CAN-CBX-COM2

CAN - RS232, RS422, RS485 oder TTY-Interface

Handbuch

zu Produkt: C.3055.02, C.3055.03

esd electronic system design 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

Hinweis

Der Inhalt dieses Handbuches wurde mit größter Sorgfalt erarbeitet und geprüft. **esd** übernimmt jedoch keine Verantwortung für Schäden, die aus Fehlern in der Dokumentation resultieren könnten. Insbesondere Beschreibungen und technische Daten sind keine zugesicherten Eigenschaften im rechtlichen Sinne.

esd hat das Recht, Änderungen am beschriebenen Produkt oder an der Dokumentation ohne vorherige Ankündigung vorzunehmen, wenn sie aus Gründen der Zuverlässigkeit oder Qualitätssicherung vorgenommen werden oder dem technischen Fortschritt dienen.

Sämtliche Rechte an der Dokumentation liegen bei **esd**. Die Weitergabe an Dritte und Vervielfältigung jeder Art, auch auszugsweise, sind nur mit schriftlicher Genehmigung durch **esd** gestattet.
© 2015 esd electronic system design gmbH, Hannover

esd electronic system design gmbH
Vahrenwalder Str. 207
30165 Hannover

Tel.: 0511/372 98-0
FAX : 0511/372 98-68
E-Mail: info@esd.eu
Internet: www.esd.eu

Trademark-Hinweise

CiA® und CANopen® sind eingetragene Gemeinschaftsmarken des CAN in Automation e.V.
Alle anderen hier aufgeführten Markenzeichen, Produktnamen, Firmennamen und Firmenlogos sind Eigentum des jeweiligen Rechteinhabers.

Handbuch-Datei:	I:\Texte\Doku\MANUALS\CAN\CBX\CAN-CBX-COM2\Deutsch\CAN-CBX-COM2_Manual_de_13.wpd
Datum der Druckvorlagen-herstellung:	2015-02-05

Beschriebene Software:	CANopen
Revision:	V1.ODJ

Änderungen in den Kapiteln

Die hier aufgeführten Änderungen im Handbuch betreffen sowohl Änderungen in der **Firmware** als auch reine Änderungen in der **Beschreibung** der Sachverhalte.

Version	Kapitel	Änderungen gegenüber Vorversion
1.3	-	Darstellungskonventionen aufgenommen
	2.	Hinweis zu Terminatorsteckern in Schritt 2 entfernt
	3.	Kapitel überarbeitet, Hinweise eingefügt
	4.5	Kapitel Software-Unterstützung verschoben
	5.1	Steckerbezeichnung Leitungsstecker ergänzt
	5.3.5	Tabelle Pin-Belegung aufgeteilt (COM A immer RS232)
	8.	Kapitel überarbeitet bzgl. Produktspezifische Eigenschaften, Beschreibung Self-Start-Feature ($1F80_h$, $1F91_h$) aufgenommen
	10.	Konformitätserklärung aktualisiert
	11.	Handbuch englisch in Bestellhinweise aufgenommen

Sicherheitshinweise

- Bitte beachten Sie im Umgang mit dem CAN-CBX-Modul die folgenden Sicherheitshinweise und lesen Sie dieses Handbuch aufmerksam durch, um Schäden am CAN-CBX-Modul und Verletzungen zu vermeiden.
- Das CAN-CBX-Modul darf nicht geöffnet werden.
- Das Gerät muss vor der Inbetriebnahme fest montiert sein.
- Lassen Sie keine Flüssigkeiten in das CAN-CBX-Modul eindringen, da sonst elektrische Schläge oder Kurzschlüsse die Folge sein können.
- Schützen Sie das CAN-CBX-Modul vor Feuchtigkeit und Dämpfen.
- Schützen Sie das CAN-CBX-Modul vor Stößen und Vibrationen.
- Das CAN-CBX-Modul wird möglicherweise während des normalen Betriebs warm. Achten Sie stets auf ausreichende Luftzufuhr, damit die Wärme abgeführt werden kann.
- Betreiben Sie das CAN-CBX-Modul nicht in unmittelbarer Nähe von Wärmequellen und setzen Sie es keiner unnötigen Wärmestrahlung aus. Die zulässige Umgebungstemperatur ist in den technischen Daten festgelegt.
- Verwenden Sie keine beschädigten Leitungen für den Anschluss des CAN-CBX-Moduls und beachten Sie die Verdrahtungshinweise zum CAN-Bus am Ende dieses Handbuchs.
- Bei Beschädigungen am Gerät, die die Sicherheit betreffen könnten, müssen unverzüglich geeignete Maßnahmen getroffen werden, die eine Gefährdung von Menschen und Haus- und Nutztieren sowie Gütern verhindern.
- Mit der Einrichtung verbundene Stromkreise müssen gegen gefährliche Spannungen ausreichend geschützt sein (SELV nach EN 60950-1).
- Das CAN-CBX-Modul darf nur an Versorgungsstromkreisen betrieben werden, die berührungssicher sind. Ein Netzteil, welches eine Schutzkleinspannung (SELV oder PELV) nach EN 60950-1 zur Verfügung stellt, erfüllt diese Bedingung.

Konformität

Das CAN-CBX-COM2 ist ein industrielles Produkt und erfüllt die in der Konformitätserklärung am Ende dieses Handbuchs angegebenen EU-Richtlinien und Normen zur EMV für industrielle Umgebungen.

