CAN-CBX-PT100/2

CANopen® Module for precise Measurement of Temperatures

Hardware Manual

For Product C.3032.04

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

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-PT100/2. 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-PT100/2 is subject to the license terms of the respective authors or rights holders. CAN-CBX-PT100/2 may only be used in accordance with these license terms!

Document Information

Document file:	I:\Texte\Doku\MANUALS\CAN\CBX\CAN-CBX-PT100-2\CAN-CBX-PT100-2_Manual_en_14.docx
Date of print:	2025-10-29
Document-type number:	QM.1012.C.3032.04

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
1.2	-	First English version of the CAN-CBX-PT100/2 manual	2025-06-17
1.3	-	Page 6: Intended use corrected	2025-09-01
	3.1	Correction/supplement in step 2., 7. and 8.	
	4.11.16	Value range and functions corrected and note on syntax error added.	
	4.11.20	Value range and description of bits changed	
	6.5	Correction of the data for the temperature measurement plug	
	12	Information on available manuals revised	
1.4	-	Minor editorial changes	2025-10-29

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-PT100/2 follow the instructions below and read the manual carefully to protect yourself from injury and the CAN-CBX-PT100/2 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-PT100/2 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-PT100/2 has only functional tasks and is not a protection against hazardous electrical voltage.
- The CAN-CBX-PT100/2 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-PT100/2 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-PT100/2 .
- The CAN-CBX-PT100/2 must be securely installed before commissioning.
- The permitted operating position is specified as shown (Figure 2). Other operating positions are not allowed.
- Never let liquids get inside CAN-CBX-PT100/2. Otherwise, electric shocks or short circuits may result.
- Protect the CAN-CBX-PT100/2 from dust, moisture, and steam.
- Protect the CAN-CBX-PT100/2 from shocks and vibrations.
- The CAN-CBX-PT100/2 may become warm during normal use. Always allow adequate ventilation around the CAN-CBX-PT100/2 and use care when handling
- Do not operate the CAN-CBX-PT100/2 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-PT100/2 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.
→ 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-PT100/2 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-PT100/2 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-PT100/2 is the operation as a CANopen module with four sensor resistor inputs for precise measurement of temperatures.
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-PT100/2 is intended for installation in a control cabinet.
- The operation of the CAN-CBX-PT100/2 in hazardous areas, or areas exposed to potentially explosive materials is not permitted.
- The operation of the CAN-CBX-PT100/2 for medical purposes is prohibited.

Service Note

The CAN-CBX-PT100/2 does not contain any parts that require maintenance by the user. The CAN-CBX-PT100/2 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-PT100/2 module.

1.2 Description of CAN-CBX-PT100/2

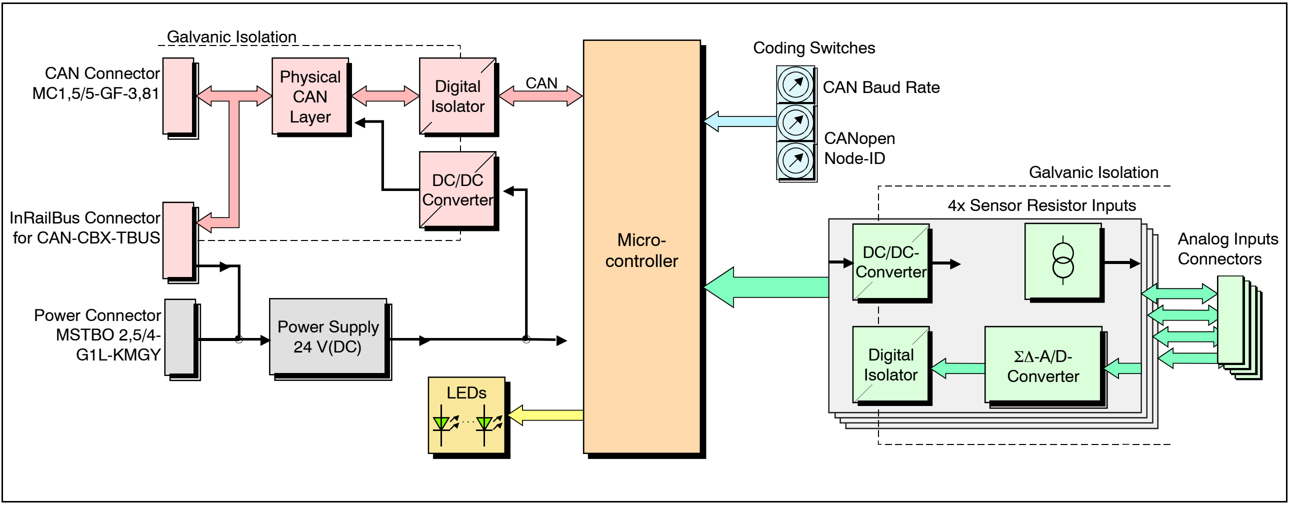


Figure 1: Block circuit diagram

The CAN-CBX-PT100/2 is a CANopen® module for precise measurement of temperatures. It is designed for the direct connection of up to four temperature sensors and provides a wide temperature application range with flexibility in the choice of Pt or Ni sensors.
The measurements can be carried out with high precision, adjustable resolution and 4-wire connection technology
A pure resistance measurement with output of the results in ohms is also possible. For these high-accuracy measurements two different sensing currents can be selected.
The conversion of the four sensor resistor inputs is realized by four independent ΣΔ-converters.

Four LEDs in the front panel clearly visualize the status of the CANopen node and the module status.

The module is connection-compatible with the predecessor module CAN-CBX-PT100 (C.3012.02) and the functionality is largely identical.

The module features a high-speed CAN interface 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® 404 "CANopen profile for measuring devices" (2)

The CAN-CBX-PT100/2 can be easily combined 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-PT100/2 can be seamlessly integrated into the DIN rail via an optional connector (TBUS connector, see page 125). 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. CAN-CBX-PT100/2 supports CAN CC. CAN FD and CAN XL are not supported
CAN CC	CAN classic	
CAN FD	CAN flexible data rate	Not supported by CAN-CBX-PT100/2
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	

2 Hardware

2.1 Connecting Diagram

2.1.1 Power Supply and CAN

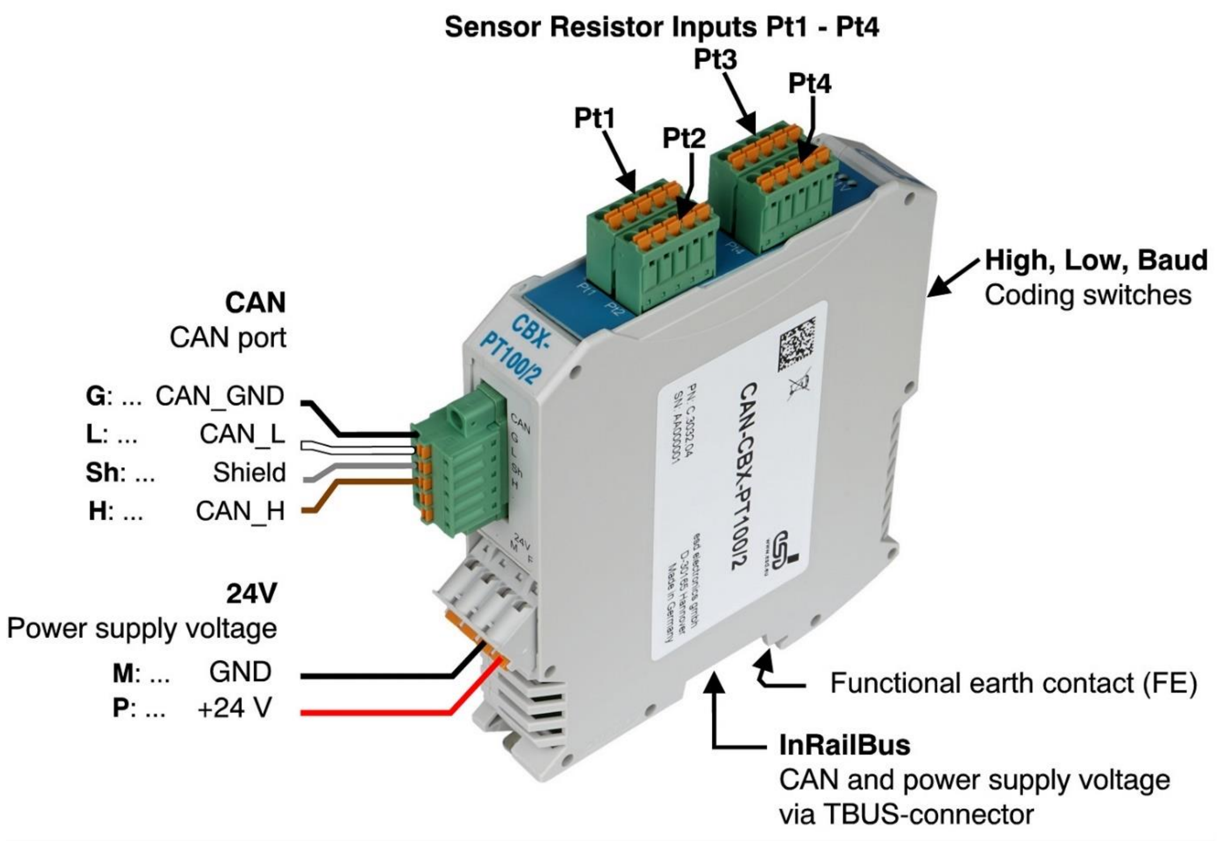


Figure 2: Connecting diagram of CAN-CBX-PT100/2

See also page 14 for examples of the configuration of the sensor resistor inputs (Pt1 – Pt4) and page 106 ff. for signal assignment of the connectors.
For conductor connection and conductor cross section see page 111

NOTICE
Read chapter "Installing and Uninstalling Hardware" from page 20, before you start with the installation or uninstallation of the hardware!

2.1.2 Temperature Sensor in 2-Wire Configuration (Example Pt2)

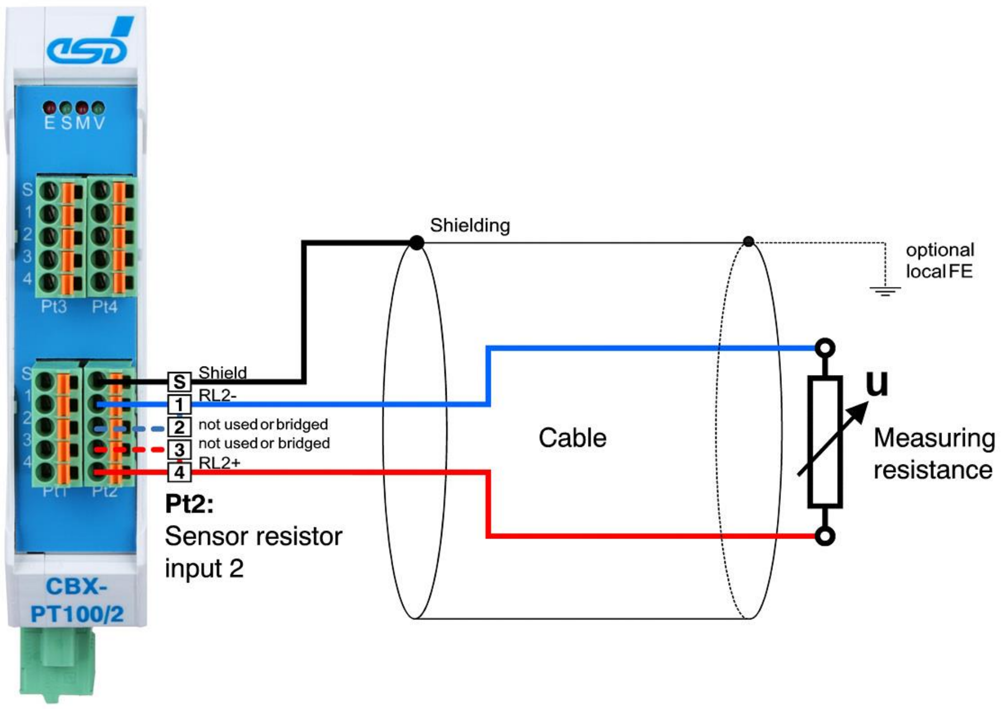


Figure 3: Temperature sensor connection in 2-wire configuration at Pt2 (Example)

For conductor connection and conductor cross section see page 111

2.1.3 Temperature Sensor in 4-Wire Configuration (Example Pt2)

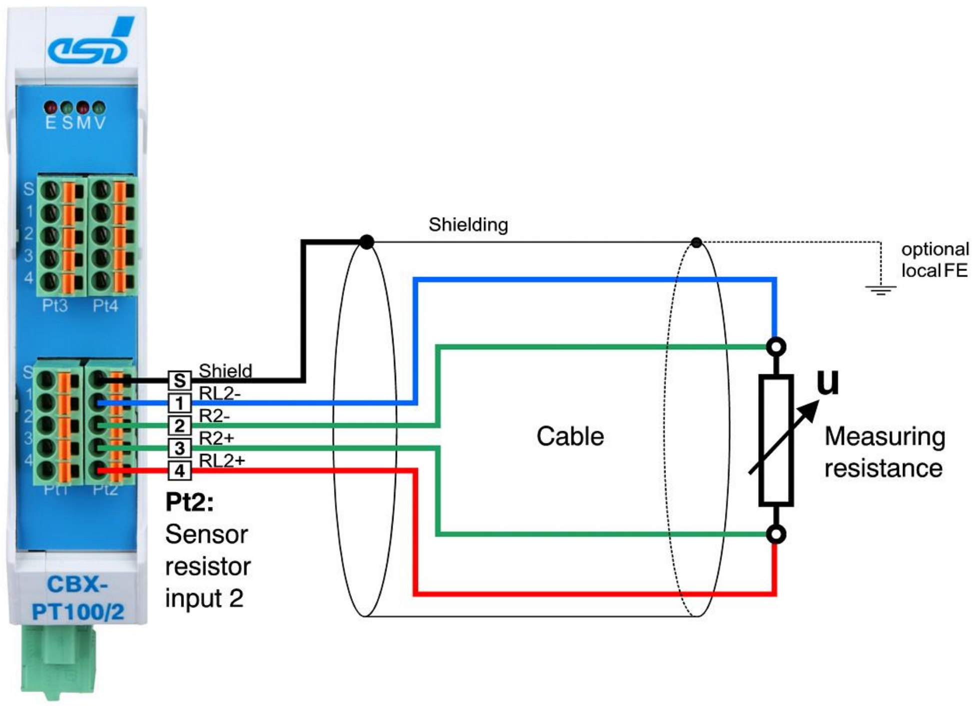


Figure 4: Temperature sensor connection in 4-wire configuration at Pt2 (Example)

For conductor connection and conductor cross section see page 111

2.2 LEDs

2.2.1 Position of the LEDs

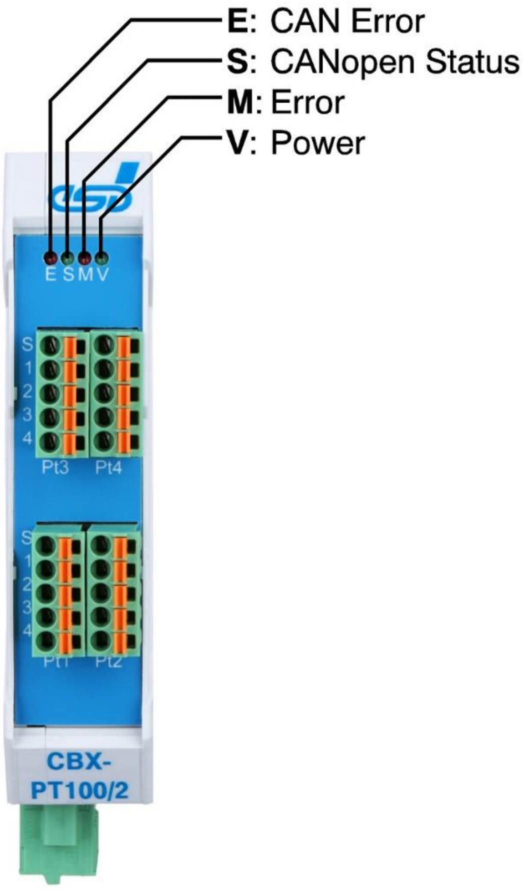


Figure 5: Position of LEDs in the front panel

The CAN-CBX-PT100/2 module is equipped with 4 status LEDs

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.

