

Emergency Message

The data off the emergency frame transmitted by the CAN-CBX-module have the following structure:

Byte:	0	1	2	3	4	5	6	7
Contents:	*emergency-error-code* (see above)		*error-register* 0x1001	*no_of_errors _ in_list* 0x1003,00	-			

An emergency message is transmitted if an error occurs. If this error occurs again, no further emergency message is generated.
If the last error message is cancelled, again an emergency message is transmitted to indicate the error disappearance.

4.9.5 COB-ID of SYNC-Message (0x1005)

INDEX	**0x1005**
Name	*COB-ID SYNC message*
Data type	unsigned 32
Access mode	rw
Default value	see chapter ‘Overview Communication Profile Objects / Product-Specific Values’ (page 36)

Structure of the parameter:

Bit-No.	Value	Meaning
31 (MSB)	-	do not care
30	0/1	0: Device does not generate SYNC message 1: Device generates SYNC message
29	0	always 0 (11-bit ID)
28...11	0	always 0 (29-bit IDs are not supported)
10...0 (LSB)	x	Bit 0...10 of the SYNC-COB-ID

The identifier can take values between 0...0x7FF.

4.9.6 Communication Cycle Period (0x1006)

INDEX	**0x1001**
Name	*Communication Cycle Period*
Data type	unsigned 32
Access mode	rw
Default value	0 [µs]

Value range of the parameter:

Value	Meaning
0	No transmission of SYNC messages
0x0000 0001 ... 0xFFFF FFFF	Cycle time in microseconds, Example: 0x4E20 corresponds to a cycle time of 20000 µs = 20 ms

4.9.7 Manufacturer Device Name (0x1008)

INDEX	**0x1008**
Name	*manufacturer device name*
Data type	visible string
Default value	see chapter ‘Overview Communication Profile Objects/ Product-Specific Values’ (page 36)

For detailed description of the SDO Uploads, please refer to CiA DS 202-2 (CMS-Protocol Specification) (4).

4.9.8 Manufacturer Hardware Version (0x1009)

INDEX	**0x1009**
Name	*manufacturer hardware version*
Data type	visible string
Default value	String, depending on hardware version

The hardware version is read similarly to reading the manufacturer's device name via the domain upload protocol. Please refer CiA DS 202-2 (CMS-Protocol Specification) (4) for a detailed description of the upload.

4.9.9 Manufacturer Software Version (0x100A)

INDEX	**0x100A**
Name	*manufacturer software version*
Data type	visible string
Default value	String, depending on software version

Reading the software version is similar to reading the manufacturer's device name via the domain upload protocol. Please refer to CiA DS 202-2 (CMS-Protocol Specification) (4) for a detailed description of the upload.

4.9.10 Guard Time (0x100C) and Life Time Factor (0x100D)

The CAN-CBX-DIO8/2 supports the node guarding or alternatively the heartbeat function (see page 55).

NOTICE
By the recommendation of the CiA, the heartbeat-function shall be used preferentially. Use the node-guarding only for existing systems and not for new developments!

Guard time and *life time factor* are evaluated together. Multiplying both values will give you the life time. The guard time is represented in milliseconds.

INDEX	**0x100C**
Name	*guard time*
Data type	unsigned 16
Access mode	rw
Default value	0 [ms]
Minimum value	0
Maximum value	0xFFFF (65.535 s)

INDEX	**0x100D**
Name	*life time factor*
Data type	unsigned 8
Access mode	rw
Default value	0
Minimum value	0
Maximum value	0xFF

4.9.11 Node Guarding Identifier (0x100E)

The module only supports 11-bit identifiers.

INDEX	**0x100E**
Name	*node guarding identifier*
Data type	unsigned 32
Access mode	ro
Default value	0x700 + Node-ID

Structure of the parameter *node guarding identifier*:

Bit-No.	Meaning
31 (MSB) 30	reserved
29 ... 11	always 0, because 29-bit-IDs are not supported
10 ... 0 (LSB)	bit 0 ... 10 of the *node guarding identifier*

The identifier can take values between 0x701...0x7FF, depending on the Node-ID set by the coding switches.

4.9.12 Store Parameters (0x1010)

INDEX	0x1010
Name	*store parameters*
Data type	unsigned 32

This object supports saving of parameters to a non-volatile memory, the EEPROM here.
The parameter groups shown below are distinguished. After they are transferred, the parameters are immediately active. The non-volatile storage of the parameters however is not carried out automatically. It must be initiated with a write access to object 0x1010 and should only be carried out if the module is in the state *pre-operational*. To avoid storage of parameters by mistake, storage is only executed when the specific signature as shown below is transmitted.

Reading the index returns information about the implemented storage functionality (refer to (1) for more information).

Index	Sub-index	Description	Value range	Data type	Access mode
0x1010	0	*number_of_entries*	4	unsigned 8	ro
	1	*save_all_parameters* (objects 0x1000 ... 0x9FFF)	no default, write: 0x65 76 61 73 (= ASCII: ‘e’ ‘v’ ‘a’ ‘s’) read: 1	unsigned 32	rw
	2	*save_communication_parameter* (objects 0x1000 ... 0x1FFF)		unsigned 32	rw
	3	*save_application_parameter* (objects 0x6000 ... 0x9FFF)		unsigned 32	rw
	4	*save_manufacturer_parameter* (objects 0x2000 ... 0x5FFF)		unsigned 32	rw

Assignment of the variables:

save all parameters
Saves the parameters of all objects (if available), which have a read/write (rw) right of access.

save_communication_parameter
Saves all communication parameters of those objects (objects 0x1000 ... 0x1FFF, if available), which have a read/write (rw) right of access (here e.g. 0x1005 ... 0x1029).

save_application_parameter
Saves all application parameters of those objects (objects 0x6000 ... 0x9FFF, if available), which have a read/write (rw) right of access (here e.g. 0x6xxx).

save_manufacturer_parameter
Saves all manufacturer parameters of those objects (objects 0x2000 ... 0x5FFF, if available), which have a read/write (rw) right of access (here e.g. 0x2xxx).

The storage mode is shown in the content of this object:
Bit 1 of object 0x1010, sub-index 1 is not set, i.e the CAN-CBX-module does not save the configuration automatically.
The storage must be initiated by writing the character string ‘save’ (0x73 61 76 65, order from CAN telegram) to object 0x1010, sub-index 1 - 4.

On read access to the appropriate sub-index, the CAN-CBX-DIO8/2 provides information about its storage functionality with the format described in the following:

Bit:	31	30 … … 2	1	0
Content:	reserved		**auto**	**cmd**
	0		0	1
	MSB			LSB

Bit	Value	Description
auto	0	CAN-CBX module does **not** save the parameters autonomously
	1	CAN-CBX module **saves** the parameters autonomously
cmd	0	CAN-CBX module does **not** save the parameters on command
	1	CAN-CBX module **saves** the parameters on command

Autonomous saving means that the CAN-CBX module stores the storable parameters non-volatile and without a user request.

4.9.13 Restore Default Parameters (0x1011)

INDEX	**0x1011**
Name	*restore default parameters*
Data type	unsigned 32

Via this command the default parameters, as valid when leaving the manufacturer, are restored.
The parameter groups as described below are distinguished.
Every individual setting stored in the EEPROM will be lost.
After a reset the default parameters will be active. The reset of the parameters however must be initiated with a write access to object 0x1011. To write the index a specific signature as shown below must be transmitted.
Reading the index provides information about its parameter restoring capability (refer to (1) for more information).

Index	Sub-index	Description	Value range	Data type	Access mode
0x1011	0	*number_of_entries*	4	unsigned 8	ro
	1	*restore_all_default_parameters* (objects 0x1000 ... 0x9FFF)	no default, write: 0x 64 61 6F 6C (= ASCII: ‘d’ ‘a’ ‘o’ ‘l’), read: 1	unsigned 32	rw
	2	*restore_communication_parameter* (objects 0x1000 ... 0x1FFF)		unsigned 32	rw
	3	*restore_application_parameter* (objects 0x6000 ... 0x9FFF)		unsigned 32	rw
	4	*restore_manufacturer_parameter* (objects 0x2000 ... 0x5FFF)		unsigned 32	rw

Assignment of the variables:

restore all parameters
Restores the default parameters of all objects (if available), which have a read/write (rw) right of access.

restore_communication_parameter
Restores all communication default parameters of those objects (objects 0x1000 ... 0x1FFF, if available, here e.g. 0x1005 ... 0x1029).

restore_application_parameter
Restores all application default parameters of those objects (objekts 0x6000 ... 0x9FFF, if available, here e.g. 0x6*xxx*).

restore_manufacturer_parameter
Loads all manufacturer default parameters of those objects (objects 0x2000 ... 0x5FFF, if available, here e.g. 0x2*xxx*).

Bit 0 of object 0x1011, sub-index 1 is set, i.e. the CAN-CBX module restores the default values initiated by writing the signature ‘load’ (0x64 61 6F 6C, sequence in CAN telegram) in object 0x1011, sub-index 1-4.

On read access to the appropriate sub-index, the CANopen device provides information about its default parameter restoring capability with the following format:

Bit:	31	30 … … 1	0
Content:	reserved		**cmd**
	0		1
	MSB		LSB

Bit	Value	Description
cmd	0	The CAN-CBX-module does **not** restore default parameters
	1	The CAN-CBX-module **restores** the default parameters

4.9.14 COB_ID Emergency Message (0x1014)

INDEX	**0x1014**
Name	*COB-ID emergency object*
Data type	unsigned 32
Default value	0x80 + Node-ID

This object defines the COB-ID of the emergency object (EMCY).

The structure of this object is shown in the following table:

Bit-No.	Value	Meaning
31 (MSB)	0/1	0: EMCY exists / is valid 1: EMCY does not exist / EMCY is not valid
30	0	reserved (always 0)
29	0	always 0 (11-bit ID)
28...11	0	always 0 (29-bit IDs are not supported)
10 ... 0 (LSB)	x	bits 0...10 of COB-ID

The identifier can take values between 0x000...0x7FF.

4.9.15 Inhibit Time EMCY (0x1015)

INDEX	**0x1015**
Name	*inhibit_time_emergency*
Data type	unsigned 16
Access mode	rw
Value range	0 ... 0xFFFF
Default value	0

The *Inhibit Time* for the EMCY message can be defined with this entry. The time is determined as a multiple of 100 µs.

4.9.16 Consumer Heartbeat Time (0x1016)

INDEX	0x1016
Name	*consumer heartbeat time*
Data type	unsigned 32
Default value	0

The heartbeat function can be used for mutual monitoring of the CANopen modules (especially to detect connection failures). Unlike node guarding/life guarding the heartbeat function does not require RTR-Frames.

Function:
A module, the so-called heartbeat producer, cyclically transmits a heartbeat message on the CAN-bus on the node-guarding identifier (see object 0x100E). One or more heartbeat consumers receive the message. The message must be received within the heartbeat time stored on the heartbeat consumer, otherwise a heartbeat event is triggered on the heartbeat-consumer module. A heartbeat event generates a heartbeat error on the CAN-CBX module.

Each module can act as a heartbeat producer and a heartbeat consumer. The CAN-CBX module can represent at most one heartbeat consumer per CAN net.

Index	Sub-index	Description	Value range	Default	Data type	Access mode
0x1016	0	*Number_of_entries*	1	1	unsigned 8	ro
	1	*Consumer_heartbeat_time*	0 ... 0x007F FFFF	0	unsigned 32	rw

Meaning of the variable *consumer-heartbeat_time_x*:

	consumer-heartbeat_time_x		
Bit	31 24	23 16	15 0
Assignment	reserved (always '0')	*Node-ID* (unsigned 8)	*heartbeat_time* (unsigned 16)

Node-ID Node-Id of the heartbeat producer to be monitored.

heartbeat_time Within this time [ms] the heartbeat producer must transmit the heartbeat on the node-guarding ID, to avoid the transmission of a heartbeat event.
The consumer-heartbeat time of the monitoring module must always be higher than the producer-heartbeat time of the heartbeat-transmitting module.

Example:

consumer-heartbeat_time = 0x0031 03E8

=> *Node-ID* = 0x31 = 49 (decimal)
=> *heartbeat time* = 0x3E8 = 1000 (decimal) => 1 s

4.9.17 Producer Heartbeat Time (0x1017)

INDEX	**0x1017**
Name	*producer heartbeat time*
Data type	unsigned 16
Default value	0 [ms]

The producer heartbeat time defines the cycle time with which the CAN-CBX- module transmits a heartbeat-frame to the node-guarding ID.

If the value of the producer heartbeat time is higher than ‘0’, it is active and stops the node-/ life-guarding (see 4.9.10 page 46).
If the value of the producer-heartbeat-time is set to ‘0’, transmitting heartbeats by this module is stopped.

Index	Sub-index	Description	Value range	Default	Data type	Access mode
0x1017	0	*producer-heartbeat_time*	0 ... 0xFFFF	0	unsigned 16	rw

producer-heartbeat_time Cycle time [ms] of heartbeat producer to transmit the heartbeat on the node-guarding ID (see object 0x100E).
The consumer-heartbeat time of the monitoring module must always be higher than the producer-heartbeat time of the heartbeat-transmitting module.

4.9.18 Identity Object (0x1018)

INDEX	0x1018
Name	*identity object*
Data type	unsigned 32
Default value	see below

This object contains general information to the CAN module.

Index	Sub-index	Description	Value range	Default	Data type	Access mode
0x1018	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*vendor_id*	0 ... 0xFFFF FFFF	0x0000 0017	unsigned 32	ro
	2	*product_code*	0 ... 0xFFFF FFFF	(see page 36 for product specific value)	unsigned 32	ro
	3	*revision_number*	0 ... 0xFFFF FFFF	0x10	unsigned 32	ro
	4	*serial_number*	0 ... 0xFFFF FFFF	-	unsigned 32	ro

Description of the variables:

vendor_id This variable contains the esd-vendor-ID. This is always 0x0000 0017.

product_code Here the esd-article number of the product is stored.
The nibbles of the long words have the following meaning:

product_code = 0x*abcd efgh*

a: 1... article number beginning with character “K”
2....article number beginning with character “C”
bcde: 4-digit hex number, which is interpreted as the integer part of the decimal number (on the left of the decimal point).
f: currently not evaluated
gh: 2-digit hex number, which is interpreted as the fraction part of the decimal number (on the right of the decimal point).

Example: ‘0x2301 0004’ corresponds to article number ‘C.3010.04’ (CAN-CBX-DIO8/2).

revision_number Here the software version is stored. In accordance with (1) the two MSB represent the revision numbers of the major changes and the two LSB show the revision number of minor corrections or changes.

revision_no	
major_revision_no	*minor_revision_no*
31 … … 16	15 … … 0
MSB	LSB

serial_number Here the serial number of the hardware is read. The first two characters of the serial number are letters which designate the manufacturing lot.

The following characters represent the actual serial number.

In the two MSB of *serial_no* the letters of the manufacturing lot are coded. They each contain the ASCII-code of the letter with the MSB set ‘1’ in order to be able to differentiate between letters and numbers:
(ASCII-Code) + 0x80 = read_byte

The two last significant bytes contain the number of the module as BCD-value.

Example:
If the value ‘0xC1C2 0105’ is being read, this corresponds to the hardware-serial number code ‘AB 0105’. This value must correspond to the serial number of the module.

See chapter ‘Overview Communication Profile Objects / Product-Specific Values’ on page 36 for the article number and the serial number of your module.

4.9.19 Synchronous Counter Overflow Value (0x1019)

INDEX	**0x1019**
Name	*Synchronous_Counter_Overflow*
Data type	unsigned 8
Default value	0

This object defines whether a counter is mapped into the SYNC message or not and further the highest value the counter can reach.

The value range of the object is described in the following table:

Value	Description
0	The SYNC message shall be transmitted as a CAN message of data length ‘0’.
1	reserved
2...240	The SYNC message shall be transmitted as a CAN message of data length ‘1’. The first data byte contains the counter.
241...255	reserved

4.9.20 Verify Configuration (0x1020)

INDEX	0x1020
Name	*verify configuration*
Data type	unsigned 32
Default value	see below

In this object the date and the time of the last configuration can be stored to check later whether the configuration complies with the expected configuration or not.
The content of the parameters is not evaluated by the firmware.

Index	Sub-index	Description	Value range	Default	Data type	Access mode
0x1020	0	*no_of_entries*	2	2	unsigned 8	ro
	1	*configuration_date*	0 ... 0xFFFF FFFF	0	unsigned 32	rw
	2	*configuration_time*	0 ... 0xFFFF FFFF	0	unsigned 32	rw

Parameter Description:

configuration_date	Date of the last configuration of the module. The value is defined in number of days since the 01.01.1984.
configuration_time	Time in ms since midnight at the day of the last configuration.

4.9.21 Error Behaviour Object (0x1029)

INDEX	0x1029
Name	*error behaviour object*
Data type	unsigned 8
Default value	see below

If an error event occurs (such as heartbeat error), the module changes into the status which has been defined in variable *communication_error* or *output_error*.

Index	Sub-index	Description	Value range	Default	Data type	Access mode
0x1029	0	*no_of_error_classes*	1	6	unsigned 8	ro
	1	*communication_error*	0...2	0	unsigned 8	rw
	2	*output_error*	0...2	0	unsigned 8	rw
	3	*input_error*	0...2	1	unsigned 8	rw
	4	*VIO_extreme_low*	0...2	1	unsigned 8	rw
	5	*VIO_low_warning*	0...2	1	unsigned 8	rw
	6	*VIO_high_warnig*	0...2	1	unsigned 8	rw

Meaning of the variables:

Variable	Meaning
no_of_error_classes	number of error-classes (here always ‘1’)
communication_error	heartbeat/lifeguard error and *Bus off*
output_error	output error
input_error	input error (not supported at the moment)
VIO_extreme_low	output driver power supply error: VIO < 5 V
VIO_low_warning	output driver power supply error: VIO is lower than the value defined in object 0x2300, sub-index 2
VIO_high_warning	output driver power supply error: VIO is higher than the value defined in object 0x2300, sub-index 3.

The module can enter the following states if an error occurs.

Variable	Module state
0	pre-operational (only if the current state is operational)
1	no state change
2	stopped

4.9.22 Receive PDO Communication Parameter (0x1400)

These objects define the parameters of receive PDOs (RPDOs).
Object ‘Receive PDO Communication Parameter 0x1400’ defines the parameters of a receive PDO (Rx-PDO).

INDEX	**0x1400**
Name	*receive PDO parameter*
Data type	PDOCommPar

Index	Sub-index	Description	Value range	Default	Data type	Access mode
0x1400	0	*no_of_entries*	2	2	unsigned 8	ro
	1	*COB_ID used by PDO1*	1 ... 0x8000 07FF	0x200 + Node-ID	unsigned 32	rw
	2	*transmission type*	0 ... 0xFF	255	unsigned 8	rw

All *transmission types* are supported.

