



CAN-CBX-A0412

4 Analogue Outputs, 12-Bit



Manual

to Product C.3040.02



NOTE

The information in this document has been carefully checked and is believed to be entirely reliable. **esd** makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document. **esd** reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

esd assumes no responsibility for the use of any circuitry other than circuitry which is part of a product of **esd gmbh**.

esd does not convey to the purchaser of the product described herein any license under the patent rights of **esd gmbh** nor the rights of others.

esd electronic system design gmbh
Vahrenwalder Str. 207
30165 Hannover
Germany

Phone: +49-511-372 98-0
Fax: +49-511-372 98-68
E-mail: info@esd-electronics.com
Internet: www.esd-electronics.com

USA / Canada:
esd electronics Inc.
525 Bernardston Road
Suite 1
Greenfield, MA 01301
USA

Phone: +1-800-732-8006
Fax: +1-800-732-8093
E-mail: us-sales@esd-electronics.com
Internet: www.esd-electronics.us

Document file:	I:\Texte\Doku\MANUALS\CAN\CBX\AO412\Englisch\CAN-CBX-AO412_11.en9
Date of print:	2008-02-12

PCB version:	CAN-CBX-AO412 Rev. 1.1
Firmware version:	Rev. 2.0

Changes in the chapters

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

Chapter	Changes versus previous version
2.4	Maximum output current specified
3.2	Indicator states of the LEDs for invalid node-ID changed
4.2	Chapter completed
8.6	Chapter replaced by chapter: "Setting and Reading the Analogue Outputs"
8.8.2	Description of the calculation of the variable and examples inserted

Technical details are subject to change without further notice.

This page is intentionally left blank.

Contents

1. Overview	7
1.1 Description of the Module	7
2. Technical Data	8
2.1 General technical Data	8
2.2 CPU-Unit	8
2.3 CAN-Interface	9
2.4 Analogue Outputs	9
2.5 Software Support	9
2.6 Order Information	10
3. Hardware Installation	11
3.1 Connecting Diagram	11
3.2 LED Display	12
3.2.1 Indicator States	12
3.2.2 Operation of the CAN-Error-LED	13
3.2.3 Operation of the CANopen-Status-LED	13
3.2.4 Operation of the Error-LED	14
3.2.5 Operation of the Power-LED	14
3.2.6 Special Indicator States	14
3.3 Coding Switches	15
3.3.1 Setting the Node-ID via Coding Switch	15
3.3.2 Setting the Baud Rate	16
3.4 Installation of the Module Using InRailBus Connector	17
3.4.1 Connecting Power Supply and CAN-Signals to CBX-InRailBus	19
3.4.2 Connection of the Power Supply Voltage	19
3.4.3 Connection of CAN	20
3.5 Remove the CAN-CBX Module from the InRailBus	20
4. Description of the Units	21
4.1 CAN Interface	21
4.2 Analogue Outputs	22
5. Connector Assignments	23
5.1 Power Supply Voltage X100	23
5.2 CAN-Bus X400	24
5.3 CAN and Power Supply Voltage via InRailBus Connector X101	25
5.4 Analogue Outputs X500	26
6. Correctly Wiring Electrically Isolated CAN Networks	27
7. CAN-Bus Troubleshooting Guide	31
7.1 Termination	31
7.2 CAN_H/CAN_L Voltage	32
7.3 Ground	32
7.4 CAN Transceiver Resistance Test	33

8. Software	34
8.1 Definition of Terms	34
8.2 NMT-Boot-up	35
8.3 The CANopen-Object Directory	35
8.3.1 Access on the Object Directory via SDOs	35
8.4 Overview of used CANopen-Identifiers	39
8.4.1 Setting the COB-ID	39
8.5 Default PDO-Assignment	40
8.6 Setting and Reading the Analogue Values	41
8.6.1 Setting the Analogue Outputs	41
8.6.2 Reading the Analogue Outputs	41
8.7 Implemented CANopen-Objects	42
8.7.1 Device Type (1000 _h)	44
8.7.2 Error Register (1001 _h)	45
8.7.3 Manufacturer Status Register (1002 _h)	46
8.7.4 Pre-defined Error Field (1003 _h)	47
8.7.5 Manufacturer's Device Name (1008 _h)	49
8.7.6 Manufacturer's Hardware Version (1009 _h)	50
8.7.7 Manufacturer's Software Version 100A _h	50
8.7.8 Guard Time (100C _h) and Life Time Factor (100D _h)	51
8.7.9 Store Parameters (1010 _h)	52
8.7.10 Restore Default Parameters (1011 _h)	53
8.7.11 COB_ID Emergency Message (1014 _h)	54
8.7.12 Inhibit Time EMCY (1015 _h)	55
8.7.13 Consumer Heartbeat Time (1016 _h)	56
8.7.14 Producer Heartbeat Time (1017 _h)	58
8.7.15 Identity Object (1018 _h)	59
8.7.16 Verify Configuration (1020 _h)	61
8.7.17 Receive PDO Communication Parameter 1401 _h , 1402 _h	62
8.7.18 Receive PDO Mapping Parameter 1601 _h , 1601 _h	63
8.8 Device Profile Area	64
8.8.1 Overview of the implemented Objects 6411 _h	64
8.8.2 Write Analogue Output 16-Bit (6411 _h)	65
8.8.2.1 Assignment of the Sub-Indices	65
8.8.2.2 Assignment of the variables <i>AOUTx_value</i> (x = 1..4)	65
8.8.2.3 Examples of CAN-Frames for the SDO-Transfer	66
8.9 Manufacturer Specific Profile Area	67
8.9.1 Overview of the implemented Objects 2000 _h	67
8.9.2 Calibrated Offset Value 16-Bit (2000 _h)	67
9. References	68



1. Overview

1.1 Description of the Module

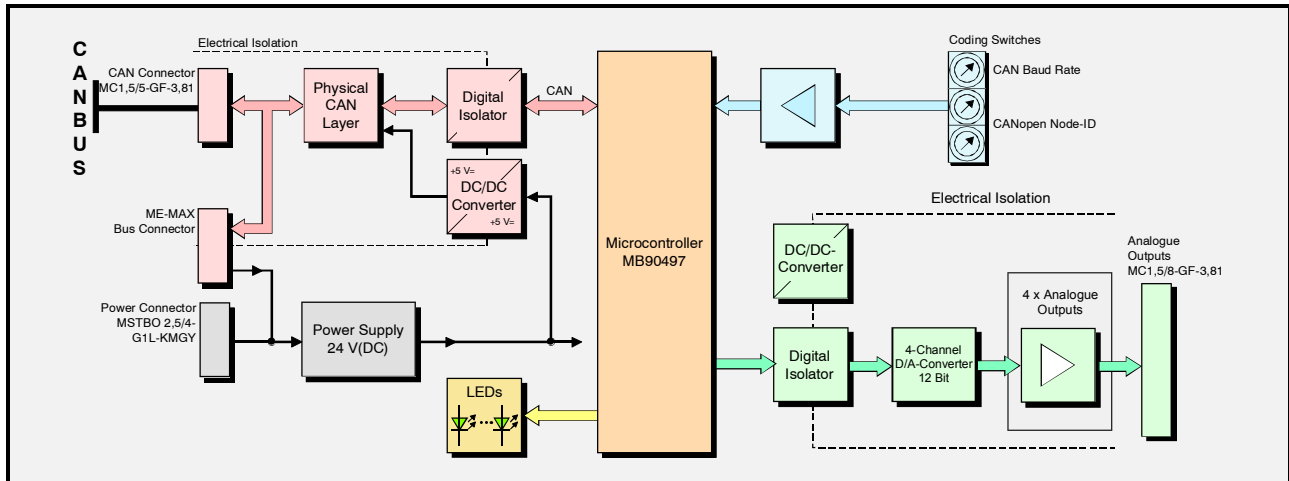


Fig. 1: Block circuit diagram of the CAN-CBX-AO412 module

The CAN-CBX-AO412 module is equipped with a MB90F497 microcontroller, which buffers the CAN data into a local SRAM. The firmware is stored in the Flash. Parameters are stored in a serial EEPROM.

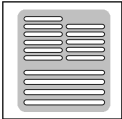
The four analogue outputs are converted via a LTC2620 12-bit-D/A-converter.

The outputs are connected via an 8-pin screw-/plug connector. The analogue outputs are electrically isolated by digital isolators for the protection of the other components.

The power supply voltage and the CAN-bus-connection can either be fed via the InRailBus connector, integrated in the top-hat-rail or via separate plugs.

The ISO 11898-compliant CAN interface allows a maximum data transfer rate of 1 Mbit/s. The CAN-interface is electrically isolated by a digital isolator and a DC/DC-converter.

The CANopen node number and the CAN-bit rate can be configured via three coding switches.



2. Technical Data

2.1 General technical Data

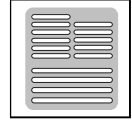
Power supply voltage	Nominal voltage: 24 V/DC Input voltage range: 24 V \pm 20% Current consumption (typically, 24 V, 20 °C): ca. 25 mA
Connectors	X100 (4-pin COMBICON connector with spring-cage connection) - 24 V-power supply voltage X101 (5-pin ME-MAX-TBUS connector, Phoenix Contact) - CAN interface and power supply voltage via InRailBus X500 (8-pin Mini-COMBICON connector) - analogue outputs X400 (5-pin Mini-COMBICON-connector) - CAN interface Only for test- and programming purposes: X200 (6-pin SMD socket strip) the connector is placed inside the case
Temperature range	-20 °C ... +70 °C ambient temperature
Humidity	max. 90%, non-condensing
Dimensions	width: 22 mm, height: 112 mm, depth: 113 mm (including mounting rail fitting and connector projection)
Weight	approx. 130 g

Table 1: General data of the module

2.2 CPU-Unit

CPU	16 bit μ C MB90F497
RAM	2 Kbyte integrated
Flash	64 Kbyte integrated
EEPROM	minimum 256 byte

Table 2: Microcontroller



2.3 CAN-Interface

Number	1
Connector	5-pin COMBICON with spring-cage connections or via Phoenix Contact TBUS-connector (InRailBus)
CAN-Controller	MB90F497, ISO 11898-1, (CANopen-software only 11-bit CAN-identifiers are supported)
Electrical isolation of the CAN interface against other units	via dual-channel digital isolator (ADUM1201BR) and DC/DC-converter
Physical CAN Layer	Physical Layer according to ISO 11898-2, transfer rate programmable from 10 Kbit/s up to 1 Mbit/s
Bustermiation	has to be set externally

Table 3: Data of the CAN-interface

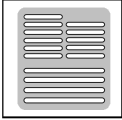
2.4 Analogue Outputs

Number	4 D/A-converter channels
Converter type	LTC2620
Resolution	12 bit + VZ
Output voltage range	± 10 V
Output current	maximum: 10 mA
Minimum load resistor	1 kΩ/channel
Electrical isolation of the interface against other units	via four-channel digital isolator (IL717-3)

Table 4: Data of the analogue outputs

2.5 Software Support

The firmware of the module supports the CANopen-standards DS-301 and DS-401.