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 17)
			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	CAN Overrun Error - The sample rate is set too high, therefore, the firmware is not able to transmit all data on the CAN bus.
			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.
			Blinking	Sensor error (emergency error code 5030h) e.g.: - sensor not connected - sensor data faulty

Table 4: Indicator state of the Error-LED

2.2.6 Operation of the Module Status LED 'V'

The display functions are described in the following table.

Label	Name	Colour	Indicator state	Description
V	Power	green	Off	No power supply voltage or application software is not running
			On	Power supply voltage is switched on and application software is running

Table 5: Indicator state of the Power-LED

2.2.7 Special Indicator States

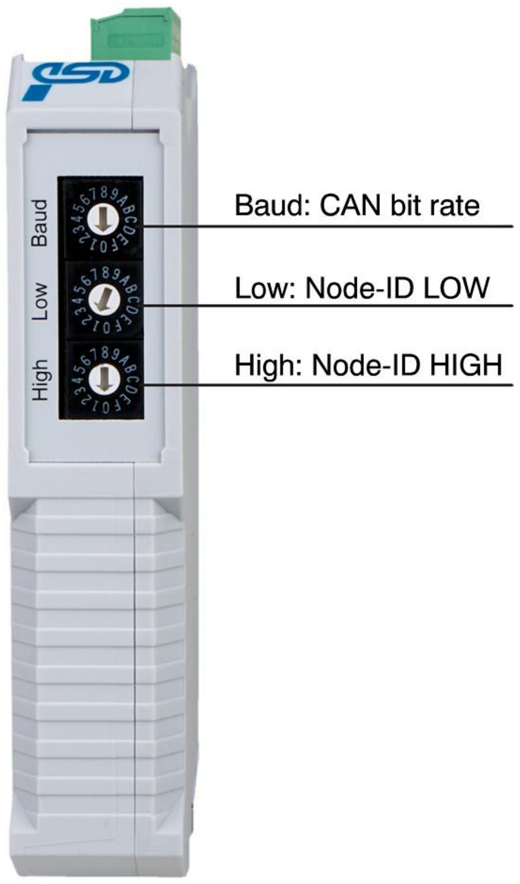
The special indicator state described in the following table is indicated by the CANopen-Status-LED and the CAN-Error-LED together:

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

2.3 Coding Switch

Figure 6: 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 setting:

- 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 7: Index of the bit rate

3 Installing and Uninstalling Hardware

To install or uninstall the CAN-CBX-PT100/2, 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-PT100/2 is to be integrated. → The CAN-CBX-PT100/2 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-PT100/2 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	20
2.	Mount the CAN-CBX-PT100/2 module. Connect the ports (power supply voltage, CAN, sensor resistor inputs).	13
	See also chapter 6 for Connector Assignments	106
	If the InRailBus is used, read and follow chapter Using InRailBus	22
❗	**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.	107, 111
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".	112

4.	Set the configuration switches according to your needs or program the CAN-CBX-PT100/2 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.	19
5.	Set the module number (Node-ID), see chapter 2.3.1. The node-ID can be configured via the coding switches LOW and 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'.	18
6.	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-PT100/2.	-
7.	Write “-1” (0xFFFFFFFF) in object 0x9133 (sub-index 1...4) (With every new A/D-conversion a TPDO with the measured value is sent.)	76
8.	Send the NMT Start Command to the module The PDOs are sent on The Process-Value (PV) is sent in the data bytes 0...3. The default unit of the PV is mOhm. E.g.: 1234567 equates to 1234.567 Ohm	32

CAN Identifier	Len	Data	
0	2 bytes	1	0 (Node-ID or 0 = start all modules)

CAN Identifier	Len	Data
0x180 + Node-ID 0x280 + Node-ID 0x380 + Node-ID 0x480 + Node-ID	4 bytes	Process Value 1 Process Value 2 Process Value 3 Process Value 4

3.2 Uninstalling the Hardware

Step	Procedure	See Page
1.	Follow the safety instructions at the beginning of chapter 3	20
2.	Make sure that all connected interfaces and power supply are switched off.	
3.	Disconnect the CAN-CBX-PT100/2 from the connected interfaces.	
4.	Use a screwdriver to pull the fastening spring of the CAN-CBX-PT100/2. downwards while swivelling the module upwards until it comes loose.	
5.	Carefully remove the CAN-CBX-PT100/2.	

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:

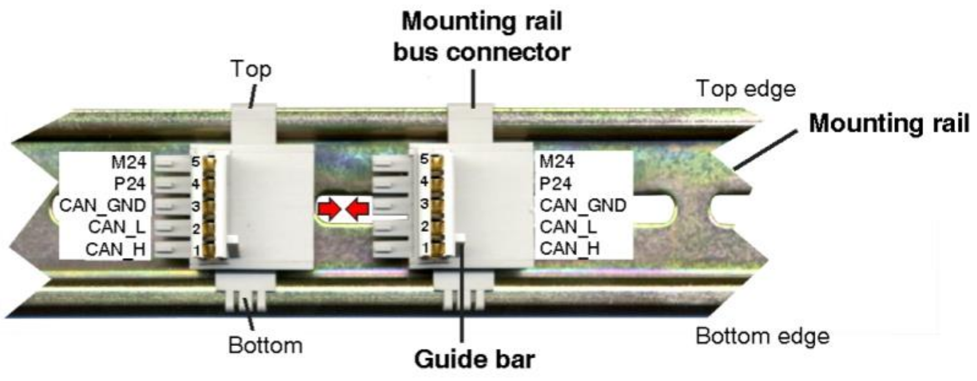


Figure 7: Mounting rail with bus connector

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.

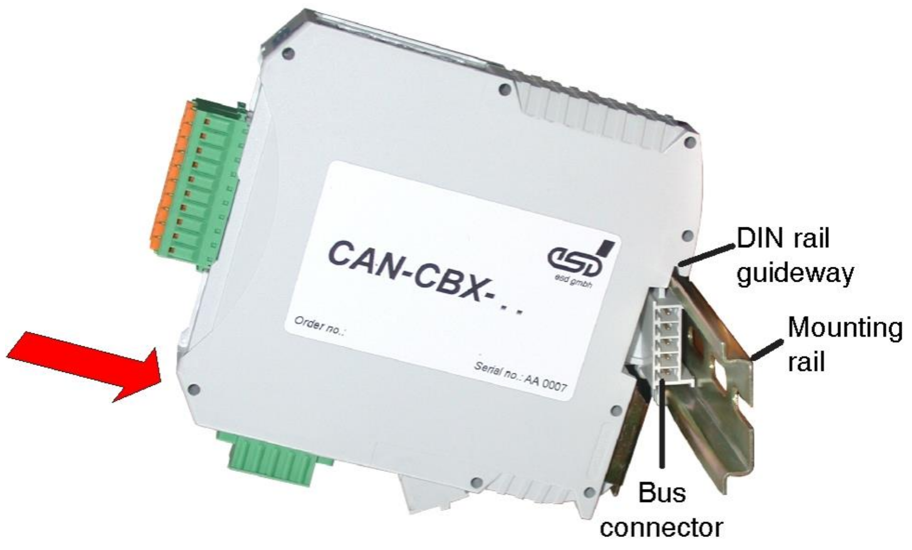


Figure 8: Mounting CAN-CBX modules

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 8. 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.

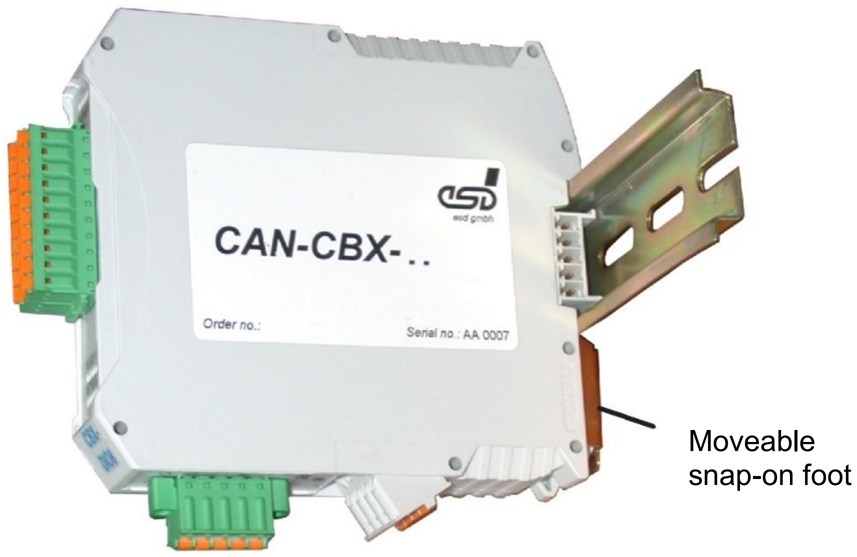


Figure 9: Mounted CAN-CBX module

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).

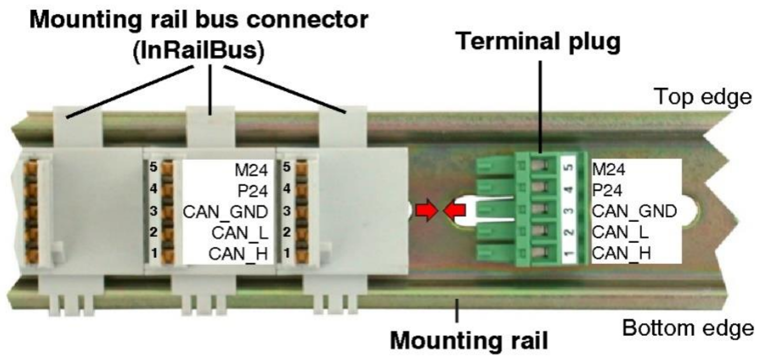


Figure 10: Mounting rail with InRailBus and terminal plug

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 10. 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-PT100/2 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.

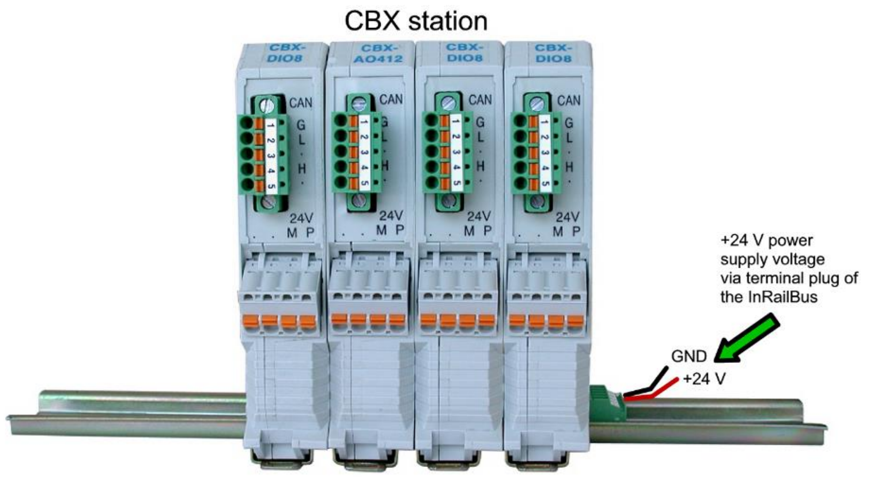

CBX station
CBX-DIO8
CBX-AO412
CBX-DIO8
CBX-DIO8
CAN
24V
M P
+24 V power supply voltage via terminal plug of the InRailBus
GND
+24 V

Figure 11: 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.

CBX station

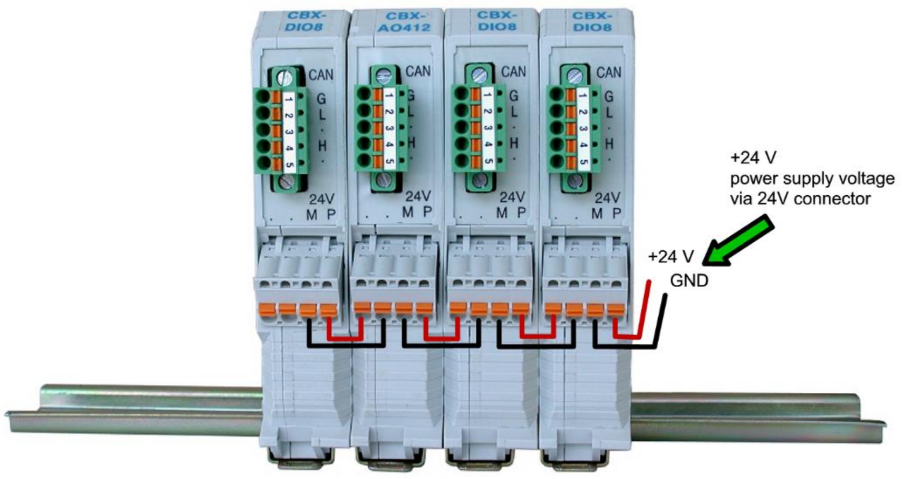


Figure 12: Connection via 24V Connector

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

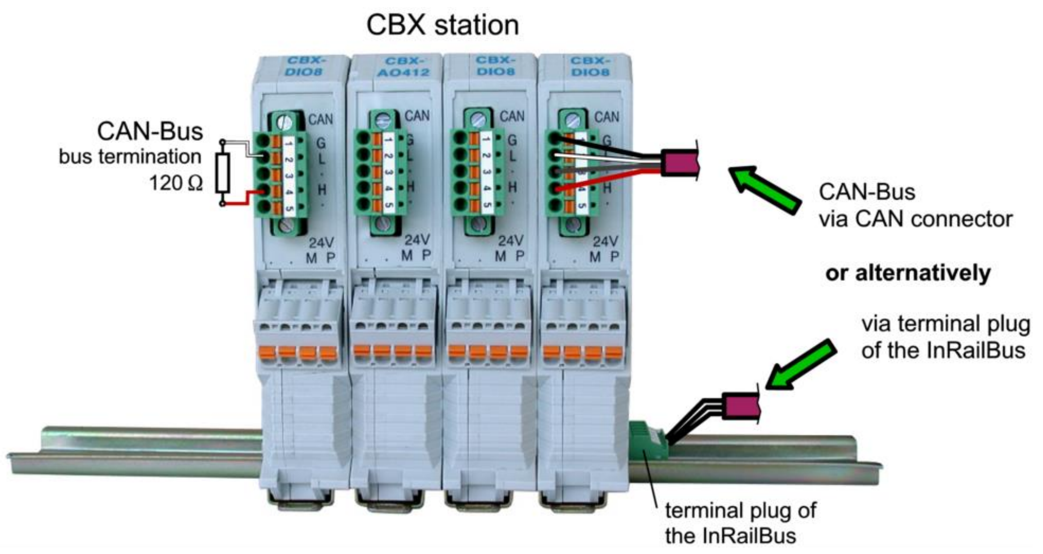


Figure 13: Connecting the CAN signals to the CAN-CBX station

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 13).