4.9.23 Receive PDO Mapping Parameter 0x1600

Object ‘Receive PDO Mapping Parameter 0x1600 defines the assignment of receive data to Rx-PDOs.

INDEX	0x1600
Name	*receive PDO mapping*
Data type	PDO Mapping

The following table shows the assignment of the Receive PDO Mapping parameters for the default configuration:

Index	Sub-index	Description	Value range	Default	Data type	Access mode
0x1600	0	*no_of_mapped_application_objects_in_PDO*	1 ... 8	1	unsigned 8	rw
	1	*1st_application_object*	0x0005 0008, 0x6200 0108	0x6200 0108	unsigned 32	rw
	2	*2nd_application_object*		0x0005 0008	unsigned 32	rw
	3	*3rd_application_object*		0x0005 0008	unsigned 32	rw
	4	*4th_application_object*		0x0005 0008	unsigned 32	rw
	5	*5th_application_object*		0x0005 0008	unsigned 32	rw
	6	*6th_application_object*		0x0005 0008	unsigned 32	rw
	7	*7th_application_object*		0x0005 0008	unsigned 32	rw
	8	*8th_application_object*		0x0005 0008	unsigned 32	rw

Parallel connection of up to 8 CAN-CBX-DIO8/2-Modules:

In general, there is the possibility to access several CAN-CBX-DIO8/2 modules synchronously. The CAN-CBX-DIO8/2 modules can be configured so that up to 8 CAN-CBX-DIO8/2 modules can be addressed with a single CAN frame simultaneously. Therefore all 64 digital outputs can be set simultaneously. The Receive COB-IDs of the CAN-CBX-DIO8/2 modules must be set to the same value using object 0x1400. Refer to (1) object 0x1600 also.
Subindex 0 contains the number of valid entries within the mapping record (see following table).
The number of valid entries shall be the same for all modules.

Value	Description
0	Mapping disabled
1	Subindex 1 valid
2	Subindex 1 and 2 valid
3	Subindex from 1 - 3 valid
:	:
8	Subindex from 1 - 8 valid

Subindices from 1 to 8 contain the information of the mapped application objects. The entry describes the content of the PDO by their index, subindex and length.
For the CAN-CBX-DIO8/2 module the value 0x6200 0108 (index 0x6200, subindex 1 and length 8) may only be contained once. The other subindices contain the value 0x0005 0008 as placeholder

for the so-called dummy mapping.

Example:
There are three CAN-CBX-DIO8/2 modules which shall be addressed via RPDO-Mapping simultaneously.
Therefore, the RPDO COB-IDs of the modules must be configured to the same value. For further information refer to s CiA 301 (1).

The object 0x1600 must be configured differently for the three modules. For the first module the *application_object* must be contained in subindex 1, for the second module in subindex 2 and for the third module in subindex 3 (see Figure 12). For a higher number of modules (here up to 8 modules) the entries must be continued respectively.

In the preceding or following subindices (here subindex 1 and 2) which are not used, the value 0x0005 0008 must be entered for the dummy-mapping. Subindex 0 contains the number of subindices, according to the number of the valid objects.

Entry in object 0x1600 for module 3:

Index [Hex]	Sub-index	Description	Value
1600	0	*no_of_mapped_application_ objects_in_PDO*	0x03
	1	*1st_application_object*	0x0005 0008
	2	*2nd_application_object*	0x0005 0008
	3	*3rd_application_object*	0x6200 0108

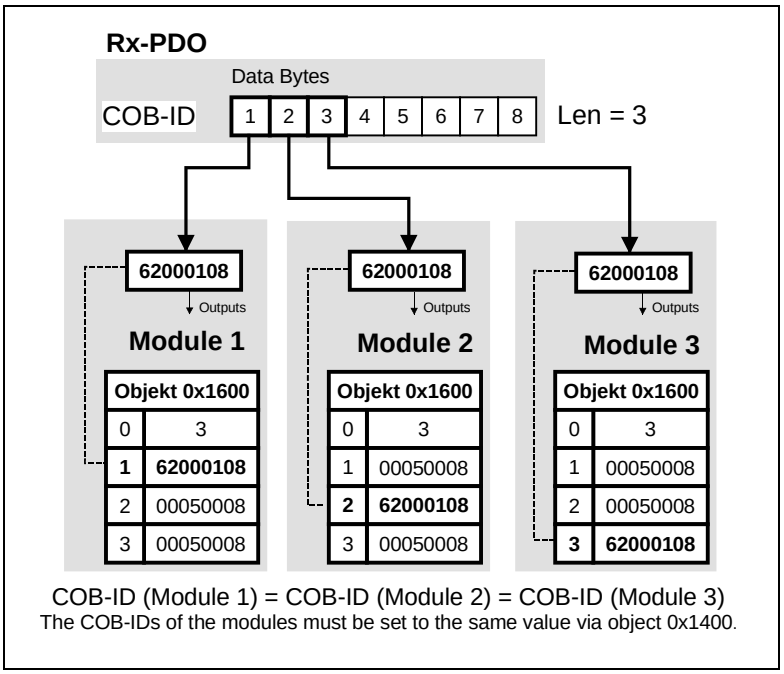


Figure 12: Example for the Rx-PDO mapping with three CAN-CBX-DIO8/2 modules

4.9.24 Object Transmit PDO1 Communication Parameter 0x1800

This object defines the parameters of a transmit PDO1.

INDEX	**0x1800**
Name	*transmit PDO1 parameter*
Data type	PDOCommPar

Index	Sub-index	Description	Value range	Default	Data type	Access mode
0x1800	0	*number_of_entries*	0 ... 0xFF	5	unsigned 8	ro
	1	*COB-ID used by PDO1*	1 ... 0x800007FF	0x**0**000 0000 +0x180 +Node-ID	unsigned 32	rw
	2	*transmission type*	0 ... 0xFF	255_d	unsigned 8	rw
	3	*inhibit time*	0 ... 0x FFFF	0	unsigned 16	rw
	4	*reserved*	0 ... 0xFF	0	unsigned 8	ro
	5	*event timer*	0 ... 0xFFFF	0	unsigned 16	rw

The *transmission types* 0, 1...240, 252, 253 and 255 are supported.

Per default the highest bit of the parameter *COB-ID used by_PDO1* is **not set**:

0x**0**000 0000 +0x180 +Node-ID

Per default this object is **valid**.

4.9.25 Object Transmit PDO2 Communication Parameter 0x1801

This object defines the parameters of a transmit PDO2.

INDEX	**0x1801**
Name	*transmit PDO2 parameter*
Data type	PDOCommPar

Index	Sub-index	Description	Value range	Default	Data type	Access mode
0x1801	0	*number_of_entries*	0 ... 0xFF	5	unsigned 8	ro
	1	*COB-ID used by PDO2*	1... 0x8000 07FF	0x8000 0000 +0x280 +Node-ID	unsigned 32	rw
	2	*transmission type*	0 ... 0xFF	255_d	unsigned 8	rw
	3	*inhibit time*	0 ... 0xFFFF	0	unsigned 16	rw
	4	*reserved*	0 ... 0xFF	0	unsigned 8	ro
	5	*event timer*	0 ... 0xFFFF	0	unsigned 16	rw

The *transmission types* 0, 1...240, 252, 253 and 255 are supported.

Per default the highest bit of the parameter *COB-ID used by_PDO2* **is set**:

0x**8**000 0000 +0x280 +Node-ID

Per default this object is **not valid**.

4.9.26 Transmit PDO1 Mapping Parameter 0x1A00

The object ‘Transmit PDO1 Mapping Parameter 0x1A00’ defines the assignment of transmit data to Tx-PDO1s.

INDEX	**0x1A00**
Name	*transmit PDO1 mapping*
Data type	PDO Mapping

The following table shows the assignment of Transmit PDO1 Mapping parameters:

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x1A00	0	*number_of_mapped objects*	0 ... 0xFF	1	unsigned 8	ro
	1	*object_to_be_mapped_1*	0 ... 0xFFFF FFFF	0x6000 0108	unsigned 32	rw
	2	*object_to_be_mapped_2*	0 ... 0xFFFF FFFF	-	unsigned 32	rw
	3	*object_to_be_mapped_3*	0 ... 0xFFFF FFFF	-	unsigned 32	rw
	4	*object_to_be_mapped_4*	0 ... 0xFFFF FFFF	-	unsigned 32	rw
	5	*object_to_be_mapped_5*	0 ... 0xFFFF FFFF	-	unsigned 32	rw
	6	*object_to_be_mapped_6*	0 ... 0xFFFF FFFF	-	unsigned 32	rw
	7	*object_to_be_mapped_7*	0 ... 0xFFFF FFFF	-	unsigned 32	rw
	8	*object_to_be_mapped_8*	0 ... 0xFFFF FFFF	-	unsigned 32	rw

The Tx-PDO mapping is freely configurable.
In the default setting the digital inputs DI1 - DI8 (object 0x6000, sub-index 1, 8-bit) are mapped.

Objects, that can be mapped (with index, sub-index and length in bit):

6000 01 08	Digital Inputs, *Read_Input_8-bit_DI8-DI1, 8-bit* (object 0x6000, sub-index 1)
6020 01 01 … 6020 08 01	Digital Inputs, *Read_Input_1-bit_DI8-DI1, 1-bit* (object 0x6020, sub-index 1 - 8),
6100 01 10	Digital Inputs, *Read_Input_16-bit_DI8-DI1, 16-bit* (object 0x6100, sub-index 1)
6120 01 20	Digital Inputs, *Read_Input_32-bit_DI8-DI1, 32-bit* (object 0x6120, sub-index 1)
2402 01 10 ... 2402 08 10	*Counter_Value_1 - Counter_Value_8*, 16-bit (obj. 0x2402, sub-index 1-8)
2403 01 20 ... 2403 08 20	*Counter_Value_1 - Counter_Value_8*, 32-bit (obj. 0x2403, sub-index 1-8)
0005 00 08	“Dummy Mapping”

INFORMATION
In the state of delivery, the PDO1-Mapping is valid per default.
The Transmit PDO1-Mapping parameters cannot be modified if the PDO1 is valid.
To change the value of the parameter *object_to_be_mapped,* the value of the parameter COB-*ID_used_by_PDO1* in object 0x1800 (4.9.24) must be set to *not valid*.

4.9.27 Transmit PDO2 Mapping Parameter 0x1A01

The object ‘Transmit PDO2 Mapping Parameter 0x1A01’ defines the assignment of transmit data to Tx-PDO2s.

INDEX	**0x1A01**
Name	*transmit PDO2 mapping*
Data type	PDO Mapping

The following table shows the assignment of Transmit PDO2 Mapping parameters:

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x1A01	0	*number of entries*	0 ... 0xFF	4	unsigned 8	rw
	1	*object_to_be_mapped_1*	0 ... 0xFFFFFFFF	0x2402 01 10	unsigned 32	rw
	2	*object_to_be_mapped_2*	0 ... 0xFFFFFFFF	0x2402 02 10	unsigned 32	rw
	3	*object_to_be_mapped_3*	0 ... 0xFFFFFFFF	0x2402 03 10	unsigned 32	rw
	4	*object_to_be_mapped_4*	0 ... 0xFFFFFFFF	0x2402 04 10	unsigned 32	rw
	5	*object_to_be_mapped_5*	0 ... 0xFFFFFFFF	-	unsigned 32	rw
	6	*object_to_be_mapped_6*	0 ... 0xFFFFFFFF	-	unsigned 32	rw
	7	*object_to_be_mapped_7*	0 ... 0xFFFFFFFF	-	unsigned 32	rw
	8	*object_to_be_mapped_8*	0 ... 0xFFFFFFFF	-	unsigned 32	rw

The Tx-PDO mapping is freely configurable.
The number of the objects, that can be mapped depends on their length. Maximum 8 byte can be transferred. From this follows that maximal four 16-bit values (see object 0x2402) or two 32-bit values (see object 0x2403) can be transferred in the PDO. Combinations are possible.
In the default setting the digital inputs DI1 - DI8 (object 0x6000, sub-index 1) are mapped.

Objects, that can be mapped (with index, sub-index and length in bit):

6000 01 08	Digital Inputs, *Read_Input_DI8-DI1, 8-bit* (object 0x6000, sub-index 1)
6020 01 01 … 6020 08 01	Digital Inputs, *Read_Input_1-bit_DI8-DI1, 1-bit* (object 0x6020, sub-index 1 - 8),
6100 01 10	Digital Inputs, *Read_Input_16-bit_DI8-DI1, 16-bit* (object 0x6100, sub-index 1)
6120 01 20	Digital Inputs, *Read_Input_32-bit_DI8-DI1, 32-bit* (object 0x6120, sub-index 1)
2402 01 10 ... 2402 08 10	*Counter_Value_1* - *Counter_Value_8*, 16-bit (obj. 0x2402, sub-index 1-8)
2403 01 20 ... 2403 08 20	*Counter_Value_1* - *Counter_Value_8*, 32-bit (obj. 0x2403, sub-index 1-8)
0005 00 08	“Dummy Mapping”

INFORMATION
In the state of delivery the PDO2-Mapping is not valid per default.
If the PDO2 is not valid the Transmit PDO2-Mapping parameters can be modified.
To activate the PDO2-Mapping the value of the parameter COB-*ID_used_by_PDO2* in object 0x1801 (page 64) must be set to *valid*.

The following table shows an example of an assignment of the Transmit PDO2 mapping parameters:

Index	Sub-index	Description	Value	Data type
0x1A01	0	*number of entries in PDO2*	4	unsigned 8
	1	*object_to_be_mapped_1*	0x0005 00 08	unsigned 8
	2	*object_to_be_mapped_2*	0x6000 01 08	unsigned 8
	3	*object_to_be_mapped_3*	0x2403 05 20	unsigned 32
	4	*object_to_be_mapped_4*	0x2402 01 10	unsigned 16

4.9.28 NMT Startup (0x1F80)

INDEX	**0x1F80**
Name	*NMT startup*
Data type	unsigned 32
Default value	2

The NMT startup is implemented to be able to start CANopen nodes in environments without NMT-manager.
Via NMT startup the auto startup of a CANopen node can be switched on or off.
Further features of the parameters *NMT startup* are currently not supported.

The value range of the object is described in the following table:

Value	**Meaning**
0x0000 0002	Auto startup disabled (default)
0x0000 0008	Auto startup enabled
all other values	reserved

4.9.29 Self-Starting Nodes Timing Parameters (0x1F91)

INDEX	**0x1F91**
Name	*Self-starting nodes timing parameters*
Data type	unsigned 16
Default value	100 ms

Index	Sub-index	Description	Value range	Default	Data type	Access mode
0x1F91	0	*number_of_entries*	1	1	unsigned 8	ro
	1	*NMT manager detection timeout*	0 ... 0xFFFF	0x0064	unsigned 16	rw

Sub-index 1 of this object contains the timeout in [ms] between the change from “preoperational” > “operational”. In default it is 100 ms.

The sub-indices 2 and 3 of this object are not supported.

4.10 Device Profile Area

4.10.1 Implemented Objects 0x6000 – 0x6207

Index	Name	Data type
0x6000	*Read Input 8-bit*	unsigned 8
0x6002	*Polarity Input 8-bit*	unsigned 8
0x6003	*Filter Constant Input 8-bit*	unsigned 8
0x6005	*Global Interrupt Enable Digital*	boolean
0x6006	*Interrupt Mask Any Change 8-bit*	unsigned 8
0x6007	*Interrupt Mask Low-to-High 8-bit*	unsigned 8
0x6008	*Interrupt Mask High-to-Low 8-bit*	unsigned 8
0x6020	*Read Input 1-bit*	boolean
0x6030	*Polarity Input 1 Bit*	boolean
0x6038	*Filter Constant Input Bit 1 Bit*	boolean
0x6050	*Interrupt Mask Input Bit Any Change 1 Bit*	boolean
0x6060	*Interrupt Mask Input Low To High 1 Bit*	boolean
0x6070	*Interrupt Mask Input High To Low 1 Bit*	boolean
0x6100	*Read Input 16 Bit*	unsigned 16
0x6102	*Polarity Input 16 Bit*	unsigned 16
0x6103	*Filter Constant Input 16 Bit*	unsigned 16
0x6106	*Interrupt Mask Any Change 16 Bit*	unsigned 16
0x6107	*Interrupt Mask Low to High 16 Bit*	unsigned 16
0x6108	*Interrupt Mask High to Low 16 Bit*	unsigned 16
0x6120	*Read Input 32 Bit*	unsigned 32
0x6122	*Polarity Input 32 Bit*	unsigned 32
0x6123	*Filter Constant Input 32 Bit*	unsigned 32
0x6126	*Interrupt Mask Input Bit Any Change 32 Bit*	unsigned 32
0x6127	*Interrupt Mask Input Low To High 32 Bit*	unsigned 32
0x6128	*Interrupt Mask Input High To Low 32 Bit*	unsigned 32
0x6200	*Write Output 8-bit*	unsigned 8
0x6202	*Polarity Output 8-bit*	unsigned 8
0x6206	*Error Mode Output 8-bit*	unsigned 8
0x6207	*Error Value Output 8-bit*	unsigned 8
0x6208	*Filter Mask Output 8 Bit*	unsigned 8
0x6220	*Write Output 1 Bit*	boolean
0x6240	*Polarity Output 1 Bit*	boolean
0x6250	*Error Mode Output 1 Bit*	boolean
0x6260	*Error Value Output 1 Bit*	boolean
0x6270	*Filter Mask Output 1 Bit*	boolean
0x6300	*Write Output 16 Bit*	unsigned 16
0x6302	*Polarity Output 16 Bit*	unsigned 16
0x6306	*Error Mode Output 16 Bit*	unsigned 16
0x6307	*Error Value Output 16 Bit*	unsigned 16
0x6308	*Filter Mask Output 16 Bit*	unsigned 16
0x6320	*Write Output 32 Bit*	unsigned 32
0x6322	*Polarity Output 32 Bit*	unsigned 32
0x6326	*Error Mode Output 32 Bit*	unsigned 32
0x6327	*Error Value Output 32 Bit*	unsigned 32
0x6328	*Filter Mask Output 32 Bit*	unsigned 32