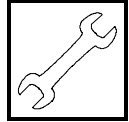


Technical Data

2.6 Order Information

Type	Features	Order-No.
CAN-CBX-AO412	CAN-CBX-AO412 4 analogue outputs, 12 bit, ± 10 V output voltage range including 1x CAN-CBX-TBUS (C.3000.01)	C.3040.02
Manuals		
CAN-CBX-AO412-ME	Manual in English	C.3040.21
Accessories		
CAN-CBX-TBUS	Mounting-rail bus connector of the CBX-InRailBus for CAN-CBX-modules, (a bus connector is included in delivery of the CAN-CBX-module)	C.3000.01
CAN-CBX-TBUS- Connector	Terminal plug of the CBX-InRailBus for the connection of the +24 V power supply voltage and the CAN interface	C.3000.02

Table 5: Order information



3. Hardware Installation

3.1 Connecting Diagram

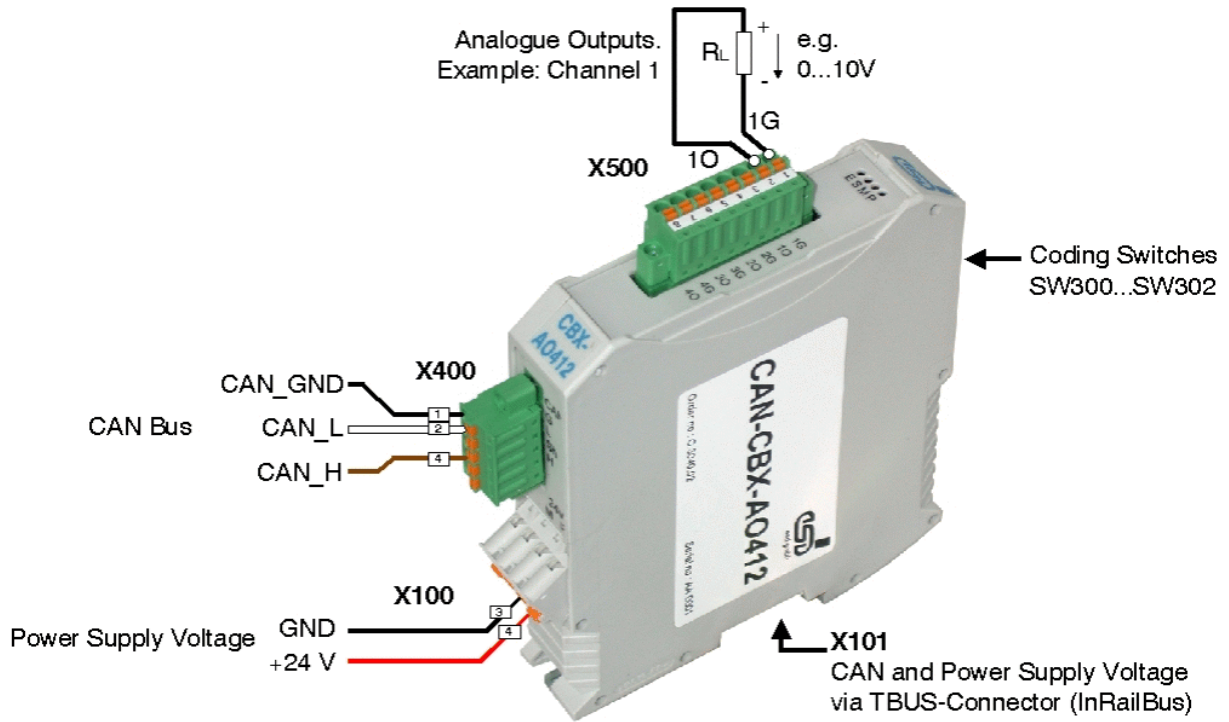
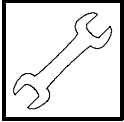


Fig. 2: Connection of the CAN-CBX-AO412 module

The connector pin assignment can be found on page 20 and the following.



Hardware Installation

3.2 LED Display

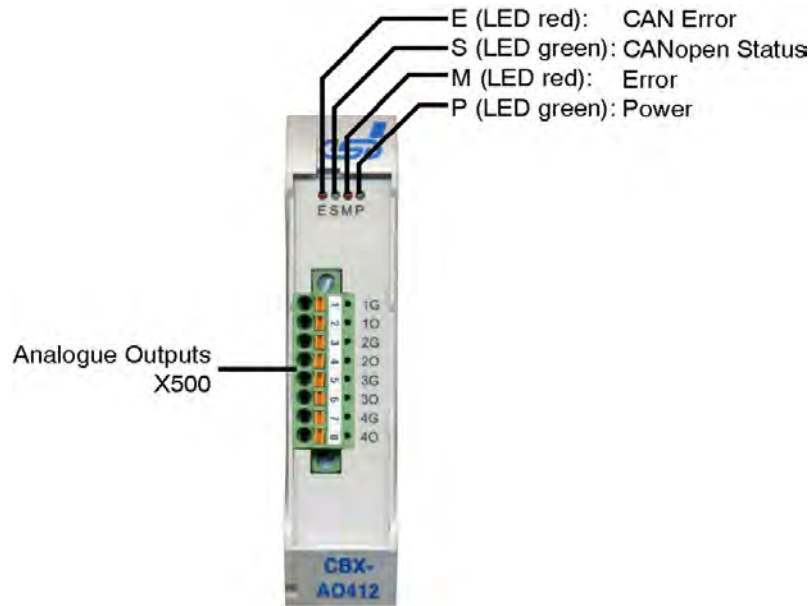


Fig. 3: Position of the LEDs in the front panel

The CAN-CBX-AO412 module is equipped with four Status-LEDs. The terms of the indicator states of the LEDs are chosen in accordance with the CANopen-Standard DS-303-3 (chapter 3.1). The indicator states are described in the following chapters.

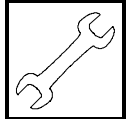
3.2.1 Indicator States

In principle there are 8 indicator states distinguished:

Indicator state	Display
on	LED on
off	LED off
blinking	LED blinking with a frequency of approx. 2.5 Hz
flickering	LED flickering with 10 Hz
1 flash	LED 200 ms on, 1400 ms off
2 flashes	LED 200 ms on, 200 ms off, 200 ms on, 1000 ms off
3 flashes	LED 2x (200 ms on, 200 ms off) + 1x (200 ms on, 1000 ms off)
4 flashes	LED 3x (200 ms on, 200 ms off) + 1x (200 ms on, 1000 ms off)

Table 6: Indicator states of the LEDs

The red LED flashes opposite in phase with the green LED!



3.2.2 Operation of the CAN-Error-LED

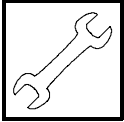
LED indication				Display function	
Label	Name	Colour	Component No.	Indicator state	Description
E	CAN Error	red	200A	off	no error
				1 flash	CAN controller is in <i>Error Active</i> state
				on	CAN controller state is <i>Bus Off</i> (or coding switch setting ID-Node > 7F _h when switching on, see page 14)
				2 flashes	Heartbeat or Nodeguard error occurred. The LED automatically turns off, if Nodeguard/Heartbeat-messages are received again.

Table 7: Indicator states of the red CAN Error-LED

3.2.3 Operation of the CANopen-Status-LED

LED indication				Display function	
Label	Name	Colour	Component No.	Indicator state	Description
S	CANopen Status	green	200B	blinking	<i>Pre-operational</i>
				on	<i>Operational</i>
				1 flash	<i>Stopped</i>
				3 flashes	Module is in bootloader mode (or coding switch setting ID-Node > 7F _h when switching on; see page 14)

Table 8: Indicator states of the CANopen Status-LED



Hardware Installation

3.2.4 Operation of the Error-LED

LED indication				Display function	
Label	Name	Colour	Component No.	Indicator state	Description
M	Error	red	200C	off	no error
				2 flashes	Internal software error e.g.: - stored data have an invalid checksum, therefore default values are loaded - internal watchdog has triggered - indicator state is continued until the module resets or an error occurs at the outputs.

Table 9: Indicator state of the Error-LED

3.2.5 Operation of the Power-LED

LED indication				Display function	
Label	Name	Colour	Component No.	Indicator state	Description
P	Power	green	200D	off	no power supply voltage or module is in bootloader mode (ID = 0)
				on	power supply voltage is on

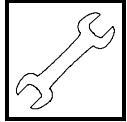
Table 10: Indicator state of the Power-LED

3.2.6 Special Indicator States

The following status is indicated by the CANopen-Status-LED and the CAN-Error-LED together:

LED indication	Description
CANopen-Status-LED: 3 flashes CAN-Error LED: on	The coding switches for the Node-ID are set to an invalid ID-value, when switching on. The firmware application will not be started.

Table 11: Special indicator states



3.3 Coding Switches

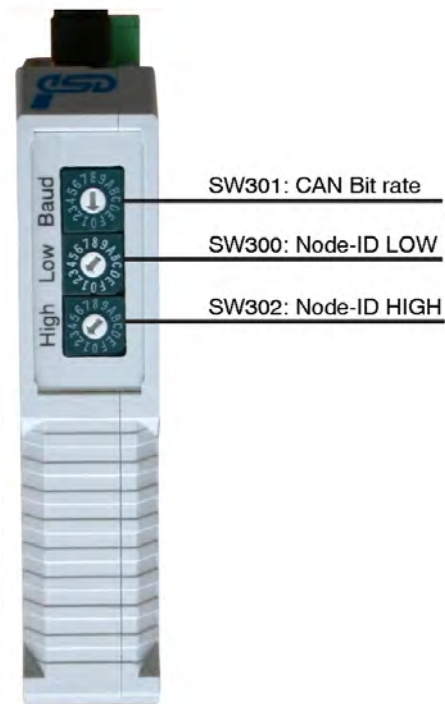


Abb. 4: Position of the coding switches



Attention!

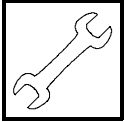
At the moment the module is switched 'on', the state of the coding switches is determined. Changes of the settings therefore have to be made **before switching on** the module, because changes of the settings are not determined during operation.

After a reset (e.g. NMT reset) the settings are read again.

3.3.1 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* from 01_h to $7F_h$.

The three higher-order bits (higher-order nibble) can be set with coding switch **HIGH**, the four lower-order bits can be set with coding switch **LOW**.



Hardware Installation

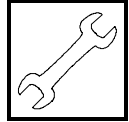
3.3.2 Setting the Baud Rate

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

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

Setting [Hex]	Baud rate [Kbit/s]
0	1000
1	666.6
2	500
3	333.3
4	250
5	166
6	125
7	100
8	66.6
9	50
A	33.3
B	20
C	12.5
D	10
E	reserved
F	reserved

Table 12: Index of the baud rate



3.4 Installation of the Module Using InRailBus Connector

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

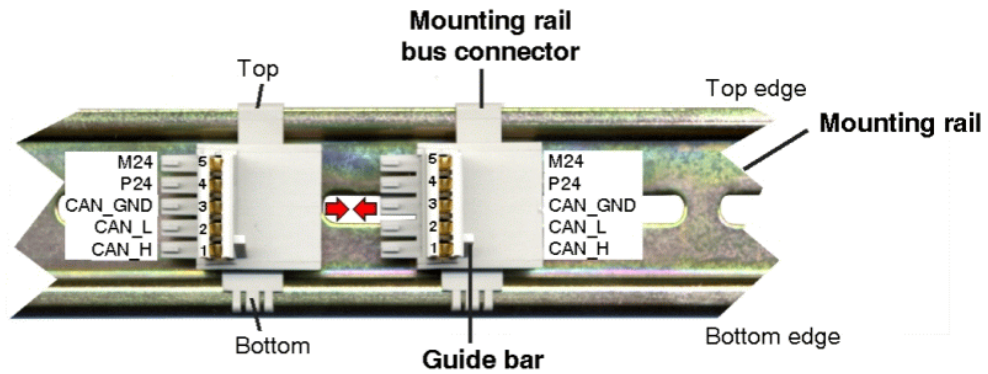
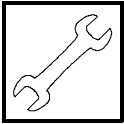


Figure 5: Mounting rail with bus connector

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



Figure 6 : Mounting CAN-CBX modules

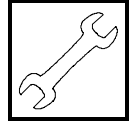


Hardware Installation

3. Swivel the CAN-CBX module onto the mounting rail in pressing the module downwards according to the arrow as shown in figure 6. The housing is mechanically guided by the DIN rail bus connector.
4. When mounting the CAN-CBX module the metal foot catch snaps on the bottom edge of the mounting rail. Now the module is mounted on the mounting rail and connected to the InRailBus via the bus connector. Connect the bus connectors and the InRailBus if not already done.