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 9) 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-PT100/2 comes with CANopen firmware according to CiA 301 (1). The generic I/O-module provides versatile integration capabilities with CiA® 404 CANopen profile (4) for profile for measuring devices

The EDS file can be downloaded under *Software Downloads* from the CAN-CBX-PT100/2 product page on esd website: https://esd.eu/en/products/can-cbx-pt100-2

For the purpose of backwards compatibility, the CAN-CBX-PT100/2 can also be operated with the EDS file of the predecessor module CAN-CBX-PT100 (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-PT100 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 = 0x0000047F			

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
0x60 (= 96 decimal):	Write Response, i.e. a parameter has been set successfully
0x80 (= 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-PT100/2 (Tx) (object 0x1800)
TPDO2	0x280 + *NodeID*	PDO from CAN-CBX-PT100/2 (Tx) (object 0x1801)
TPDO3	0x380 + *NodeID*	PDO from CAN-CBX-PT100/2 (Tx) (object 0x1802)
TPDO4	0x480 + *NodeID*	PDO from CAN-CBX-PT100/2 (Tx) (object 0x1803)
Client-SDO	0x580 + *Node-ID*	SDO from CAN-CBX-PT100/2 (Tx)
Server-SDO	0x600 + *Node-ID*	SDO to CAN-CBX-PT100/2 (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
TPDO1	0x180 + Node-ID	4 bytes	From CAN-CBX-PT100/2 (Tx/Transmit PDO)	Process value 1 (0x9130, 1)
TPDO2	0x280 + Node-ID	4 bytes		Process value 2 (0x9130, 2)
TPDO3	0x380 + Node-ID	4 bytes		Process value 3 (0x9130, 3)
TPDO4	0x480 + Node-ID	4 bytes		Process value 4 (0x9130, 4)

TPDO1 (CAN-CBX-PT100/2 ->)
CAN Identifier: 0x180 + Node-ID

Byte	0	1	2	3
Parameter	Process value 1			

TPDO2 (CAN-CBX-PT100/2 ->)
CAN Identifier: 0x280 + Node-ID

Byte	0	1	2	3
Parameter	Process value 2			

TPDO3 (CAN-CBX-PT100/2 ->)
CAN Identifier: 0x380 + Node-ID

Byte	0	1	2	3
Parameter	Process value 3			

TPDO4 (CAN-CBX-PT100/2 ->)
CAN Identifier: 0x480 + Node-ID

Byte	0	1	2	3
Parameter	Process value 4			

4.7 Reading the analog Values

4.7.1 Message of the Analog Inputs

The following transmission types are available at the CAN-CBX-PT100/2:

- 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).
- event controlled, asynchronous: The transmission is initiated if the state of selected inputs has changed (PDO-transmission type 254, 255).

4.7.2 Supported Transmission Types Based on CiA 301

Trans-mission Type	PDO-Transmission					supported by CAN-CBX-PT100/2
	cyclic	acyclic	synchro-nous	asynchro-nous	RTR	
0		X	X			x
1...240	X		X			x
241...253	reserved					-
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	Highest usable Sub-index	Description	Data type	Access mode	Product-specific properties
0x1000	0x0	Device Type	unsigned 32	ro	Value of device type: 0x0003 0191
0x1001	0x0	Error Register	unsigned 8	ro	Error bits supported by CAN-CBX-PT100/2: 4: communication error 7: manufacturer
0x1003	0xA	Pre-Defined-Error-Field	unsigned 32	rw	0x00
0x1005	0x0	COB-ID-Sync	unsigned 32	rw	0x80
0x1006	0x0	Communication Cycle Period	unsigned 32	rw	0x00
0x1008	0x0	Manufacturer Device Name	visible string	ro	“CAN-CBX-PT100/2”
0x1009	0x0	Manufacturer Hardware Version	visible string	ro	x.yy (depending on version)
0x100A	0x0	Manufacturer Software Version	visible string	ro	x.yy (depending on version)
0x100B	0x0	Node-ID	unsigned 32	ro	-
0x100C	0x0	Guard Time	unsigned 16	rw	0x0000
0x100D	0x0	Life Time Factor	unsigned 8	rw	0x00
0x100E	0x0	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	0x0	COB-ID Emergency Object	unsigned 32	rw	0x80 + Node-ID
0x1015	0x0	Inhibit Time EMCY	unsigned 16	rw	0x00
0x1016	0x1	Consumer Heartbeat Time	unsigned 32	rw	0x00
0x1017	0x0	Producer Heartbeat Time	unsigned 16	rw	0x00
0x1018	0x4	Identity Object	unsigned 32	ro	Vendor Id: 0x00000017 Prod. code: 0x23032004, (C.3032.04) Rev. number: e.g. 0x10, Serial number depending on individual module
0x1019	0x0	Synchronous Counter Overflow	unsigned 8	rw	0x00
0x1020	0x2	Verify Configuration	unsigned 32	rw	Configuration date and time
0x1029	0x1	Error Behaviour	unsigned 8	rw	Supported error classes: 1

Index	Highest usable Sub-index	Description	Data type	Access Mode
0x1800	0x5	1. Transmit PDO Parameter	PDO CommPar (0x20)	rw
0x1801	0x5	2. Transmit PDO Parameter	PDO CommPar (0x20)	rw
0x1802	0x5	3. Transmit PDO Parameter	PDO CommPar (0x20)	rw
0x1803	0x5	4. Transmit PDO Parameter	PDO CommPar (0x20)	rw
0x1A00	0x8	1. Transmit PDO Mapping	PDO Mapping (0x21)	rw
0x1A01	0x8	2. Transmit PDO Mapping	PDO Mapping (0x21)	rw
0x1A02	0x8	3. Transmit PDO Mapping	PDO Mapping (0x21)	rw
0x1A03	0x8	4. Transmit PDO Mapping	PDO Mapping (0x21)	rw

Index	Highest usable Sub-index	Description	Data type	Access mode	Product-specific properties
0x1F80	0x0	NMT startup	unsigned 32	rw	default: 2 (autostart disabled)
0x1F91	0x1	Self-starting nodes timing parameters	unsigned 16	rw	default: 0x64 (= 100 ms)

4.9.2 Device Type (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 35)

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 Read Request	0x00	0x10	0x00 Sub-Index	0x00	0x00	0x00	0x00
				Index=0x1000						

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 Read Response	0x43	0x00	0x10	0x00 Sub Index	0x94	0x01	0x02	0x00
				Index=0x1000			Example here: Device Profile No. 0x0194		Example here: Input 0x0002	

Value of the device type in the example above: 0x00020194.
The value of the device type of the CAN-CBX-PT100/2 module is printed in chapter 'Overview Communication Profile Objects / Product-Specific Values' (page 35).

The data field is always structured following the rule: 'LSB first, MSB last', see **Data Field** page 29.

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

Structure of the error register as defined in CiA 301:

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

For a list of the error bits supported by the CAN-CBX-PT100/2 module see in chapter 'Overview Communication Profile Objects / Product-Specific Values' (page 35).

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 ... 0xFFFFFFFF	0	unsigned 32	ro
	2	*error-code (n-1)*	0 ... 0xFFFFFFFF	0	unsigned 32	ro
	:	*:*	:	:	:	:
	0xA	*error-code (n-9)*	0 ... 0xFFFFFFFF	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

Emergency Message

The data of 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 35)

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	**0x1006**
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
0x00000001 ... 0xFFFFFFFF	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 35)

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.

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.

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

The CAN-CBX-PT100/2 supports the node guarding or alternatively the heartbeat function (see page 53).

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-PT100/2 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 ... 0x007FFFFF	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 44).
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 ... 0xFFFFFFFF	0x00000017	unsigned 32	ro
	2	*product_code*	0 ... 0xFFFFFFFF	(see page 35 for product specific value)	unsigned 32	ro
	3	*revision_number*	0 ... 0xFFFFFFFF	0x10	unsigned 32	ro
	4	*serial_number*	0 ... 0xFFFFFFFF	-	unsigned 32	ro

Description of the variables:

vendor_id This variable contains the esd-vendor-ID. This is always 0x00000017.

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 35 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 ... 0xFFFFFFFF	0	unsigned 32	rw
	2	*configuration_time*	0 ... 0xFFFFFFFF	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	1	unsigned 8	ro
	1	*communication_error*	0...2	0	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*

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 Transmit PDO1 Communication Parameter 0x1800 - 0x1803

This object defines the parameters of a transmit PDO1.

INDEX	**0x1800 – 0x1803**
Name	*transmit PDO parameter*
Data type	PDOCommPar

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x1800	0	*number_of_entries*	5	5		unsigned 8	ro
	1	*COB-ID used by PDO1*	1 ... 0xC00007FF	0x40000180 +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
0x1801	0	*number_of_entries*	5	5		unsigned 8	ro
	1	*COB-ID used by PDO2*	1 ... 0xC00007FF	0x40000280 +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
0x1802	0	*number_of_entries*	5	5		unsigned 8	ro
	1	*COB-ID used by PDO3*	1 ... 0xC00007FF	0x40000380 +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
0x1803	0	*number_of_entries*	5	5		unsigned 8	ro
	1	*COB-ID used by PDO4*	1 ... 0xC00007FF	0x40000480 +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, 254 and 255 are supported.

NOTICE
The value of *COB-ID used by PDOx* is RTR-disabled (0x40000xxx) by default!

4.9.23 Transmit PDO1 Mapping Parameter 0x1A00 – 0x1A03

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

INDEX	**0x1A00 - 0x1A03**
Name	*transmit PDO 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_entries*	0 ... 8	1	unsigned 8	rw
	1	*process_value_1*	0 ... 0xFFFFFFFF	0x91300120	unsigned 32	rw
	2	*(field_value_1)*	0 ... 0xFFFFFFFF	(0x91000120)	unsigned 32	rw
	3	*(mapping_for_object_3)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	4	*(mapping_for_object_4)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	5	*(mapping_for_object_5)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	6	*(mapping_for_object_6)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	7	*(mapping_for_object_7)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	8	*(mapping_for_object_8)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
0x1A01	0	*number_of_entries*	0 ... 8	1	unsigned 8	rw
	1	*process_value_2*	0 ... 0xFFFFFFFF	0x91300220	unsigned 32	rw
	2	*(field_value_2)*	0 ... 0xFFFFFFFF	(0x91000220)	unsigned 32	rw
	3	*(mapping_for_object_3)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	4	*(mapping_for_object_4)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	5	*(mapping_for_object_5)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	6	*(mapping_for_object_6)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	7	*(mapping_for_object_7)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	8	*(mapping_for_object_8)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
0x1A02	0	*number_of_entries*	0 ... 8	1	unsigned 8	rw
	1	*process_value_3*	0 ... 0xFFFFFFFF	0x91300320	unsigned 32	rw
	2	*(field_value_3)*	0 ... 0xFFFFFFFF	(0x91000320)	unsigned 32	rw
	3	*(mapping_for_object_3)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	4	*(mapping_for_object_4)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	5	*(mapping_for_object_5)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	6	*(mapping_for_object_6)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	7	*(mapping_for_object_7)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	8	*(mapping_for_object_8)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
0x1A03	0	*number_of_entries*	0 ... 8	1	unsigned 8	rw
	1	*process_value_4*	0 ... 0xFFFFFFFF	0x91300420	unsigned 32	rw
	2	*(field_value_4)*	0 ... 0xFFFFFFFF	(0x91000420)	unsigned 32	rw
	3	*(mapping_for_object_3)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	4	*(mapping_for_object_4)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	5	*(mapping_for_object_5)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	6	*(mapping_for_object_6)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	7	*(mapping_for_object_7)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw
	8	*(mapping_for_object_8)*	0 ... 0xFFFFFFFF	0	unsigned 32	rw

Only the following objects can be mapped:	0x9100, 0x9130, 0x2801, 0x2802

Structure of the PDO mapping using the example of object 0x1A00, sub-index 1:

31 16	15 8	7 0
Index e.g.: 0x9130	Subindex e.g.: 0x01	Length e.g.: 0x20
MSB		LSB

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 (4.9.22) must be set to *not valid*.

4.9.24 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**
0x00000002	Auto startup disabled (default)
0x00000008	Auto startup enabled
all other values	reserved

4.9.25 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 0x6110 – 0x9135

Index	Name	Data type
0x6110	*AI_sensor_type*	unsigned 16
0x6111	*AI_autocalibration*	unsigned 32
0x6112	*AI_operating_mode*	unsigned 8
0x6114	*AI_ADC_sampling_rate*	unsigned 32
0x6131	*AI_physical_unit_PV*	unsigned 32
0x6132	*AI_decimal_digits_PV*	unsigned 8
0x6150	*AI_status*	unsigned 8
0x9100	*AI_FV*	integer 32
0x9103	*AI_interrupt_delta_input_FV*	integer 32
0x9130	*AI_PV*	integer 32
0x9133	*AI_interrupt_delta_input_PV*	integer 32
0x9134	*AI_interrupt_lower_limit_PV*	integer 32
0x9135	*AI_interrupt_upper_limit_PV*	integer 32

An overview of the cooperation of the objects of the “Device Profile Area” (0x61xx and 0x91xx) and the
“Manufacturer Specific Profile Area” (0x24xx) is shown in the diagram on the following page.

4.10.2 Interrelation of the Implemented Objects

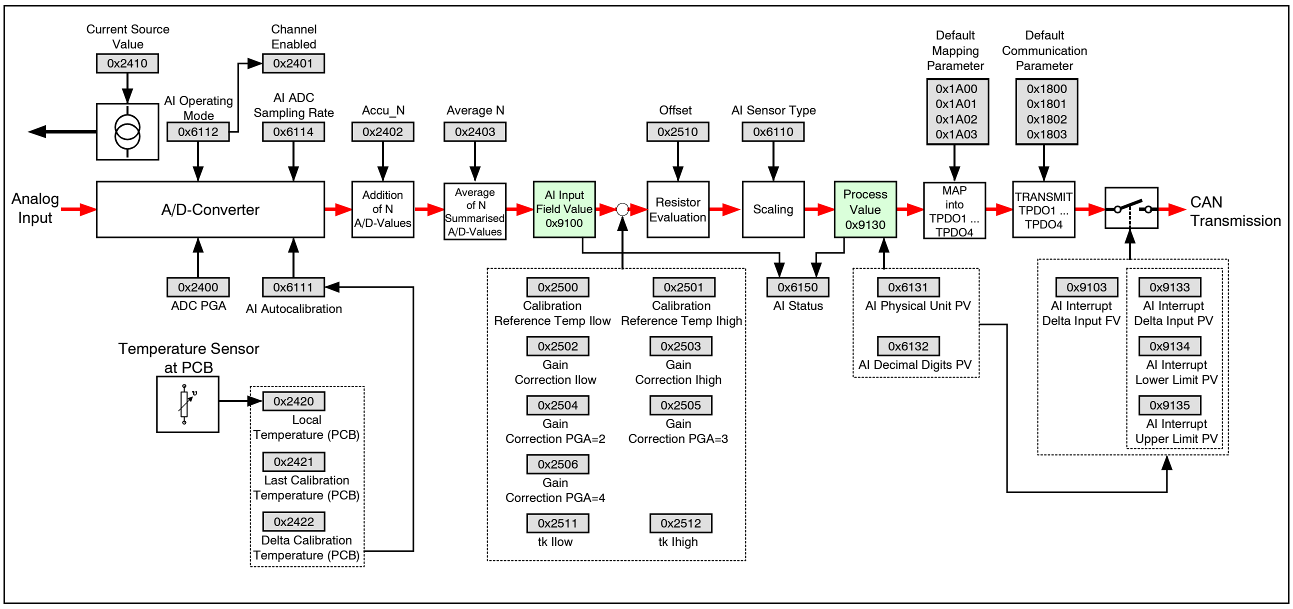


Figure 14: Relationship between the implemented objects

Example here: The module is in the “operational” status

4.10.3 AI Sensor Type (0x6110)

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x6110	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*AI_sensor_type_1*	0x1E ... 0x2732 (30 … 10034)	0x2710 (10000)	unsigned 16	rw
	2	*AI_sensor_type_2*	0x1E ...0x2732	0x2710	unsigned 16	rw
	3	*AI_sensor_type_3*	0x1E ... 0x2732	0x2710	unsigned 16	rw
	4	*AI_sensor_type_4*	0x1E ... 0x2732	0x2710	unsigned 16	rw

Description of the parameter *AI_sensor_type_x* (x = 1...4):

The parameter contains the type of the connected temperature sensor. The end number of the variable name is the number of the measuring channel.