Warnung: In Wohnbereich, Geschäfts- und Gewerbebereichen sowie Kleinbetrieben kann das CAN-CBX-COM2 Funkstörungen verursachen. In diesem Fall ist es erforderlich, dass der Anwender angemessene Maßnahmen ergreift.

Qualifiziertes Personal

Diese Dokumentation wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuer- und Automatisierungstechnik. Die Installation und Inbetriebnahme des Produkts darf nur von qualifiziertem Personal vorgenommen werden, das berechtigt ist, Geräte, Systeme und Stromkreise gemäß den Standards der Sicherheitstechnik in Betrieb zu nehmen.

Bestimmungsgemäßer Gebrauch

Die bestimmungsgemäße Verwendung ist der Einsatz CANopen Modul mit zwei seriellen Schnittstellen . Der Hersteller haftet nicht für Schäden, die durch unsachgemäßen Gebrauch, nicht bestimmungsgemäße Verwendung oder in Folge von Nichtbeachtung der Sicherheitshinweise und Warnungen verursacht werden.

- Das CAN-CBX-COM2 ist nur für die Anwendung in geschlossenen Räumen vorgesehen.
- Das CAN-CBX-COM2 darf nicht in explosionsgefährdeten Bereichen und Zonen für Gase und Stäube sowie in explosivstoffgefährdeten Bereichen eingesetzt werden.
- Der Einsatz zu medizinischen Zwecken ist nicht zulässig.

Wartungshinweis

Innerhalb und außerhalb des CAN-CBX-COM2 befinden sich keine vom Anwender zu wartenden Komponenten. Jeder Eingriff in das Gerät durch nicht von esd autorisierte Personen führt zum Verlust aller Garantieansprüche.

Umwelthinweis

Auf Dauer unbrauchbar gewordene Geräte sind in geeigneter Weise zu entsorgen oder dem Hersteller zur Entsorgung zu übergeben. Bitte leisten auch Sie Ihren Beitrag zum Schutz unserer Umwelt.

Inhalt

1. Übersicht 9
1.1 Beschreibung des Moduls 9

2. Schnelle Inbetriebnahme 10

3. Hardware-Installation 12
3.1 Anschlussplan 12
3.2 LED-Anzeigen 13
3.2.1 Blinkzustände der Activity-LEDs 13
3.2.2 Blinkzustände der Status-LEDs 14
3.2.3 Bedeutung der CAN-Error-LED 14
3.2.4 Bedeutung der CANopen-Status-LED 15
3.2.5 Bedeutung der Error-LED 15
3.2.6 Bedeutung der Power-LED 16
3.2.7 Besondere Blinkzustände 16
3.2.8 Zuordnung der LED-Bezeichnung zur Bezeichnung im Schaltplan 16
3.3 Kodierschalter 17
3.3.1 Einstellung der Node-ID über die Kodierschalter Low und High 17
3.3.2 Einstellung der CAN-Bitrate über den Kodierschalter Baud 18
3.3.3 Zuordnung der Kodierschalterbezeichnung zur Bezeichnung im Schaltplan 18

4. Technische Daten 19
4.1 Allgemeine technische Daten 19
4.2 CPU 20
4.3 CAN-Schnittstelle 20
4.4 Serielle Schnittstellen 21
4.5 Software-Unterstützung 21
4.6 Einbau des Moduls bei Verwendung des InRailBus-Verbinders 22
4.6.1 Anschluss über den CBX-InRailBus 23
4.6.2 Anschluss der Versorgungsspannung 24
4.6.3 Anschluss von CAN 25
4.7 Ausbau des CAN-CBX-Moduls vom InRailBus 25

5. Schnittstellen und Steckerbelegungen 26
5.1 Spannungsversorgung 24V (X100) 26
5.2 CAN 27
5.2.1 CAN-Interface-Schaltung 27
5.2.2 CAN-Stecker (X400) 28
5.2.3 CAN und Versorgungsspannung über InRailBus X101 29
5.3 Serielle Schnittstellen COM A und COM B 30
5.3.1 RS232-Schnittstelle 30
5.3.2 Option: RS422-Schnittstelle 31
5.3.3 Option: RS485-Schnittstelle 31
5.3.4 Option: TTY(20 mA)-Schnittstelle 32
5.3.5 Steckerbelegung 33
5.4 Zuordnung der Stecker-Bezeichnung zur Bezeichnung im Schaltplan 35
5.5 Leiteranschluss/Leiterquerschnitt 36

6. Korrekte Verdrahtung galvanisch getrennter CAN-Netze 37
6.1 Standards zur CAN-Verdrahtung 37
6.2 Leicht störbehaftete Industrieumgebung (zweiadrig verdrillte Leitung) 38
6.2.1 Grundregeln 38
6.2.2 Verkabelung 39
6.2.3 Abschlusswiderstand 39
6.3 Stark störbehaftete Industrieumgebung (vieradrig verdrillte Leitung) 40
6.3.1 Grundregeln 40
6.3.2 Verkabelung 41
6.3.3 Abschlusswiderstand 41
6.4 Erdung 42
6.5 Buslänge 42
6.6 Beispiele für CAN-Kabel 43
6.6.1 Kabel für leicht störbehaftete Industrieumgebung (zweiadrig) 43
6.6.2 Kabel für stark störbehaftete Industrieumgebung (vieradrig) 43