Figure 7: Mounted CAN-CBX module



3.4.1 Connecting Power Supply and CAN-Signals to CBX-InRailBus

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

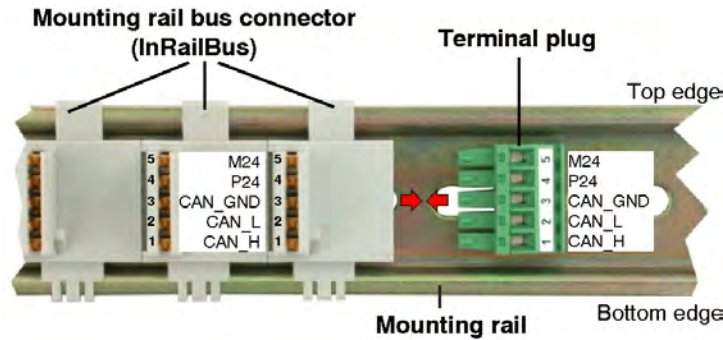


Fig. 8: Mounting rail with InRailBus and terminal plug

Plug the terminal plug into the socket on the right of the mounting-rail bus connector of the InRailBus, as described in Fig. 8. Then connect the CAN interface and the power supply voltage via the terminal plug.

3.4.2 Connection of the Power Supply Voltage



Attention!

It is **not permissible** to feed through the power supply voltage through the CBX station and to supply it to another CBX station via connector X100! A feed through of the +24 V power supply voltage can cause damage on the CBX modules.

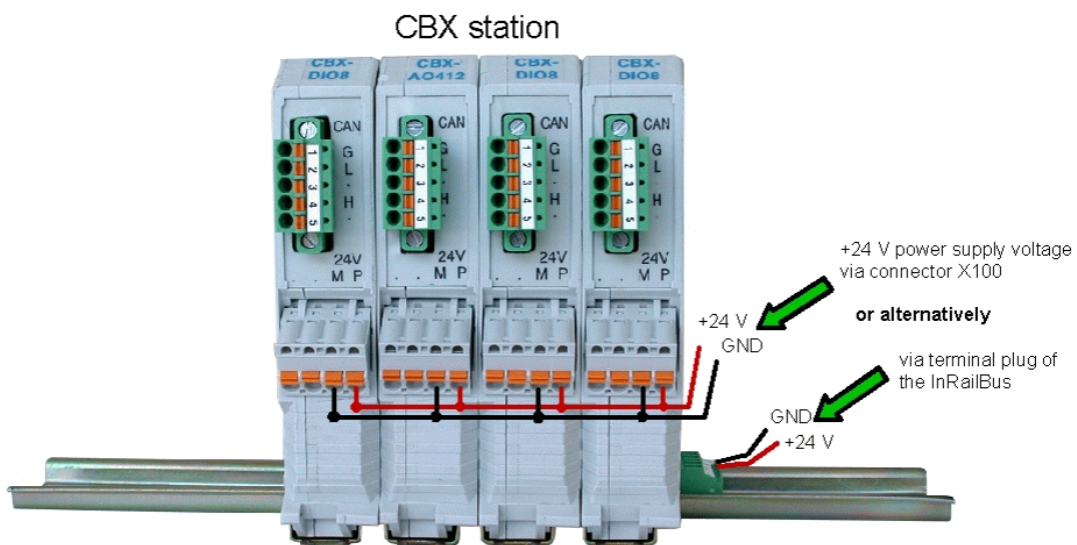
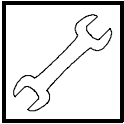


Fig. 9: Connecting the power supply voltage to the CAN-CBX station



3.4.3 Connection of CAN

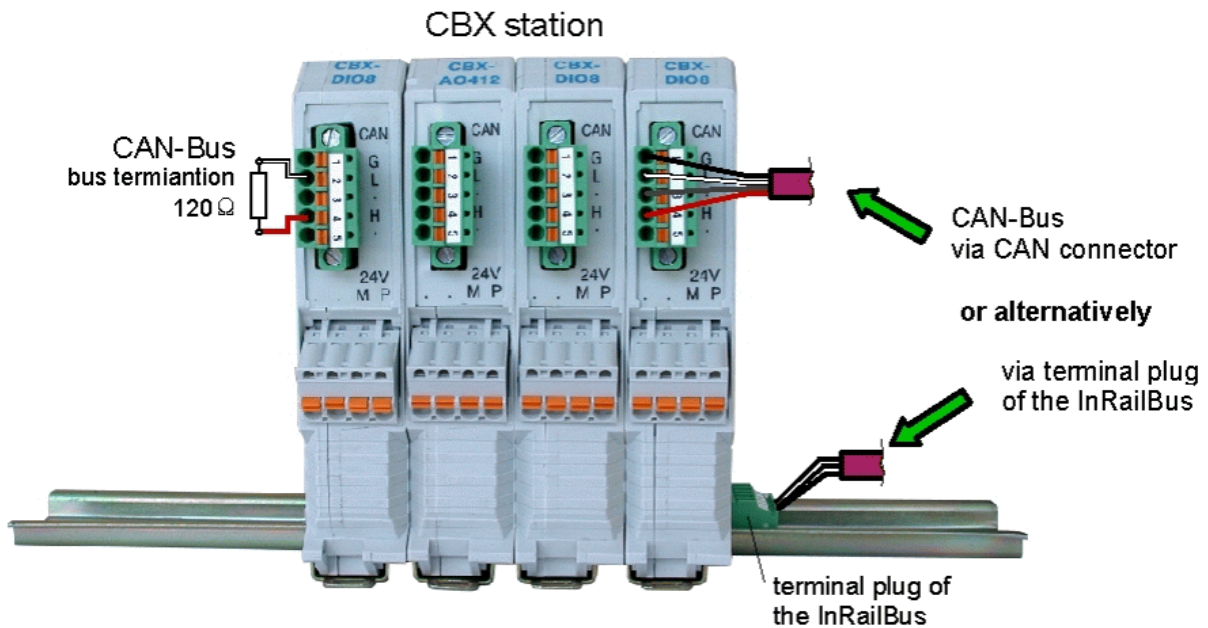


Fig. 10: Connecting the CAN signals to the CAN-CBX station

Generally the CAN signals can be fed via the CAN connector of the first CAN-CBX module of the CBX station. The signals are then connected through the CAN-CBX station via the InRailBus. To lead through the CAN signals the CAN bus connector of the last CAN-CBX module of the CAN-CBX station has to be connected to the CAN bus, because this would cause incorrect branching.

A bus termination must be connected to the CAN connector of the CAN-CBX module at the end of the CBX-InRailBus (see Fig. 10), if the CAN bus ends there.

3.5 Remove the CAN-CBX Module from the InRailBus

If the CAN-CBX module is connected to the InRailBus please proceed as follows:

Release the module from the mounting rail in moving the foot catch (see Fig. 7) downwards (e.g. with a screwdriver). Now the module is detached from the bottom edge of the mounting rail and can be removed.



Note:

It is possible to remove entire individual devices from the whole without interrupting the InRailBus connection, because the contact chain will not be interrupted.



4. Description of the Units

4.1 CAN Interface

An 82C251 is used as driver unit. The differential CAN Bus signals are electrically isolated from the other signals via dual digital converter and DC/DC-converter.