Value range:

Value of the parameters		Type of sensor resistor
30	(0x1E)	Pt100 temperature sensor
31		Pt200 temperature sensor
32		Pt500 temperature sensor
33		Pt1000 temperature sensor
34		Pt5000 temperature sensor
10000	(0x2710)	Resistance measurement
10030		Ni100 temperature sensor
10031		Ni200 temperature sensor
10032		Ni500 temperature sensor
10033		Ni1000 temperature sensor
10034	(0x2732)	Ni5000 temperature sensor

Table 8: Type of sensor resistor

NOTICE
If you connect temperature sensors, please ensure that you set the type of temperature sensor in this parameter.
By default, the parameters contain the value for the resistance measurement (*AI_sensor_type_x* = 10000 (0x2710)).

4.10.4 AI Autocalibration (0x6111)

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x6111	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*AI_autocalibration_1*	-	no default, write: 0x69 6C 61 63 (= ASCII: ‘i’ ‘l’ ‘a’ ‘c’)	unsigned 32	wo
	2	*AI_autocalibration_2*	-		unsigned 32	wo
	3	*AI_autocalibration_3*	-		unsigned 32	wo
	4	*AI_autocalibration_4*	-		unsigned 32	wo

Function:
Writing the ASCII-strings ‘cali’ leads to the automatic-calibration of the corresponding channel.

The automatic calibration starts automatically:

- After detection of new temperature sensors

- After change of the parameter
  - Sampling-Rate *AI_ADC_sampling_rate* (object 0x6114) or
  - Gain *ADC_PGA* (object 0x2400)
  - Measuring current *current_source_value* (object 0x2410)

- After exceeding a defined change in temperature of the circuit board since the last calibration (if enabled)
  - Temperature measured during the last calibration
    last_calibration_temp (object 0x2421)
  - Maximum allowed temperature difference since the last calibration
    delta_calibration_temp (object 0x2422)

4.10.5 AI Operating Mode (0x6112)

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x6112	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*AI_operating_mode_1*	0, 1	1	unsigned 8	rw
	2	*AI_operating_mode_2*	0, 1	1	unsigned 8	rw
	3	*AI_operating_mode_3*	0, 1	1	unsigned 8	rw
	4	*AI_operating_mode_4*	0, 1	1	unsigned 8	rw

Description of parameter *AI_operating_mode_x* (x = 1...4):

The parameter selects the desired operation mode. It contains the “nominal-value” of the operating mode. The “actual value” can be read via object 0x2401.

Value range:

Value of the parameter	Operation mode of the channel
0	channel disabled
1	normal operation

4.10.6 AI ADC Sampling Rate (0x6114)

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x6114	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*AI_ADC_sampling_rate_1*	0x03E8 ... 0x61A80	0x61A80	unsigned 32	rw
	2	*AI_ADC_sampling_rate_2*	0x03E8 ... 0x61A80	0x61A80	unsigned 32	rw
	3	*AI_ADC_sampling_rate_3*	0x03E8 ... 0x61A80	0x61A80	unsigned 32	rw
	4	*AI_ADC_sampling_rate_4*	0x03E8 ... 0x61A80	0x61A80	unsigned 32	rw

Description of the parameter *AI_ADC_sampling_rate_x* (x = 1...4):

This parameter can be used to set the sampling rates of the individual channels independently of each other. The resolution is 1 µs. The following values are supported:

Value range:

Allowed parameter values *1)		Sample time	Sampling rate
1000	(0x03E8)	1 ms	1000 Hz
2000		2 ms	500 Hz
10000		10 ms	100 Hz
16667		16.667 ms	60 Hz
20000	(0x4E20)	20 ms	50 Hz
33333		33.333 ms	30 Hz
40000		40 ms	25 Hz
66667		66.667 ms	15Hz
100000		100 ms	10 Hz
200000		200 ms	5 Hz
400000	(0x61A80)	400 ms	2.5 Hz

*1) If a value is transmitted which is not included in the list of permissible parameters, it is rounded to the next permissible value.

NOTICE
To achieve a high resolution, it is advisable to use sampling times that are a multiple of 20 ms (50 Hz), as this reduces the interference caused by the mains frequency!

4.10.7 AI Physical Unit (0x6131)

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x6131	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*AI_physical_unit_1*	0x280000, 0x2D0000	$\text{unit}_{\text{SensorType}}$_1	unsigned 32	ro
	2	*AI_physical_unit_2*	0x280000, 0x2D0000	$\text{unit}_{\text{SensorType}}$_2	unsigned 32	ro
	3	*AI_physical_unit_3*	0x280000, 0x2D0000	$\text{unit}_{\text{SensorType}}$_3	unsigned 32	ro
	4	*AI_physical_unit_4*	0x280000, 0x2D0000	$\text{unit}_{\text{SensorType}}$_4	unsigned 32	ro

The default values $\text{unit}_{\text{SensorType}}$_x (x = 1 - 4) depend on the set sensor-type: With the default-setting of object 0x6110 =10000 (0x2710= resistance measurement), the value is 2621440 = resistance in Ohm.

Description of the variable *AI_physical_unit_x* (x = 1...4):
These variables contain the physical units and prefixes for the measuring channels used. The value of the variables is determined by the sensor type specified in object "*AI_sensor_type*" (object 0x6110).

The structure of the variable is defined in CiA 404. The coding of the physical units and prefixes is done according to CiA 303-2 (7).

Structure of the variable:

31 … … 24	23 … … 16	15 … … 8	7 … … 0
Prefix	SI Numerator	SI Denominator	reserved
MSB			LSB

Coding of the prefix for physical units:

Prefix	Factor	Notation index
-	10^0	0x00

The SI Numerator and the SI Denominator contain the physical units.

Coding of the physical units:

International symbol	Name of unit	Notation index
Ω	Ohm	0x28
°C	degree Celsius	0x2D
non-dimensional	none	0x00

Value range:

Sensor type	Physical unit
Pt100 ... Pt1000	Degree Celsius (Variable value = 0x002D0000)
Ni100 ... Ni5000	
electric resistance measurement	Ohm (Variable value = 0x00280000)

4.10.8 AI Decimal Digits PV (0x6132)

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x6132	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*AI_decimal_digits_PV_1*	0 ... 3	3	unsigned 8	rw
	2	*AI_decimal_digits_PV_2*	0 ... 3	3	unsigned 8	rw
	3	*AI_decimal_digits_PV_3*	0 ... 3	3	unsigned 8	rw
	4	*AI_decimal_digits_PV_4*	0 ... 3	3	unsigned 8	rw

Description of the parameter *AI_decimal_digits_PV_x* (x = 1...4):

This parameter contains the number of fractional digits of the process value (PV) in object "*AI_PV*" (0x9130).
The value of the parameter gives the number of the decimal digits.

Example:

AI_decimal_digits_PV_1 = 3
Value in object *AI_PV_1* = 123456 (decimal)

Measured value: 123.456 Ω (at default setting (electric resistance measurement) see object 0x6110).

4.10.9 AI Status (0x6150)

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x6150	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*AI_status_1*	0 ... 5	-	unsigned 8	ro
	2	*AI_status_2*	0 ... 5	-	unsigned 8	ro
	3	*AI_status_3*	0 ... 5	-	unsigned 8	ro
	4	*AI_status_4*	0 ... 5	-	unsigned 8	ro

Description of the variable *AI_status_x* (x = 1...4):
These variables return the state of the corresponding analog input channel.

Bit:	7	6	5	4	3	2	1	0
Meaning:	reserved *2)	not supported *2)			reserved *2)	Negative Overload	Positive Overload	Not Valid

*2) These bits are always returned as ‘0’.

Value range:

Description	Negative overload limit	Positive overload limit
PT-sensors	-200.715 °C	853.498 °C
Ni-sensors	-200.000 °C	399.138 °C
electric resistance measurement at default setting	-	-

A bit is active (set to ‘1’), if the lower or the upper limit is exceeded. The bits 1 and 2 cannot be set at the same time.

If no sensor is connected, only bit “Not Valid” is set.
If the temperature exceeds the range, bits “Not Valid” AND the corresponding “Overload”-bit are set.

Valid return messages are e.g.:

AI_Status_1 = 0x00: no error
AI_Status_1 = 0x03: positive overload and sensor value not valid (e.g. no sensor identified)

4.10.10 Analog Input Field Value (0x9100)

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x9100	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*AI_FV_1*	0x80000000 ... 0x7FFFFFFF	-	integer 32	ro
	2	*AI_FV_2*	0x80000000 ... 0x7FFFFFFF	-	integer 32	ro
	3	*AI_FV_3*	0x80000000 ... 0x7FFFFFFF	-	integer 32	ro
	4	*AI_FV_4*	0x80000000 ... 0x7FFFFFFF	-	integer 32	ro

Description of the variable *AI_FV_x* (x = 1...4):

These variables contain the uncorrected “Raw values” of the A/D-converter.

Value range:

AI_FV_x = *0x*80000000 ... 0x7FFFFFFF

4.10.11 AI Interrupt Delta Input FV (0x9103)

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x9103	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*AI_interrupt_delta_input_FV_1*	0x80000000 ... 0x7FFFFFFF	0	integer 32	rw
	2	*AI_interrupt_delta_input_FV_2*	0x80000000 ... 0x7FFFFFFF	0	integer 32	rw
	3	*AI_interrupt_delta_input_FV_3*	0x80000000 ... 0x7FFFFFFF	0	integer 32	rw
	4	*AI_interrupt_delta_input_FV_4*	0x80000000 ... 0x7FFFFFFF	0	integer 32	rw

Description of the parameter *AI_interrupt_delta_input_FV_x* (x = 1...4):

If a field-value is mapped on a PDO and the deviation of the field value is higher than specified in *AI_interrupt_delta_input_FV_*x, this PDO is transmitted.

Value range:

AI_interrupt_delta_input_FV_x	Meaning
0x80000000 ... 0xFFFFFFFF	For negative values of the parameter a PDO-transmission is initiated with every change of the field value.
0	No comparison of the field values and thus no transmission
0x00000001 .... 0x7FFFFFFF	A PDO-transmission will only be initiated, if the deviation of the field value exceeds the value specified here.

4.10.12 Analog Input Process Value (0x9130)

NOTICE
This object contains the important data (resistance or temperature values)!

Index	Sub-index	Description	Value range	Default value	Data type	PDO Mapping	Access mode
0x9130	0	*number_of_entries*	4	4	unsigned 8	-	ro
	1	*AI_PV_1*	0x80000000 ... 0x7FFFFFFF	-	integer 32	1	ro
	2	*AI_PV_2*	0x80000000 ... 0x7FFFFFFF	-	integer 32	1	ro
	3	*AI_PV_3*	0x80000000 ... 0x7FFFFFFF	-	integer 32	1	ro
	4	*AI_PV_4*	0x80000000 ... 0x7FFFFFFF	-	integer 32	1	ro

Description of the variable *AI_PV_x* (x = 1...4):

Theses variables return the corrected, measured values of the four channels. The physical units of the values are defined in "*AI_physical_unit_x*" (object 0x6131, see page 70).

The number of decimal places of the variable is defined in object "*AI_decimal_digits_PV_x*" (object 0x6132, see page 71).

Value range:

AI_PV_x = *0x*80000000 ... 0x7FFFFFFF

4.10.13 AI Interrupt Delta Input PV (0x9133)

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x9133	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*AI_interrupt_delta_input_PV_1*	0x80000000 ... 0x7FFFFFFF	0	integer 32	rw
	2	*AI_interrupt_delta_input_PV_2*	0x80000000 ... 0x7FFFFFFF	0	integer 32	rw
	3	*AI_interrupt_delta_input_PV_3*	0x80000000 ... 0x7FFFFFFF	0	integer 32	rw
	4	*AI_interrupt_delta_input_PV_4*	0x80000000 ... 0x7FFFFFFF	0	integer 32	rw

Description of the parameter *AI_interrupt_delta_input_PV_x* (x = 1...4):

If a process variable is mapped on a PDO and the deviation of the process variable is higher than specified in *AI_interrupt_delta_input_PV_*x, this PDO is transmitted.

The number of decimal places of the variable is defined in object "*AI_decimal_digits_PV_x*" (object 0x6132, see page 71).

Value range:

*AI_interrupt_delta_input_PV_*x	Meaning
0x80000000 ... 0xFFFFFFFF	For negative values of the parameter a PDO-transmission is initiated with every change of the process values
0	No comparison of the process values and thus no transmission
0x00000001 .... 0x7FFFFFFF	A PDO-transmission will only be initiated, if the deviation of the process values exceeds the value specified here.

4.10.14 AI Interrupt Lower Limit PV (0x9134)

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x9134	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*AI_interrupt_lower_limit_PV_1*	0x80000000 ... 0x7FFFFFFF	0x80000000	integer 32	rw
	2	*AI_interrupt_lower_limit_PV_2*	0x80000000 ... 0x7FFFFFFF	0x80000000	integer 32	rw
	3	*AI_interrupt_lower_limit_PV_3*	0x80000000 ... 0x7FFFFFFF	0x80000000	integer 32	rw
	4	*AI_interrupt_lower_limit_PV_4*	0x80000000 ... 0x7FFFFFFF	0x80000000	integer 32	rw

Description of the parameter *AI_interrupt_lower_limit_PV_x* (x = 1...4):

If a process variable is mapped on a PDO and the deviation of the process variable is lower than specified in *AI_interrupt_lower_limit_PV_*x, this PDO is transmitted.

The number of decimal places of the variable is defined in object "*AI_decimal_digits_PV_x*" (object 0x6132, see page 71).

Value range:

*AI_interrupt_lower_limit_PV_*x = *0x*80000000 ... 0x7FFFFFFF

4.10.15 AI Interrupt Upper Limit PV (0x9135)

Index	Sub-index	Description	Value range	Default value	Data type	Access mode
0x9135	0	*number_of_entries*	4	4	unsigned 8	ro
	1	*AI_interrupt_upper_limit_PV_1*	0x80000000 ... 0x7FFFFFFF	0x7FFFFFFF	integer 32	rw
	2	*AI_interrupt_upper_limit_PV_2*	0x80000000 ... 0x7FFFFFFF	0x7FFFFFFF	integer 32	rw
	3	*AI_interrupt_upper_limit_PV_3*	0x80000000 ... 0x7FFFFFFF	0x7FFFFFFF	integer 32	rw
	4	*AI_interrupt_upper_limit_PV_4*	0x80000000 ... 0x7FFFFFFF	0x7FFFFFFF	integer 32	rw

Description of the parameter *AI_interrupt_upper_limit_PV_x* (x = 1...4):

If a process variable is mapped on a PDO and the deviation of the process variable is higher than specified in *AI_interrupt_upper_limit_PV_*x, this PDO is transmitted.