7. CAN-Bus Troubleshooting Guide 44
7.1 Bus-Abschluss 44
7.2 Erdung 45
7.3 Kurzschluss in der CAN-Verdrahtung 45
7.4 CAN_H/CAN_L-Spannungen 45
7.5 CAN-Transceiver Widerstandstest 46
7.6 Support bei esd 46

8. CANopen-Firmware 47
8.1 Begriffsdefinition 47
8.2 NMT-Boot-up 48
8.3 Das CANopen-Objektverzeichnis 48
8.4 Communication Parameter 49
8.4.1 Zugriff auf das Objektverzeichnis über SDOs 49
8.4.2 Nichtflüchtiges Speichern von Parametern im EEPROM 51
8.4.3 Wiederherstellen des Default-Zustandes der Parameter 52
8.5 Übersicht der verwendeten CANopen-Identifier 53
8.5.1 Einstellung der COB-ID 53
8.6 Default PDO-Belegung 54
8.7 Communication Profile Area 55
8.7.1 Verwendete Bezeichnungen und Abkürzungen 55
8.8 Implementierte CANopen-Objekte 56
8.8.1 Device Type (1000_h) 58
8.8.2 Error Register (1001_h) 59
8.8.3 Manufacturer Status Register 1002_h 60
8.8.4 Pre-defined Error Field (1003_h) 61
8.8.5 COB-ID of SYNC-Message (1005_h) 63
8.8.6 Communication Cycle Period (1006_h) 64
8.8.7 Manufacturer Device Name (1008_h) 65
8.8.8 Manufacturer Hardware Version (1009_h) 66
8.8.9 Manufacturer Software Version ($100A_h$) 66
8.8.10 Guard Time ($100C_h$) und Life Time Factor ($100D_h$) 67
8.8.11 Store Parameters (1010_h) 68
8.8.12 Restore Default Parameters (1011_h) 70

8.8.13 COB_ID Emergency Message (1014_h) 72
8.8.14 Inhibit Time EMCY (1015_h) 73
8.8.15 Consumer Heartbeat Time (1016_h) 74
8.8.16 Producer Heartbeat Time (1017_h) 76
8.8.17 Identity Object (1018_h) 77
8.8.18 Synchronous Counter Overflow Value (1019_h) 79
8.8.19 Verify Configuration (1020_h) 80
8.8.20 Error Behaviour Object (1029_h) 81
8.8.21 NMT Startup ($1F80_h$) 82
8.8.22 Self Starting Nodes Timing Parameters ($1F91_h$) 83
8.8.23 RPDO Communication Parameter 1400_h, 1402_h 84
8.8.24 RPDO Mapping Parameter 1600_h, 1602_h 86
8.8.25 TPDO Communication Parameter 1800_h, 1802_h 87
8.8.26 TPDO Mapping Parameter $1A00_h$, $1A02_h$ 89
8.9 Übertragungsarten 90
8.9.1 Unterstützte Übertragungsarten nach DSP-301, Tabelle 55 90
8.10 Datentransfer und PDO-Belegung 91
8.10.1 Funktionsweise der lokalen Firmware 91
8.11 Device Profile Area 93
8.11.1 Parameter-Übersicht 6000_h...$9FFF_h$ 93
8.12 Manufacturer Specific Profile Area 94
8.12.1 Parameter-Übersicht 2800_h - 2981_h 94
8.12.2 Serielle Kommunikationsparameter 2800_h, 2900_h 95
8.12.3 *Tx_Cycle_String* 2810_h, 2910_h 117
8.12.4 *Dummy_Rx_Object* 2880_h, 2980_h 118
8.12.5 *Dummy_Tx_Object* 2881_h, 2981_h 118
8.13 Programm Download über CANopen-Objekte 119
8.13.1 Download-Steuerung über Objekt $1F51_h$ 120
8.13.2 Download-Steuerung über Objekt $1F52_h$ 121

9. Referenzen 122

10. Konformitätserklärung 123

11. Bestellhinweise 124

Darstellungskonventionen

In diesem Handbuch werden die folgenden Darstellungen zur Unterscheidung und Hervorhebung der aufgelisteten Programmbestandteile verwendet.

Darstellung von	**Beispiel**
Datei- und Pfadname	`/dev/null or <stdio.h>`
Funktionsnamen	***open()***
Konstanten	`NULL`
Datentypen	`uint32_t`
Variablennamen	*Count*

Das Dokument enthält die folgenden Textboxen um wichtige Informationen und Warn- und Gefahrenhinweise hervorzuheben.

Achtung:
Warn- und Gefahrenhinweise, die sie zu ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen.

Hinweis:
Hinweise enthalten wichtige oder nützliche Informationen.

Zahlendarstellung
Alle Zahlenangaben in diesem Dokument sind Dezimalzahlen, sofern nicht anders angegeben. Hexadezimalzahlen sind mit einem angehängten $_h$ dargestellt. Zum Beispiel, wird 42 hexadezimal als $2A_h$ dargestellt.