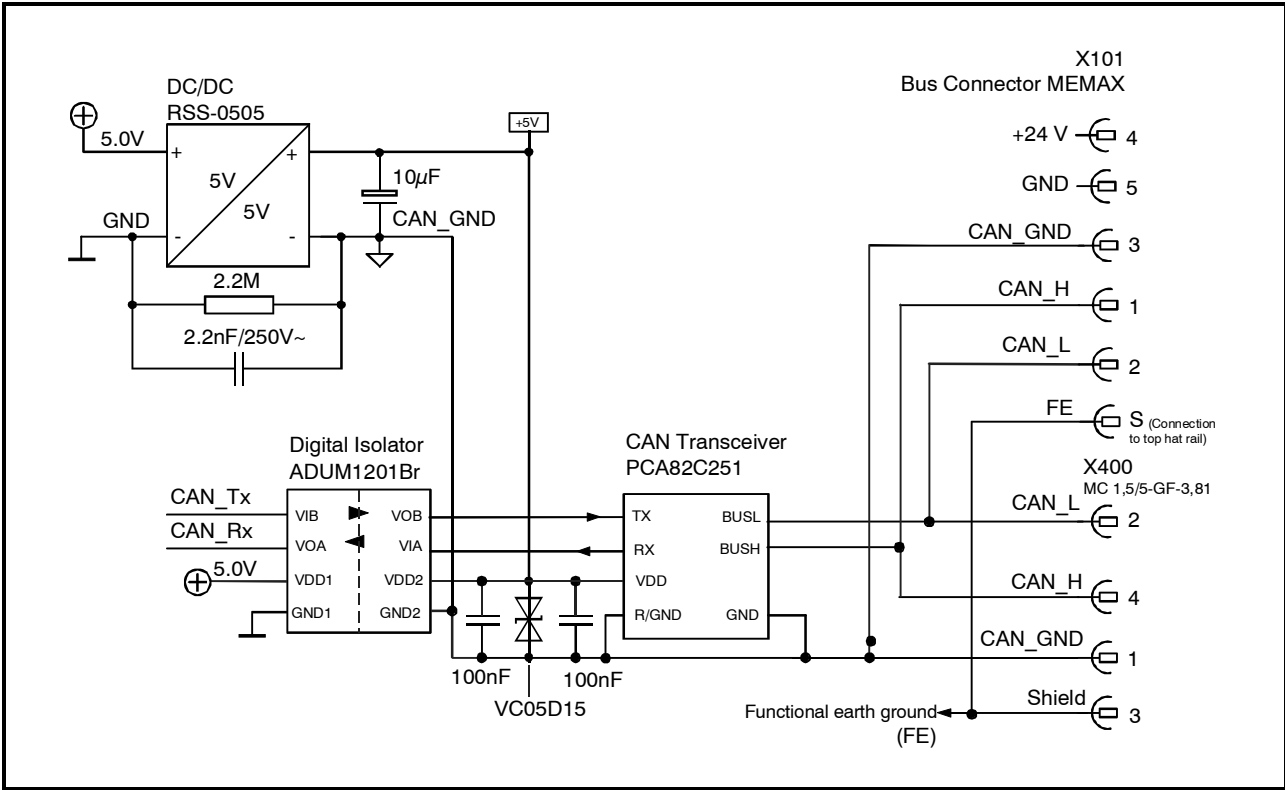


Fig. 11: CAN interface



4.2 Analogue Outputs

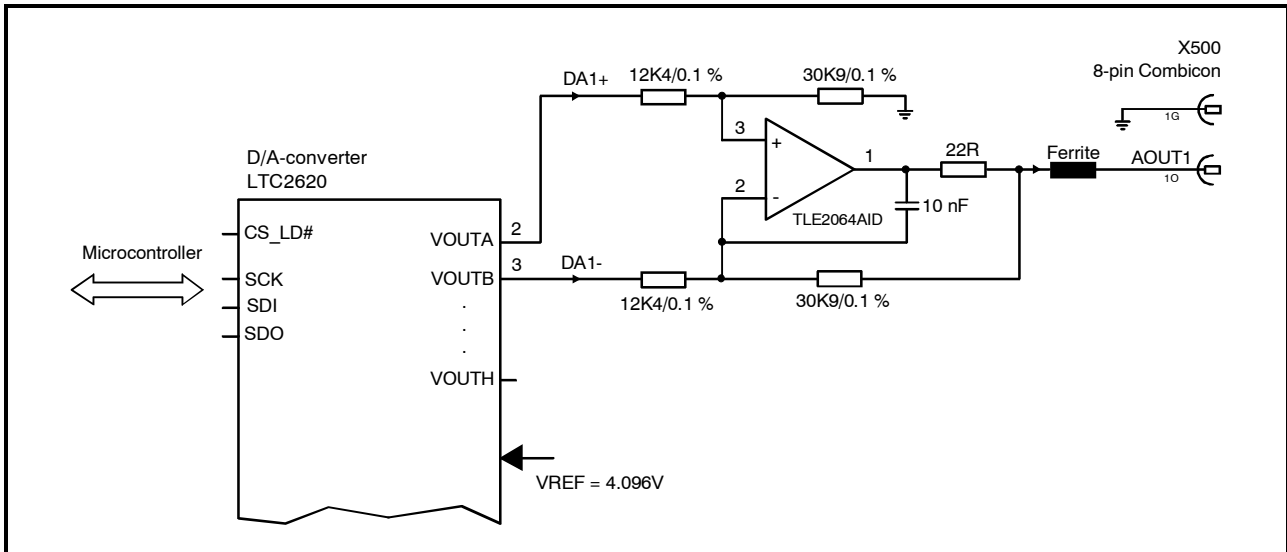


Fig. 12: Analogue output circuit (example: channel 1)

Special features of the circuit

The voltage at the output of the CAN-CBX-AO412 module is generated by two outputs of the internal digital/analogue-converter LTC[®]2620 of Linear Technology and a post-connected differential amplifier TLE2064AID of Texas Instruments.

Calculation of the measurement range

For the internal D/A-converter the voltage value of the LSB ($V_{LSB_D/A}$) is determined by the reference voltage $V_{REF} = 4,096 \text{ V}$ and its resolution $n = 12$.

$$V_{LSB_D/A} = \frac{V_{REF}}{2^n} = \frac{4,096 \text{ V}}{2^{12}} = 1 \text{ mV} \quad [1]$$

The amplification K of the post-connected differential amplifier is: $K = \frac{30,9 \text{ k}\Omega}{12,4 \text{ k}\Omega} \approx 2,492$

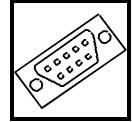
The resolution of the output voltage of the module results as: $V_{LSB} = V_{LSB_D/A} \cdot K \approx 2,492 \text{ mV}$

The voltage range of the module is determined by the maximum output voltage of the D/A-converter and the gain of the differential amplifier.

The maximum output voltage of the D/A-converter is the reference voltage $V_{REF} = 4,096 \text{ V}$ minus the voltage of a bit ($V_{LSB_D/A}$).

$$V_{max} = \left(V_{REF} - V_{LSB_D/A} \right) K = \left(4,096 \text{ V} - \frac{4,069 \text{ V}}{2^{12}} \right) \cdot \frac{30,9 \text{ k}\Omega}{12,4 \text{ k}\Omega} \approx 10,205 \text{ V}$$

As a result the voltage range of the module is $V_{MIN} = -10,205 \text{ V}$ to $V_{MAX} = +10,205 \text{ V}$ (rounded). See also object 6411_h on page 65.



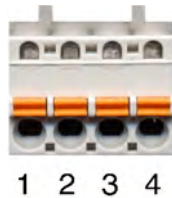
5. Connector Assignments

5.1 Power Supply Voltage X100

Device connector: COMBICON MSTBO 2,5/4-G1L-KMGY

Line connector: COMBICON FKCT 2,5/4-ST, 5.0 mm pitch, spring-cage connection, PHOENIX-CONTACT order no.: 19 21 90 0 (included in the scope of delivery)

Pin Position:



Pin Assignment:

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

Please refer also to the connecting diagram on page 11.

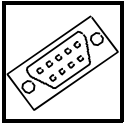


Note: The pins 1 and 4 are connected to each other at the PCB.
The pins 2 and 3 are connected to each other at the PCB.

Signal Description:

P24... power supply voltage +24 V

M24... reference potential



Connector Assignments

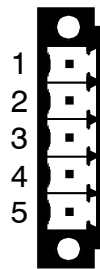
5.2 CAN-Bus X400

Device Connector: COMBICON MC 2,5/5-GF-3,81

Line Connector: COMBICON FK-MCP ,5/5-STF-3,81, spring-cage connection (included in the scope of delivery)

Pin Position:

(Illustration of device connector)



Pin-Assignment:

Pin	Signal
1	CAN_GND
2	CAN_L
3	Shield
4	CAN_H
5	-

Signal description:

CAN_L, CAN_H ... CAN signals
 CAN_GND ... reference potential of the local CAN physical layer
 Shield ... pin for line shield connection (using hat rail mounting direct contact to the mounting rail potential)
 - ... not connected

Recommendation of an adapter cable from 5-pin Combicon (here line connector FK-MCP1,5/5-STF-3,81 with spring-cage-connection) to 9-pin DSUB:

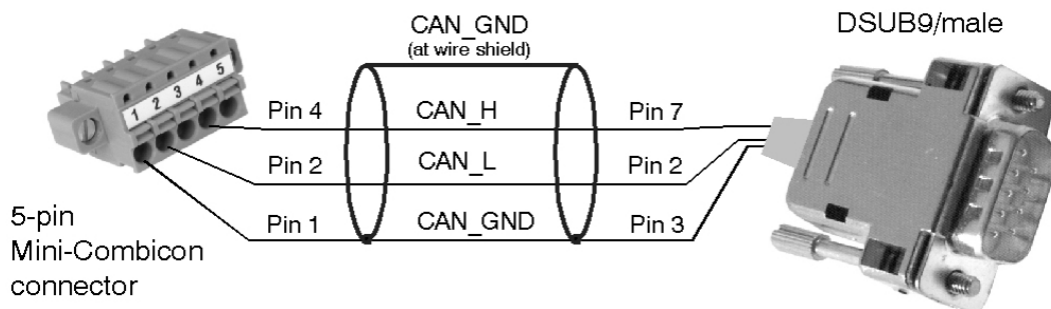
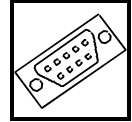


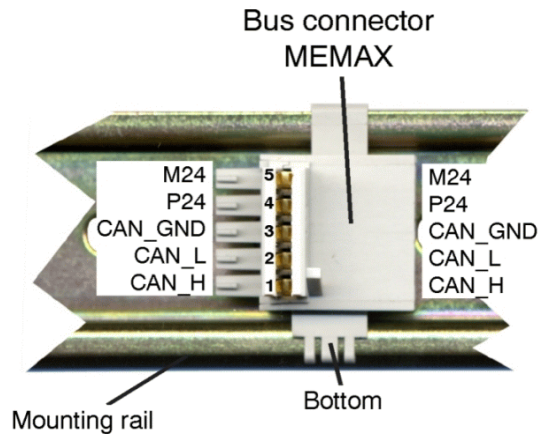
Figure 13: Assignment of the 9-pin DSUB-connector according to CiA DS 102.



5.3 CAN and Power Supply Voltage via InRailBus Connector X101

Connector type: Bus connector MEMAX
ME 22,5 TBUS 1,5/5-ST-3,81 KMGY

Pin Position:

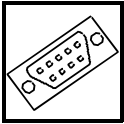


Pin Assignment:

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)



Connector Assignments

5.4 Analogue Outputs X500

Device Connector: COMBICON MC 1,5/8-GF-3,81

Line Connector: COMBICON MC 1,5/8-STF-3,81 (screw connection)
(included in the scope of delivery)

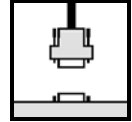
Pin Position: (view of device connector)	Pin Assignment	
	Pin:	Signal:
1	1	1G
2	2	1O
3	3	2G
4	4	2O
5	5	3G
6	6	3O
7	7	4G
8	8	4O

Signal description:

xO... Plus pin of the analogue output x

xG ... Reference potential analogue-ground (connected to functional earth ground contacts of the module)

(x = 1, 2, 3, 4)



6. Correctly Wiring Electrically Isolated CAN Networks

Generally all instructions applying for wiring regarding an electromagnetic compatible installation, wiring, cross sections of wires, material to be used, minimum distances, lightning protection, etc. have to be followed.

The following **general rules** for the CAN wiring must be followed:

1.	A CAN net must not branch (exception: short dead-end feeders) and has to be terminated by the wave impedance of the wire (generally $120\ \Omega \pm 10\%$) at both ends (between the signals CAN_L and CAN_H and not at GND)!
2.	A CAN data wire requires two twisted wires and a wire to conduct the reference potential (CAN_GND)! For this the shield of the wire should be used!
3.	The reference potential CAN_GND has to be connected to the earth potential (PE) at one point. Exactly one connection to earth has to be established!
4.	The bit rate has to be adapted to the wire length.
5.	Dead-end feeders have to kept as short as possible ($l < 0.3\ \text{m}$)!
6.	When using double shielded wires the external shield has to be connected to the earth potential (PE) at one point. There must be not more than one connection to earth.
7.	A suitable type of wire (wave impedance ca. $120\ \Omega \pm 10\%$) has to be used and the voltage loss in the wire has to be considered!
8.	CAN wires should not be laid directly next to disturbing sources. If this cannot be avoided, double shielded wires are preferable.