The number of decimal places of the variable is defined in object "*AI_decimal_digits_PV_x*" (object 0x6132, see page 71).

Value range:

*AI_interrupt_upper_limit_PV_*x = *0x*80000000 ... 0x7FFFFFFF

4.11 Manufacturer Specific Profile Area

4.11.1 Implemented Objects 0x2210 – 0x2510

Index	Name	Data Type
0x2400	*ADC_PGA*	unsigned 8
0x2401	*Channel_enabled*	unsigned 8
0x2402	*Accu_N*	unsigned 8
0x2403	*Average_N*	unsigned 8
0x2410	*Measuring_current*	unsigned 8
0x2420	*Local_temperature (PCB)*	integer 16
0x2421	*Last_calibration_temperature (PCB)*	integer 16
0x2422	*Delta_calibration_temperature (PCB)*	integer 16
0x2500	*Ref_temp_Ilow*	integer 16
0x2501	*Ref_temp_Ihigh*	integer 16
0x2502	*Gain_correction_Ilow*	integer 16
0x2503	*Gain_correction_Ihigh*	integer 16
0x2504	*Gain_correction_PGA=2*	integer 16
0x2505	*Gain_correction_PGA=3*	integer 16
0x2506	*Gain_correction_PGA=4*	integer 16
0x2510	*Offset*	integer 16
0x2511	*tk_Ilow*	integer 16
0x2512	*tk_Ihigh*	integer 16
0x2800	*Calc_Expression*	string (80)
0x2801	*Calc_Values_Float*	real 32
0x2802	*Calc_Value_Int64*	integer 64
0x2803	*Calc_Const*	real 32
0x2804	*Calc_used_Var_Expr*	unsigned 32
0x2805	*Calc_Duration*	unsigned 32

See also diagram “Figure 14: Relationship between the implemented objects” on page 65.

4.11.2 ADC_PGA (0x2400)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2400	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*ADC_PGA_1*	1 ... 4	3	0	unsigned 8	rw
	2	*ADC_PGA_2*	1 ... 4	3	0	unsigned 8	rw
	3	*ADC_PGA_3*	1 ... 4	3	0	unsigned 8	rw
	4	*ADC_PGA_4*	1 ... 4	3	0	unsigned 8	rw

Description of the parameter *ADC_PGA_x* (x = 1...4):
This parameter sets the gain (PGA) of the A/D-converters (ADS1255).

Value range:
The gain V_x is:

$$\boxed{V_x = 2^{ADC\_PGA\_x}}$$

The gain 2, 4, 8 and 16 can be adjusted. The gain is transparent for the user and already included in the calculation of the firmware.

NOTICE
For the connection of measuring resistors, the following voltage limits have to be considered: Voltage limitation by input circuit: U_{maxph} = 1.25 V

Depending on *ADC_PGA_x* the maximum permissible resistor values are:

Gain (ADC_PGA_x)	Input voltage U_{FS} [V]	Max. permissible electrical resistance at a measuring current of	
		ca. 400 µA	ca. 40 µA
1	± 2.5	3 kΩ	34 kΩ
2	± 1.25	3 kΩ	34 kΩ
3	± 0.6125	1.5 kΩ	17 kΩ
4	± 0.306	773 Ω	8.5 kΩ

NOTICE
The recommended gain is *ADC_PGA_x* =3.

4.11.3 Channel Enabled (0x2401)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2401	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*channel_enabled_1*	0, 1	1	0	boolean	ro
	2	*channel_enabled_2*	0, 1	1	0	boolean	ro
	3	*channel_enabled_3*	0, 1	1	0	boolean	ro
	4	*channel_enabled_4*	0, 1	1	0	boolean	ro

Description of the variable *channel_enabled_x* (x = 1...4):

If a sensor has been detected for the measuring channel, this variable returns the “actual value”of the object “*AI_Operating_mode_x*” (object 0x6112). It indicates if the corresponding channel is active.

With object 0x6150 “*AI_status_x*” it can be determined if a sensor has been detected.

Value range:

channel_enable_x	Binary value	Meaning
false	0	A/D-converter channel is off (*AI_operation_mode_x* = 0)
true	1	A/D-converter channel is on (*AI_operation_mode_x* = 1)

4.11.4 Accu N (0x2402)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2402	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*accu_count_1*	0...8	0	0	unsigned 8	rw
	2	*accu_count_2*	0...8	0	0	unsigned 8	rw
	3	*accu_count_3*	0...8	0	0	unsigned 8	rw
	4	*accu_count_4*	0...8	0	0	unsigned 8	rw

Description of the parameter *accu_count_x* (x = 1...4):

This parameter defines the number of analog values to be added. The parameter accu_count reduces the data rate while improving the resolution.
The number of accumulations is:

n ... Number of accumulations

$$\boxed{n = 2^{(accu\_count\_x)}}$$

Up to 256 values can be summed up.

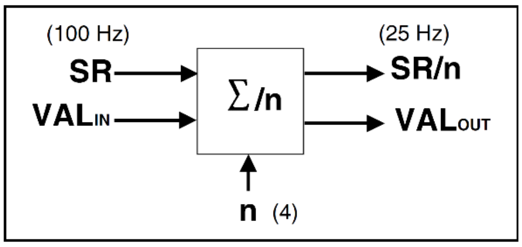
Features: • Filter with decimation
• Improvement of the resolution
• Reduction of the data rate

The data rate of the analog value is calculated as:

$$\boxed{T_{Data} = 2^n \times T\_sampling\_ADC \qquad \text{(object 0x6114)}}$$

Description of the Filter:

SR ... sample rate (e.g.:100 Hz)
n ... number of additions (e.g.:4)
SR/n... sample rate/ number of summations
(e.g.:25 Hz)



$$\boxed{VAL_{OUT}(t) = \frac{1}{n}\sum_{x=1}^{n} VAL_{IN}\,(t-n+x)}$$

4.11.5 Average N (0x2403)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2403	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*average_count_1*	0 ... 4	0	0	unsigned 8	rw
	2	*average_count_2*	0 ... 4	0	0	unsigned 8	rw
	3	*average_count_3*	0 ... 4	0	0	unsigned 8	rw
	4	*average_count_4*	0 ... 4	0	0	unsigned 8	rw

Description of the parameter *average_count_x* (x = 1...4):

This parameter defines how many analog values are used to calculate the floating average.
After every conversion a new average is available, because the A/D-values are buffered in a ring buffer.

The number of averaged values is:

m ... Number of averaged values

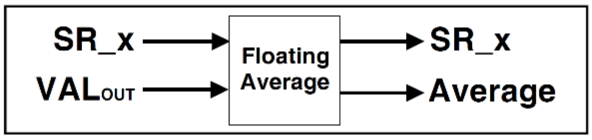
$$\boxed{m = 2^{(average\_count\_x)}}$$

Thus, it can be averaged over the last 1, 2, 4, 8 or 16 values.

Features: • Filter with decimation
• Improvement of the resolution
• A new average is available after every conversion
• Step response is $2^m \cdot T_{Data}$ (see object 0x2402)

NOTICE
The input for “Average” (object 0x2403) is “Val$_{OUT}$” configured by object 0x2402!

Description of the Filter:
SR_x ... sample rate



4.11.6 Measuring Current (0x2410)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2410	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*current_source_value_1*	0, 1	0	0	boolean	rw
	2	*current_source_value_2*	0, 1	0	0	boolean	rw
	3	*current_source_value_3*	0, 1	0	0	boolean	rw
	4	*current_source_value_4*	0, 1	0	0	boolean	rw

Description of the parameter *current_source_value_x* (x = 1...4):

With this parameter the measuring current can be selected individually for each channel.

For measuring resistors up to approximately 500 Ω resistance the measuring current can be 400 µA. For higher values the self-heating of the sensors should be noted and if necessary, the lower measuring current must be selected.
If the lower measuring current is selected, the parameter *ADC_PGA_x* (object 2400_h) must be set to “4” (for information about the selection of measuring current and measuring resistance please refer to page 81).

Value range:

current_source_value_x = 0 :	measuring current = ca. 400 µA
current_source_value_x = 1 :	measuring current = ca. 40 µA

4.11.7 Local Temperature at PCB (0x2420)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2420	0	*number_of_entries*	1	1	0	unsigned 8	ro
	1	*local_temperature*	0x8000 ... 0x7FFF		0	integer 16	ro

Description of the variable *local_temperature*:

This variable contains the value of the temperature on the circuit board in steps of 1/256 °C.

The value is coded as described in the following:

$$Temperature[°C] = \frac{Int16\ Value}{256}$$

Example:

local_temperature = 0x2640 => Temperature = 38.25 °C

4.11.8 Local Temperature at the Last Calibration (0x2421)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2421	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*last_calibration_temp_1*	0x8000 ... 0x7FFF	$\text{temp}_{\text{last_cal}}$_1	0	integer 16	ro
	2	*last_calibration_temp_2*	0x8000 ... 0x7FFF	$\text{temp}_{\text{last_cal}}$_2	0	integer 16	ro
	3	*last_calibration_temp_3*	0x8000 ... 0x7FFF	$\text{temp}_{\text{last_cal}}$_3	0	integer 16	ro
	4	*last_calibration_temp_4*	0x8000... $0x7FFF_h$	$\text{temp}_{\text{last_cal}}$_4	0	integer 16	ro

The default value $\text{temp}_{\text{last_cal}}$_x (x = 1 - 4) contains the temperature value which is measured at the last calibration.

Description of the variable *last_calibration_temp_x* (x = 1...4):

This variable contains the value of the temperature on the circuit board in steps of 1/256 °C, determined at the last calibration of the corresponding channel.

The value is coded as described in the following:

$$\boxed{Temperature[°C] = \frac{Int16\ Value}{256}}$$

Example:

last_calibration_temp_1 = 0x2640   => Temperature = 38.25 °C

4.11.9 Calibration Delta Temperature (0x2422)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2422	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*delta_calibration_temp_1*	0 ... 0xFFFF	0	0	unsigned 16	rw
	2	*delta_calibration_temp_2*	0 ... 0xFFFF	0	0	unsigned 16	rw
	3	*delta_calibration_temp_3*	0 ... 0xFFFF	0	0	unsigned 16	rw
	4	*delta_calibration_temp_4*	0 ... 0xFFFF	0	0	unsigned 16	rw

Description of the variable *delta_calibration_temp_x* (x = 1 … 4):

The ADS1255 performs a self-calibration, if the temperature of the PCB (0x2420) deviates more than the value specified in *delta_calibration_temp_x* from the *last_calib_temp_x* .

Value *delta_calibration_temp_x*	Function
0	no automatic self-calibration at deviation of the temperature
0x0001 ... 0xFFFF	automatic self-calibration at exceeding the allowed deviation of the temperature

The value is coded as:

$$\boxed{Temperature[°C] = \frac{Int16\ Value}{256}}$$

Example:

delta_calibration_temp_1 = 0x0A80 => Temperature deviation = 10.50 °C

4.11.10 Calibration Data

The analog inputs of the CAN-CBX_Pt100 module have been calibrated by the manufacturer before delivery. The following calibrations have been made:

- the higher measuring current (ca. 400µA)
- the lower measuring current (ca. 40 µA)
- ADC gain PGA=2
- ADC gain PGA=3
- ADC gain PGA=4

The parameters determined during calibration are contained in the following objects and can be changed. Normally, an adjustment of the objects by the user is not necessary.
An adjustment by the user is only advisable in the object “offset” (0x2510), to compensate the line resistances.

With command *Restore Default Parameters* (0x1011) the initial setting of the calibration data adjusted by esd are reactivated.

4.11.10.1 Calibration and Process/Field-Value Calculation

In the following formulas the parameter names are shown without the channel identifier at the end of the parameter names to improve the clarity.

The calculations shown in the following are automatically done by the local firmware. For the user the corrected measuring values are accessible by the process variables PV.

Field-Value Calculation

$$\boxed{AI\_FV = \frac{adc\ raw\ value}{2^{ADC\ PGA}}}$$

with

adc_raw value:	A/D-converter value after addition and averaging
AI_FV [Object 0x9100]:	Field Value
ADC_PGA [Object 0x2400]:	ADC-gain

Process-Value Calculation

1) Correction FV

$$\boxed{FV_{corr} = AI - FV\left(1 + \frac{corr}{2^{22}}\right)}$$

with

FVFV_{corr}:	internal corrected field value in ADC-units
AI_FV [Object 0x9100]:	analog input field value
corr:	internal correction factor

$$\boxed{corr = gc\_lh + gc\_PGAy + (local\_temp\_ref\_temp\_lh) \times \frac{tk\ Ih}{256}}$$

With

gc_lh [objects 0x2502, 0x2503]:	Gain-correction factor (**g**ain **c**orrection **l**low/**l**high) for the low and the high measuring current
gc_PGAy [objects 0x2504, 0x2505, 0x2506]: (*y*= 2, 3, 4)	Gain correction for the gains: PGA=2 [0x2504], PGA=3 [0x2505] and PGA=4 [0x2506] (no correction at PGA=1)
local_temp [object 0x2420]:	Temperature of the units of this channel on the PCB, measured during calibration
ref_temp_lh [objects 0x2500, 0x2501]:	Reference temperature for the high and the low measuring current determined during calibration
tk_lh [objects 0x2511, 0x2512]:	Temperature coefficient for the high and the low measuring current determined during calibration

2) Calculation PV_R (resistance value)

$$\boxed{PV_R = (k\_Ih \times FV_{corr}) + offset}$$

with

PV_R :	Internal resistance value (only accessible for the user in the mode resistance (see Object 0x6110))
k_Ih:	Internal conversion factor for the high and the low measuring current (not accessible for the user); *k_Ilow/Ihigh* = f (*current_source_value* [Objekt 0x2410])
offset [Object 0x2510]:	Offset value for the compensation of the systematic measuring error caused by the line resistance

3) Temperature Calculation

$$\boxed{PV_T = Tab(PV_R)}$$

with

PV_T [Object 0x9130]:	Process value (mapped to PDOs)
Tab:	Application of the sensor-type conversion table *Tab* = f (*AI_sensor_type* [object 0x6110])

4.11.11 Reference Temperature at Calibration (0x2500, 0x2501)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2500	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*ref_temp_llow_1*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	2	*ref_temp_llow_2*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	3	*ref_temp_llow_3*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	4	*ref_temp_llow_4*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
0x2501	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*ref_temp_lhigh_1*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	2	*ref_temp_lhigh_2*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	3	*ref_temp_lhigh_3*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	4	*ref_temp_lhigh_4*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw

*3) The default values have been determined individually for every module at the calibration. Thus, these values are not described in this general documentation.

Description of the parameter *ref_temp_llow_x* and *ref_temp_lhigh_x* (x = 1 ... 4):

These objects contain the reference temperatures, determined at calibration. The temperature values are saved in steps of 1/256 °C.

The value is coded as described below:

$$\boxed{\mathit{Temperature}[°C] = \frac{\mathit{Int16\ Value}}{256}}$$

Example:

ref_temp_llow = 0x2640 => Temperature = 38.25 °C

4.11.12 Gain Correction at I_{LOW} and I_{HIGH} (0x2502, 0x2503)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2502	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*gain_correction_Ilow_1*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	2	*gain_correction_Ilow_2*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	3	*gain_correction_Ilow_3*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	4	*gain_correction_Ilow_4*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
0x2503	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*gain_correction_Ihigh_1*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	2	*gain_correction_Ihigh_2*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	3	*gain_correction_Ihigh_3*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	4	*gain_correction_Ihigh_4*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw

*3) The default values have been determined individually for every module at the calibration. Thus, these values are not described in this general documentation.