1. Übersicht

1.1 Beschreibung des Moduls

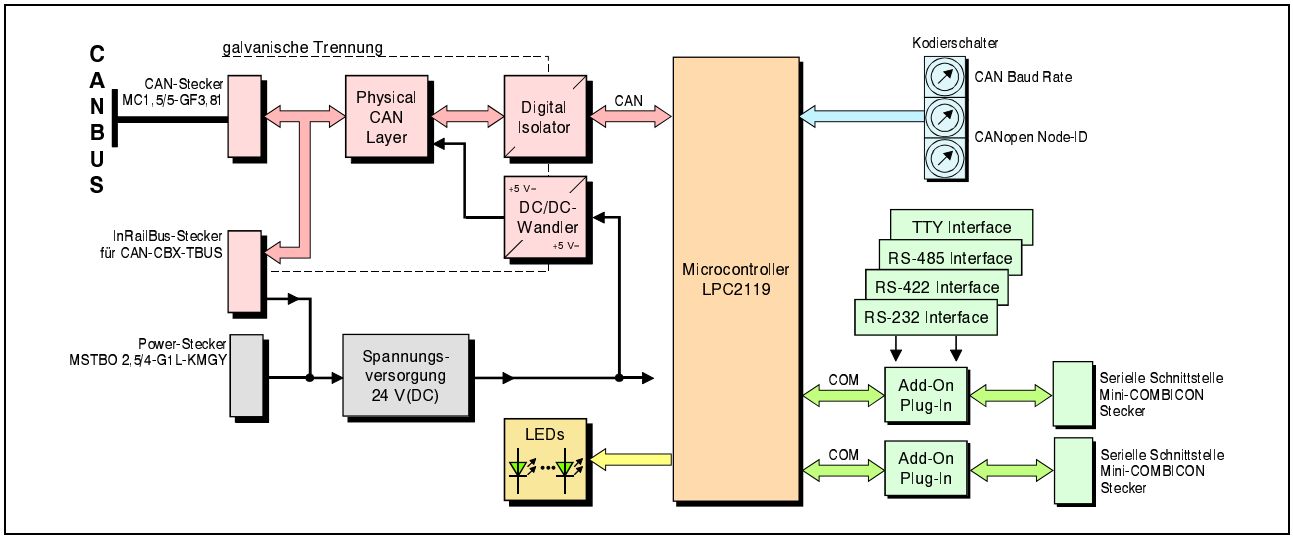


Abb. 1: Blockschaltbild des CAN-CBX-COM2-Modul

Das Modul CAN-CBX-COM2 ist ein CANopen® Modul, das es ermöglicht die Kommunikation zweier serieller Schnittstellen in ein CANopen Netzwerk zu integrieren.
In der Grundversion sind zwei RS232-Treiber bestückt.
Die serielle Schnittstelle COM B läßt sich über Add-On-Platinen (Piggybacks) als RS232-, RS422-, RS485- oder TTY-Interface konfigurieren.

Als CBX-Modul kann das CAN-CBX-COM2 als einzelnes Modul betrieben oder an den InRailBus angeschlossen werden.

Das CAN-CBX-COM2-Modul verfügt über einen ARM7®-Microcontroller, der als Controller für die seriellen Schnittstellen und den CAN-Bus arbeitet. Die CAN-Daten werden im lokalen SRAM zwischengespeichert. Die Firmware wird im Flash gehalten. Zur Speicherung von Parametern dient ein serielles EEPROM.

Die Versorgungsspannung und der CAN-Bus-Anschluss werden entweder über den in die Hutschiene integrierten InRailBus-Verbinder oder über separate Stecker zugeführt.

Die nach ISO 11898-2 ausgelegte CAN-Schnittstelle und gestattet eine maximale Datenübertragungsrate von 1 MBit/s. Das CAN-Interface ist durch einen digitalen Datenkoppler und DC/DC-Wandler galvanisch getrennt.

Die CANopen Knotennummer und die CAN-Bitrate können über drei außen liegende Drehschalter eingestellt werden. Die Firmware unterstützt CANopen gemäß CiA® 301 [1].

2. Schnelle Inbetriebnahme

Schritt	Aktion	siehe Seite
!	**Bevor Sie mit der Inbetriebnahme beginnen lesen Sie die Sicherheitshinweise am Anfang des Handbuches sorgfältig!**	4
!	**Beachten Sie auch die Kapitel: “Einbau und Verdrahtung des Moduls” sowie “Korrekte Verdrahtung galvanisch getrennter CAN-Netze”!**	22, 37
1	Montieren und verdrahten Sie das CAN-CBX-COM2-Modul (Spannungsversorgung, CAN-Bus, Serielle Schnittstellen)	12
2	Beachten Sie beim Anschluss des CAN-Bus bitte, dass der CAN-Bus an beiden Enden abgeschlossen werden muss. esd bietet hierzu T-Stücke und Terminatoren an. Das CAN-GND-Signal ist außerdem an genau einem Punkt im CAN-Netz zu erden. Ein CAN-Teilnehmer, dessen CAN-Interface nicht galvanisch getrennt ist, ist mit einer Erdung des CAN-GND gleichzusetzen.	-
3	Einstellen der Baudrate (Nur wenn eine andere als die Default-Einstellung gewünscht wird.) Die Default-Bitrate beträgt 1 MBit/s. Die Bitrate kann über den Kodierschalter BAUD, wie im Kapitel: “Einstellung der CAN-Bitrate über den Kodierschalter” beschrieben, gesetzt werden.	18
4	Einstellen der Modul-Nummer (Node-ID). Die Node-ID wird mit den Kodierschaltern LOW und HIGH, gesetzt. Sie darf Werte zwischen 1 und 127 (01-7F_h) annehmen.	17
5	Legen sie die Versorgungsspannung an	-
6	Modul starten mit NMT-Start Kommando CAN-Identifier: 0 Len: 2 Byte Data: 1, 0 (Node-ID, 0 = alle Module starten)	-