4.10.2 Interrelation of the Implemented Objects of the Digital Inputs

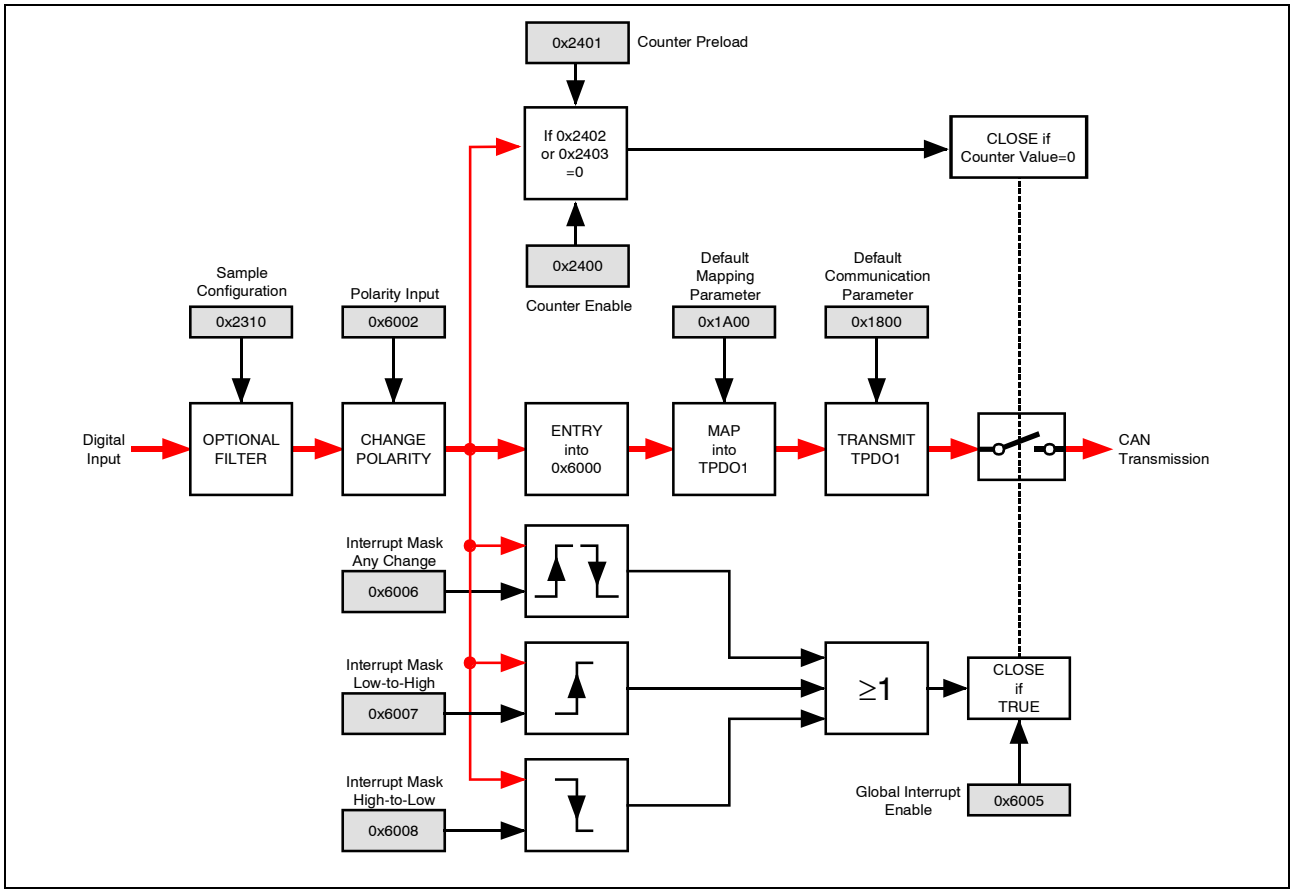


Figure 13: Overview of the objects for the digital inputs (Example 8-bit objects)

The overview is shown as an example with 8-bit objects. The 1-bit, 16-bit or 32-bit objects for the digital inputs, as shown in Table 9, can be used alternatively.

Name	8-bit object (Example)	Objects that can be used as an alternative		
		1-bit objects	16-bit objects	32-bit objects
Read Input	0x6000	0x6020	0x6100	0x6120
Polarity Input	0x6002	0x6030	0x6102	0x6122
Global Interrupt Enable	0x6005	-	-	-
Interrupt Mask Any Change	0x6006	0x6050	0x6106	0x6126
Interrupt Mask Low-to-High	0x6007	0x6060	0x6107	0x6127
Interrupt Mask High-to-Low	0x6008	0x6070	0x6108	0x6128

Table 9: Alternative objects for digital inputs

4.10.3 Interrelation of the Implemented Objects of the Digital Outputs

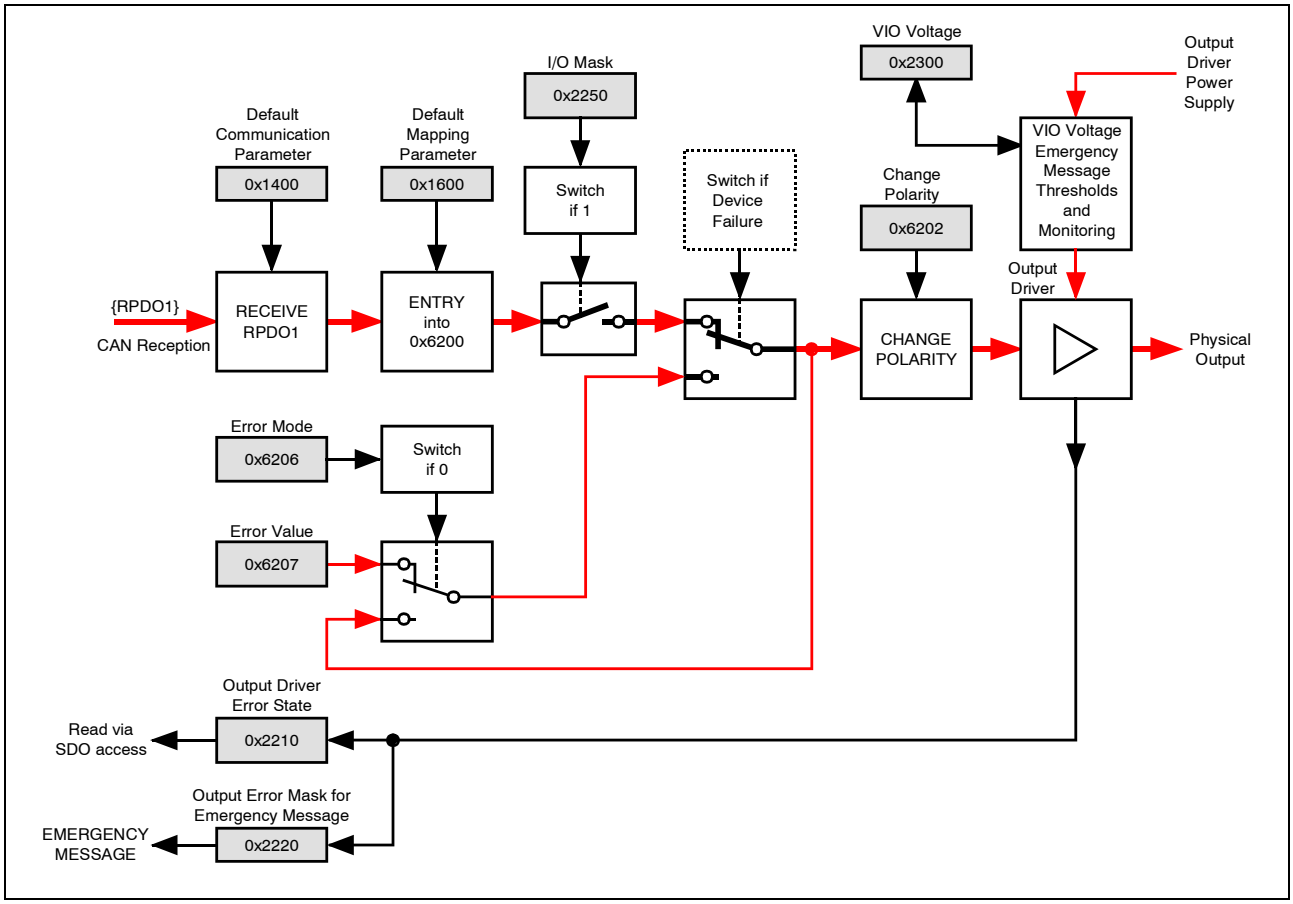


Figure 14: Overview of the objects for the digital outputs (Example 8-bit objects)

The overview is shown as an example with 8-bit objects. The 1-bit, 16-bit or 32-bit objects for the digital outputs, as shown in Table 10, can be used alternatively

Name	8-bit object (Example)	Objects that can be used as an alternative		
		1-bit objects	16-bit objects	32-bit objects
Write Output	0x6200	0x6220	0x6300	0x6320
Change Polarity Output	0x6202	0x6240	0x6302	0x6322
Error Mode Output	0x6206	0x6250	0x6306	0x6326
Error Value Output	0x6207	0x6260	0x6307	0x6327
Filter Mask Output	0x6208	0x6270	0x6308	0x6328

Table 10: Alternative objects for digital outputs

4.10.4 Read Input 8-Bit (0x6000)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6000	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*read_input_8-bit_DI8-DI1*	0 ... 0xFF	-	1	unsigned 8	ro

Assignment of the variables *read_input_8-bit_DI8-DI1*:

Index: 0x6000, Sub-index: 1

Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

An input bit is read as ‘1’, if the corresponding input is active, i.e. voltage is ‘on’ (if the according bit in object 0x6002 ‘Polarity Input 8-Bit’ is set to ‘0’).

Bit value *DIx*	Input *DIx*
1	Input *DIx* is active
0	Input *DIx* is not active

(x = 1 … 8)

4.10.5 Polarity Input 8-Bit (0x6002)

With this object the polarity of the 8 digital inputs can be set individually.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6002	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*polarity_input_8-bit_DI8-DI1*	0 ... 0xFF	0	0	unsigned 8	ro

Assignment of the parameter *polarity_input_8-bit_DI8-DI1*:

Index: 0x6002, Sub-index: 1

Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

Bit value *DIx*	Input *DIx*
1	Input *DIx* is inverted
0	Input *DIx* is not inverted

(x = 1 … 8)

4.10.6 Filter Constant Input 8-Bit (0x6003)

With this parameter the digital filter can be individually activated for each input. Via the object 0x2310 the sample rate of the digital inputs is defined.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6003	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*filter constant_8-bit_DI8-DI1*	0 ... 0xFF	0	0	unsigned 8	ro

Assignment of the parameter *filter constant_8-bit_DI8-DI1*:

Index: 0x6003, Sub-index: 1

Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

Bit value *DIx*	Input *DIx*
1	filter of *DIx* enabled
0	filter of *DIx* disabled

(x = 1 … 8)

4.10.7 Global Interrupt Enable/Disable

Index	Sub-index	Description	Value range [Boolean]	Default	PDO Mapping	Data type	Access mode
0x6005	0	*Global Interrupt Enable-Bit*	True, False	1	0	boolean	rw

Global Interrupt Enable-Bit	**Value**	**Description**
true	1	Interrupts enabled
false	0	Interrupts disabled

4.10.8 Interrupt Mask Any Change 8-Bit (0x6006)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6006	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*IRQ-mask_any_8-bit_DI8-DI1*	0 ... 0xFF	0xFF	0	unsigned 8	rw

Assignment of parameter *IRQ-mask_any_8-bit_DI8-DI1*:

Index: 0x6006, Sub-index: 1

Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

This object determines, which input port lines shall activate an interrupt by **rising or falling edge** detection.

Bit-value *DIx*	Interrupt *DIx*-enable
0	interrupt of *DIx* disabled
1	interrupt of *DIx* enabled

(x = 1 … 8)

4.10.9 Interrupt Mask Low to High 8-Bit (0x6007)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6007	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*IRQ-mask_LH_8-bit_DI8-DI1*	0 ... 0xFF	0	0	unsigned 8	rw

Assignment of the parameter *IRQ-mask_LH_8-bit_DI8-DI1*:

Index: 0x6007, Sub-index: 1

Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

For activated IRQ an interrupt is generated by **rising edge** detection of the input signal.

Bit-value *DIx*	Interrupt *DIx*-enable
0	interrupt of *DIx* disabled
1	interrupt of *DIx* enabled

(x = 1 … 8)

4.10.10 Interrupt Mask High to Low 8-Bit (0x6008)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6008	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*IRQ-mask_HL_8_DI8-DI1*	0 ... 0xFF	0	0	unsigned 8	rw

Assignment of the parameter *IRQ-mask_HL_8-bit_DI8-DI1*:

Index: 0x6008, Sub-index: 1

Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

For activated IRQ an interrupt is generated by **falling edge** detection of the input signal.

Bit-value *DIx*	Interrupt *DIx* enable
0	interrupt of *DIx* disabled
1	interrupt of *DIx* enabled

(x = 1 … 8)

4.10.11 Read Input 1-Bit (0x6020)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6020	0	*no_of_entries*	-	8	0	unsigned 8	ro
	1	*read_input_1-bit_DI1*	true, false	-	1	boolean	ro
	2	*read_input_1-bit_DI2*	true, false	-	1	boolean	ro
	3	*read_input_1-bit_DI3*	true, false	-	1	boolean	ro
	4	*read_input_1-bit_DI4*	true, false	-	1	boolean	ro
	5	*read_input_1-bit_DI5*	true, false	-	1	boolean	ro
	6	*read_input_1-bit_DI6*	true, false	-	1	boolean	ro
	7	*read_input_1-bit_DI7*	true, false	-	1	boolean	ro
	8	*read_input_1-bit_DI8*	true, false	-	1	boolean	ro

Assignment of the variables *read_input_1-bit_DIx*:
(x = 1 …8)

read_input_1-bit_DIx	Value	Description
true	1	Input *DIx* is active
false	0	Input *DIx* is not active

An input (DIx) is read as ‘true’, if the corresponding input is active, i.e. voltage is ‘on’ (if the according bit in object 0x6030 ‘Polarity Input 1-Bit’ is set to ‘0’).

4.10.12 Polarity Input 1-Bit (0x6030)

With this object the polarity of the 8 digital inputs can be set individually.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6030	0	*no_of _entries*	-	8	-	unsigned 8	ro
	1	*polarity_input_1-bit_DI1*	true, false	0	-	boolean	rw
	2	*polarity_input_1-bit_DI2*	true, false	0	-	boolean	rw
	3	*polarity_input_1-bit_DI3*	true, false	0	-	boolean	rw
	4	*polarity_input_1-bit_DI4*	true, false	0	-	boolean	rw
	5	*polarity_input_1-bit_DI5*	true, false	0	-	boolean	rw
	6	*polarity_input_1-bit_DI6*	true, false	0	-	boolean	rw
	7	*polarity_input_1-bit_DI7*	true, false	0	-	boolean	rw
	8	*polarity_input_1-bit_DI8*	true, false	0	-	boolean	rw

Assignment of the parameter *polarity_input_1-bit_DIx*:
(x = 1 … 8)

polarity_input_1-bit_DIx	Value	Description
true	1	Input *DIx* is inverted
false	0	Input *DIx* is not inverted

(x = 1 …8)

4.10.13 Filter Constant Input 1-Bit (0x6038)

With this parameter the digital filter can be individually activated for each input port.
Via the object 0x2310 the sample rate of the digital inputs is defined.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6038	0	*no_of _entries*	-	8	-	unsigned 8	ro
	1	*filter_constant_input_1-bit_DI1*	true, false	0	-	boolean	rw
	2	*filter_constant_input_1-bit_DI2*	true, false	0	-	boolean	rw
	3	*filter_constant_input_1-bit_DI3*	true, false	0	-	boolean	rw
	4	*filter_constant_input_1-bit_DI4*	true, false	0	-	boolean	rw
	5	*filter_constant_input_1-bit_DI5*	true, false	0	-	boolean	rw
	6	*filter_constant_input_1-bit_DI6*	true, false	0	-	boolean	rw
	7	*filter_constant_input_1-bit_DI7*	true, false	0	-	boolean	rw
	8	*filter_constant_input_1-bit_DI8*	true, false	0	-	boolean	rw

Assignment of the parameter *polarity_input_1-bit_DIx*:
(x = 1 … 8)

filter_constant_input_1-bit_DIx	Value	Description
true	1	Filter of digital input x is enabled
false	0	Filter of digital input x is disabled

4.10.14 Interrupt Mask Input Bit Any Change 1-Bit (0x6050)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6050	0	*no_of _entries*	-	8	-	unsigned 8	ro
	1	*IRQ-mask_any_1-bit_DI1*	true, false	1	-	boolean	rw
	2	*IRQ-mask_any_1-bit_DI2*	true, false	1	-	boolean	rw
	3	*IRQ-mask_any_1-bit_DI3*	true, false	1	-	boolean	rw
	4	*IRQ-mask_any_1-bit_DI4*	true, false	1	-	boolean	rw
	5	*IRQ-mask_any_1-bit_DI5*	true, false	1	-	boolean	rw
	6	*IRQ-mask_any_1-bit_DI6*	true, false	1	-	boolean	rw
	7	*IRQ-mask_any_1-bit_DI7*	true, false	1	-	boolean	rw
	8	*IRQ-mask_any_1-bit_DI8*	true, false	1	-	boolean	rw

This object determines, which input shall activate an interrupt by **rising or falling edge** detection

Assignment of the parameter *IRQ-mask_any_1-bit_DIx*:
(x = 1 … 8)

IRQ-mask_any_1-bit_DIx	Value	Description
true	1	interrupt of DIx enabled
false	0	interrupt of DIx disabled

4.10.15 Interrupt Mask Low to High 1-Bit (0x6060)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6060	0	*no_of _entries*	-	8	-	unsigned 8	ro
	1	*IRQ-mask_LH_1-bit_DI1*	true, false	0	-	boolean	rw
	2	*IRQ-mask_LH_1-bit_DI2*	true, false	0	-	boolean	rw
	3	*IRQ-mask_LH_1-bit_DI3*	true, false	0	-	boolean	rw
	4	*IRQ-mask_LH_1-bit_DI4*	true, false	0	-	boolean	rw
	5	*IRQ-mask_LH_1-bit_DI5*	true, false	0	-	boolean	rw
	6	*IRQ-mask_LH_1-bit_DI6*	true, false	0	-	boolean	rw
	7	*IRQ-mask_LH_1-bit_DI7*	true, false	0	-	boolean	rw
	8	*IRQ-mask_LH_1-bit_DI8*	true, false	0	-	boolean	rw

Assignment of the parameter *IRQ-mask_LH_1-bit_DIx*:
For activated *IRQ-mask_LH_1-bit_DIx* an interrupt is generated by **rising edge** detection of the input signal of port DIx.