Description of the parameter *gain_correction_Ilow_x and gain_correction_Ihigh_x* (x = 1 ... 4):

These objects contain the gain-correction factors, that are determined during calibration of the individual channels for I_{low} and I_{high} at PGA=1.

Value range: 0x8000 ... 0x7FFF

4.11.13 Gain Correction PGA=2/3/4 (0x2504 – 0x2506)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2504	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*gain_correction_PGA=2_1*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	2	*gain_correction_PGA=2_2*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	3	*gain_correction_PGA=2_3*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	4	*gain_correction_PGA=2_4*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
0x2505	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*gain_correction_PGA=3_1*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	2	*gain_correction_PGA=3_2*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	3	*gain_correction_PGA=3_3*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	4	*gain_correction_PGA=3_4*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
0x2406	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*gain_correction_PGA=4_1*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	2	*gain_correction_PGA=4_2*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	3	*gain_correction_PGA=4_3*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw
	4	*gain_correction_PGA=4_4*	0x8000 ... 0x7FFF	*3)	0	integer 16	rw

*3) The default values have been determined individually for every module at the calibration. Thus, these values are not described in this general documentation.

Description of the parameter *gain_correction_PGA=y_x* (y = 1, 2, 3, 4; x = 1 ... 4):

These objects contain the gain-correction factors for the PGA-values 2, 3 and 4, as determined during calibration.

Value range: 0x8000 ... 0x7FFF

4.11.14 Offset (0x2510)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2510	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*offset_1*	0x8000 ... 0x7FFF	0	0	integer 16	rw
	2	*offset_2*	0x8000 ... 0x7FFF	0	0	integer 16	rw
	3	*offset_3*	0x8000 ... 0x7FFF	0	0	integer 16	rw
	4	*offset_4*	0x8000 ... 0x7FFF	0	0	integer 16	rw

Description of the parameter *offset_x* (x = 1 ... 4):

By means of these objects the line resistance can be compensated.
It contains the offset in steps of 0.1 mOhm.

If the compensation of a resistance is necessary, the negative value of the resistance (see PV(R), object 9130_h) must be entered.

Example:

R_{Line_x} = 123.4 mΩ   => 1234_d => *offset_x* = -1234
                                                  => *offset_x* = 0xFB2E

4.11.15 Temperature Coefficient at I_{LOW} and I_{HIGH} (0x2511, 0x2512)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2511	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*tk_Ilow_1*	0x8000 ... 0x7FFF	0	0	integer 16	rw
	2	*tk_Ilow_2*	0x8000 ... 0x7FFF	0	0	integer 16	rw
	3	*tk_Ilow_3*	0x8000 ... 0x7FFF	0	0	integer 16	rw
	4	*tk_Ilow_4*	0x8000 ... 0x7FFF	0	0	integer 16	rw
0x2512	0	*number_of_entries*	4	4	0	unsigned 8	ro
	1	*tk_Ihigh_1*	0x8000 ... 0x7FFF	0	0	integer 16	rw
	2	*tk_Ihigh_2*	0x8000 ... 0x7FFF	0	0	integer 16	rw
	3	*tk_Ihigh_3*	0x8000 ... 0x7FFF	0	0	integer 16	rw
	4	*tk_Ihigh_4*	0x8000 ... 0x7FFF	0	0	integer 16	rw

Description of the parameters *tk_Ilow_x* and *tk_Ihigh_x* (x = 1 ... 4):

The temperature coefficients can be set by these parameters. The default value of the temperature coefficient is ‘0’. The temperature compensation can be set by the user. The formulas containing the temperature compensation are printed on page 90.

Value range: 0x8000 ... 0x7FFF

4.11.16 Calc Expression (0x2800)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2800	0	*number_of_entries*	8	8	0	unsigned 8	ro
	1	*calc_expression_1*	*4)	“”	0	string (80)	rw
	2	*calc_expression_2*	*4)	“”	0	string (80)	rw
	3	*calc_expression_3*	*4)	“”	0	string (80)	rw
	4	*calc_expression_4*	*4)	“”	0	string (80)	rw
	5	*calc_expression_5*	*4)	“”	0	string (80)	rw
	6	*calc_expression_6*	*4)	“”	0	string (80)	rw
	7	*calc_expression_7*	*4)	“”	0	string (80)	rw
	8	*calc_expression_8*	*4)	“”	0	string (80)	rw

*4) Syntax-correct string with up to 80 characters

“” … empty string

Description of the parameters *calc_expression_x (x = 1 … 8)*
Placeholders in the formula that are replaced by real (float) values for the calculation:

calc_expression_x can contain:

- A0 … A3 for the PVs of the 4 channels:
  A0 … Measured value of channel 1 (float in Ω or °C)
  A1 … Measured value of channel 2 (float in Ω or °C)
  A2 … Measured value of channel 3 (float in Ω or °C)
  A3 … Measured value of channel 4 (float in Ω or °C)

- and C0 … C7 for the constants of 0x2803:
  C0 … Constant from 0x2803, sub-index 1
  C1 … Constant from 0x2803, sub-index 2
  :   :
  C7 … Constant from 0x2803, sub-index 8

The following functions are understood:		
+	atan2(x,y)	pow(x,y)
-	ceil(x)	sin(x)
*	cos(x)	sinh(x)
/	cosh(x)	sqrt(x)
()	e()	tan(x)
abs(x)	exp(x)	tanh(x)
acos(x)	floor(x)	min(x,y)
asin(x)	ln(x)	max(x,y)
atan(x)	log(x)	
	pi()	

Float constants (immediates) need to use "." as decimal dot, not ",".

Syntax error in your expression (e.g. unknown function, unmatched brackets) will lead to an SDO error when writing the string: VALUE_RANGE_EXEEDED (0x06090030), whereas writing a string longer than the allowed length will lead to PARA_TO_LONG (0x06070012).

4.11.17 Calc Values Float (0x2801)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2801	0	*number_of_entries*	8	8	0	unsigned 8	ro
	1	*calc_value_float_1*	*5)	-	1	real 32	ro
	2	*calc_value_float_2*	*5)	-	1	real 32	ro
	3	*calc_value_float_3*	*5)	-	1	real 32	ro
	4	*calc_value_float_4*	*5)	-	1	real 32	ro
	5	*calc_value_float_5*	*5)	-	1	real 32	ro
	6	*calc_value_float_6*	*5)	-	1	real 32	ro
	7	*calc_value_float_7*	*5)	-	1	real 32	ro
	8	*calc_value_float_8*	*5)	-	1	real 32	ro

*5) Float constant (from 1.175494e-38 up to 3.402823e+38), Measured values in Ohm/°C can of course not reach such extreme values.

The default value depends on the values and formulas used.

Description of the parameters *calc_value_float_x*:

This index contains the results of the 8 calculations, in 32-bit float format.

4.11.18 Calc Value Int64 (0x2802)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2802	0	*number_of_entries*	8	8	0	unsigned 8	ro
	1	*calc_value_int_64_1*	*6)	-	1	integer 64	ro
	2	*calc_value_int_64_2*	*6)	-	1	integer 64	ro
	3	*calc_value_int_64_3*	*6)	-	1	integer 64	ro
	4	*calc_value_int_64_4*	*6)	-	1	integer 64	ro
	5	*calc_value_int_64_5*	*6)	-	1	integer 64	ro
	6	*calc_value_int_64_6*	*6)	-	1	integer 64	ro
	7	*calc_value_int_64_7*	*6)	-	1	integer 64	ro
	8	*calc_value_int_64_8*	*6)	-	1	integer 64	ro

*6) Integer 64

The default value depends on the values and formulas used.

Description of the parameters *calc_value_int_64_x*:
This index contains the results of the 8 calculations, in 64-bit integer format.
You can also map only 32 bit or 16 or 8. Make sure if the result can go beyond the mapped range, to use min/max in the expression to limit to values in range of the bits mapped.

Example:
min(A0+C0, 65535)

4.11.19 Calc Constant (0x2803)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2803	0	*number_of_entries*	8	8	0	unsigned 8	ro
	1	*calc_constant_1*	*7)	0	0	real 32	rw
	2	*calc_constant_2*	*7)	0	0	real 32	rw
	3	*calc_constant_3*	*7)	0	0	real 32	rw
	4	*calc_constant_4*	*7)	0	0	real 32	rw
	5	*calc_constant_5*	*7)	0	0	real 32	rw
	6	*calc_constant_6*	*7)	0	0	real 32	rw
	7	*calc_constant_7*	*7)	0	0	real 32	rw
	8	*calc_constant_8*	*7)	0	0	real 32	rw

*7) Float constant (from 1.175494e-38 up to 3.402823e+38), Measured values in Ohm/°C can of course not reach such extreme values.

Description of the parameters *calc_constant_x (x = 1 - 8)*:

Here 8 float constants can be used, that can be separated from the calc expressions (see 4.11.16). They can be used in the expressions as C0 ... C7.

Example:
min(A0+C0, 65535)

4.11.20 Calc used Vars Expr (0x2804)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2804	0	*number_of_entries*	8	8	0	unsigned 8	ro
	1	*calc used vars expr_1*	0 ... 0xFFF	0	0	unsigned 32	ro
	2	*calc used vars expr_2*	0 ... 0xFFF	0	0	unsigned 32	ro
	3	*calc used vars expr_3*	0 ... 0xFFF	0	0	unsigned 32	ro
	4	*calc used vars expr_4*	0 ... 0xFFF	0	0	unsigned 32	ro
	5	*calc used vars expr_5*	0 ... 0xFFF	0	0	unsigned 32	ro
	6	*calc used vars expr_6*	0 ... 0xFFF	0	0	unsigned 32	ro
	7	*calc used vars expr_7*	0 ... 0xFFF	0	0	unsigned 32	ro
	8	*calc used vars expr_8*	0 ... 0xFFF	0	0	unsigned 32	ro

Description of the parameters *calc_used_vars_expr_x (x = 1 - 8)*:

This is more for debugging/checking your expressions (see 4.11.16). It shows whether the respective expressions use the analog values of the 4 channels.

Bit:	11	10	9	8	7	6	5	4	3	2	1	0
Meaning:	C7	C6	C5	C4	C3	C2	C1	C0	A3	A2	A1	A0

4.11.21 Calc Duration (0x2805)

Index	Sub-index	Description	Value range	Default	PDO Mapping	Data type	Access mode
0x2805	0	*number_of_entries*	8	8	0	unsigned 8	ro
	1	*calc duration expr_1*	0 ... 0xFFFFFFFF	-	0	unsigned 32	ro
	2	*calc duration expr_2*	0 ... 0xFFFFFFFF	-	0	unsigned 32	ro
	3	*calc duration expr_3*	0 ... 0xFFFFFFFF	-	0	unsigned 32	ro
	4	*calc duration expr_4*	0 ... 0xFFFFFFFF	-	0	unsigned 32	ro
	5	*calc duration expr_5*	0 ... 0xFFFFFFFF	-	0	unsigned 32	ro
	6	*calc duration expr_6*	0 ... 0xFFFFFFFF	-	0	unsigned 32	ro
	7	*calc duration expr_7*	0 ... 0xFFFFFFFF	-	0	unsigned 32	ro
	8	*calc duration expr_8*	0 ... 0xFFFFFFFF	-	0	unsigned 32	ro

Description of the parameters *calc_duration_expression_x (x = 1 - 8)*:

This is more for debugging/checking your expressions (See 4.11.16). It shows how long your expressions take to calculate (in microseconds). This gives you an idea of what sampling rates are possible. This is not to be used cyclically.

4.12 Firmware Management via CiA 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 cannot 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
0x1F52	2	Verify Application Software		

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 included 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).

4.12.2 Verify Application Software 0x1F52

This object is only implemented for compatibility reasons.

INDEX	**0x1F52**
Name	Program Control

Index	Sub-index	Description	Value range	Default value	PDO Mapping	Data type	Access mode
0x1F52	0	*number of entries*	2	2	0	unsigned 8	ro
	1	*Application_Software_Date*	0 ... 0xFFFFFFFF	-	0	unsigned 32	ro
	2	*Application_Software_Time*	0 ... 0x05265C00	-	0	unsigned 32	ro

Description of the variables:

Application_Software_Date	Date of the generation of the firmware used, specified in number of days since 1. January 1984
Application_Software_Time	Time of the generation of the firmware used, specified in milliseconds since midnight.

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} = 95 mA, I_{MAX_24V} = 120 mA
Power consumption	Typical: 2.3 W Maximum: 2.9 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 9: 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
Pt1- Pt4	Inputs	4 x Phoenix Contact PCB header MCDN 1,5/5-G1-3,5, 5-pos., with cable connector FMC 1,5/5-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 and must be ordered separately. See accessories page 125.)	25 mating cycles of TBUS-connector

Table 10: Connectors, accessible from outside

5.3 CAN

Number of CAN ports	1x CAN CC
CAN controller	Acc. to ISO 11898-1 (CAN CC), contained in CPU
CAN protocol	According to ISO 11898-1
Physical CAN Layer	High-speed CAN CC interface according to ISO 11898-2, bit rate from 10 kbit/s up to 1 Mbit/s
Galvanic isolation	Separation by means of digital isolators and DC/DC-converters. Voltage over CAN isolation (CAN to housing/EARTH or “PE (mounting rail)”, CAN to Host/System Ground): 1kV DC @ 1s (I < 1 mA)
Bus termination	None, Terminating resistor must be set externally if required
Connector	CAN connector or via InRailBus, see 5.2

Table 11: Data of the CAN interface

5.4 Sensor Resistor Inputs

Number	4 independent ΣΔ A/D-converter channels
Sensor types	- Temperature sensors: Pt100, Pt200, Pt500, Pt1000 and Pt5000, Ni100, Ni200, Ni500, Ni1000 and Ni5000 - Electric resistance measurement
Measuring current	400 µA or 40 µA selectable
Connection technology	2-wire or 4-wire configuration
Measurement ranges	Temperature: Pt sensors: -250 °C ... +850° C Ni sensors: -200 °C ... +400 °C Resistance: 0 KΩ ... +50 KΩ
Conversion rate	2.5 Hz ... 1000 Hz
Resolution	< 1 µV at 25 Hz conversion rate < 0.01 °C
Measuring error	Depending on sensor type and measuring temperature e.g.: At Pt100 sensors with 4-wire configuration: Measuring current = ca. 400 µA: measuring error < 0.1 Ω Measuring current = ca. 40 µA: measuring error < 0.5 Ω The data of the characteristic curve are taken from the standard: DIN EN 60751:2008
Electrical resolution	electrical resolution = f(measuring current, sample rate) e.g.: measuring current = ca. 400 µA sample rate = 400 msec resulting resolution = approx. 1 mΩ
Galvanic isolation	Galvanic isolation of temperature sensor inputs against each other and against power supply
Input filter	Adjustable via CiA 404 objects (4)
Connector	4x connector, see 5.2

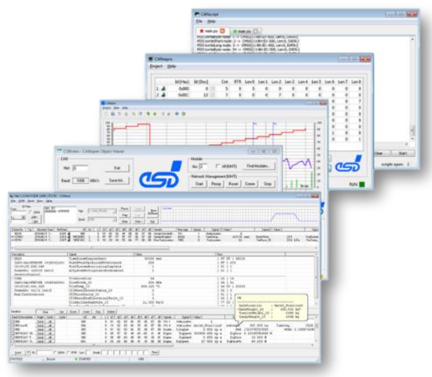
Table 12: Data of the sensor resistor inputs

5.5 Software Support

The CAN-CBX-PT100/2 presents itself as a CANopen responder device. The "Analog In" functions conform to the CiA® 404 (4) profile for measuring devices.
The firmware of the module comes with CANopen® firmware according to CiA 301 (1).
See chapter 4 for further information about the CANopen Firmware.