<table>
<tr><th>Schritt</th><th>Aktion</th><th>siehe Seite</th></tr>
<tr><td>7</td><td>Daten CAN -> seriell senden
<table>
<tr><th>CAN-Identifier</th><th>Len</th><th>Data</th></tr>
<tr><td>200_h + Node-ID</td><td>8 Byte</td><td>0...8 Byte Nutzdaten COM A</td></tr>
<tr><td>400_h + Node-ID</td><td>8 Byte</td><td>0...8 Byte Nutzdaten COM B</td></tr>
</table></td><td>-</td></tr>
<tr><td>8</td><td>Daten seriell -> CAN empfangen
<table>
<tr><th>CAN-Identifier</th><th>Len</th><th>Data</th></tr>
<tr><td>180_h + Node-ID</td><td>8 Byte</td><td>0...8 Byte Nutzdaten COM A</td></tr>
<tr><td>380_h + Node-ID</td><td>8 Byte</td><td>0...8 Byte Nutzdaten COM B</td></tr>
</table></td><td>-</td></tr>
</table>

3. Hardware-Installation

3.1 Anschlussplan

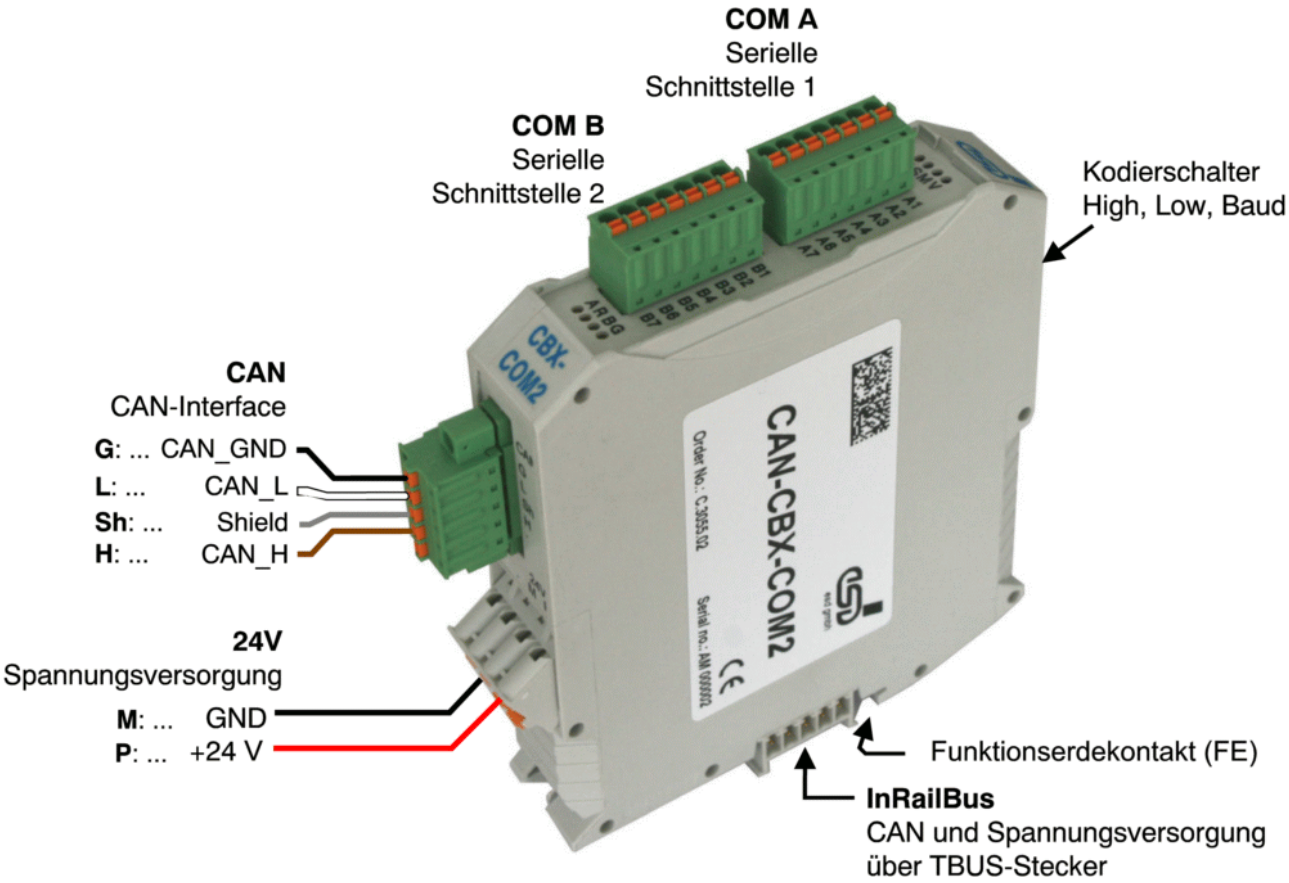


Abb. 2: Spannungsversorgungs- und CAN-Anschluss des CAN-CBX-COM2-Moduls

Hinweis:
Hinweise zum Leiteranschluss/Leiterquerschnitt entnehmen Sie bitte Seite 36.
Die Signalbelegungen der Steckverbinder sind in tabellarischer Form ab Seite 26 abgedruckt.