IRQ-mask_LH_1-bit_DIx	Value	Description
true	1	interrupt on **rising edge** of DIx enabled
false	0	interrupt on **rising edge** of DIx disabled

(x = 1 … 8)

4.10.16 Interrupt Mask High to Low 1-Bit (x6070)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6070	0	*no_of _entries*	-	8	0	unsigned 8	ro
	1	*IRQ-mask_HL_1-bit_DI1*	true, false	0	0	boolean	rw
	2	*IRQ-mask_HL_1-bit_DI2*	true, false	0	0	boolean	rw
	3	*IRQ-mask_HL_1-bit_DI3*	true, false	0	0	boolean	rw
	4	*IRQ-mask_HL_1-bit_DI4*	true, false	0	0	boolean	rw
	5	*IRQ-mask_HL_1-bit_DI5*	true, false	0	0	boolean	rw
	6	*IRQ-mask_HL_1-bit_DI6*	true, false	0	0	boolean	rw
	7	*IRQ-mask_HL_1-bit_DI7*	true, false	0	0	boolean	rw
	8	*IRQ-mask_HL_1-bit_DI8*	true, false	0	0	boolean	rw

Assignment of the parameter *IRQ-mask_HL_1-bit_DIx*:
For activated *IRQ-mask_HL_1-bit_DIx* an interrupt is generated by **falling edge** detection of the input signal of port DIx.

IRQ-mask_HL_1-bit_DIx	Value	Description
true	1	interrupt on **falling edge** of DIx enabled
false	0	interrupt on **falling edge** of DIx disabled

(x = 1 …8)

4.10.17 Read Input 16-Bit (0x6100)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6100	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*read_input_16-bit_-DI8-DI1*	0 ... 0xFFFF	-	1	unsigned 16	ro

Assignment of the variables *read_input_16-bit_DI8-DI1*:

Index: 0x6100, Sub-index: 1

Bit:	**15 …**	**…8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	-		*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

An input bit is read as ‘1’, if the corresponding input is active, i.e. voltage is ‘on’ (if the according bit in object 0x6102 ‘Polarity Input 16-Bit’ is set to ‘0’).

Bit value *DIx*	Input *DIx*
1	Input *DIx* is active
0	Input *DIx* is not active

(x = 1 …8)

4.10.18 Polarity Input 16-Bit (0x6102)

With this object the polarity of the 8 digital inputs can be set individually.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6102	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*polarity_input_16-bit_DI8-DI1*	0 ... 0xFFFF	0	0	unsigned 16	rw

Assignment of the parameter *polarity_input_16-bit_DI8-DI1*:

Index: 0x6102, Sub-index: 1

Bit:	**15 …**	**… 8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	-		*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

Bit value *DIx*	Input *DIx*
1	Input *DIx* is inverted
0	Input *DIx* is not inverted

(x = 1 …8)

4.10.19 Filter Constant Input 16-Bit (0x6103)

With this parameter the digital filter can be individually activated for each input Via the object 0x2310 the sample rate of the digital inputs is defined.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6103	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*filter constant_16-bit_DI8-DI1*	0 ... 0xFFFF	0	0	unsigned 16	rw

Assignment of the parameter *filter constant_16-bit_DI8-DI1*:

Index: 0x6103, Sub-index: 1

Bit:	**15 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	-	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

Bit value *DIx*	Input *DIx*
1	Filter of input *DIx* is enabled
0	Filter of input *DIx* is disabled

(x = 1 …8)

4.10.20 Interrupt Mask Any Change 16-Bit (0x6106)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6106	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*IRQ-mask_any_16-bit_DI8-DI1*	0 ... 0xFFFF	0x00FF	0	unsigned 16	rw

Assignment of parameter *IRQ-mask_any_change_16-bit_DI8-DI1*:

Index: 0x6106, Sub-index: 1

Bit:	**15 …**	**…8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	-		*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

This object determines, which inputs shall activate an interrupt by **rising or falling edge** detection.

Bit-value *DIx*	Interrupt-enable
0	interrupt on DIx disabled
1	interrupt on DIx enabled

(x = 1 … 8)

4.10.21 Interrupt Mask Low to High 16-Bit (0x6107)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6107	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*IRQ-mask_LH_16-bit_DI8-DI1*	0 ... 0xFFFF	0	0	unsigned 16	rw

Assignment of the parameter *IRQ-mask_LH_16-bit_DI8-DI1*:

Index: 0x6107, Sub-index: 1

Bit:	**15 …**	**…8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	-		*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

For activated IRQ an interrupt is generated by **rising edge** detection of the input signal.

Bit-value *DIx*	Interrupt-enable
0	interrupt on **rising edge** of DIx disabled
1	interrupt on **rising edge** of DIx enabled

(x = 1 …8)

4.10.22 Interrupt Mask High to Low 16-Bit (0x6108)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6108	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*IRQ-mask_HL_16-Bit_DI8-DI1*	0 ... 0xFFFF	0	0	unsigned 16	rw

Assignment of the parameter *IRQ-mask_HL_16-bit_DI8-DI1*:

Index: 0x6108, Sub-index: 1

Bit:	**15 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	-	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

For activated IRQ an interrupt is generated by **falling edge** detection of the input signal.

Bit-value *DIx*	Interrupt-enable
0	interrupt on **falling edge** of DIx disabled
1	interrupt on **falling edge** of DIx enabled

(x = 1 … 8)

4.10.23 Read Input 32-Bit (0x6120)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6120	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*read_input_32-bit_-DI8-DI1*	0 ... 0xFFFF FFFF	-	1	unsigned 32	ro

Assignment of the variables *read_input_32-bit_DI8-DI1*:

Index: 0x6120, Sub-index: 1

Bit:	**31… …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	-	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

An input bit is read as '1', if the corresponding input is active, i.e. voltage is 'on' (if the according bit in object 0x6122 'Polarity Input 32-Bit' is set to '0').

Bit value *DIx*	Input *DIx*
1	Input *DIx* is active
0	Input *DIx* is inactive

(x = 1 …8)

4.10.24 Polarity Input 32-Bit (0x6122)

With this object the polarity of the 8 digital inputs can be set individually.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6122	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*polarity_input_32-bit_DI8-DI1*	0 ... 0xFFFF FFFF	0	0	unsigned 32	rw

Assignment of the parameter *polarity_input_32-bit_DI8-DI1*:

Index: 0x6122, Sub-index: 1

Bit:	**31 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	-	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

Bit value *DIx*	Input *DIx*
1	Input *DIx* is inverted
0	Input *DIx* is not inverted

(x = 1 …8)

4.10.25 Filter Constant Input 32-Bit (0x6123)

With this parameter the digital filter can be individually activated for each input. Via the object 0x2310 the sample rate of the digital inputs is defined.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6123	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*filter constant_32-bit_DI8-DI1*	0 ... 0xFFFF FFFF	0	0	unsigned 32	rw

Assignment of the parameter *filter constant_32-bit_DI8-DI1*:

Index: 0x6123, Sub-index: 1

Bit:	**31 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	-	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

Bit value *DIx*	Input
1	filter of input *DIx* is enabled
0	filter of Input *DIx* is disabled

(x = 1 … 8)

4.10.26 Interrupt Mask Any Change 32-Bit (0x6126)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6126	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*IRQ-mask_any_32-bit_DI8-DI1*	0 ... 0xFFFF FFFF	0x0000 00FF	0	unsigned 32	rw

Assignment of parameter *IRQ-mask_any_32-bit_DI8-DI1*:

Index: 0x6126, Sub-index: 1

Bit:	**31 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	-	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

This object determines, which inputs shall activate an interrupt by **rising or falling edge** detection.

Bit-value *DIx*	Interrupt-enable
0	interrupt on DIx disabled
1	interrupt on DIx enabled

(x = 1 …8)

4.10.27 Interrupt Mask Low to High 32-Bit (0x6127)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6127	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*IRQ-mask_LH_32-bit_DI8-DI1*	0 ... 0xFFFF FFFF	0	0	unsigned 32	rw

Assignment of the parameter *IRQ-mask_LH_32-bit_DI8-DI1*:

Index: 0x6127, Sub-index: 1

Bit:	**31… …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	-	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

For activated IRQ an interrupt is generated by **rising edge** detection of the input signal.

Bit-value *DIx*	Interrupt-enable
0	interrupt on **rising edge** of DIx disabled
1	interrupt on **rising edge** of DIx enabled

(x = 1 … 8)

4.10.28 Interrupt Mask High to Low 32-Bit (0x6128)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6128	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*IRQ-mask_HL_32-Bit_DI8-DI1*	0 ... 0xFFFF FFFF	0	0	unsigned 32	rw

Assignment of the parameter *IRQ-mask_HL_32-bit_DI8-DI1*:

Index: 0x6128, Sub-index: 1

Bit:	**31 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Input:	-	*DI8*	*DI7*	*DI6*	*DI5*	*DI4*	*DI3*	*DI2*	*DI1*

For activated IRQ an interrupt is generated by **falling edge** detection of the input signal.

Bit-value *DIx*	Interrupt-enable
0	interrupt on **falling edge** of DIx disabled
1	interrupt on **falling edge** of DIx enabled

(x = 0 … 8))

4.10.29 Write Output 8-Bit (0x6200)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6200	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*write_output_8-bit_DO8-DO1*	0 ... 0xFF	-	1	unsigned 8	rw

Assignment of the variable *write_output_8-bit_DO8-DO1*:

Index: 0x6200, Sub-index: 1

Bit:	7	6	5	4	3	2	1	0
Output:	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

If an output bit is set to ‘1’, the corresponding output is activated, i.e. the output voltage is ‘on’.

Bit value *DOx*	Output
1	output *DOx* is turned on (24 V)
0	output *DOx* is turned off (0 V/open)

(x = 1 …8)

4.10.30 Polarity Output 8-Bit (0x6202)

This object defines the state of the 8 digital outputs. Every output can be inverted individually.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6202	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*polarity output_8-bit_DO8-DO1*	0 ... 0xFF	0	0	unsigned 8	rw

Assignment of the parameter *polarity_output_8-bit_DO8-DO1*:

The parameter determines which digital output can be inverted.

Bit:	7	6	5	4	3	2	1	0
Output:	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

Setting an output bit to ‘1’ inverts the corresponding output.

Bit value *DOx*	Output *DOx*
1	output *DOx* is inverted
0	output *DOx* is not inverted

(x = 1 …8)

4.10.31 Error Mode Output 8-Bit (0x6206)

The error mode is evaluated if the module is in the *Stopped* state. This object in combination with object 0x1029 indicates, whether an output is set to an error-value defined in object 0x6207 in case of an internal device error.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6206	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*error_mode_output_8-bit_DO8-DO1*	0 ... 0xFF	0xFF	0	unsigned 8	rw

Assignment of the parameter *error_mode_output_8-bit_DO8-DO1*:

Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Output:	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

Bit value *DOx*	Output *DOx*
1	Output *DOx* shall take the pre-defined value specified in object 0x6207.
0	Output *DOx* shall be kept unchanged if an error occurs.

(x = 0 …8)

4.10.32 Error Value Output 8-Bit (0x6207)

Provided that the corresponding error mode (object 0x6206) is active, device failures shall set the output to the value configured by this object.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6207	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*error_value_output_8-bit_DO8-DO1*	0 ... 0xFF	0	0	unsigned 8	rw

Assignment of the parameter *error_value_output_8-bit_DO8-DO1*:

Index: 0x6207, Sub-index: 1

Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Output:	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

The parameter contains the value, the outputs shall be set to, in case of fault.

Bit value *DOx*	Output *DOx*
1	Output shall be set to ‘1’ (enabled) in case of fault, if object 0x6206 is enabled.
0	Output shall be set to ‘0’ (disabled) in case of fault, if object 0x6206 is enabled.

4.10.33 Filter Mask Output 8-Bit (0x6208)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6208	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*filter_mask_output_8-bit_DO8-DO1*	0 ... 0xFF	0	0	unsigned 8	rw

Assignment of the parameter *filter_mask_output_8-bit_DO8-DO1*:

Index: 0x6208, Sub-index: 1

Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Output:	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

This parameter contains the filter mask

Bit value *DOx*	Output *DOx*
1	filter of output *DOx* is enabled
0	filter of output *DOx* is disabled

4.10.34 Write Output 1-Bit (0x6220)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6220	0	*no_of_entries*	-	8	0	unsigned 8	ro
	1	*write_output_1-bit_DO1*	true, false	0	1	boolean	rww
	2	*write_output_1-bit_DO2*	true, false	0	1	boolean	rww
	3	*write_output_1-bit_DO3*	true, false	0	1	boolean	rww
	4	*write_output_1-bit_DO4*	true, false	0	1	boolean	rww
	5	*write_output_1-bit_DO5*	true, false	0	1	boolean	rww
	6	*write_output_1-bit_DO6*	true, false	0	1	boolean	rww
	7	*write_output_1-bit_DO7*	true, false	0	1	boolean	rww
	8	*write_output_1-bit_DO8*	true, false	0	1	boolean	rww

Assignment of the variable *write_output_1-bit_DOx*:

write_output_1-bit_DOx	Value	Description
true	1	output *DOx* is turned on (24 V)
false	0	output *DOx* is turned off (0 V/open)

(x = 1 …8)

4.10.35 Polarity Output 1-Bit (0x6240)

This object defines the state of the 8 digital outputs. Every output can be inverted individually.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6240	0	*no_of_entries*	-	8	0	unsigned 8	ro
	1	*polarity output_1-bit_DO1*	true, false	0	0	boolean	rw
	2	*polarity output_1-bit_DO2*	true, false	0	0	boolean	rw
	3	*polarity output_1-bit_DO3*	true, false	0	0	boolean	rw
	4	*polarity output_1-bit_DO4*	true, false	0	0	boolean	rw
	5	*polarity output_1-bit_DO5*	true, false	0	0	boolean	rw
	6	*polarity output_1-bit_DO6*	true, false	0	0	boolean	rw
	7	*polarity output_1-bit_DO7*	true, false	0	0	boolean	rw
	8	*polarity output_1-bit_DO8*	true, false	0	0	boolean	rw

Assignment of the variable *polarity_output_1-bit_DOx*:

The parameter determines which digital output can be inverted.
Setting an output bit to ‘1’ inverts the corresponding output.

***change_polarity_output_1-bit_DOx*:**	**Value**	**Description**
true	1	output DOx is inverted
false	0	output DOx is not inverted

(x = 1 …8)

4.10.36 Error Mode Output 1-Bit (0x6250)

The error mode is evaluated if the module is in the *Stopped* state. This object in combination with object 0x1029 indicates, whether an output is set to an error-value defined in object 0x6260 in case of an internal device error.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6250	0	*no_of_entries*	-	8	0	unsigned 8	ro
	1	*error_mode_output_1-bit_DO1*	true, false	1	0	boolean	rw
	2	*error_mode_output_1-bit_DO2*	true, false	1	0	boolean	rw
	3	*error_mode_output_1-bit_DO3*	true, false	1	0	boolean	rw
	4	*error_mode_output_1-bit_DO4*	true, false	1	0	boolean	rw
	5	*error_mode_output_1-bit_DO5*	true, false	1	0	boolean	rw
	6	*error_mode_output_1-bit_DO6*	true, false	1	0	boolean	rw
	7	*error_mode_output_1-bit_DO7*	true, false	1	0	boolean	rw
	8	*error_mode_output_1-bit_DO8*	true, false	1	0	boolean	rw

Assignment of the parameter *error_mode_output_1-bit_DOx*:

error_mode_output_1-bit_DOx	Value	Description
true	1	output DOx shall take the pre-defined value specified in object 0x6260, sub-index x
false	0	output DOx shall be kept unchanged if an error occurs

(x = 1 … 8)

4.10.37 Error Value Output 1-Bit (0x6260)

Provided that the corresponding error mode of output DIOx (object 0x6250, sub-index x) is active, device failures shall set the output DIOx to the value configured by this object.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6260	0	*no_of_entries*	-	8	0	unsigned 8	ro
	1	*error_value_output_1-bit_DO1*	true, false	0	0	boolean	rw
	2	*error_value_output_1-bit_DO2*	true, false	0	0	boolean	rw
	3	*error_value_output_1-bit_DO3*	true, false	0	0	boolean	rw
	4	*error_value_output_1-bit_DO4*	true, false	0	0	boolean	rw
	5	*error_value_output_1-bit_DO5*	true, false	0	0	boolean	rw
	6	*error_value_output_1-bit_DO6*	true, false	0	0	boolean	rw
	7	*error_value_output_1-bit_DO7*	true, false	0	0	boolean	rw
	8	*error_value_output_1-bit_DO8*	true, false	0	0	boolean	rw

Assignment of the parameter *error_value_output_1-bit_DOx*:
The parameter contains the value, the outputs *DIOx* shall be set to, in case of fault.

error_value_output_1-bit_DOx	Value	Description
true	1	output DOx shall be set to ‘1’ (enabled) in case of fault if *error_mode_output_1-bit_DOx* (object 0x6250, sub-index x) is enabled
false	0	output DOx shall be set to ‘0’ (disabled) in case of fault if *error_mode_output_1-bit_DOx* (object 0x6250, sub-index x) is enabled

(x = 1 … 8)

4.10.38 Filter Mask Output 1-Bit (0x6270)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6270	0	*no_of_entries*	-	8	0	unsigned 8	ro
	1	*filter_mask_output_1-bit_DO1*	true, false	0	0	boolean	rw
	2	*filter_mask_output_1-bit_DO2*	true, false	0	0	boolean	rw
	3	*filter_mask_output_1-bit_DO3*	true, false	0	0	boolean	rw
	4	*filter_mask_output_1-bit_DO4*	true, false	0	0	boolean	rw
	5	*filter_mask_output_1-bit_DO5*	true, false	0	0	boolean	rw
	6	*filter_mask_output_1-bit_DO6*	true, false	0	0	boolean	rw
	7	*filter_mask_output_1-bit_DO7*	true, false	0	0	boolean	rw
	8	*filter_mask_output_1-bit_DO8*	true, false	0	0	boolean	rw

Assignment of the parameter *filter_mask_output_1-bit_DOx*:
The parameter contains the filter mask of output DIOx.

filter_mask_output_1-bit_DOx	Value	Description
true	1	filter of output DOx is enabled
false	0	filter of output DOx is disabled

(x = 1 … 8)

4.10.39 Write Output 16-Bit (0x6300)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6300	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*write_output_16-bit_DO8-DO1*	0 ... 0xFFFF	0	1	unsigned 16	rww

Assignment of the variable *write_output_16-bit_DO8-DO1*:

Index: 0x6300, Sub-index: 1

Bit:	**15 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Output:	-	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

If an output bit is set to ‘1’, the corresponding output is activated, i.e. the output voltage is ‘on’.

Bit value *DOx*	Output
1	output *DOx* is turned on (24 V)
0	output *DOx* is turned off (0 V/open)

(x = 1 …8)

4.10.40 Polarity Output 16-Bit (0x6302)

This object defines the state of the 8 digital outputs. Every output can be inverted individually.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6302	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*polarity output_16-bit_DO8-DO1*	0 ... 0xFFFF	0	0	unsigned 16	rw

Assignment of the parameter *change_polarity_output_16-bit_DO8-DO1*:

The parameter determines which digital output can be inverted.