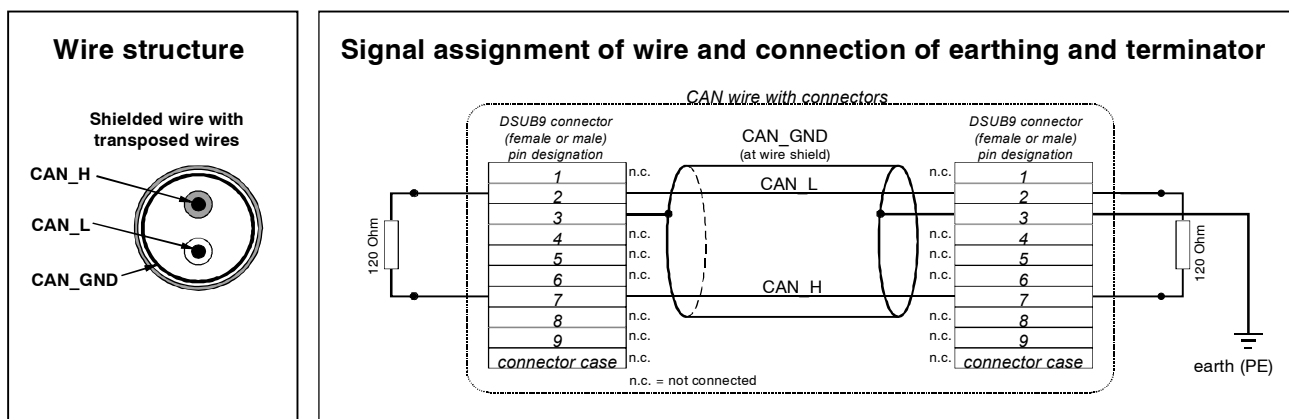
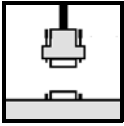


Figure: Structure and connection of wire



Wiring

Cabling

- for devices which have only one CAN connector per net use T-connector and dead-end feeder (shorter than 0.3 m) (available as accessory)

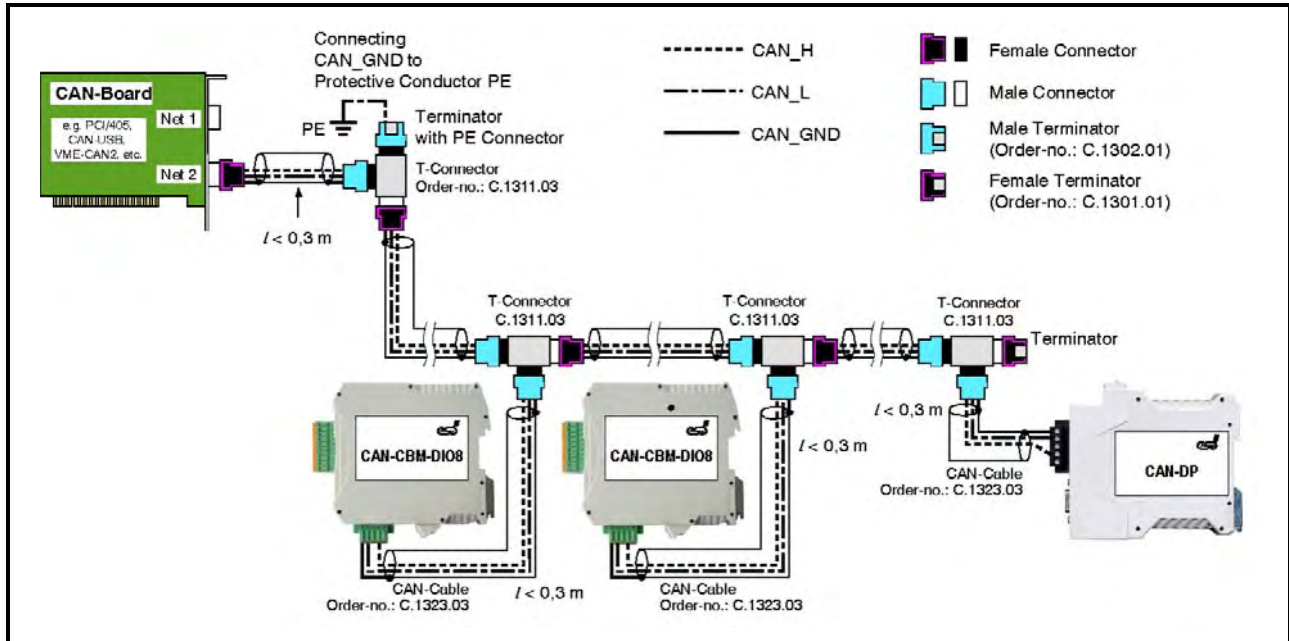


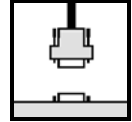
Figure: Example for correct wiring (when using single shielded wires)

Terminal Resistance

- use **external** terminator, because this can later be found again more easily!
- 9-pin DSUB-terminator with male and female contacts and earth terminal are available as accessories

Earthing

- CAN_GND has to be conducted in the CAN wire, because the individual esd modules are electrically isolated from each other!
- CAN_GND has to be connected to the earth potential (PE) at **exactly one** point in the net!
- each CAN user without electrically isolated interface works as an earthing, therefore: do not connect more than one user without potential separation!
- Earthing CAN e.g. be made at a connector

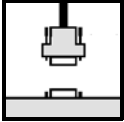


Wire Length

- Optical couplers are delaying the CAN signals. By using fast optical couplers and testing each board at 1 Mbit/s, however, esd CAN guarantee a reachable length of 37 m at 1 Mbit/s for most esd CAN modules within a closed net without impedance disturbances like e.g. longer dead-end feeders. (Exception: CAN-CBM-DIO8, -AI4 and AO4 (these modules work only up to 10 m with 1 Mbit/s))

Bit rate [Kbit/s]	Typical values of reachable wire length with esd interface l_{\max} [m]	CiA recommendations (07/95) for reachable wire lengths l_{\min} [m]
1000	37	25
800	59	50
666.6	80	-
500	130	100
333.3	180	-
250	270	250
166	420	-
125	570	500
100	710	650
66.6	1000	-
50	1400	1000
33.3	2000	-
20	3600	2500
12.5	5400	-
10	7300	5000

Table: Reachable wire lengths depending on the bit rate when using esd-CAN interfaces

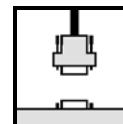


Wiring

Examples for CAN Wires

Manufacturer	Type of wire
U.I. LAPP GmbH Schulze-Delitzsch-Straße 25 70565 Stuttgart Germany www.lappkabel.de	e.g. UNITRONIC ®-BUS CAN UL/CSA (UL/CSA approved) UNITRONIC ®-BUS-FD P CAN UL/CSA (UL/CSA approved)
ConCab GmbH Äußerer Eichwald 74535 Mainhardt Germany www.concab.de	e.g. BUS-PVC-C (1 x 2 x 0.22 mm ²) Order No.: 93 022 016 (UL appr.) BUS-Schleppflex-PUR-C (1 x 2 x 0.25 mm ²) Order No.: 94 025 016 (UL appr.)
SAB Bröckskes GmbH&Co. KG Grefrather Straße 204-212b 41749 Viersen Germany www.sab-brockskes.de	e.g. SABIX® CB 620 (1 x 2 x 0.25 mm ²) Order No.: 56202251 CB 627 (1 x 2 x 0.25 mm ²) Order No.: 06272251 (UL appr.)

Note: Completely configured CAN wires can be ordered from **esd**.



7. CAN-Bus Troubleshooting Guide

The CAN-Bus Troubleshooting Guide is a guide to find and eliminate the most frequent hardware-error causes in the wiring of CAN-networks.

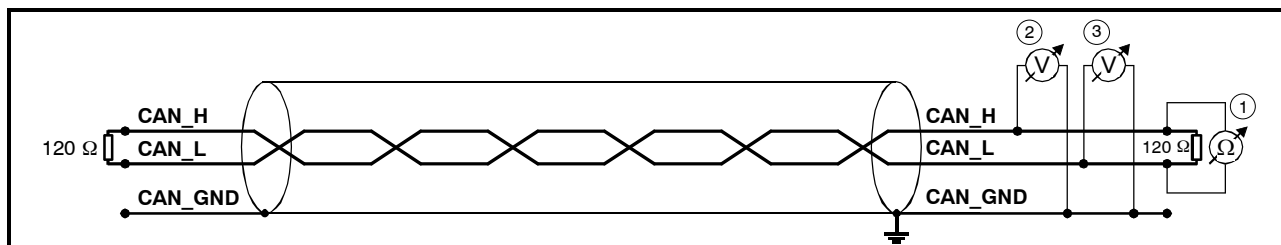


Figure: Simplified diagram of a CAN network

7.1 Termination

The termination is used to match impedance of a node to the impedance of the transmission line being used. When impedance is mismatched, the transmitted signal is not completely absorbed by the load and a portion is reflected back into the transmission line. If the source, transmission line and load impedance are equal these reflections are eliminated. This test measures the series resistance of the CAN data pair conductors and the attached terminating resistors.

To test it, please

1. Turn off all power supplies of the attached CAN nodes.
2. Measure the DC resistance between CAN_H and CAN_L at the middle and ends of the network (1) (see figure above).

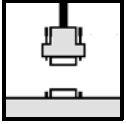
The measured value should be between 50 and 70 Ω. The measured value should be nearly the same at each point of the network.

If the value is below 50 Ω, please make sure that:

- there is no short circuit between CAN_H and CAN_L wiring
- there are not more than two terminating resistors
- the nodes do not have faulty transceivers.

If the value is higher than 70 Ω, please make sure that:

- there are no open circuits in CAN_H or CAN_L wiring
- your bus system has two terminating resistors (one at each end) and that they are 120 Ω each.



7.2 CAN_H/CAN_L Voltage

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 volts. Faulty transceivers can cause the idle voltages to vary and disrupt network communication.

To test for faulty transceivers, please

1. Turn on all supplies.
2. Stop all network communication.
3. Measure the DC voltage between CAN_H and GND **②** (see figure above).
4. Measure the DC voltage between CAN_L and GND **③** (see figure above).

Normally the voltage should be between 2.0 V and 4.0 V.

If it is lower than 2.0 V or higher than 4.0 V, it is possible that one or more nodes have faulty transceivers. For a voltage lower than 2.0 V please check CAN_H and CAN_L conductors for continuity. For a voltage higher than 4.0 V, please check for excessive voltage.

To find the node with a faulty transceiver please test the CAN transceiver resistance (see next page).

7.3 Ground

The shield of the CAN network has to be grounded at only one location. This test will indicate if the shielding is grounded in several places. To test it, please

1. Disconnect the shield wire (Shield) from the ground.
2. Measure the DC resistance between Shield and ground (see picture on the right hand).
3. Connect Shield wire to ground.

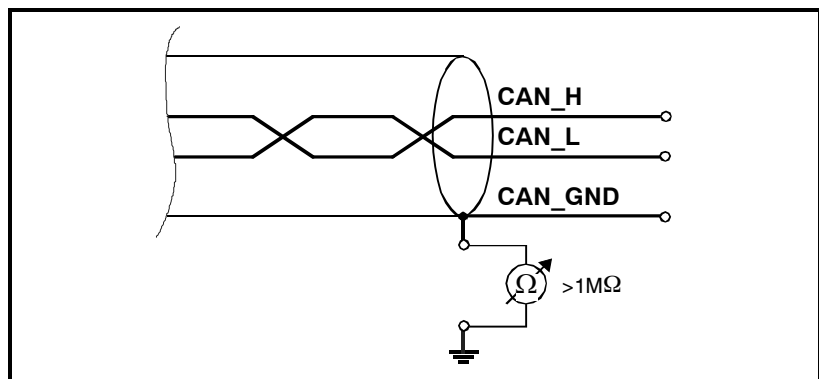
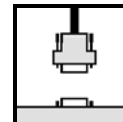


Fig.: Simplified schematic diagram of ground test measurement

The resistance should be higher than 1 MΩ. If it is lower, please search for additional grounding of the shield wires.



7.4 CAN Transceiver Resistance Test

CAN transceivers have one circuit that controls CAN_H and another circuit that controls CAN_L. Experience has shown that electrical damage to one or both of the circuits may increase the leakage current in these circuits.

To measure the current leakage through the CAN circuits, please use an resistance measuring device and:

1. Disconnect the node from the network. Leave the node unpowered (4) (see figure below).
2. Measure the DC resistance between CAN_H and CAN_GND (5) (see figure below).
3. Measure the DC resistance between CAN_L and CAN_GND (6) (see figure below).

Normally the resistance should be between 1 M Ω and 4 M Ω or higher. If it is lower than this range, the CAN transceiver is probably faulty.