3.2 LED-Anzeigen

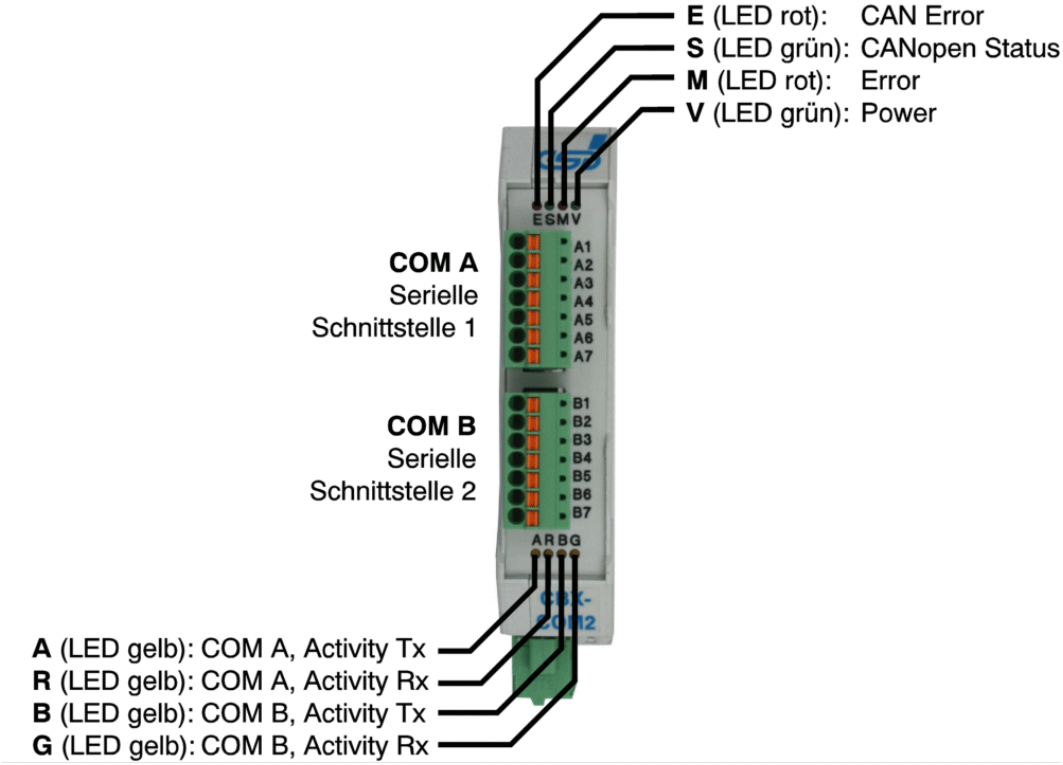


Abb. 3: Position der LEDs in der Frontplatte

Das CAN-CBX-COM2-Modul verfügt über vier Status-LEDs und vier Activity-LEDs.

3.2.1 Blinkzustände der Activity-LEDs

LED	Funktion	Farbe	Zustand	Bedeutung
A	COM A, Activity Tx	gelb	aus	CAN-CBX-COM2 sendet keine Daten auf COM A
			blinken	CAN-CBX-COM2 sendet Daten auf COM A
R	COM A, Activity Rx	gelb	aus	CAN-CBX-COM2 empfängt keine Daten auf COM A
			blinken	CAN-CBX-COM2 empfängt Daten auf COM A
B	COM B, Activity Tx	gelb	aus	CAN-CBX-COM2 sendet keine Daten auf COM B
			blinken	CAN-CBX-COM2 sendet Daten auf COM B
G	COM B, Activity Rx	gelb	aus	CAN-CBX-COM2 empfängt keine Daten auf COM B
			blinken	CAN-CBX-COM2 empfängt Daten auf COM B

Tabelle 2: Anzeigefunktion der Activity-LEDs

3.2.2 Blinkzustände der Status-LEDs

Die Bezeichnung der Blinkzustände der einzelnen Status-LEDs ist in Anlehnung an die von der CiA empfohlenen Bezeichnungen [3] gewählt. Die Bedeutung der Blinkzustände ist in den folgenden Kapiteln beschrieben. Es gibt prinzipiell 8 Blinkzustände, die die Status-LEDs annehmen können:

Blinkzustand	Anzeige
an	LED an
aus	LED aus
Blinken	LED blinkt mit 2,5 Hz
Flackern	LED flackert mit 10 Hz
1 Blitz	LED 200 ms an, 1400 ms aus
2 Blitze	LED 200 ms an, 200 ms aus, 200 ms an, 1000 ms aus
3 Blitze	LED 2x (200 ms an, 200 ms aus) + 1x (200 ms an, 1000 ms aus)
4 Blitze	LED 3x (200 ms an, 200 ms aus) + 1x (200 ms an, 1000 ms aus)

Tabelle 3: Anzeigefunktion der Status-LEDs

Hinweis:
Rote und grüne LEDs werden entsprechend der CANopen Spezifikation [3] grundsätzlich gegenphasig angesteuert.
Bei bestimmten Blinkzuständen kann es bei gleichzeitiger Betrachtung aller LEDs zu einer Fehlinterpretation von Blinkzuständen nebeneinanderliegender LEDs kommen. Es wird daher ggf. empfohlen die Blinkzustände nebeneinanderliegender LEDs einzeln zu betrachten, indem die anderen LEDs verdeckt werden.