Index: 0x6302, Sub-index: 1

Bit:	**15 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Content:	-	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

Bit value *DOx*	Output
1	output *DOx* is inverted
0	output *DOx* is not inverted

(x = 1 …8)

4.10.41 Error Mode Output 16-Bit (0x6306)

The error mode is evaluated if the module is in the *Stopped* state. This object in combination with object 0x1029 indicates, whether an output is set to an error-value defined in object 0x6307 in case of an internal device error.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6306	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*error_mode_output_16-bit_ DO8-DO1*	0 ... 0xFFFF	0x00FF	0	unsigned 16	rw

Assignment of the parameter *error_mode_output_16-bit_DO8-DO1*:

Index: 0x6306, Sub-index: 1

Bit:	**15 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Content:	-	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

Bit value *DOx*	Output *DOx*
1	Output *DOx* shall take the pre-defined value specified in *error_value_output_16-bit_DO8-DO1 (*object 0x6307, subindex x).
0	Output *DOx* shall be kept unchanged if an error occurs.

(x = 0 …8)

4.10.42 Error Value Output 16-Bit (0x6307)

Provided that the corresponding error mode of *Dox* (object 0x6306, sub-index x) is active, device failures shall set the output *DOx* to the value configured by this object.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6307	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*error_value_output_16-bit_DO8-DO1*	0 ... 0xFFFF	0	0	unsigned 16	rw

Assignment of the parameter *error_value_output_16-bit_DO8-DO1*:

Index: 0x6307, Sub-index: 1

Bit:	**15 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Content:	-	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

The parameter contains the value, the outputs shall be set to, in case of fault.

Bit value *DOx*	Output *DOx*
1	Output *DOx* shall be set to ‘1’ (enabled) in case of fault if object 0x6306, subindex x is enabled.
0	Output *DOx* shall be set to ‘0’ (disabled) in case of fault if object 0x6306 sub-index x is enabled.

4.10.43 Filter Mask Output 16-Bit (0x6308)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6308	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*filter_mask_output_8-bit_DO8-DO1*	0 ... 0xFFFF	0	0	unsigned 16	rw

Assignment of the parameter *filter_mask_output_16-bit_DO8-DO1*:

Index: 0x6308, Sub-index: 1

Bit:	**15 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Content:	-	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

This parameter contains the filter mask

Bit value *DOx*	Output *DOx*
1	filter of output *DOx* is enabled
0	filter of output *DOx* is disabled

4.10.44 Write Output 32-Bit (0x6320)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6320	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*write_output_32-bit_DO8-DO1*	0 ... 0xFFFF FFFF	0	1	unsigned 32	rww

Assignment of the variable *write_output_32-bit_DO8-DO1*:

Index: 0x6320, Sub-index: 1

Bit:	**31 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Output:	-	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

If an output bit is set to ‘1’, the corresponding output is activated, i.e. the output voltage is ‘on’.

Bit value *DOx*	Output
1	output *DOx* is turned on (24 V)
0	output *DOx* is turned off (0 V/open)

(x = 1 …8)

4.10.45 Polarity Output 32-Bit (0x6322)

This object defines the state of the 8 digital outputs. Every output can be inverted individually.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6322	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*polarity output_32-bit_DO8-DO1*	0 ... 0xFFFF FFFF	0	0	unsigned 32	rw

Assignment of the parameter *change_polarity_output_32-bit_DO8-DO1*:

The parameter determines which digital output can be inverted.

Index: 0x6322, Sub-index: 1

Bit:	**31 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Content:	-	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

Bit value *DOx*	Output
1	output *DOx* is inverted
0	output *DOx* is not inverted

(x = 1 …8)

4.10.46 Error Mode Output 32-Bit (0x6326)

The error mode is evaluated if the module is in the *Stopped* state. This object in combination with object 0x1029 indicates, whether an output is set to an error-value defined in object 0x6327 in case of an internal device error.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6326	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*error_mode_output_32-bit_DO8-DO1*	0 ... 0xFFFF FFFF	0xFF	0	unsigned 32	rw

Assignment of the parameter *error_mode_output_32-bit_DO8-DO1*:

Index: 0x6326, Sub-index: 1

Bit:	**31 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Content:	-	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

Bit value *DOx*	Output *DOx*
1	Output *DOx* shall take the pre-defined value specified in *error_value_output_32-bit_DO8-DO1 (*object 0x6327, subindex x).
0	Output *DOx* shall be kept unchanged if an error occurs.

(x = 0 …8)

4.10.47 Error Value Output 32-Bit (0x6327)

Provided that the corresponding error mode of *Dox* (object 0x6326, sub-index x) is active, device failures shall set the output *DOx* to the value configured by this object.

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6327	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*error_value_output_32-bit_DO8-DO1*	0 ... 0xFFFFFFFF	0	0	unsigned 32	rw

Assignment of the parameter *error_value_output_32-bit_DO8-DO1*:

Index: 0x6327, Sub-index: 1

Bit:	**31 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Content:	-	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

The parameter contains the value, the outputs shall be set to, in case of fault.

Bit value *DOx*	Output *DOx*
1	Output *DOx* shall be set to ‘1’ (enabled) in case of fault if object 0x6326, subindex x is enabled.
0	Output *DOx* shall be set to ‘0’ (disabled) in case of fault if object 0x6326 sub-index x is enabled.

(x = 0 …8)

4.10.48 Filter Mask Output 32-Bit (0x6328)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x6328	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*filter_mask_output_32-bit_DO8-DO1*	0 ... 0xFFFFFFFF	0	0	unsigned 32	rw

Assignment of the parameter *filter_mask_output_32-bit_DO8-DO1*:

Index: 0x6328, Sub-index: 1

Bit:	**31 … …8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Content:	-	*DO8*	*DO7*	*DO6*	*DO5*	*DO4*	*DO3*	*DO2*	*DO1*

This parameter contains the filter mask

Bit value *DOx*	Output *DOx*
1	Filter of output *DOx* is enabled
0	Filter of output *DOx* is disabled

(x = 0 …8)

4.11 Manufacturer Specific Profile Area

4.11.1 Implemented Objects 0x2210 – 0x2403

Index	Name	Data Type
0x2250	*Input_Output 8-Bit*	unsigned 8
0x2300	*VIO Voltages*	unsigned 16
0x2310	*Sample Configuration*	unsigned 16
0x2400	*Counter Enable*	unsigned 8
0x2401	*Counter Preload*	unsigned 32
0x2402	*Counter Value 16- Bit*	unsigned 16
0x2403	*Counter Value 32-Bit*	unsigned 32
0x2404	*Counter Value 8-Bit*	unsigned 8

4.11.2 Input - Output (0x2250)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2250	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*output_enable_lines_DIO8-DIO1*	0 ... 0xFF	0	0	unsigned 8	rw

This object defines a bit mask, that determines which of the 8 I/O ports shall be used as input and which as output.

Assignment of the parameter *output_enable_lines_DIO8-DIO1*:

Index: 0x2250, Sub-index: 1

Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Output:	*DIO8*	*DIO7*	*DIO6*	*DIO5*	*DIO4*	*DIO3*	*DIO2*	*DIO1*

Value	Meaning
1	Port is used as output
0	Port is used as input

This object has the same functionality as 0x6208. You can choose which to use but it does not make sense to use both. They both hold the exact same value. There is even a third object 0x5FF5 Subindex 0 which also holds the same value (for backward compatibility)

4.11.3 VIO Voltage 16-Bit (0x2300)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2300	0	*no_of_entries*	0 ... 0xFF	3	0	unsigned 8	ro
	1	*VIO_voltage*	0 ... 0xFFFF	-	1	unsigned 16	ro
	2	*VIO_low_emergency_voltage*	0 ... 0xFFFF	0xB4	0	unsigned 16	rw
	3	*VIO_high_emergency_voltage*	0 ... 0xFFFF	0x118	0	unsigned 16	rw

With this object the operating voltage for the digital outputs can be read as analog value.
Additionally, the voltage limits for the operating voltage of the output drivers (that generate an Emergency Message if exceeded) can be defined.

Evaluation of the voltage values for sub-index 1, 2, 3:

VIO_voltage = (Voltage in V) x 10

Assignment of the variable *VIO_voltage* (Index: 0x2300, Subindex:1):
This variable returns the measured value of the supply voltage of the digital outputs.
Example:
Value of the variable *VIO_voltage* = 0xF0 = 240
(Voltage in V) = *VIO_voltage* / 10 = 240 / 10 = 24.0 V

Assignment of the parameters *VIO_low_emergency_voltage* (Index: 0x2300, Subindex:2):
This variable defines the voltage limit that initiates an Emergency Message, if the value falls **below**.
0 = disable
Example:
VIO_low_emergency_voltage = 0xB4 = 180 (value of the variable)
180/10 = 18 V voltage limit (min.); below this limit an Emergency Message is initiated

Assignment of the parameters *VIO_high_emergency_voltage* (Index: 0x2300, Subindex:3):
This variable defines the voltage limit that initiates an Emergency Message, if the value falls below.
0 = disable
Example:
VIO_high_emergency_voltage = 0x118 = 280 (value of the variable)
280/10 = 28 V voltage limit (max.); above this limit an Emergency Message is initiated

4.11.4 Sample Configuration (0x2310)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2310	0	*no_of_entries*	-	1	0	unsigned 8	ro
	1	*sample_rate*	0xC8 ... 0xFFFF	0xC8	0	unsigned 16	rw

This object enables and defines the filter of the digital inputs.

Assignment of the parameter *sample_rate*:
This parameter defines the sample rate that is used to read the digital inputs. The sample rate is set in steps of 0.5 µs. The minimum value is 100 µs.

Value of parameter *sample_rate*	Resulting sample rate [µs]
≤199	current parameter value is not changed (old value will stay valid + SDO error message)
200	100 (default value)
201	100.5
:	:
65535	32767.5

4.11.5 Function of the Counter

A 32-bit counter can be enabled (object 0x2400) for each input. In the default setting the edge-sensitive counters count the rising edges of the input signals.
For falling edge detection, the parameter *Polarity_Input* (object 0x6002) has to be adapted.

The initial value of the 32-bit counter is *counter_preload_x*, which is defined in object 0x2401. With every step the counter will be decremented by '1'. The *counter_value_x* (see objects 0x2402 and 0x2403) contains the current value of the counter x (x = 1...8).

The 32-bit counter can be used as event timer for Transmit-PDOs.
If the 32-bit counter (*counter_value_x_32-bit*) results in '0' (i.e. the event timer has expired), an event is generated, if the following conditions are met:

- The counter must be mapped.
  This setting is contained in the Transmit-PDO2-Mapping parameter per default (see object 0x1A01, *object_to_be_mapped_1-4*).

- The PDO2-Mapping must be enabled via object 0x1801.
  Therefore, for the parameter *COB-ID used by_PDO2* the most significant bit **must not be set**.
  Per default this bit is set, see object 0x1801, sub-index 1, i.e. the event-timer is disabled per default.

- The transmission type event controlled, asynchronous must be configured for the Tx-PDO2, i.e. the PDO transmission type must be 254 or 255.
  Per Default the transmission type is 255, see object 0x1801, sub-index 2.

After expiry of the counters (*counter_value_x_32-bit* = '0') and thus of the event timer the *counter_value_x_32-bit* is set to the *counter_preload_x*-value again (x = 1...8).

The interrelation of the objects of the digital inputs are described in Figure 13 on page 72.

NOTICE
If the counter function of an input is used, it is strongly recommended to disable the local interrupts via the *Interrupt Mask*-objects 0x6006, 0x6007 and 0x6008 (reduces bus load of the CAN bus).

The maximum attainable input clock rate results from the sample rate of the inputs (see object *Sample Configuration* 0x2310):

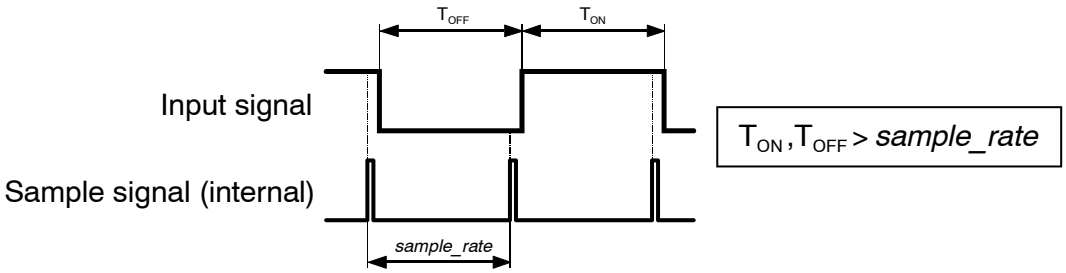


Figure 15: Interrelation of maximum count frequency and sample rate

So, the absolute maximum of the countable input frequency is 5 kHz.

4.11.6 Counter Enable (0x2400)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2400	0	*counter_enable*	0 ... 0xFF	0	0	unsigned 8	rw

With this object the counters 1 - 8 can be enabled. The counters count the number of changes of the digital inputs 1 – 8. In this process either changes with rising or with falling edges are counted.
By default, rising edges are counted. To count the falling edges, adapt the parameter *Polarity_Input* (object 0x6002).

Assignment of the variable *counter enable*:

Index: 0x2400, Subindex: 0

Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Port:	*counter8*	*counter7*	*counter6*	*counter5*	*counter4*	*counter3*	*counter2*	*counter1*

Value	Description
1	counter enabled
0	counter disabled

4.11.7 Counter Preload (0x2401)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2401	0	*no_of_ objects*	8	8	0	unsigned 8	ro
	1	*counter_preload_1*	0 ... 0xFFFF FFFF	0	0	unsigned 32	rw
	2	*counter_preload_2*		0	0	unsigned 32	rw
	3	*counter_preload_3*		0	0	unsigned 32	rw
	4	*counter_preload_4*		0	0	unsigned 32	rw
	5	*counter_preload_5*		0	0	unsigned 32	rw
	6	*counter_preload_6*		0	0	unsigned 32	rw
	7	*counter_preload_7*		0	0	unsigned 32	rw
	8	*counter_preload_8*		0	0	unsigned 32	rw

In this object the initial values *counter_preload_1...8* of the counters 1 ... 8 are defined.
Beginning from the initial value *counter_preload_x* the counter value is decremented by ‘1’ if a change occurs.

4.11.8 Counter Value 16-Bit (0x2402)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2402	0	*no_of_ objects*	-	8	0	unsigned 8	ro
	1	*counter_value_1_16-bit*	0 ... 0xFFFF	0	1	unsigned 16	ro
	2	*counter_value_2_16-bit*		0	1	unsigned 16	ro
	3	*counter_value_3_16-bit*		0	1	unsigned 16	ro
	4	*counter_value_4_16-bit*		0	1	unsigned 16	ro
	5	*counter_value_5_16-bit*		0	1	unsigned 16	ro
	6	*counter_value_6_16-bit*		0	1	unsigned 16	ro
	7	*counter_value_7_16-bit*		0	1	unsigned 16	ro
	8	*counter_value_8_16-bit*		0	1	unsigned 16	ro

The parameters *counter_value_x_16-bit* contain the least significant 16 bit of the current 32-bit values of the counters 1 - 8 (see object 0x2400 also) (*x* = 1 ... 8).

The following table shows the position of the 16-bit counter value *counter_value_x_16-bit* in the 32-bit counter value *counter_value_x_32-bit:*

Bit 31...			...Bit 0
counter_preload_x			
counter_value_x_32-bit			
Bit 31...	...Bit 16	**Bit 15...**	**...Bit 0**
-		***counter_value_x_16-bit***	

4.11.9 Counter Value 32-Bit (0x2403)

Index	Sub-index	Description	Value range	Default	PDO-Mapping	Data type	Access mode
0x2403	0	*no_of_ objects*	-	8	0	unsigned 8	ro
	1	*counter_value_1_32-bit*	0 ... 0xFFFF FFFF	0	1	unsigned 32	ro
	2	*counter_value_2_32-bit*		0	1	unsigned 32	ro
	3	*counter_value_3_32-bit*		0	1	unsigned 32	ro
	4	*counter_value_4_32-bit*		0	1	unsigned 32	ro
	5	*counter_value_5_32-bit*		0	1	unsigned 32	ro
	6	*counter_value_6_32-bit*		0	1	unsigned 32	ro
	7	*counter_value_7_32-bit*		0	1	unsigned 32	ro
	8	*counter_value_8_32-bit*		0	1	unsigned 32	ro

This object contains the current 32-bit values *counter_value_x_32-bit* of the counters 1...8 (see object 0x2400 also).

The counters count the number of changes of the digital inputs 1..8. Beginning with the 32-bit initial value *counter_preload_x* the value is decremented by ‘1’ if a change occurs (*x* = 1...8).

An event is generated, if the value of the **32-bit** counter results in ‘0’.
If the value of *counter_preload_x* is set to ‘0000.0000’, the preload is 2^{32}!

Bit 31...	...Bit 0
counter_preload_x	
counter_value_x_32-bit	

4.11.10 Counter Value 8-Bit (0x2404)

Index	Sub-index	Description	Value range	Default	PDO-Mapping	Data type	Access mode
0x2404	0	*no_of_ objects*	-	8	0	unsigned 8	ro
	1	*counter_value_1_8-bit*	0 ... 0xFF	0	1	unsigned 8	ro
	2	*counter_value_2_8-bit*		0	1	unsigned 8	ro
	3	*counter_value_3_8-bit*		0	1	unsigned 8	ro
	4	*counter_value_4_8-bit*		0	1	unsigned 8	ro
	5	*counter_value_5_8-bit*		0	1	unsigned 8	ro
	6	*counter_value_6_8-bit*		0	1	unsigned 8	ro
	7	*counter_value_7_8-bit*		0	1	unsigned 8	ro
	8	*counter_value_8_8-bit*		0	1	unsigned 8	ro

The parameters *counter_value_x_8-bit* contain the least significant 8 bit of the current 32-bit values of the counters 1 - 8 (see object 0x2400 also) (*x* = 1 ... 8).

The following table shows the position of the 8-bit counter value *counter_value_x_8-bit* in the 32-bit counter value *counter_value_x_32-bit:*

Bit 31...			...Bit 0
counter_preload_x			
counter_value_x_32-bit			
Bit 31... ...Bit 16	Bit 15... ...Bit 8	**Bit 7... ...Bit 0**	
-		***counter_value_x_8-bit***	

4.12 Firmware Management via CiA DSP-302-Objects

The objects described below are used for program updates via the object dictionary.