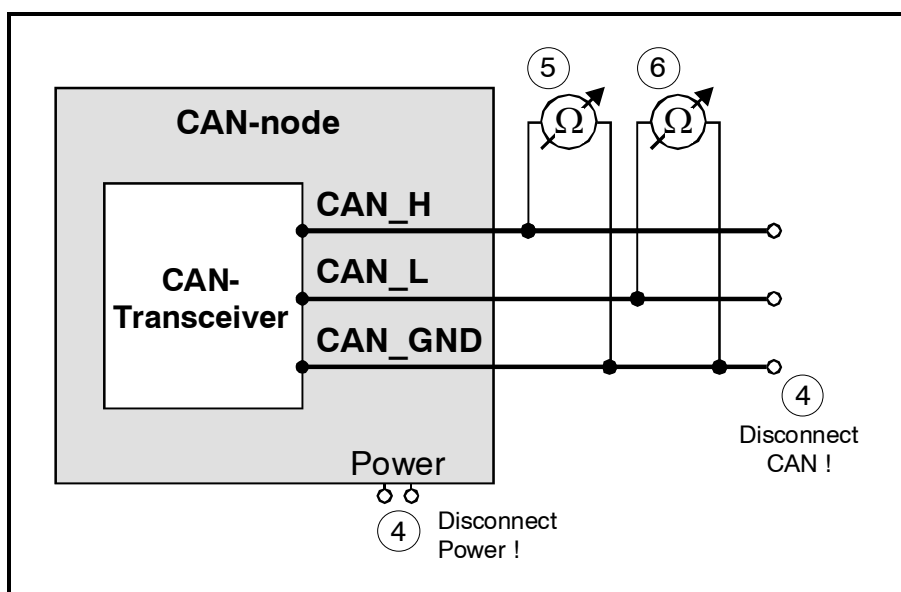
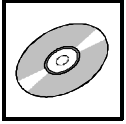


Figure: Simplified diagram of a CAN node



8. Software

Apart from basic descriptions of the 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 CAL/CANopen documentation 'CiA Draft Standard 301, V 4.02' and 'CiA Draft Standard Proposal 401, V 2.1'.

8.1 Definition of Terms

COB ...	Communication Object
Emergency-Id...	Emergency Data Object
NMT...	Network Management (Master)
Rx...	receive
SDO...	Service Data Object
Sync...	Sync(frame) Telegram
Tx...	transmit

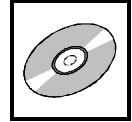
PDOs (Process Data Objects)

PDOs are used to transmit process data.

In the 'Receive'-PDO (RxPDO) the CAN-CBX-AO412 module receives data from the CANopen-net.

SDOs (Service Data Objects)

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.



8.2 NMT-Boot-up

The CAN-CBX-AO412- module can be initialized with the ‘Minimum Capability Device’ boot-up as described in CiA-Draft Standard 301 in chapter 9.4.

Usually a telegram to switch from *Pre-Operational* status to *Operational* status after boot-up is sufficient. For this the 2-byte telegram ‘01_h’, ‘00_h’, for example, has to be transmitted to CAN-identifier ‘0000_h’ (= Start Remote Node all Devices).

8.3 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 (CiA-Draft DS 301) or a manufacturer-specific code. By means of the MSBs of the index the object class of the parameter is defined.

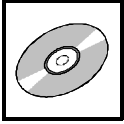
Part of the object directory are among others:

Index [Hex]	Object	Example
0001 ... 009F	Definition of data types	-
1000 ... 1FFF	Communication Profile Area	1001 _h : Error register
2000 ... 5FFF	Manufacturer Specific Profile Area	2000 _h : Calibrated offset
6000 ... 9FFF	Standardised Device Profile Area	according to Device Profile DS-40x
A000 ... FFFF	reserved	-

8.3.1 Access on the Object Directory via SDOs

The SDOs (Service Data Objects) are used to get access to the object directory of a device. An SDO therefore represents a ‘channel’ to access the parameter of the device. The access via this channel can be made in CAN-CBX-AO412 module state *Operational* and *Pre-operational*.

The SDOs are transmitted on ID ‘600_h + NodeID’ (request). The receiver acknowledges the parameter on ID ‘580_h + NodeID’ (response).



An SDO is structured as follows:

Identifier	Command code	Index		Sub-index	LSB	Data field		MSB
		(low)	(high)					

Example:

600 _h + Node-ID	23 _h (write)	00 (Index=1400 _h) (Receive-PDO-Comm-Para)	14 _h	01 (COB-def.)	7F _h	04 _h	00	00
					COB = 047F _h			

Description of the SDOs:

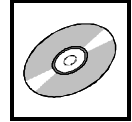
Identifier The parameters are transmitted on ID ‘600_h + NodeID’ (request).
The receiver acknowledges the parameters on ID ‘580_h + NodeID’ (response).

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

- 40_h = 64_{dec} : Read Request, i.e. a parameter is to be read
- 23_h = 35_{dec} : Write Request with 32-bit data, i.e. a parameter is to be set

The CAN-CBX-AO412 module responds to every received telegram with a response telegram. This can contain the following command codes:

- 43_h = 67_{dec} : Read Response with 32 bit data, this telegram contains the parameter requested
- 60_h = 96_{dec} : Write Response, i.e. a parameter has been set successfully
- 80_h = 128_{dec} : Error Response, i.e. the CAN module reports a communication error



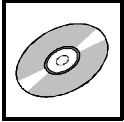
Frequently Used Command Codes

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 CiA DS 301, chapter “Service Data Object”.

Command	Number of data bytes	Command code [Hex]
Write Request (Initiate Domain Download)	1	2F
	2	2B
	3	27
	4	23
Write Response (Initiate Domain Download)	-	60
Read Request (Initiate Domain Upload)	-	40
Read Response (Initiate Domain Upload)	1	4F
	2	4B
	3	47
	4	43
Error Response (Abort Domain Transfer)	-	80

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

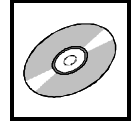
Data Field The data field has got a size of a maximum of 4 bytes and is always structured ‘LSB first, MSB last’. The least significant byte is always in ‘Data 1’. With 16-bit values the most significant byte (bits 8...15) is always in ‘Data 2’, and with 32-bit values the MSB (bits 24...31) is always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



Error Codes of the SDO Domain Transfer

The following error codes might occur (according to CiA DS 301, chapter “Abort SDO Transfer Protocol”):

Abort code [Hex]	Description
0x05040001	wrong command specifier
0x06010002	wrong write access
0x06020000	wrong index
0x06040041	object can not be mapped to PDO
0x06060000	access failed due to an 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



8.4 Overview of used CANopen-Identifiers

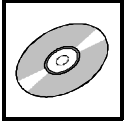
Function	Identifier [Hex]	Description
Network management	0	NMT
SYNC	80	Sync to all, (configurable via object 1005 _h)
Emergency Message	80 + <i>NodeID</i>	configurable with object 1014 _h
Tx-SDO	580 + <i>Node-ID</i>	SDO from CAN-CBX-AO412 (Tx)
Rx-SDO	600 + <i>Node-ID</i>	SDO from CAN-CBX-AO412 (Rx)
Node Guarding	700 + <i>NodeID</i>	configurable with object 100E _h

NodeID: CANopen-address set [1_h...7F_h]

8.4.1 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 page 15). 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 have to be restored (object 1011_h)



8.5 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	Default assignment
RxPDO1	n.a.	n.a.	n.a.	RxPDO1 is not used
RxPDO2	300 _h + Node-ID	8 byte	to CAN-CBX-AO412 (Rx/Receive-PDO)	D/A-values channel 1 to 4 as 16-bit values
RxPDO3	400 _h + Node-ID	8 byte	to CAN-CBX-AO412 (Rx/Receive-PDO)	Dummy mapping

Rx-PDO2 (-> CAN-CBX-AO412)

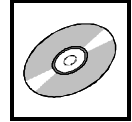
CAN-Identifier: 300_h + Node-ID

Byte	0	1	2	3	4	5	6	7
Parameter	<i>Analog_Output_16-Bit_1</i>		<i>Analog_Output_16-Bit_2</i>		<i>Analog_Output_16-Bit_3</i>		<i>Analog_Output_16-Bit_4</i>	

Rx-PDO3 (-> CAN-CBX-AO412)

CAN-Identifier: 400_h + Node-ID

Byte	0	1	2	3	4	5	6	7
Parameter	<i>Dummy_Mapping</i>		<i>Dummy_Mapping</i>		<i>Dummy_Mapping</i>		<i>Dummy_Mapping</i>	



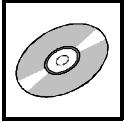
8.6 Setting and Reading the Analogue Values

8.6.1 Setting the Analogue Outputs

The analogue outputs are set, as soon as an object for setting the outputs of the CAN-CBX-AO412 is received.

8.6.2 Reading the Analogue Outputs

Differing from the CANopen Specification DS-301 the values of the analogue outputs can be read. If the CAN-CBX-AO412 module receives an RTR-frame, in response a Tx-PDO will be transmitted, which contains the D/A-values of the channels 1 to 4.

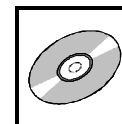


Implemented CANopen Objects

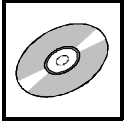
8.7 Implemented CANopen-Objects

A detailed Description of the objects can be taken from CiA DS-301.

Index [Hex]	Sub-index (max.) [Dec]	Description	Data type	R/W	Default value
1000	-	Device Type	unsigned 32	ro	00080191 _h
1001	-	Error Register	unsigned 8	ro	00 _h
1002	-	Manufacturer Status Register (internal use only!)	unsigned32	ro	00 _h
1003	10	Pre-Defined-Error-Field	unsigned32	rw	00 _h
1008	-	Manufacturer's Device Name	visible string	ro	"CAN-CBX-AO412"
1009	-	Manufacturer's Hardware Version	visible string	ro	x.yy (depending on version)
100A	-	Manufacturer's Software Version	visible string	ro	x.yy (depending on version)
100C	-	Guard Time	unsigned 16	rw	0000 _h
100D	-	Life Time Factor	unsigned 8	rw	00 _h
1010	3	Store Parameter	unsigned 32	rw	
1011	3	Restore Parameter	unsigned 32	rw	
1014	-	COB-ID Emergency Object	unsigned 32	rw	80 _h + Node-ID
1015	-	Inhibit Time EMCY	unsigned 16	rw	00 _h
1016	1	Consumer Heartbeat Time	unsigned 32	rw	00 _h
1017	-	Producer Heartbeat Time	unsigned 16	rw	00 _h
1018	4	Identity Object	unsigned 32	ro	Vendor Id: 00000017 _h Prod. Code: 23040002 _h
1020	0-2	Verify Configuration	unsigned 32	ro	n.a.



Index [Hex]	Sub-index [Hex]	Description	Data type	R/W
1401	2	2. Receive PDO-Parameter	PDO CommPar (20 _h)	rw
1402	2	3. Receive PDO-Parameter	PDO CommPar (20 _h)	rw
1601	0	2. Receive PDO-Mapping	PDO Mappping (21 _h)	ro
1602	0	3. Receive PDO-Mapping	PDO Mappping (21 _h)	ro



Implemented CANopen Objects

8.7.1 Device Type (1000_h)

INDEX	1000_h
Name	<i>device type</i>
Data Type	unsigned 32
Access Type	ro
Default Value	0008 0191

The value of the *device type* is: 0008.0191_h (Analogue output: 0008_h
Digital profile number: 0191_h)

Example: Reading the Device Type

The CANopen Master transmits the read request on identifier '603_h' (600_h + Node-ID) to the CAN-CBX-AO412-module with the module no. 3 (Node-ID=3_h):

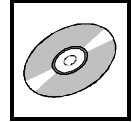
ID	RTR	LEN	DATA							
			1	2	3	4	5	6	7	8
603 _h	0 _h	8 _h	40 _h	00 _h	10 _h	00 _h	00 _h	00 _h	00 _h	00 _h
			Read Request	Index=1000 _h		Sub Index				

The CAN-CBX-AO412 module no. 3 responds to the master by means of read response on identifier '583_h' (580_h + Node-ID) with the value of the device type:

ID	RTR	LEN	DATA							
			1	2	3	4	5	6	7	8
583 _h	0 _h	8 _h	43 _h	00 _h	10 _h	00 _h	91 _h	01 _h	08 _h	00 _h
			Read Response	Index=1000 _h		Sub Index	dig. Profile Nr.191		Analog Output	

value of device type: 0008.0191_h

The data field is always structured following the rule 'LSB first, MSB last' (see page 37, data field).