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

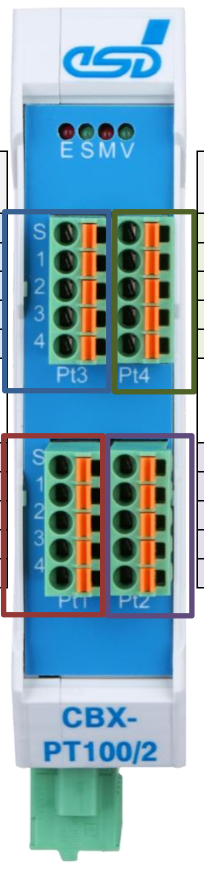
6.1 Temperature Sensor Inputs Pt1 – Pt4

Device socket: Phoenix Contact header MCDN 1,5/5-G1-3,5 P26THR
(4 connectors of 5 positions each)

Cable plug: Phoenix Contact pluggable connectors, 4x FMC 1,5/5-ST-3,5,
Push-in spring-connection, Phoenix Contact Order No.:1952296 (included in delivery)
For conductor connection and conductor cross section see page 111.

Pin position and pin assignment

Label/ Pin	Port	Signal	Signal	Port
S	**Pt3**	Shield	Shield	**Pt4**
1		RL3-	RL4-	
2		R3	R4	
3		R3+	R4+	
4		RL3+	RL4+	
S	**Pt1**	Shield	Shield	**Pt2**
1		RL1-	RL2-	
2		R1	R2	
3		R1+	R2+	
4		RL1+	RL2+	



Signal Description:

Pin	Signal name	Function at 4-wire configuration	Function at 2-wire configuration
S	Shield	Connection for Shielding (At top hat rail mounting direct contact with mounting rail potential)	
1	RLx-	Current source – of port x	Negative potential for temperature sensor of port x
2	Rx-	Negative potential for temperature sensor of port x	Not connected or bridged with RLx-
3	Rx+	Positive potential for temperature sensor of port x	Not connected or bridged to RLx+
4	RLx+	Current source + of port x	Positive potential for temperature sensor of port x

(x = 1, 2, 3 or 4)

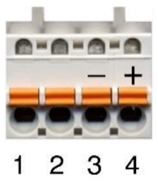
INFORMATION
Examples for the connection of the temperature sensor in 2-wire and 4-wire configuration can be found on page 14

6.2 24V Power Supply Voltage

DANGER
The CAN-CBX-PT100/2 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 111.

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 13.

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.

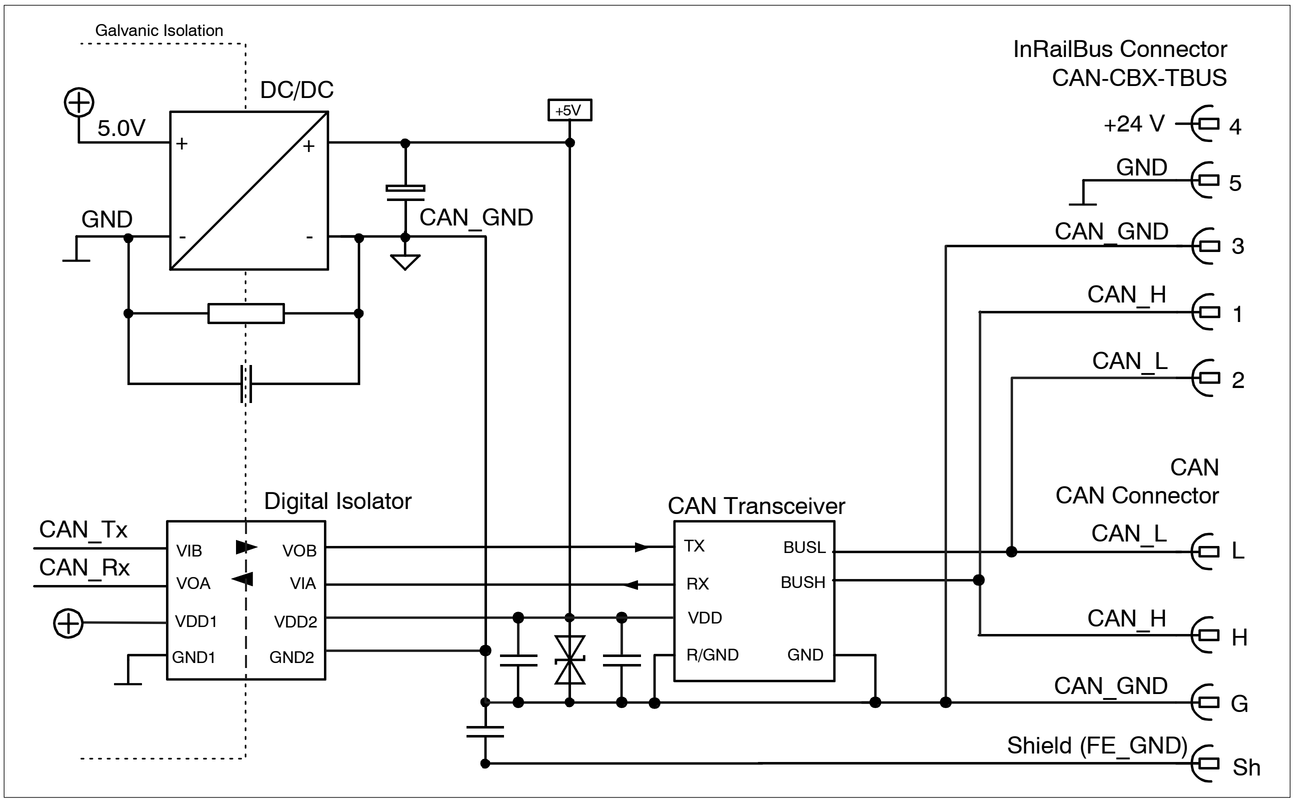


Figure 15: CAN port

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

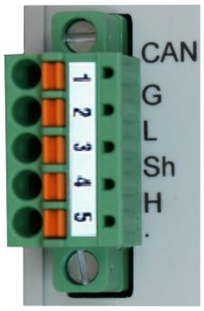
6.3.2 CAN Connector

The CAN bus is connected via the CAN connector on the upper side of the module or optional via the InRailbus (see page 110)

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 111

Pin Position:

(Cable plug)



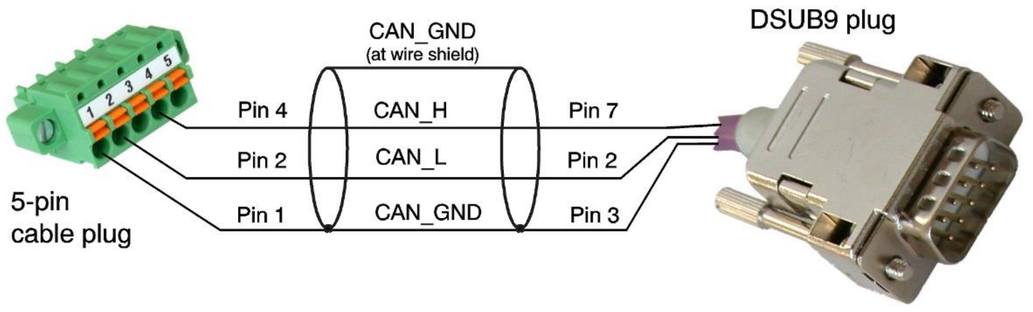
Pin Assignment:

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 125.

6.4 24 V and CAN via InRailBus

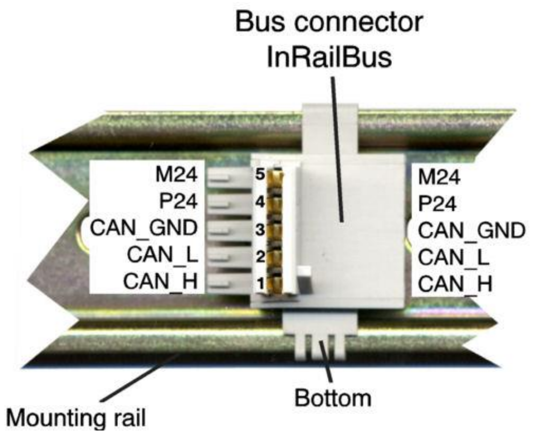
Power supply voltage and CAN can optionally be fed via the InRailBus. Use the mounting-rail bus connector (CAN-CBX-TBus) for the connection via the InRailBus, see Order Information (page 125).Read and follow the instructions for connecting power supply and CAN signals via InRailBus (see from page 22 and page 107)!

6.4.1 Connector Assignment 24V and CAN via InRailBus

DANGER
The CAN-CBX-PT100/2 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	Temperature Sensor Inputs
Connector type plug component	FKCT 2,5/..-ST KMGY	FK-MCP 1,5/5-STF-3,81	FMC 1,5/5-ST-3,5
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	0.14 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.

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

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

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

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.

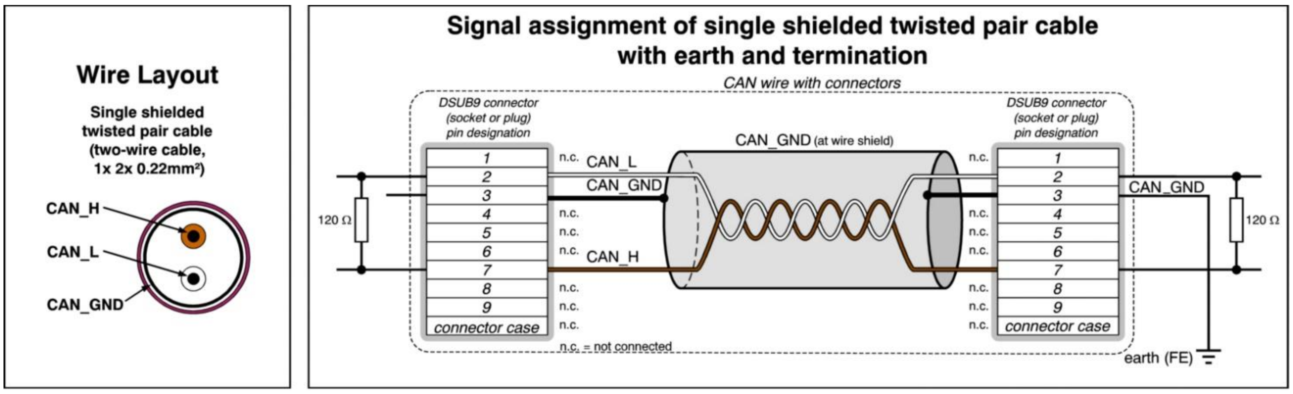

Wire Layout
Single shielded twisted pair cable (two-wire cable, 1x 2x 0.22mm²)
CAN_H
CAN_L
CAN_GND
Signal assignment of single shielded twisted pair cable with earth and termination
CAN wire with connectors
DSUB9 connector (socket or plug) pin designation
CAN_GND (at wire shield)
1
2
3
4
5
6
7
8
9
connector case
n.c.
CAN_L
CAN_GND
CAN_H
120 Ω
n.c. = not connected
earth (FE)

Figure 16: 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.

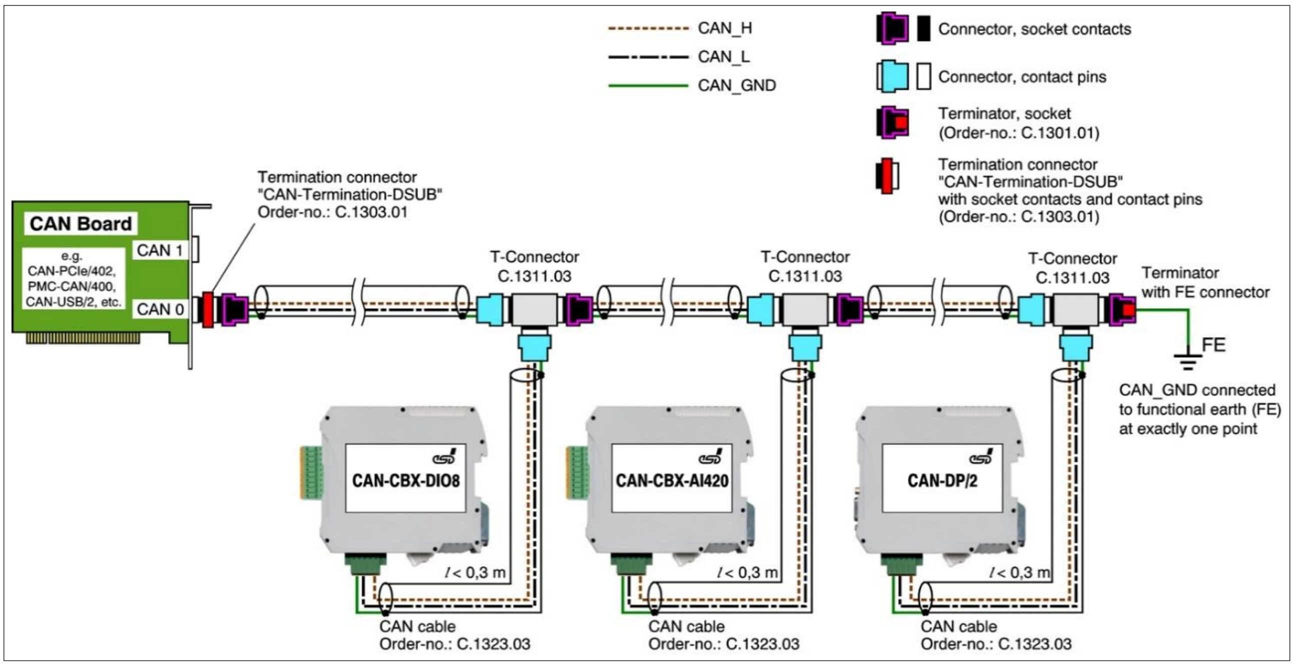

CAN_H
CAN_L
CAN_GND
Connector, socket contacts
Connector, contact pins
Terminator, socket (Order-no.: C.1301.01)
Termination connector "CAN-Termination-DSUB" with socket contacts and contact pins (Order-no.: C.1303.01)
Termination connector "CAN-Termination-DSUB" Order-no.: C.1303.01
CAN Board
e.g. CAN-PCIe/402, PMC-CAN/400, CAN-USB/2, etc.
CAN 1
CAN 0
T-Connector C.1311.03
T-Connector C.1311.03
T-Connector C.1311.03
Terminator with FE connector
FE
CAN_GND connected to functional earth (FE) at exactly one point
CAN-CBX-DIO8
CAN-CBX-AI420
CAN-DP/2
l < 0,3 m
l < 0,3 m
l < 0,3 m
CAN cable Order-no.: C.1323.03
CAN cable Order-no.: C.1323.03
CAN cable Order-no.: C.1323.03

Figure 17: 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.