3.2.3 Bedeutung der CAN-Error-LED

LED-Kennung			Anzeigefunktion	
Auf-druck	Name	Farbe	Blinkzustand	Bedeutung
E	CAN-Error	rot	aus	kein Fehler
			1 Blitz	CAN-Controller ist in *Error Active*
			an	CAN-Controller ist *Bus Off* (oder Kodierschalterposition ID-node > 7Fh beim einschalten; siehe 'Besondere Blinkzustände' auf Seite 16)
			2 Blitze	Heartbeat- oder Nodeguard-Fehler aufgetreten. Die LED geht automatisch in einen anderen Status über, wenn wieder Nodeguard/Heartbeat-Messages empfangen werden.

Tabelle 4: Anzeigefunktion der roten CANopen Error-LED

3.2.4 Bedeutung der CANopen-Status-LED

LED-Kennung			Anzeigefunktion	
Auf-druck	Name	Farbe	Blinkzustand	Bedeutung
S	CANopen-Status	grün	Blinken	*Preoperational*
			an	*Operational*
			1 Blitz	*Stopped*
			3 Blitze	Modul ist im Bootloader-Mode (oder Kodierschalterposition ID-Node > 7Fh beim Einschalten; siehe Seite 16)

Tabelle 5: Anzeigefunktion der CANopen Status-LED

3.2.5 Bedeutung der Error-LED

LED-Kennung			Anzeigefunktion	
Auf-druck	Name	Farbe	Blinkzustand	Bedeutung
M	Error	rot	aus	kein Fehler
			an	CAN Overrun Error Die Sample-Rate ist so hoch eingestellt, dass die Firmware nicht in der Lage ist, alle Daten auf dem CAN-Bus zu senden.
			2 Blitze	Internal Software Error z.B.: - gespeicherte Daten hatten eine ungültige Checksumme und Default-Werte wurden geladen - interner Watchdog hat ausgelöst - Blinkzustand bleibt, bis das Modul einen Reset durchführt oder ein Fehler an den Ausgängen auftritt.
			Blinken	Sendedaten fehlerhaft

Tabelle 6: Anzeigefunktion der Error-LED

3.2.6 Bedeutung der Power-LED

LED-Kennung			Anzeigefunktion	
Auf-druck	Name	Farbe	Zustand	Bedeutung
V	Power	grün	aus	keine Versorgungsspannung oder Modul ist im Bootloader-Modus (ID = 0), dieser Zustand wird durch die CANopen-Status-LED angezeigt (3 Blitze)
			an	Versorgungsspannung liegt an

Tabelle 7: Anzeigefunktion der Power-LED

3.2.7 Besondere Blinkzustände

Für die folgenden Blinkzustände werden zwei Status-LEDs gemeinsam ausgewertet:

LED-Anzeige	Bedeutung
- rote CAN-Error-LED: an und - CANopen-Status-LED: 3 Blitze	Ungültige Node-ID: Die Kodierschalter für die Node-ID stehen auf einer ungültigen ID, das Modul wird angehalten.

Tabelle 8: Besondere Blinkzustände

3.2.8 Zuordnung der LED-Bezeichnung zur Bezeichnung im Schaltplan

Bezeichnung auf CAN-CBX-COM2	Bezeichnung im Schaltplan*[1]
E	LED240A
S	LED240B
M	LED240C
V	LED240D
A	LED320A
R	LED320B
B	LED320C
G	LED320D

*[1] Der Schaltplan ist nicht Bestandteil dieses Handbuchs.

3.3 Kodierschalter

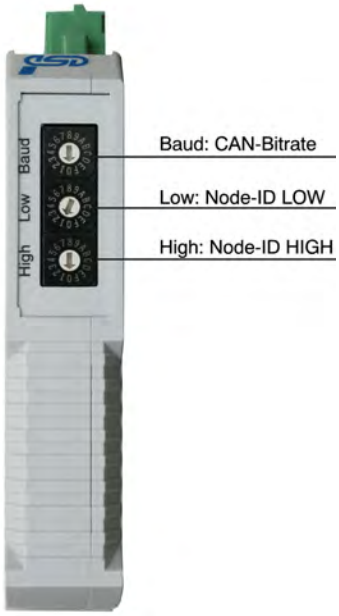


Abb. 4: Position der Kodierschalter

Achtung!
Die Auswertung der Kodierschalter durch die Firmware erfolgt beim Einschalten des Moduls. Änderungen der Einstellungen müssen daher **vor dem Einschalten** durchgeführt werden, da Änderungen während des Betriebes keine Auswirkungen haben.
Auch nach einem CANopen NMT-Reset werden die Einstellungen erneut eingelesen.

3.3.1 Einstellung der Node-ID über die Kodierschalter Low und High

Der Adress-Bereich des CAN-CBX-Moduls ist *dezimal* von 1 bis 127 bzw. *hexadezimal* von 01_h bis $7F_h$ einstellbar.
Der Kodierschalter **HIGH** dient zur Einstellung der 3 höherwertigen Bits (höherwertiges Nibble), der Kodierschalter **LOW** zur Einstellung der 4 niederwertigen Bits.