8.7.2 Error Register (1001_h)

The CAN-CBX-AO412-module uses the error register to indicate error messages.

INDEX	1001_h
Name	<i>error register</i>
Data Type	unsigned 8
Access Type	ro
Default Value	0

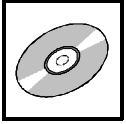
The following bits of the error register are being supported at present:

Bit	Meaning
0	<i>generic</i>
1	<i>current</i>
2	<i>voltage</i>
3	-
4	<i>communication error</i> (overrun, error state)
5	-
6	-
7	-

Bits which are not supported (-) are always returned as '0'. If an error is active, the according 'error' bit is set to '1'.

The following messages are supported:

00 _h	no errors
01 _h	generic error
02 _h	current error
04 _h	voltage error
10 _h	communication error

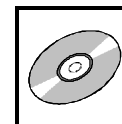


Implemented CANopen Objects

8.7.3 Manufacturer Status Register (1002_h)

This object is intended only for internal use!

INDEX	1002_h
Name	<i>Manufacturer status register</i>
Data Type	unsigned 32
Access Type	ro
Default Value	0



8.7.4 Pre-defined Error Field (1003_h)

INDEX	1003_h
Name	<i>pre-defined error field</i>
Data Type	unsigned 32
Access Type	rw
Default Value	0

The *pre-defined error field* provides an error history of the errors that have occurred on the device. 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.

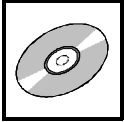
This module supports a maximum of 10 error entries. When the 11th error occurs the oldest error entry is deleted. In order to delete the entire error list, sub-index '0' has to be set to '0'. This is the only permissible write access to the object.

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

Index [Hex]	Subindex [Dec]	Description	Value range [Hex]	Default	Data type	R/W
1003	0	<i>no_of_errors_in_list</i>	0, 1...10	-	unsigned 8	rw
	1	<i>error-code n</i>	0...FFFFFFFF	-	unsigned 32	ro
	2	<i>error-code (n-1)</i>	0...FFFFFFFF	-	unsigned 32	ro
	:	:	:	:	:	ro
	10	<i>error-code (n-9)</i>	0...FFFFFFFF	-	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 error which occurred last
- in order to delete the error list this variable has to be set to '0'
- if *no_of_errors_in_list* ≠ 0, the error register (Object 1001_h) is set



Implemented CANopen Objects

error-code x The 32-bit long error code consists off the CANopen-emergency error code described in DS-301, Table 21 and the error code described by esd (manufacturer-specific error field).

Bit:	31 16	15 0
Contents:	<i>manufacturer-specific error field</i>	<i>emergency-error-code</i>

manufacturer-specific error field: for CAN-CBX-AO412 always '00', unless *emergency-error-code* = 2300_h (see below)

emergency-error-code: The following error-codes are supported:
 8120_h - CAN in Error Passive Mode
 8130_h - Lifeguard Error / Heartbeat Error
 8140_h - Recovered from "Bus Off"
 6000_h - Software-Error:
 -EEPROM Checksum error (no transmission of this message as emergency message)
 8210_h - PDO-length error (only an even number of bytes is valid)

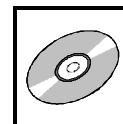
Emergency Message

The data of the emergency-frame transmitted by the CAN-CBX-AO412 have the following structure:

Byte:	0	1	2	3	4	5	6	7
Contents:	<i>emergency-error-code</i> (see above)		<i>error-register</i> 1001 _h	<i>no_of_errors_in_list</i> 1003,00 _h	-			

An emergency message is transmitted, if an error occurs. If this error occurs again, no further emergency message is generated.

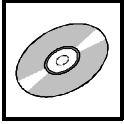
If the last error message disappears, again an emergency message is transmitted to indicate the error disappearance.



8.7.5 Manufacturer's Device Name (1008_h)

INDEX	1008_h
Name	<i>manufacturer's device name</i>
Data Type	visible string
Access Type	ro
Default Value	string: 'CAN-CBX-AO412'

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



Implemented CANopen Objects

8.7.6 Manufacturer's Hardware Version (1009_h)

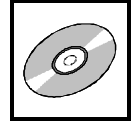
INDEX	1009_h
Name	<i>manufacturer's hardware version</i>
Data Type	visible string
Default Value	string: e.g. '1.0'

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

8.7.7 Manufacturer's Software Version 100A_h

INDEX	100A_h
Name	<i>manufacturer's software version</i>
Data Type	visible string
Default Value	string: e.g.: '1.0'

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



8.7.8 Guard Time (100C_h) and Life Time Factor (100D_h)

The CAN-CBX-AO412-module supports the node guarding or alternatively the heartbeat function (see page 58).



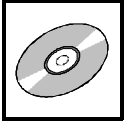
Note:

On 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 factors are evaluated together. Multiplying both values will give you the life time. The guard time is represented in milliseconds.

INDEX	100C _h
Name	<i>guard time</i>
Data Type	unsigned 16
Access Type	rw
Default Value	0 [ms]
Minimum Value	0
Maximum Value	FFFF _h (65,535 s)

INDEX	100D _h
Name	<i>life time factor</i>
Data Type	unsigned 8
Access Type	rw
Default Value	0
Minimum Value	0
Maximum Value	FF _h



Implemented CANopen Objects

8.7.9 Store Parameters (1010_h)

This object supports saving of parameters to the EEPROM.

In order 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 functionalities (refer to CiA DS 301 for more information).

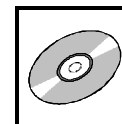
INDEX	1010_h
Name	<i>store parameters</i>
Data Type	unsigned 32

Index [Hex]	Subindex	Description	Value range	Data type	R/W
1010	0	<i>number_of_entries</i>	4 _h	unsigned 8	ro
	1	<i>save_all_parameters</i> (objects 1000 _h ... 9FFF _h)	no default, write: 65 76 61 73 _h (= ASCII: 'e' 'v' 'a' 's')	unsigned 32	rw
	2	<i>save_communication_parameter</i> (objects 1000 _h ... 1FFF _h)		unsigned 32	rw
	3	<i>save_application_parameter</i> (objects 6000 _h ... 9FFF _h)		unsigned 32	rw
	4	<i>save_manufacturer_parameter</i> (objects 2000 _h ... 5FFF _h)		unsigned 32	rw

Parameters which can be saved or loaded:

Communication parameter of the objects 1005_h ... 1020_h

Application parameter of the objects 6411_h



8.7.10 Restore Default Parameters (1011_h)

Via this command the default parameters, valid when leaving the manufacturer, are activated again. Every individual setting stored in the EEPROM will be lost. Only command 'Restore all Parameters' is being supported.

In order to avoid restoring of default parameters by mistake, restoring is only executed when a specific signature as shown below is transmitted.

Reading the index provides information about its parameter restoring capability (refer to CiA DS 301 for more information).

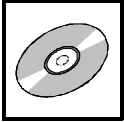
INDEX	1011_h
Name	<i>restore default parameters</i>
Data Type	unsigned 32

Index [Hex]	Subindex	Description	Value range	Data type	R/W
1011	0	<i>number_of_entries</i>	4	unsigned 8	ro
	1	<i>load_all_default_parameters</i> (objects 1000 _h ... 9FFF _h)	no default, write: 64 61 6F 6C _h (= ASCII: 'd' 'a' 'o' 'l')	unsigned 32	rw
	2	<i>load_communication_parameter</i> (objects 1000 _h ... 1FFF _h)		unsigned 32	rw
	3	<i>load_application_parameter</i> (objects 6000 _h ... 9FFF _h)		unsigned 32	rw
	4	<i>load_manufacturer_parameter</i> (objects 2000 _h ... 5FFF _h)		unsigned 32	rw

Parameters which can be saved or loaded:

Communication parameters of the objects 1005_h ... 1020_h

Application parameters of the object 6411_h



Implemented CANopen Objects

8.7.11 COB_ID Emergency Message (1014_h)

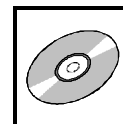
INDEX	1014_h
Name	<i>COB-ID emergency object</i>
Data Type	unsigned 32
Default Value	80 _h + Node-ID

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

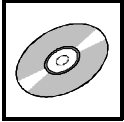
The identifier can take values between 0...7FF_h.



8.7.12 Inhibit Time EMCY (1015_h)

INDEX	1015_h
Name	<i>inhibit_time_emergency</i>
Data Type	unsigned 16
Access Type	rw
Value Range	0-FFFF _h
Default Value	0

The *Inhibit Time* for the EMCY message can be adjusted via this entry. The time is determined as a multiple of 100 μ s.



Implemented CANopen Objects

8.7.13 Consumer Heartbeat Time (1016_h)

INDEX	1016_h
Name	<i>consumer heartbeat time</i>
Data Type	unsigned 32
Default Value	No

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 (700_h + Node-ID). One or more heartbeat consumer receive the message. It has to 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-AO412 module.

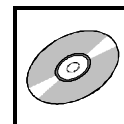
Each module can act as a heartbeat producer and a heartbeat consumer. The CAN-CBX-AO412 module supports maximum one heartbeat producer per CAN net.

Index [Hex]	Subindex [Dec]	Description	Value range [Hex]	Default [Dec]	Data type	R/W
1016	0	<i>number_of_entries</i>	1	1	unsigned 8	ro
	1	<i>consumer-heartbeat_time</i>	0...007FFFFFFF	0	unsigned 32	rw

Meaning of the variable *consumer-heartbeat_time_x*:

<i>consumer-heartbeat_time_x</i>			
Bit	3124	2316	150
Assignment	reserved (always '0')	<i>Node-ID</i> (unsigned 8)	<i>heartbeat_time</i> (unsigned 16)

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



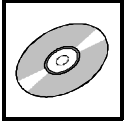
heartbeat_time Cycle time of heartbeat producer to transmit the heartbeat on the node-guarding ID (see object 100E_h).

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 = 0031 03E8_h

=> *Node-ID* = 31_h = 49_d
=> *heartbeat time* = 3E8_h = 1000_d => 1 s



Implemented CANopen Objects

8.7.14 Producer Heartbeat Time (1017_h)

INDEX	1017_h
Name	<i>producer heartbeat time</i>
Data Type	unsigned 16
Default Value	0 ms

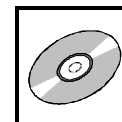
The producer heartbeat time defines the cycle time with which the CAN-CBX-AO412 module transmits a heartbeat-frame to the node-guarding ID.

If the value of the producer heartbeat time is higher than '0', it is active and stops the node-/ life-guarding (see page 51).

If the value of the producer-heartbeat-time is set to '0', transmitting heartbeats by this module is stopped.