NOTICE
The firmware update must be carried out only by qualified personnel!
Faulty program update can result in deleting of the memory and loss of the firmware. The module then can not be operated further!

INFORMATION
esd offers the program CANfirmdown for a firmware update.
Please, contact our support for this.

Object 0x1F50 shall be used for program download to the CANopen device and object 0x1F51 for the control of the programs downloaded (Program data, object 0x1F50).
For further information about the objects and the firmware-update please refer to (5).

Index	Sub-index	Description	Data type	Access mode
0x1F50	1	Program data	domain	rw
0x1F51	1	Program control	unsigned 8	rw

4.12.1 Program Control via Object 0x1F51

This object is only implemented for compatibility reasons.

INDEX	**0x1F51**
Name	Program Control

Index	Sub-index	Description	Value range	Default value	PDO Mapping	Data type	Access mode
0x1F51	0	*number of programs*	1	1	0	unsigned 8	ro
	1	*program_number_1*	-	0	0	unsigned 8	rw

INFORMATION
This object is only contained for compatibility reasons.
It is not necessary to erase the firmware beforehand, just send the firmware to object 0x1F50 sub-index 1.
For further information about object 0x1F51 and the firmware-update please refer to the standard (5).

5 Technical Data

5.1 General Technical Data

Power supply voltage	Nominal voltage: 24 V Input voltage range: 12 V … 32 V DC Current consumption: I_{TYP_24V} = 23 mA, I_{MAX_24V} = 41 mA
Power consumption	Maximum: 1 W
Protective circuits	Reverse voltage protection, Overvoltage protection (triggering voltage= 32 V) Polyfuse in the input
Temperature range	-20 ... +70 °C ambient temperature
Humidity	Operation: max. 90%, non-condensing
Protection class	IP20
Pollution degree	Maximum permissible according to DIN EN 61131-2: Pollution Degree 2
Housing	Plastic housing for carrier rail mounting NS35/7,5 DIN EN 60715
Form factor / Dimensions	Width: 22.5 mm, height: 99 mm, depth: 114.5 mm (without connectors)
Weight	Approx. 125 g

Table 11: General Data of the module

5.2 Connectors accessible from Outside

Name/ Labelling	**Function, Ports**	**Type**	**Durability** (e.g. grade, contact surface, mating cycles)
CAN	CAN	5-pos. Phoenix Contact PCB header MC 1,5/5-GF-3,81 with cable connector FK-MCP 1,5/5-STF-3,81 with push-in spring connection	25 mating cycles
1 – 8 P, M,	Digital IO, Power supply of the I/Os	10-pos. (20-pin) Phoenix Contact PCB header MCDN 1,5/10-G1-3,5 with 2x cable connector FMC 1,5/10-ST-3,5 with push-in spring connection	25 mating cycles
24V	24V-power supply	4-pos. Phoenix Contact PCB header MSTBO 2,5/ 4-G1L KMGY with cable connector FKCT 2,5/4-ST KMGY with push-in spring connection	25 mating cycles
InRailBus	CAN and 24V power supply via InRailBus	(Not a physical component but contact surfaces on the printed circuit board. A 5-pos.) The CAN-CBX-TBus connector (C.3000.01) can be used, see accessories page 141.)	25 mating cycles of TBUS-connector

Table 12: Connectors, accessible from outside

5.3 CAN Port

Number of CAN ports	1x CAN CC
CAN controller	acc. to ISO 11898-1 (CAN 2.0 A/B), contained in CPU
CAN protocol	According to ISO 11898-1
Physical CAN Layer	High-speed CAN port according to ISO 11898-2, bit rate from 10 kbit/s up to 1 Mbit/s
Galvanic isolation	Separation by means of a digital isolator (transformer-based) and DC/DC-converter. Voltage over CAN isolation (CAN to slot bracket/EARTH; CAN to Host/System Ground): 1kV DC @ 1s (I < 1 mA)
Bus termination	Terminating resistor must be set externally, if required
Connector	CAN connector or via InRailBus, see 5.2

Table 13: Data of the CAN port

5.4 Digital Inputs/Outputs

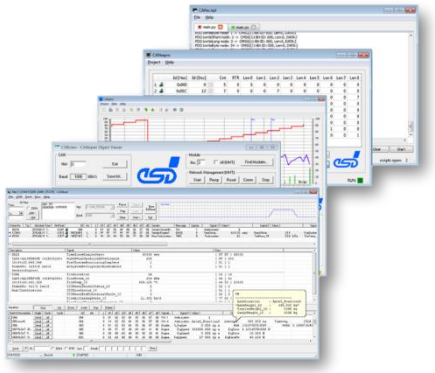
Number	8 independent ports, each programmable as input or output (Digital outputs whose output status is readable can be used like inputs when the output port is switched off.)
Specification of digital inputs	Input voltage (nominal value): 24 V DC, R_{IN} ~ 10 kΩ Input voltage is limited to VIO
Special input functions	Level, edge, counter
Input switching threshold	U_{ON} = ≥ 9 V, U_{OFF} = ≤ 5 V, $U_{HYSTERESIS}$ = 1 V
Specification of digital outputs	High-side power switches IO power supply (nominal): 12 ... 32 V DC, Nominal output current: : I_{NOM} = 0.5 A (70 °C, 24 V DC), Overcurrent limit: > 0.7 A
Galvanic isolation	none
Protective circuit	Short circuit and over temperature protection with output shutdown and auto-restart with hysteresis,
Connector	Digital I/O connector, see 5.2

Table 14: Data of the digital input/output

5.5 Software Support

CAN Tools
esd offers additional free-of-charge tools which support efficient setup and analysis of CAN applications and networks.
The CAN Tools are operational with all esd PC-CAN interfaces (e.g. PCIe, USB, EtherCAN/2 ...)

The following CAN Tools are available:



CANreal	Display and record of CAN message frames
CANplot	Graphical display of CAN data
CANrepro	Replay of pre-recorded CAN messages
CANscript	Python based scripting tool
COBview	Analysis and diagnostics of CANopen® nodes

System Requirements:
- Windows 32-bit or 64-bit system
- esd CAN driver installed

As part of the esd software development kit (CAN SDK) of the NTCAN-API the CAN Tools are included in delivery of the CAN-CD.
The CAN SDK can also be downloaded free-of-charge from the esd website: https://esd.eu/

6 Connector Assignments

6.1 Digital Inputs/Outputs

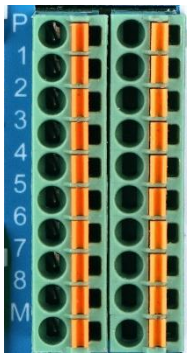
Device socket: Phoenix Contact header MCDN 1,5/10-G1-3,5 RNP26THR 20
(2 rows of 10 positions each)

Cable plug: Phoenix Contact pluggable connectors, 2x FMC 1,5/10-ST-3,5,
Push-in spring-connection, Phoenix Contact Order No.:1952348 (included in delivery)
For conductor connection and conductor cross section see page 127.

Pin position on cable plug:

Connector **Row1** **Row2**

Imprint on front panel:



P
1
2
3
4
5
6
7
8
M

Pin Assignment

Imprint on front panel	Row 1 Signal	Pin Position (device socket) Row 1 Row 2	Row 2 Signal
P	**P** (24 V IO Supply)		**P** (24 V IO Supply)
1	**M24** (Reference Potential of 24 V IO supply)		IO1
2			IO2
3			IO3
4			IO4
5			IO5
6			IO6
7			IO7
8			IO8
M			**M24**

Signal Description:

P (24 V IO Supply) ... 24 V supply voltage of the digital inputs and outputs
The pins are internally connected

M24... Reference potential of 24 V IO supply voltage (GND)
The pins are internally connected

IO*x*... Signal line of the digital input/output *x* (*x* = 1 - 8)

NOTICE
The **P** (24 V IO Supply) pins (pin **P** of row 1 and 2) are connected internally!
The **M24** pins (pin **1-8** and **M** of row 1 and **M** of row 2) are connected internally!

NOTICE
The maximum current load of the connector pins is 6 A/pin. If all 8 outputs are to be operated with the maximum admissible load, the supply voltage must be connected to the two **P** (24 V IO Supply) -pins of row 1 and row 2. These pins are connected to each other on the PCB.

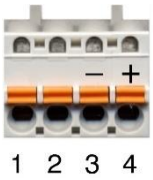
NOTICE
To ensure EU Conformity a cable with a maximum wire length of 30 m must be used for the digital inputs and digital outputs.

6.2 24V Power Supply Voltage

DANGER
The CAN-CBX-DIO8/2 is a device of protection class III according to DIN EN 61140 and may only be operated on supply circuits that offer sufficient protection against dangerous voltages.

Device socket: Phoenix Contact PCB header MSTBO 2,5/4-G1L-KMGY
Cable plug: Phoenix Contact pluggable connector FKCT 2,5/4-ST, 5.0 mm pitch, Push-in spring-connection, included in the scope of delivery (Phoenix Contact order No.: 19 21 90 0)
For conductor connection, cross section and stripping length see page 127.

Pin Position (cable plug):



Pin Assignment:

Device housing label			24V	
	.	.	M	P
Connector label	(none)	(none)	-	+

Pin	1	2	3	4
Signal	P24 (+ 24 V)	M24 (GND)	M24 (GND)	P24 (+ 24 V)

Please refer to the connecting diagram page 14.

NOTICE
The P24 pins (pin 1, 4) are connected internally!
The M24 pins (pin 2, 3) are connected internally!

NOTICE
There is a connection between the 24V plug and the InRailBus so that the module can be supplied via the InRailBus. Note that this connection is not designed to feed the 24V supply voltage via the plug to the InRailBus.
Feeding through the 24V power supply voltage can cause damage on the module!
Further wiring via the plug is possible if the modules are mounted on the DIN rail without InRailBus connectors. Use pin 1 and 2 as input wiring and pins 3 and 4 as output to the next module for example. Note that the limit values of the plug must not be exceeded and that voltage drops may occur in the plug.

→ Make absolutely sure to connect the cables correctly to the cable plug!
→ Use only suitable cables for the line plug.

Signal Description:
P24... Power supply voltage (12 V … 32 V DC, Nominal voltage = +24 V)
M24... Reference potential of P24

6.3 CAN

6.3.1 CAN Port

The CAN bus signals are electrically isolated from the other signals via digital isolator and DC/DC-converter.

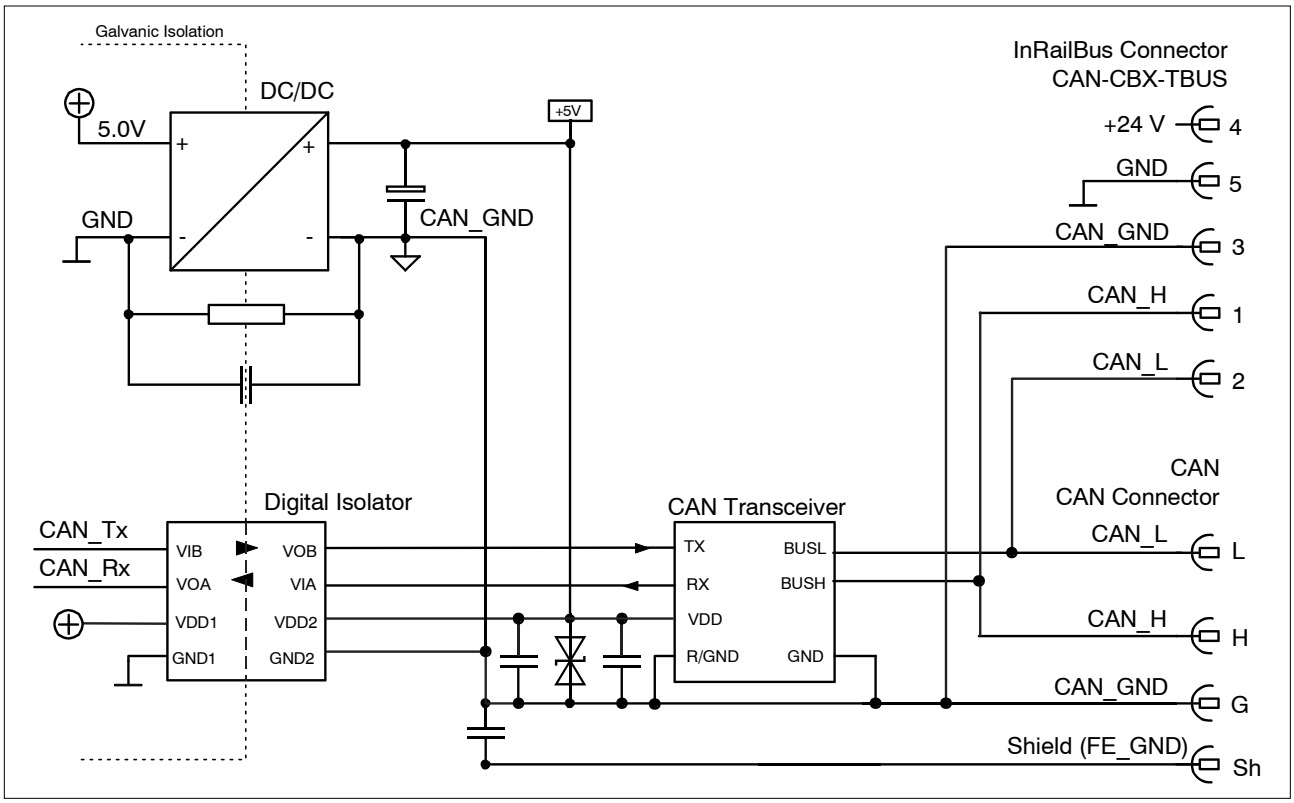


Figure 16: CAN Port

The CAN port can be connected via the CAN connector (see chapter 6.3.2) or optionally via the InRailBus (see chapter 6.4).

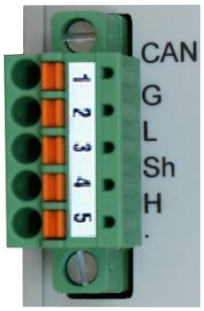
6.3.2 CAN Connector

The CAN bus is connected via the via this CAN connector on the upper side of the module or optional via the InRailbus (see page 126)

Device connector: Phoenix Contact PCB header MC 1,5/5-GF-3,81
Cable plug: Phoenix Contact pluggable connector FK-MCP 1,5/5-STF-3,81,
Push-in spring-connection, 3.81 mm pitch
Phoenix Contact Order No.: 1851261 (included in delivery)
For conductor connection, cross section and stripping length see page 127

Pin Position:

(Cable plug)



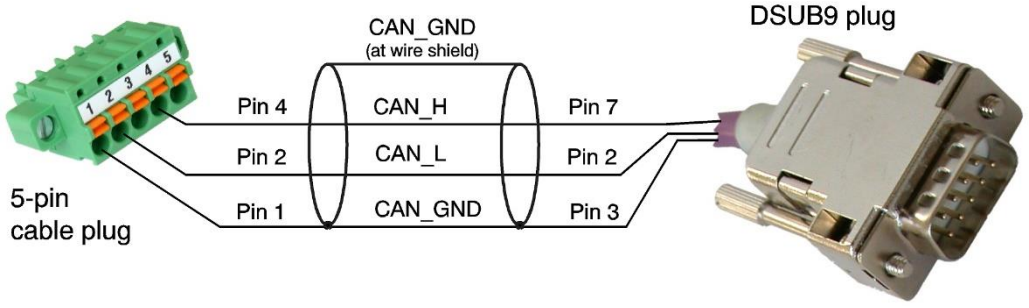
Pin Assignment:

Imprint	Signal	Pin
G	CANx_GND	1
L	CANx_L	2
Sh	Shield	3
H	CANx_H	4
●	-	5

Signal Description:

CANx_L, CANx_H …	CAN signal lines of CAN port x (x = 0, 1)
CANx_GND …	Reference potential of the local CAN physical layer of CAN x (x = 0, 1) To ensure reliable CAN communication, this pin must always be connected!
Shield …	Pin for line shield connection (using hat rail mounting direct contact to the mounting rail potential)
- …	Reserved, do not connect

Recommendation of an adapter cable from 5-pin cable plug (here Phoenix Contact FK-MCP1,5/5-STF_3,81 with spring-cage-connection) to 9-pin DSUB:



The assignment of the 9-pin DSUB-connector and the cable plug is designed according to CiA 106 .

INFORMATION
esd offers assembled CAN cables according to recommendations of CiA 303 part1 and CiA 106 (6) as accessories, see Order Information, page 141.

6.4 24 V and CAN via InRailBus

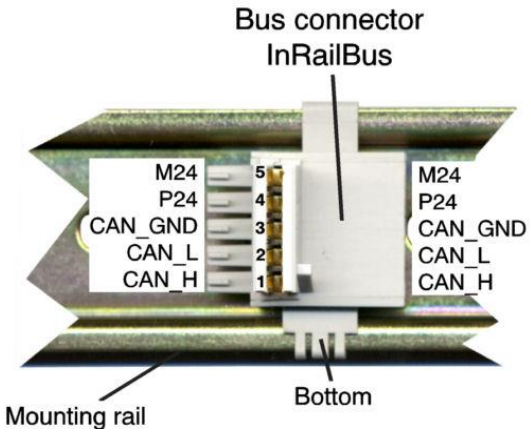
Power supply voltage and CAN can optionally be fed via the InRailBus. Use the mounting-rail bus connector (CAN-CBX-TBus) for the connection via the InRailBus, see Order Information (page 141).Read and follow the instructions for connecting power supply and CAN signals via InRailBus (see from page 23 and page 123)!

6.4.1 Connector Assignment 24V and CAN via InRailBus

DANGER
The CAN-CBX-DIO8/2 is a device of protection class III according to DIN EN IEC 61010-2-201 and may only be operated on supply circuits that offer sufficient protection against dangerous voltages.

Connector type: Mounting-rail bus connector of the CBX-InRailBus
Phoenix Contact ME 22,5 TBUS 1,5/5-ST-3,81 KMGY

Connector View:



Pin Assignment:

Pin	Signal
5	M24 (GND)
4	P24 (+24 V)
3	CAN_GND
2	CAN_L
1	CAN_H

S	FE (PE_GND)

Signal Description:

CAN_L, CAN_H ... CAN signals
CAN_GND ... reference potential of the local CAN-Physical layers
P24... power supply voltage +24 V
M24... reference potential
FE... functional earth contact (EMC) (connected to mounting rail potential)

6.5 Conductor Connection/Conductor Cross Section

NOTICE
The user must ensure that the cables used are designed for the connected voltages and currents in terms of dielectric strength, conductor cross-section and temperature range!