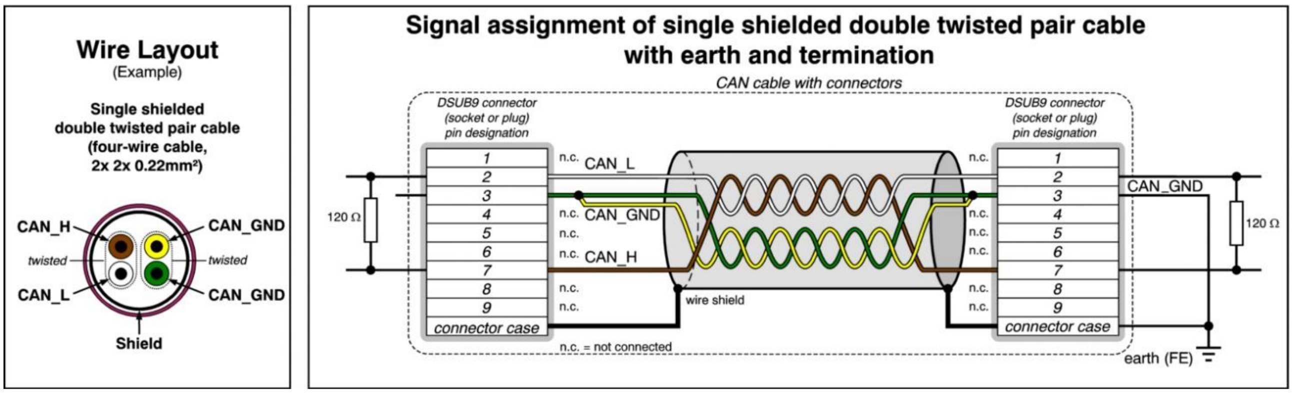

Wire Layout
(Example)
Single shielded double twisted pair cable (four-wire cable, 2x 2x 0.22mm²)
CAN_H
twisted
CAN_L
CAN_GND
twisted
CAN_GND
Shield
Signal assignment of single shielded double twisted pair cable with earth and termination
CAN cable with connectors
DSUB9 connector (socket or plug) pin designation
1
2
3
4
5
6
7
8
9
connector case
n.c.
CAN_L
CAN_GND
CAN_H
120 Ω
wire shield
n.c. = not connected
earth (FE)

Figure 18: CAN wiring for heavy industrial environment

7.3.2 Device Cabling

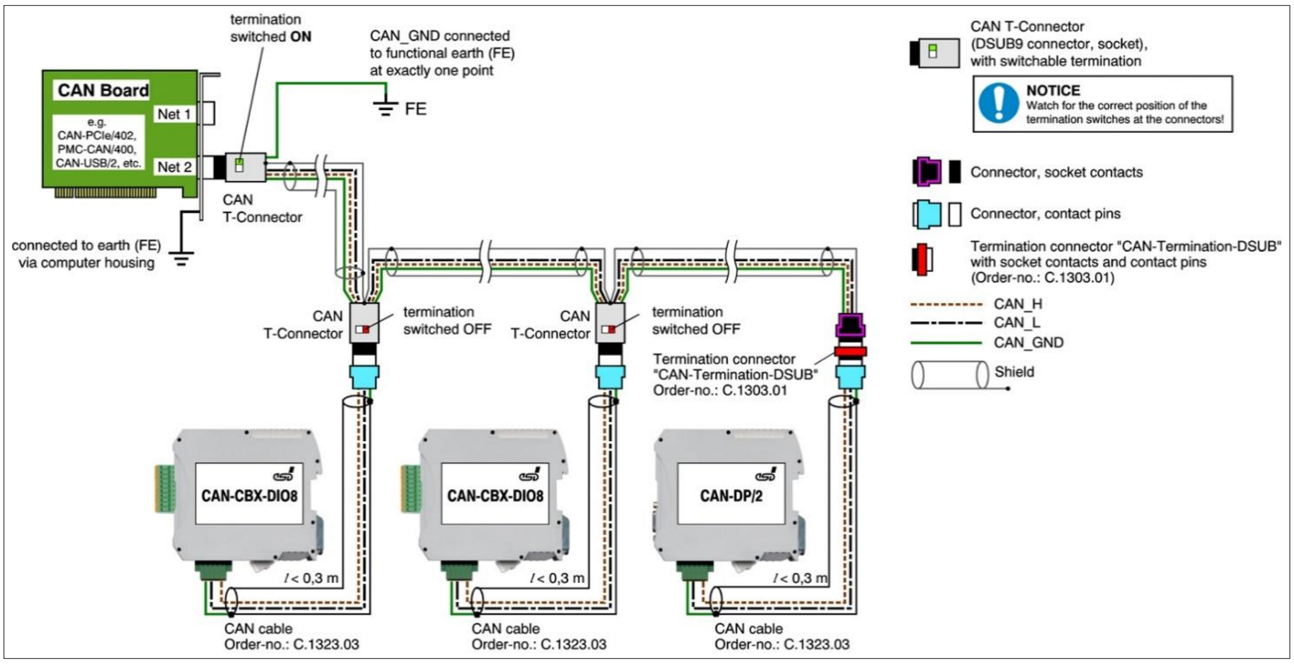

termination switched ON
CAN_GND connected to functional earth (FE) at exactly one point
FE
CAN Board
Net 1
Net 2
e.g. CAN-PCIe/402, PMC-CAN/400, CAN-USB/2, etc.
CAN T-Connector
connected to earth (FE) via computer housing
CAN T-Connector
termination switched OFF
CAN T-Connector
termination switched OFF
Termination connector "CAN-Termination-DSUB" Order-no.: C.1303.01
CAN-CBX-DIO8
CAN-CBX-DIO8
CAN-DP/2
l < 0,3 m
CAN cable Order-no.: C.1323.03
CAN T-Connector (DSUB9 connector, socket), with switchable termination
NOTICE Watch for the correct position of the termination switches at the connectors!
Connector, socket contacts
Connector, contact pins
Termination connector "CAN-Termination-DSUB" with socket contacts and contact pins (Order-no.: C.1303.01)
CAN_H
CAN_L
CAN_GND
Shield

Figure 19: 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 13: 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) UNITRONIC ®-BUS-FD P CAN UL/CSA (1x 2x 0.25) (UL/CSA approved)	 Part No.: 2170260 Part No.: 2170272
ConCab GmbH Äußerer Eichwald 74535 Mainhardt Germany www.concab.de	e. g. BUS-PVC-C (1x 2x 0.22 mm²) BUS-Schleppflex-PUR-C (1x 2x 0.25 mm²)	 Order No.: 93 022 016 (UL appr.) 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) UNITRONIC ®-BUS-FD P CAN UL/CSA (2x 2x 0.25) (UL/CSA approved)	 Part No.: 2170261 Part No.: 2170273
ConCab GmbH Äußerer Eichwald 74535 Mainhardt Germany www.concab.de	e. g. BUS-PVC-C (2x 2x 0.22 mm²) BUS-Schleppflex-PUR-C (2x 2x 0.25 mm²)	 Order No.: 93 022 026 (UL appr.) 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.

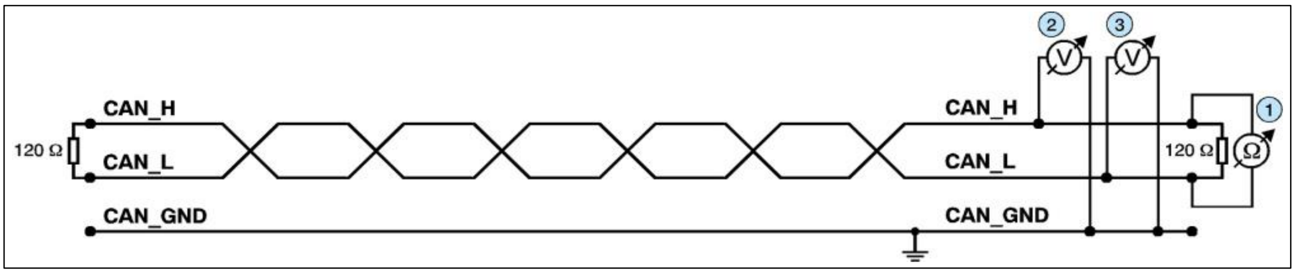


Figure 20: Simplified diagram of a CAN network

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!

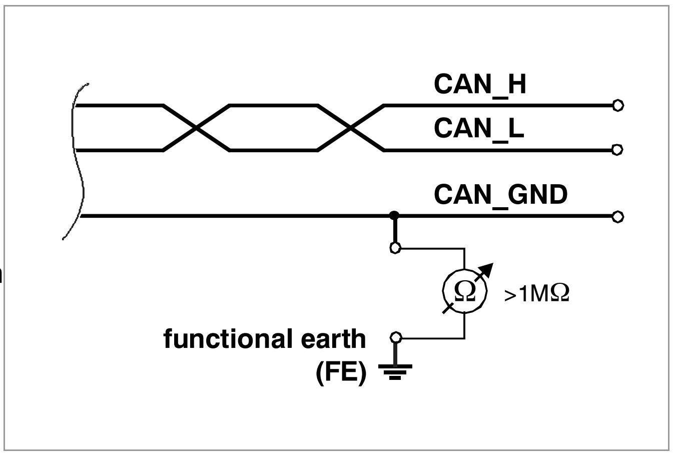


Figure 21: Simplified schematic diagram of ground test measurement

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.)

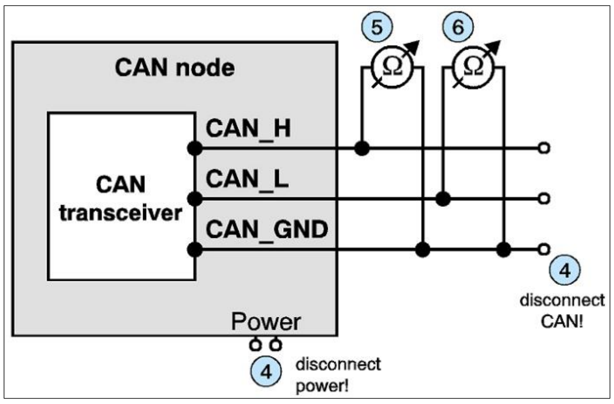


Figure 22: Measuring the internal resistance of CAN transceivers

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-PT100/2 is subject to the license terms of the respective authors or rights holders. CAN-CBX-PT100/2 may only be used in accordance with these license terms!
By using the CAN-CBX-PT100/2 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 14: 3rd party software license terms

- The CAN-CBX-PT100/2 uses the opensource operating system FreeRTOS^{TM}.
  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 14.

- 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-PT100/2 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 14.

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.

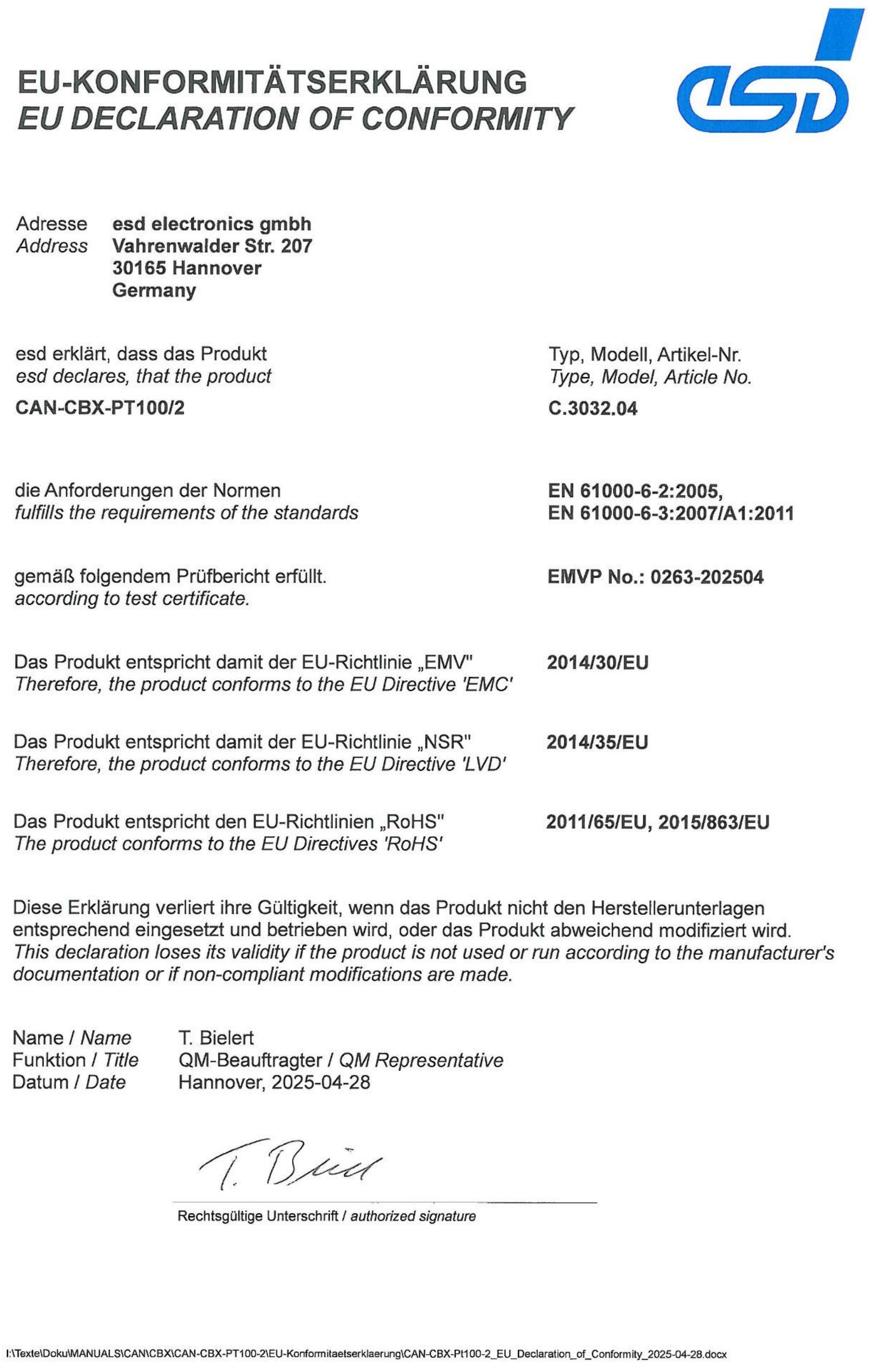
(4) CiA 404, Public Available Specification V 1.2.0 (5.2002), CANopen profile for measuring devices and closed-loop controllers

(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.

(7) Device and network design - Part 2: Representation of SI units and prefixes

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-PT100/2	**C.3032.04**
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.: 0263-202504**
Das Produkt entspricht damit der EU-Richtlinie „EMV" *Therefore, the product conforms to the EU Directive 'EMC'*	**2014/30/EU**
Das Produkt entspricht damit der EU-Richtlinie „NSR" *Therefore, the product conforms to the EU Directive 'LVD'*	**2014/35/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, 2025-04-28

[illegible]

Rechtsgültige Unterschrift / *authorized signature*

I:\Texte\Doku\MANUALS\CAN\CBX\CAN-CBX-PT100-2\EU-Konformitaetserklaerung\CAN-CBX-Pt100-2_EU_Declaration_of_Conformity_2025-04-28.docx

12 Order Information

Type	Properties	Order No.
CAN-CBX-PT100/2	CANopen module with 4 inputs for Pt and Ni temperature sensors (Pt100–Pt5000, Ni100–Ni5000) and resistance measurement. Measuring range -250 °C … +850 °C with 20-bit resolution and galvanic isolation. CANopen according to CiA 301 and CiA 404 Device Profile for Measurement Devices. Connectors for wire end ferrules included in the scope of delivery.	C.3032.04
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 CAN (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 CAN (IMC 1,5/ 5-ST-3,81 AU), pin contacts	C.3000.03

Table 15: Order information

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

Manuals	
CAN-CBX-PT100/2-ME	CAN-CBX-PT100/2 Manual in English

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.