Index [Hex]	Subindex	Description	Value range [Hex]	Default	Data type	R/W
1017	0	<i>producer-heartbeat_time</i>	0...FFFF	0 ms	unsigned 16	rw

producer-heartbeat_time Cycle time of heartbeat producer to transmit the heartbeat on the node-guarding ID (see object 100E_h).
The consumer-heartbeat time of the monitoring module must always be higher than the producer-heartbeat time of the heartbeat-transmitting module.



8.7.15 Identity Object (1018_h)

INDEX	1018_h
Name	<i>identity object</i>
Data Type	unsigned 32
Default Value	No

This object contains general information to the CAN module.

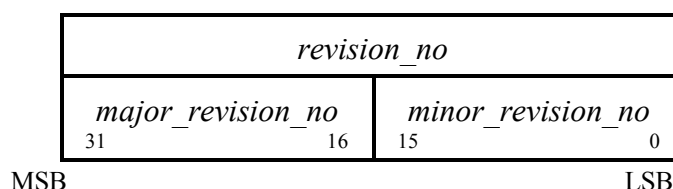
Index [Hex]	Subindex	Description	Value range [Hex]	Default	Data type	R/W
1018	0	<i>no_of_entries</i>	4	4	unsigned 8	ro
	1	<i>vendor_id</i>	0...FFFFFFFF	0000 0017 _h	unsigned 32	ro
	2	<i>product_code</i>	0...FFFFFFFF	2304 0002 _h	unsigned 32	ro
	3	<i>revision_number</i>	0...FFFFFFFF	0	unsigned 32	ro
	4	<i>serial_number</i>	0...FFFFFFFF	-	unsigned 32	ro

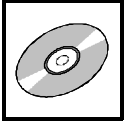
Meaning of the variable:

vendor_id This variable contains the esd-Vendor-ID. This is always 00000017_h.

product_code Here the esd-article-number of the product is stored.
 Example:
 The value '2304 0002_h' corresponds to the article number 'C.3040.02'.

revision_number Here the software version is stored. In accordance with DS 301 the two MSB represent the revision numbers of the major changes and the two LSB show the revision number of minor corrections or changes.





Implemented CANopen Objects

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.

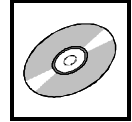
In the two MSB of *serial_no* the letters of the manufacturing lot are coded. They each contain the ASCII-code of the letter with the MSB set '1' in order to be able to differentiate between letters and numbers:

$$(\text{ASCII-Code}) + 80_{\text{h}} = \text{read_byte}$$

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

Example:

If the value 'C1C2 0105_h' is being read, this corresponds to the hardware-serial number code 'AB 0105'. This value has to correspond to the serial number of the module.



8.7.16 Verify Configuration (1020_h)

INDEX	1020_h
Name	<i>verify configuration</i>
Data Type	unsigned 32
Default Value	No

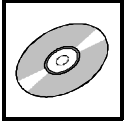
In this object date and the time of the last configuration can be stored to check whether the configuration complies with the expected configuration or not in the future.

Index [Hex]	Subindex [Dec]	Description	Value range [Hex]	Default [Dec]	Data type	R/W
1020	0	<i>no_of_entries</i>	2	2	unsigned 8	ro
	1	<i>configuration_date</i>	0...FFFFFFFF	0	unsigned 32	rw
	2	<i>configuration_time</i>	0...FFFFFFFF	0	unsigned 32	rw

Meaning of the variables:

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.



Implemented CANopen Objects

8.7.17 Receive PDO Communication Parameter 1401_h, 1402_h

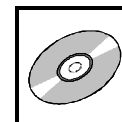
The objects ‘Receive PDO Communication Parameter 1401_h, 1402_h’ contain the communication parameters for the PDOs the CAN-CBX-AO412 is able to receive.

INDEX	1401_h
Name	<i>2. receive PDO parameter</i>
Data Type	PDOCommPar

INDEX	1402_h
Name	<i>3. receive PDO parameter</i>
Data Type	PDOCommPar

Index [Hex]	Subindex	Description	Value range [Hex]	Default	Data type	R/W
1401	0	<i>no_of_entries</i>	2	2	unsigned 8	ro
	1	<i>COB_ID used by PDO2</i>	1... 800007FF	300 _h + Node-ID	unsigned 32	rw
	2	<i>transmission type</i>	0...FF	255 _d	unsigned 8	rw
1402	0	<i>no_of_entries</i>	2	2	unsigned 8	ro
	1	<i>COB_ID used by PDO3</i>	1... 800007FF	400 _h + Node-ID	unsigned 32	rw
	2	<i>transmission type</i>	0...FF	255 _d	unsigned 8	rw

All *transmission types* are supported.



8.7.18 Receive PDO Mapping Parameter 1601_h, 1601_h

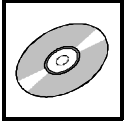
This object contains the PDO mapping parameters for the PDOs the CAN-CBX-AO412 is able to receive.

INDEX	1601_h
Name	<i>2. receive PDO mapping</i>
Data Type	PDO Mapping

INDEX	1602_h
Name	<i>3. receive PDO mapping</i>
Data Type	PDO Mapping

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

Index [Hex]	Subindex	Description	Value range [Hex]	Default	Data type	R/W
1601	0	<i>no_of_mapped_application_objects_in_PDO_2</i>	4	4	unsigned 8	ro
	1	<i>1st_application_object</i>	6411 0110 _h	6411 0110 _h	unsigned 32	rw
	2	<i>2nd_application_object</i>	6411 0210 _h , 6411 0310 _h	6411 0210 _h	unsigned 32	rw
	3	<i>3rd_application_object</i>	6411 0410 _h , 0003 0010 _h	6411 0310 _h	unsigned 32	rw
	4	<i>4th_application_object</i>		6411 0410 _h	unsigned 32	rw
1602	0	<i>no_of_mapped_application_objects_in_PDO_3</i>	4	4	unsigned 8	ro
	1	<i>1st_application_object</i>	6411 0110 _h	0003 0010 _h	unsigned 32	rw
	2	<i>2nd_application_object</i>	6411 0210 _h , 6411 0310 _h	0003 0010 _h	unsigned 32	rw
	3	<i>3rd_application_object</i>	6411 0410 _h , 0003 0010 _h	0003 0010 _h	unsigned 32	rw
	4	<i>4th_application_object</i>		0003 0010 _h	unsigned 32	rw

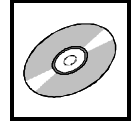


Device Profile Area

8.8 Device Profile Area

8.8.1 Overview of the implemented Objects 6411_h

Index [HEX]	Name	Data Type
6411	<i>Write Analogue Outputs 16-Bit</i>	signed16



8.8.2 Write Analogue Output 16-Bit (6411_h)

8.8.2.1 Assignment of the Sub-Indices

Index [Hex]	Subindex [Dec]	Description	Value range [Hex]	Default [Dec]	Data type	R/W
6411	0	<i>Number of entries</i>	4	4	unsigned 8	ro
	1	<i>AOUT1_value</i>	8000...7FFF	0	signed 16	rw
	2	<i>AOUT2_value</i>	8000...7FFF	0	signed 16	rw
	3	<i>AOUT3_value</i>	8000...7FFF	0	signed 16	rw
	4	<i>AOUT4_value</i>	8000...7FFF	0	signed 16	rw

8.8.2.2 Assignment of the variables *AOUTx_value* (x = 1...4)

The number in the name of the variable indicates the number of the analogue output channel.

With the variable *AOUTx_value* the output values of the channels can be specified individually. The analogue value contains the voltage as 16-bit value. The value is represented as two's complement. The firmware rounds up or down to the 13 bit (12 bit + sign) of the two digital-analog-converters (see page 22).

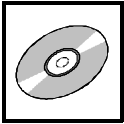
The bit value is calculated according to the following formula:

$$n_{\text{Bit}} \approx V_{\text{OUT}} \cdot \frac{2^{15}}{K \cdot 4,096\text{V}}$$

with K = 2,492
 V_{OUT...} output voltage
 n_{Bit...} Bit value

The analogue value is transmitted in two's complement representation.

Two's complement representation	Output voltage
8000 _h	-10,205 V
⋮	⋮
FFFB _h	-2,492 mV
FFFC _h ...0003 _h	0,000V
0004 _h	+2,492mV
⋮	⋮
7FFF _h	+10,205 V



8.8.2.3 Examples of CAN-Frames for the SDO-Transfer

For SDO transfer the 8-byte data section of the CAN telegram has the following structure:

Data byte	d1	d2	d3	d4	d5	d6	d7	d8
Content	2B _h	11 _h	64 _h	01h to 04h	LSB _{OutputX}	MSB _{OutputX}	various	various

Description of the data bytes:

- d1: Command code (here 2B_h = „Write Request for 2 data bytes“)
- d2 and d3: **Index** (here 6411_h = „Write_Analogue_Outputs_16-Bit“)
- d4: Subindex: Number of the output X (X = 1 to 4)
- d5 and d6: Data field, containing the voltage value for output X.
- d7 and d8: Not defined, entry optional

Telegram Example 1:

The output voltage of the output channel 1 shall be set to V_{OUT} = 5,0 V.

The bit value is calculated according to the following formula:

$$n_{\text{Bit}} \approx 5,0\text{V} \cdot \frac{2^{15}}{\text{K} \cdot 4,096\text{V}} \approx 16051_{\text{d}} = 3\text{EB}3_{\text{h}}$$

The following table shows the entries in the data section of the CAN telegram.

For output channel 1 the subindex has to be d4 = 01_h.

Data byte	d1	d2	d3	d4	d5	d6	d7	d8
Content	2B _h	11 _h	64 _h	01h	B3_h	3E_h	various	various

Telegram Example 2:

The output voltage of the output channel 2 shall be set to V_{OUT} = -5,0 V.

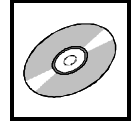
Under consideration of the representation in the two's complement, the bit value is calculated according to the following formula:

$$n_{\text{Bit}} \approx -5,0\text{V} \cdot \frac{2^{15}}{\text{K} \cdot 4,096\text{V}} \approx -16051_{\text{d}} = \text{C14D}_{\text{h}}$$

The following table shows the entries in the data section of the CAN telegram.

For output channel 2 the subindex has to be d4 = 02_h.

Data byte	d1	d2	d3	d4	d5	d6	d7	d8
Content	2B _h	11 _h	64 _h	02h	4D_h	C1_h	various	various



8.9 Manufacturer Specific Profile Area

8.9.1 Overview of the implemented Objects 2000_h

Index [HEX]	Name	Data Type
2000	<i>Calibrated Offset Value 16-Bit</i>	signed16

8.9.2 Calibrated Offset Value 16-Bit (2000_h)

Index [Hex]	Subindex [Dec]	Description	Value range [Hex]	Default [Dec]	Data type	R/W
2000	0	<i>Number_of_entries</i>	4	4	unsigned 8	ro
	1	<i>AOUT1_offset_value</i>	8000...7FFF	0	signed 16	rw
	2	<i>AOUT2_offset_value</i>	8000...7FFF	0	signed 16	rw
	3	<i>AOUT3_offset_value</i>	8000...7FFF	0	signed 16	rw
	4	<i>AOUT4_offset_value</i>	8000...7FFF	0	signed 16	rw

Assignment of the variable *AOUTx_offset_value* (x = 1...4):

The number in the name of the variable indicates the number of the corresponding analogue output channel.

9. References

- [1] Linear Technology, Data sheet: LTC 2600/LTC2610/LTC2620 Octal 16-/14-/12-Bit Rail-to-Rail DACs in Lead SSOP, USA, 2600fa, LT/TP1103 1K RevA