The following table contains an extract of the technical data of the cable plugs:

Characteristics	Connector Type [1]		
	Power Supply Voltage 24 V Cable connector	CAN Cable connector	Digital I/Os
Connector type plug component	FKCT 2,5/..-ST KMGY	FK-MCP 1,5/5-STF-3,81	FMC 1,5/..-MCDN-1,5
Connection method	Push-in spring-connection	Push-in spring-connection	Push-in spring-connection
Conductor material	Copper	Copper	Copper
Stripping length	10 mm	9 mm	10 mm
Nominal cross section	2.5 mm²	1.5 mm²	1.5 mm²
Conductor cross section rigid.	0.2 mm² … 2.5 mm²	0.14 mm² … 1.5 mm²	0.2 mm² …1.5 mm²
Conductor cross section flexible	0.2 mm² … 2.5 mm²	0.14 mm² … 1.5 mm²	0.2 mm² … 1.5 mm²
Conductor cross section AWG	24 … 12	26 … 16	24 … 16
Conductor cross section flexible, with ferrule without plastic sleeve	0.25 mm² … 2.5 mm²	0.25 mm² … 1.5 mm²	0.25 mm² … 1.5 mm²
Conductor cross section flexible, with ferrule with plastic sleeve	0.25 mm² … 2.5 mm²	0.25 mm² … 0.75 mm²	0.25 mm² … 0.75 mm²
2 conductors with same cross section, stranded, TWIN ferrules with plastic sleeve, min./max.	0.5 mm² … 1.5 mm²	n.a.	n.a.

[1] Technical Data from Phoenix Contact website, printed circuit board connector, plug component

The following table contains an extract of the technical data of the optional InRail Bus Connectors:

Characteristics of optional InRailBus Connectors	Connector Type [1]	
	CAN-CBX-TBus-Connector-Socket	CAN-CBX-TBus-Connector-Plug
Connector type plug component	MCVR 1,5/5-ST-3,81 AU	IMC 1,5/ 5-ST-3,81 AU
Connection method	Screw connection with tension sleeve	Screw connection with tension sleeve
Conductor material	Copper	Copper
Stripping length	7 mm	7 mm
Nominal cross section	1.5 mm²	1.5 mm²
Conductor cross section rigid.	0.14 mm² … 1.5 mm²	0.14 mm² … 1.5 mm²
Conductor cross section flexible	0.14 mm² … 0.5 mm²	0.14 mm² … 1.5 mm²
Conductor cross section AWG	28 … 16	28 … 16
Conductor cross section flexible, with ferrule without plastic sleeve	0.25 mm² … 1.5 mm²	0.25 mm² … 1.5 mm²
Conductor cross section flexible, with ferrule with plastic sleeve	0.25 mm² … 0.5 mm²	0.25 mm² … 0.5 mm²
2 conductors with same cross section, stranded, TWIN ferrules with plastic sleeve, min./max.	0.5 mm² ... 0.5 mm²	0.5 mm² ... 0.5 mm²

[1] Technical Data from Phoenix Contact website, printed circuit board connector, plug component

INFORMATION
For further information or other conductor connections see technical data of the connectors on the Phoenix Contact web site.

7 Correct Wiring of Electrically Isolated CAN Networks

NOTICE
This chapter applies to CAN networks with bit rates up to 1 Mbit/s.
If you work with higher bit rates, as for example used for CAN FD, the information given in this chapter must be examined for applicability in each individual case.
For further information refer to the CiA® CAN FD guidelines and recommendations (https://www.can-cia.org/).

For the CAN wiring all applicable rules and regulations (EU, DIN), such as regarding electromagnetic compatibility, security distances, cable cross-section or material, must be obeyed.

7.1 CAN Wiring Standards

The flexibility in CAN network design is a major strength of the various extensions based on the original CAN standard ISO 11898-2, such as CANopen®, ARINC825, DeviceNet® and NMEA2000. However, taking advantage of this flexibility absolutely requires a network design that considers the interactions of all network parameters.

In some cases, the CAN organizations have adapted the scope of CAN in their specifications to enable applications outside the ISO 11898 standard. They have imposed system-level restrictions on data rate, line length and parasitic bus loads.

However, when designing CAN networks, a margin must always be planned for signal losses over the entire system and cabling, parasitic loads, network imbalances, potential differences against earth potential, and signal integrities. **Therefore, the maximum achievable number of nodes, bus lengths and stub lengths may differ from the theoretically possible number!**

esd has limited its recommendations for CAN wiring to the specifications of ISO 11898-2.
A description of the special features of the derived specifications CANopen, ARINC825, DeviceNet, and NMEA2000 is omitted here.

The consistent compliance with the ISO 11898-2 standard offers significant advantages:

- Reliable operation due to proven design specifications
- Minimization of error sources due to sufficient distance to the physical limits.
- Easy maintenance because there are no "special cases" to consider for future network modifications and troubleshooting.

Of course, reliable networks can be designed according to the specifications of CANopen, ARINC825, DeviceNet and NMEA2000, **however it is strictly not recommended to mix the wiring guidelines of the various specifications!**

7.2 Light Industrial Environment (*Single* Twisted Pair Cable)

7.2.1 General Rules

NOTICE
esd grants the EU Conformity of the product if the CAN wiring is carried out with at least single shielded **single** twisted pair cables that match the requirements of ISO 11898-2. Single shielded *double* twisted pair cable wiring as described in chapter 7.3 ensures the EU Conformity as well.

The following **general rules** for CAN wiring with single shielded *single* twisted pair cable should be followed:

1	A suitable cable type with a wave impedance of about 120 Ω ±10% with an adequate conductor cross-section (≥ 0.22 mm²) must be used. The voltage drop over the wire must be considered.
2	For light industrial environment use at least a two-wire CAN cable, the wires of which must be assigned as follows: • Two twisted wires must be assigned to the data signals (CAN_H, CAN_L). • The cable shield must be connected to the reference potential (CAN_GND).
3	The reference potential CAN_GND must be connected to the functional earth (FE) at exactly **one** point.
4	A CAN bus line must not branch (exception: short cable stubs) and must be terminated with the characteristic impedance of the line (generally 120 Ω ±10%) at both ends (between the signals CAN_L and CAN_H and **not** at CAN_GND).
5	Keep cable stubs as short as possible (l < 0.3 m).
6	Select a working combination of bit rate and cable length.
7	Keep away cables from disturbing sources. If this cannot be avoided, double shielded wires are recommended.

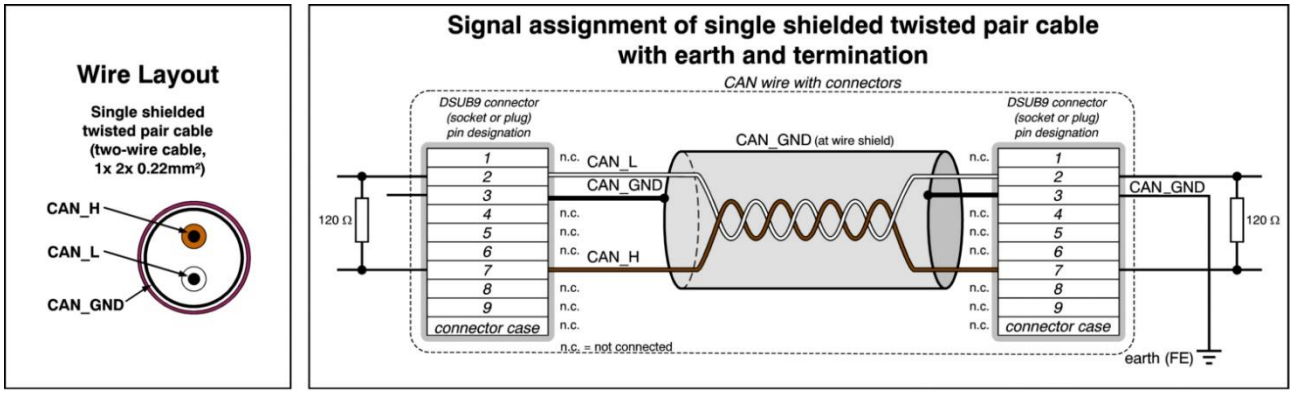


Figure 17: CAN wiring for light industrial environment

7.2.2 Cabling

- To connect CAN devices with just one CAN connector per net use a short stub (< 0.3 m) and a T-connector (available as accessory). If these devices are located at the end of the CAN network, the CAN terminator “CAN-Termination-DSUB9” can be used.

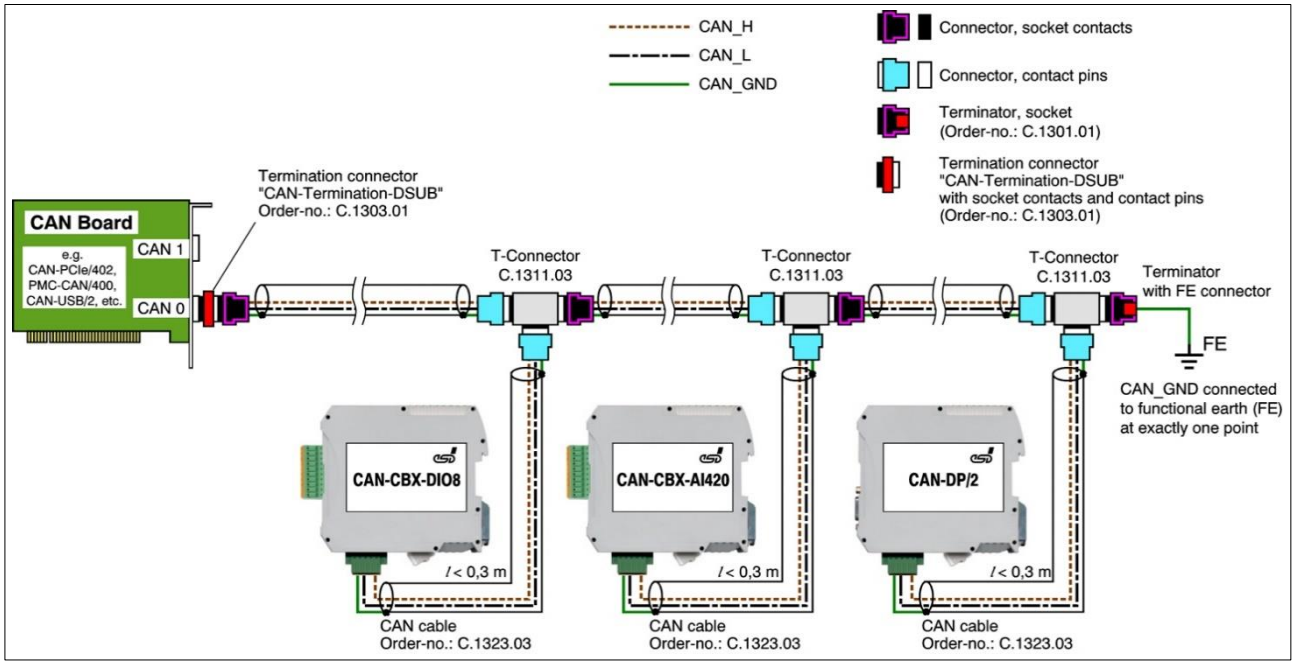


Figure 18: Example for proper wiring with single shielded single twisted pair wires

7.2.3 Branching

- In principle the CAN bus must be realized in a line. The nodes are connected to the main CAN bus line via short cable stubs. This is normally realised by so called T-connectors. esd offers the CAN-T-Connector (Order No.: C.1311.03)
- If a mixed application of single twisted and double twisted cables cannot be avoided, ensure that the CAN_GND line is not interrupted!
- Deviations from the bus structure can be realized by using repeaters.

7.2.4 Termination Resistor

- A termination resistor must be connected at both ends of the CAN bus.
  If an integrated CAN termination resistor is connected to the CAN interface at the end of the CAN bus, this integrated termination must be used instead of an external CAN termination resistor.
- 9-pole DSUB-termination connectors with integrated termination resistor and pin contacts and socket contacts are available from esd (order no. C.1303.01).
- For termination of the CAN bus and grounding of the CAN_GND, DSUB terminators with pin contacts (order no. C.1302.01) or socket contacts (order no. C.1301.01) and with additional functional earth contact are available.

7.3 Heavy Industrial Environment (Double Twisted Pair Cable)

7.3.1 General Rules

The following **general rules** for the CAN wiring with single shielded *double* twisted pair cable should be followed:

1	A suitable cable type with a wave impedance of about 120 Ω ±10% with an adequate conductor cross-section (≥ 0.22 mm²) must be used. The voltage drop over the wire must be considered.
2	For heavy industrial environment use a four-wire CAN cable, the wires of which must be assigned as follows: • Two twisted wires must be assigned to the data signals (CAN_H, CAN_L) and • The other two twisted wires must be assigned to the reference potential (CAN_GND). • The cable shield must be connected to functional earth (FE) at least at one point.
3	The reference potential CAN_GND must be connected to the functional earth (FE) at exactly **one** point.
4	A CAN bus line must not branch (exception: short cable stubs) and must be terminated with the characteristic impedance of the line (generally 120 Ω ±10%) at both ends (between the signals CAN_L and CAN_H and **not** to CAN_GND).
5	Keep cable stubs as short as possible (l < 0.3 m).
6	Select a working combination of bit rate and cable length.
7	Keep away CAN cables from disturbing sources. If this cannot be avoided, double shielded cables are recommended.

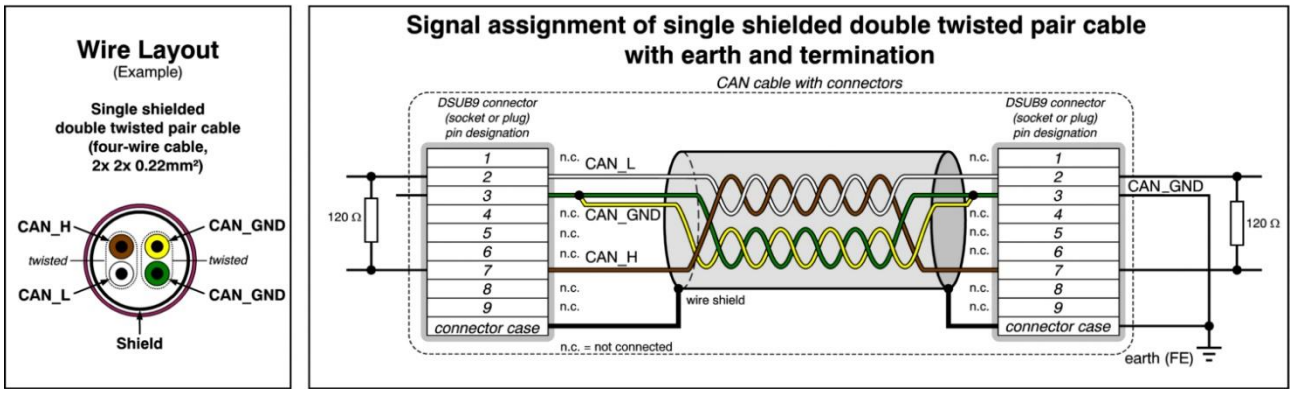


Figure 19: CAN wiring for heavy industrial environment

7.3.2 Device Cabling

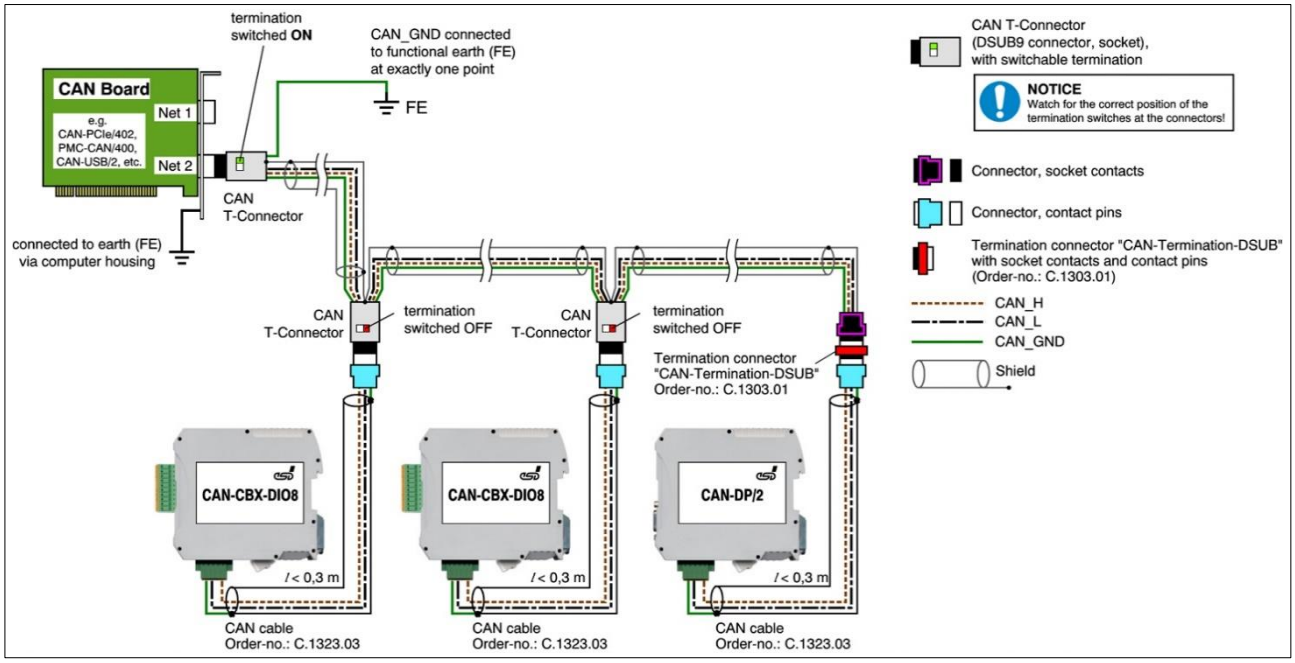


Figure 20: Example of proper wiring with single shielded double twisted pair cables

7.3.3 Branching

- In principle, the CAN bus must be realized in a line. The nodes are connected to the main CAN bus line via short cable stubs. This is usually realised via so called T-connectors. When using esd's CAN-T-Connector (order no.: C.1311.03) in heavy industrial environment and with four-wire twisted cables, it must be noted that the shield potential of the conductive DSUB housing is not looped through this type of T-connector. This interrupts the shielding. Therefore, you must take appropriate measures to connect the shield potentials, as described in the manual of the CAN-T-Connector. For further information on this, please refer to the CAN-T-Connector Manual (order no.: C.1311.21).
  Alternatively, a T-connector can be used, in which the shield potential is looped through, for example the DSUB9 connector from ERNI (ERBIC CAN BUS MAX, order no.:154039).
- If a mixed application of single twisted and double twisted cables cannot be avoided, ensure that the CAN_GND line is not interrupted!
- Deviations from the bus structure can be realized by using repeaters.

7.3.4 Termination Resistor

- A termination resistor must be connected at both ends of the CAN bus.
  If an integrated CAN termination resistor is connected to the CAN interface at the end of the CAN bus, this integrated termination must be used instead of an external CAN termination resistor.
- 9-pole DSUB-termination connectors with integrated termination resistor and pin contacts and socket contacts are available from esd (order no. C.1303.01).
- 9-pole DSUB-connectors with integrated switchable termination resistor can be ordered for example from ERNI (ERBIC CAN BUS MAX, socket contacts, order no.:154039).

7.4 Electrical Grounding

- For CAN devices with galvanic isolation the CAN_GND must be connected between the CAN devices.
- CAN_GND should be connected to the earth potential (FE) at **exactly one** point of the network.
- Each *CAN interface with electrical connection to earth potential* acts as a grounding point. For this reason, it is recommended not to connect more than one *CAN device with electrical connection to earth potential.*
- Grounding can be done for example at a termination connector (e.g. order no. C.1302.01 or C.1301.01).

7.5 Bus Length

The bus length of a CAN network must be adapted to the set bit rate. The maximum values result from the fact that the time required for a bit to be transmitted in the bus system is shorter the higher the transmission rate is. However, as the line length increases, so does the time it takes for a bit to reach the other end of the bus. It should be noted that the signal is not only transmitted, but the receiver must also respond to the transmitter within a certain time. The transmitter, in turn, must detect any change in bus level from the receiver(s). Delay times on the line, the transceiver, the controller, oscillator tolerances and the set sampling time must be considered.
In the following table you will find guide values for the achievable bus lengths at certain bit rates.

Bit Rate [kbit/s]	Theoretical values of reachable wire length **with esd interface** l_{max} [m]	**CiA recommendations** (07/95) for reachable wire lengths l_{min} [m]	Standard values of the cross-section according to CiA 303-1 [mm²]
1000	37	25	0.25 to 0.34
800	59	50	0.34 to 0.6
666.$\overline{6}$	80	-	
500	130	100	
333.$\overline{3}$	180	-	
250	270	250	
166	420	-	0.5 to 0.6
125	570	500	
100	710	650	0.75 to 0.8
83.$\overline{3}$	850	-	
66.$\overline{6}$	1000	-	
50	1400	1000	
33.$\overline{3}$	2000	-	not defined in CiA 303-1
20	3600	2500	
12.5	5400	-	
10	7300	5000	

Table 15: Recommended cable lengths at typical bit rates (with esd-CAN interfaces)

Optical couplers are delaying the CAN signals. esd modules typically achieve a wire length of 37 m at 1 Mbit/s within a proper terminated CAN network without impedance disturbances, such as those caused by cable stubs > 0.3 m.

NOTICE
Please note that the cables, connectors, and termination resistors used in CANopen networks shall meet the requirements defined in ISO 11898-2.
In addition, further recommendations of the CiA, like standard values of the cross section, depending on the cable length, are described in the CiA recommendation CiA 303-1 (see CiA 303 CANopen Recommendation - Part 1: “Cabling and connector pin assignment,” Version 1.9.0, Table 2). Recommendations for pin-assignment of the connectors are described in CiA 106: “Connector pin-assignment recommendations ”.

7.6 Examples for CAN Cables

esd recommends the following two-wire and four-wire cable types for CAN network design. These cable types are used by esd for ready-made CAN cables, too.

7.6.1 Cable for Light Industrial Environment Applications (Two-Wire)

Manufacturer	Cable Type
U.I. LAPP GmbH Schulze-Delitzsch-Straße 25 70565 Stuttgart Germany www.lappkabel.com	e.g. UNITRONIC ®-BUS CAN UL/CSA (1x 2x 0.22) (UL/CSA approved) Part No.: 2170260 UNITRONIC ®-BUS-FD P CAN UL/CSA (1x 2x 0.25) (UL/CSA approved) Part No.: 2170272
ConCab GmbH Äußerer Eichwald 74535 Mainhardt Germany www.concab.de	e. g. BUS-PVC-C (1x 2x 0.22 mm²) Order No.: 93 022 016 (UL appr.) BUS-Schleppflex-PUR-C (1x 2x 0.25 mm²) Order No.: 94 025 016 (UL appr.)

7.6.2 Cable for Heavy Industrial Environment Applications (Four-Wire)

Manufacturer	Cable Type
U.I. LAPP GmbH Schulze-Delitzsch-Straße 25 70565 Stuttgart Germany www.lappkabel.com	e.g. UNITRONIC ®-BUS CAN UL/CSA (2x 2x 0.22) (UL/CSA approved) Part No.: 2170261 UNITRONIC ®-BUS-FD P CAN UL/CSA (2x 2x 0.25) (UL/CSA approved) Part No.: 2170273
ConCab GmbH Äußerer Eichwald 74535 Mainhardt Germany www.concab.de	e. g. BUS-PVC-C (2x 2x 0.22 mm²) Order No.: 93 022 026 (UL appr.) BUS-Schleppflex-PUR-C (2x 2x 0.25 mm²) Order No.: 94 025 026 (UL appr.)

INFORMATION
Ready-made CAN cables with standard or custom length can be ordered from **esd**.

8 CAN Troubleshooting Guide

The CAN Troubleshooting Guide is a guide to finding and eliminating the most common problems and errors when setting up CAN bus networks and CAN-based systems.

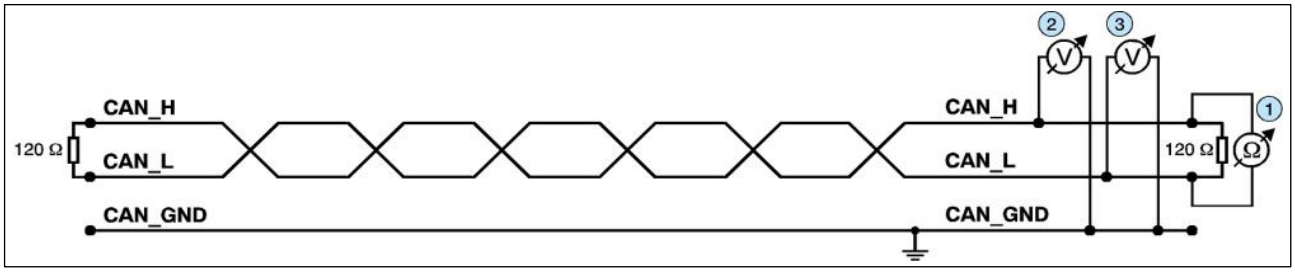


Figure 21: Simplified diagram of a CAN network

Termination
The bus termination is used to match impedance of a node to the impedance of the bus line used. If the impedance is mismatched, the transmitted signal is not completely absorbed by the load and will be partially reflected back into the transmission line.
If the impedances of the sources, transmission lines and loads are equal, the reflections are avoided. This test measures the total resistance of the two CAN data lines and the connected terminating resistors.

To **test this, please proceed as follows:**

1. Switch off the supply voltages of all connected CAN nodes.
2. Measure the DC resistance between CAN_H and CAN_L at one end of the network, measuring point ① (see figure above).

Expected result:
The measured value should be between 50 Ω and 70 Ω.

Possible causes of error:

- If the determined value is below 50 Ω, please make sure that:
  - There is no **short circuit** between CAN_H and CAN_L wiring.
  - **No more than two** terminating resistors are connected.
  - The transceivers of the individual nodes are not defective.

- If the determined value is higher than 70 Ω, please make sure that:
  - All CAN_H and CAN_L lines are correctly connected.
  - Two terminating resistors of 120 Ω each are connected to your CAN network (one at each end).

8.1 Electrical Grounding

The CAN_GND of the CAN network should be connected to the functional earth potential (FE) at only **one** point. This test indicates whether the CAN_GND is grounded at one or more points.

Please note that this test can only be performed with galvanically isolated CAN nodes.

To test this, please proceed as follows:

1. Disconnect the CAN_GND from the earth potential (FE).
2. Measure the DC resistance between CAN_GND and earth potential (see figure on the right).

Do not forget to reconnect CAN_GND to earth potential after the test!

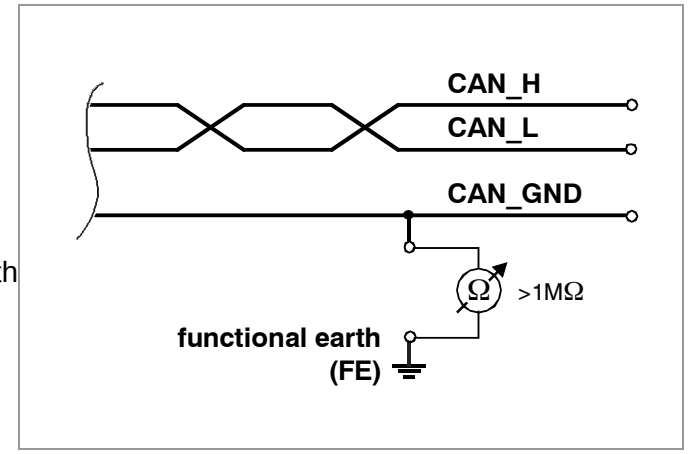


Figure 22: Simplified schematic diagram of ground test measurement

Expected result:
The measured resistance should be greater than 1 MΩ. If it is smaller, please search for additional grounding of the CAN_GND wires.

8.2 Short Circuit in CAN Wiring

A CAN bus might possibly still be able to transmit data even if CAN_GND and CAN_L are short-circuited. However, this will usually cause the error rate to rise sharply.
Ensure that there is no short circuit between CAN_GND and CAN_L!

8.3 Correct Voltage Levels on CAN_H and CAN_L

Each node contains a CAN transceiver that outputs differential signals. When the network communication is idle the CAN_H and CAN_L voltages are approximately 2.5 V measured to CAN_GND. Defective transceivers can cause the idle voltages to vary and disrupt network communication.

To test for defective transceivers, please proceed as follows:

1. Switch on all supply voltages.
2. Terminate all network communication.
3. Measure the DC voltage between CAN_H and CAN_GND, measuring point ②.
   (See “Simplified diagram of a CAN network” on previous page).
4. Measure the DC voltage between CAN_L and CAN_GND, measuring point ③.
   (See “Simplified diagram of a CAN network” on previous page).

Expected result:
The measured voltage should be between 2.0 V and 3.0 V.

Possible causes of error:

- If the voltage is lower than 2.0 V or higher than 3.0 V, it is possible that one or more nodes have defective transceivers.
    - If the voltage is lower than 2.0 V, please check the connections of the CAN_H and CAN_L lines.
- To find a node with a defective transceiver within a network, please check individually the resistances of the CAN transceivers of the nodes (see next section).

8.4 CAN Transceiver Resistance Test

CAN transceivers have circuits that control CAN_H and CAN_L. Experience shows that electrical damage can increase the leakage current in these circuits.

To measure the current leakage through the CAN circuits, please use an ohmmeter and proceed as follows:

1. Switch **off** the node ④ and **disconnect** it from the CAN network.
   (See figure below.)
2. Measure the DC resistance between CAN_H and CAN_GND, measuring point ⑤
   (See figure below.)
3. Measure the DC resistance between CAN_L and CAN_GND, measuring point ⑥
   (See figure below.)

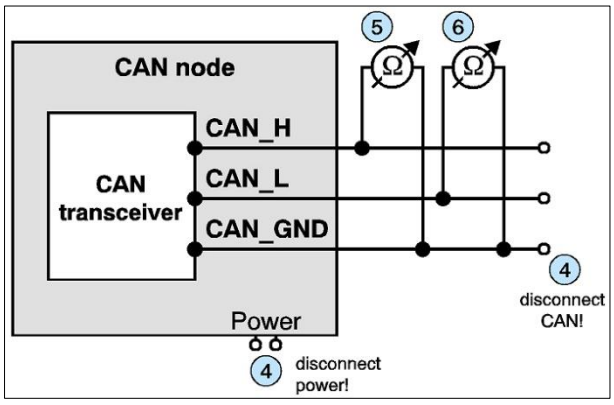


Figure 23: Measuring the internal resistance of CAN transceivers

Expected result:
The measured resistance should be greater than 10 kΩ for each measurement.

Possible causes of error:

- If the resistance is significantly lower, the CAN transceiver may be defective.
- Another indication of a defective CAN transceiver is a very high deviation of the two measured input resistances (>> 200 %).

8.5 Support by esd

If you have followed the troubleshooting steps in this troubleshooting guide and still cannot find a solution to your problem, our support team can help.
Please contact our support by email to support@esd.eu or by phone **+49-511-37298-130.**

9 Software Licenses

NOTICE
The software from esd and from third parties used in the CAN-CBX-DIO8/2 is subject to the license terms of the respective authors or rights holders. CAN-CBX-DIO8/2 may only be used in accordance with these license terms!
By using the CAN-CBX-DIO8/2 you agree to the terms of these software licenses.

You can also download the licenses from our website, see the following chapters.

9.1 3rd Party Software License Terms

License Name	Identifier (from SPDX License List)
MIT License	MIT
BSD 3-Clause "New" or "Revised" License	BSD-3-Clause

Table 16: 3rd Party Software License Terms

- The CAN-CBX-DIO8/2 uses the opensource operating system FreeRTOS™.
  For the full license text see FreeRTOS, Amazon.com, Inc., Licence Details:
  https://www.freertos.org/a00114.html#license_comparison

  This also includes the MIT open source license.
  You can also download the text of the MIT License from our homepage, see **Table 16**.

- CMSIS End User License Agreement, For the full text of the End user licence agreement for the cortex microcontroller software interface standard (CMSIS) deliverables see:
  CMSIS_END_USER_LICENCE_AGREEMENT.pdf

- The CAN-CBX-DIO8/2 uses the libraries of ST Microelectronics
  The library of ST is subject to the 3rd Party Software License Terms of BSD-3-Clause, see **Table 16**.

9.2 Open-Source Software Copy

You may obtain a copy of the open-source source code, if and as required under the license by sending a mail to oss-compliance@esd.eu

You may also obtain a copy of the open-source source code, if and as required under the license, by sending a check or money of EUR 25.00 to:
esd electronics gmbh
Vahrenwalder Str. 207
30165 Hannover, Germany

10 References

(1) CiA 301, CANopen Application Layer and Communication Profile V4.2.0 (02.2011), CAN in Automation (CiA) e. V.
(2) CiA 401 Draft Standard Proposal V3.0 (10.2006) CANopen Device profile for generic IO modules, CAN in Automation (CiA) e. V., Nürnberg, Germany,
(3) CiA 303-3 CANopen LEDs, CANopen Additional Specification V1.4.0 (04.2012), CAN in Automation (CiA) e. V.
(4) CAN Application Layer for Industrial Applications CiA/DS202-2 February 1996 CMS Protocol Specification
(5) CiA Draft Standard Proposal 302 V4.1 (04.2010) Additional Application Layer functions, Part 3: Configuration and program download
(6) CiA 106, Connector Pin-assignment Recommendations, Technical Report V1.1.0 (07. 2023), CAN in Automation (CiA) e. V.

11 Declaration of Conformity

EU-KONFORMITÄTSERKLÄRUNG
EU DECLARATION OF CONFORMITY

Adresse / *Address*
esd electronics gmbh
Vahrenwalder Str. 207
30165 Hannover
Germany

esd erklärt, dass das Produkt *esd declares, that the product*	Typ, Modell, Artikel-Nr. *Type, Model, Article No.*
CAN-CBX-DIO8/2	**C.3010.04**
die Anforderungen der Normen *fulfills the requirements of the standards*	**EN 61000-6-2:2005,** **EN 61000-6-3:2007/A1:2011**
gemäß folgendem Prüfbericht erfüllt. *according to test certificate.*	**EMVP No.: 0233-202307**
Das Produkt entspricht damit der EU-Richtlinie „EMV" *Therefore, the product conforms to the EU Directive 'EMC'*	**2014/30/EU**
Das Produkt entspricht den EU-Richtlinien „RoHS" *The product conforms to the EU Directives 'RoHS'*	**2011/65/EU, 2015/863/EU**

Diese Erklärung verliert ihre Gültigkeit, wenn das Produkt nicht den Herstellerunterlagen entsprechend eingesetzt und betrieben wird, oder das Produkt abweichend modifiziert wird.
This declaration loses its validity if the product is not used or run according to the manufacturer's documentation or if non-compliant modifications are made.

Name / *Name* T. Bielert
Funktion / *Title* QM-Beauftragter / *QM Representative*
Datum / *Date* Hannover, 2025-04-30

Rechtsgültige Unterschrift / *authorized signature*

I:\Texte\Doku\MANUALS\CAN\CBX\CAN-CBX-DIO8-2\EU-Konformitaetserklaerung\CAN-CBX-DIO8-2_EU_Declaration_of_Conformity_2025-04-30.docx

12 Order Information

Type	Properties	Order No.
CAN-CBX-DIO8/2	CANopen module with 8 Digital I/O ports. CANopen according to CiA 301 and CiA 401 Device Profile for Generic I/O Modules. Connectors for wire end ferrules included in the scope of delivery.	C.3010.04
Accessories		
CAN-Cable-S, 0.3 m (plug)	CAN cable assembly, 0.3 m length, 1 x DSUB-9 plug and 3 wire end sleeves, metallized plastic housing	C.1323.03
CAN-Cable-B, 0.3 m (socket)	CAN cable assembly, 1 x DSUB-9 socket and 3 wire end sleeves, length 0.3 m, metallized plastic housing	C.1323.04
Accessories for InRailBus		
 **CAN-CBX-TBus**	DIN-rail bus connector of the CBX-InRailBus for CAN-CBX modules (ME 22,5 TBUS 1,5/ 5-ST-3,81 KMGY)	C.3000.01
 **CAN-CBX-TBus-Connector-Socket**	Terminal plug of the CBX-InRailBus for the connection of the +24V power supply voltage and the CAN port (MCVR 1,5/5-ST-3,81 AU), socket contacts	C.3000.02
 **CAN-CBX-TBus-Connector-Plug**	Terminal plug of the CBX-InRailBus for the connection of the +24V power supply voltage and the CAN-Port (IMC 1,5/ 5-ST-3,81 AU), pin contacts	C.3000.03

Table 17: Order information

PDF Manuals
Please download the manuals as PDF documents from our esd website https://www.esd.eu for free.

Manuals		Order No.
CAN-CBX-DIO8/2-ME	CAN-CBX-DIO8/2 Manual in English	C.3010.21
CAN-API-ME	CAN-API Manual in English NTCAN-API, Part 1: Application Developers Manual NTCAN-API, Part 2: Driver Installation Guide	C.2001.21
CANopen-ME	CANopen Manuals in English	C.2002.21

Printed Manuals
If you need a printout of the manual additionally, please contact our sales team (sales@esd.eu) for a quotation. Printed manuals may be ordered for a fee.