Hinweis:
Die folgenden Einstellungen sollten vermieden werden:
- Einstellungen der Kodierschalter höher als $7F_h$ führen zu Fehlermeldungen, die rote CAN-Error-LED leuchtet dauerhaft.
- Bei Einstellung der Kodierschalter auf 00_h befindet sich das Modul im Bootlader-Modus.

3.3.2 Einstellung der CAN-Bitrate über den Kodierschalter Baud

Die CAN-Bitrate wird mit dem Kodierschalter **Baud** eingestellt.

Über den Kodierschalter können Werte von 0_h bis F_h eingestellt werden. Die Einstellung der Bitrate erfolgt entsprechend der folgenden Tabelle:

Kodierschalter-Stellung	Bitrate [kBit/s]
0	1000
1	$666,\overline{6}$
2	500
3	$333,\overline{3}$
4	250
5	166
6	125
7	100
8	$66,\overline{6}$
9	50
A_h	$33,\overline{3}$
B_h	20
C_h	12,5
D_h	10
E_h	800
F_h	$83,\overline{3}$ *2

*2 ab FW 1.02

Tabelle 9: Index der Bitrate

3.3.3 Zuordnung der Kodierschalterbezeichnung zur Bezeichnung im Schaltplan

Bezeichnung auf CAN-CBX-COM2	Bezeichnung im Schaltplan*3
Baud	SW401
Low	SW400
High	SW402

*3 Der Schaltplan ist nicht Bestandteil dieses Handbuchs.

4. Technische Daten

4.1 Allgemeine technische Daten

Versorgungsspannung	Nennspannung: 24 V/DC Eingangsspannungsbereich: 24 V ±20% Stromaufnahme (24 V, 20 ˚C): 30 mA
Steckverbinder	24V (4-pol. Leiterplattensteckverbinder) - 24V-Spannungsversorgung COM A (7-pol. Leiterplattensteckverbinder, X300) - Serielle Schnittstelle 1 COM B (7-pol. Leiterplattensteckverbinder, X310) - Serielle Schnittstelle 2 CAN (5-pol. Leiterplattensteckverbinder, X400) - CAN-Schnittstelle InRailBus (5-pol. TBUS-Verbinder, Phoenix Contact, X101) - CAN-Schnittstelle und Spannungsversorgung über InRailBus Nur für werkseitige Test und Programmierzwecke (intern): X220 (10-pol. Buchsenleiste)
Temperaturbereich	0 ˚C ... +60 ˚C Umgebungstemperatur
IP-Schutzklasse	IP20
Verschmutzungsgrad	maximal zulässig nach DIN EN 61131-2: Verschmutzungsgrad 2
Gehäuse	Kunststoffgehäuse für Tragschienenmontage NS35/7,5 DIN EN 60715
Luftfeuchtigkeit	max. 90%, nicht kondensierend
Abmessungen	Breite: 22,5 mm, Höhe: 99 mm, Tiefe: 114.5 mm (ohne Stecker)
Gewicht	ca. 135 g

Tabelle 10: Allgemeine Daten des Moduls

4.2 CPU

CPU	32-Bit µC LPC2119FBD64
RAM	16 kByte integriert
Flash	128 kByte integriert
EEPROM	4 kByte

Tabelle 11: Microcontroller

4.3 CAN-Schnittstelle

Anzahl	1
Anschluss	5-pol. Leiterplattensteckverbinder oder über TBUS-Verbinder (InRailBus)
CAN-Controller	im LPC2119 integriert, ISO 11898-1
Galvanische Trennung des CAN-Interfaces gegenüber den anderen Baugruppen	Isolationsspannung U = 500 V (= Steh-Stoßspannung gemäß DIN EN 60664-1)
Physical CAN Layer	Physical Layer gemäß ISO 11898-2, Übertragungsrate programmierbar von 10 kBit/s bis 1 MBit/s
Busabschluss	Abschlusswiderstand ist bei Bedarf extern anzuschließen.

Tabelle 12: Daten der CAN-Schnittstelle

4.4 Serielle Schnittstellen

Anzahl der seriellen Schnittstellen	zwei unabhängig konfigurierbare serielle Schnittstellen
Galvanische Trennung	keine
Interface	Standard: - CAN-CBX-COM2 2x RS232 (Hardware-Handshake)(esd-Bestell-Nr: C.3055.02) - CAN-CBX-COM2 1x RS232 1x RS485 (esd-Bestell-Nr: C.3055.03) Optionen: RS422, RS485, TTY-aktiv/passiv
Bitraten	RS232: bis zu 115,2 kBit/s RS422: bis zu 115,2 kBit/s RS485: bis zu 115,2 kBit/s TTY: bis zu 38,4 kBit/s
Anschluss	7-poliger Leiterplattensteckverbinder mit Federkraftanschluss-Kontakten

Tabelle 13: Daten der seriellen Schnittstellen

4.5 Software-Unterstützung

Die Firmware des Moduls unterstützt die CiA® CANopen Spezifikation CiA 301 [1].

Die EDS-Datei des CAN-CBX-COM2 kann von der esd-Webseite unter www.esd.eu heruntergeladen werden.

4.6 Einbau des Moduls bei Verwendung des InRailBus-Verbinders

Sollen der CAN-Bus und die Versorgungsspannung über den InRailBus angeschlossen werden, gehen Sie bitte wie folgt vor:

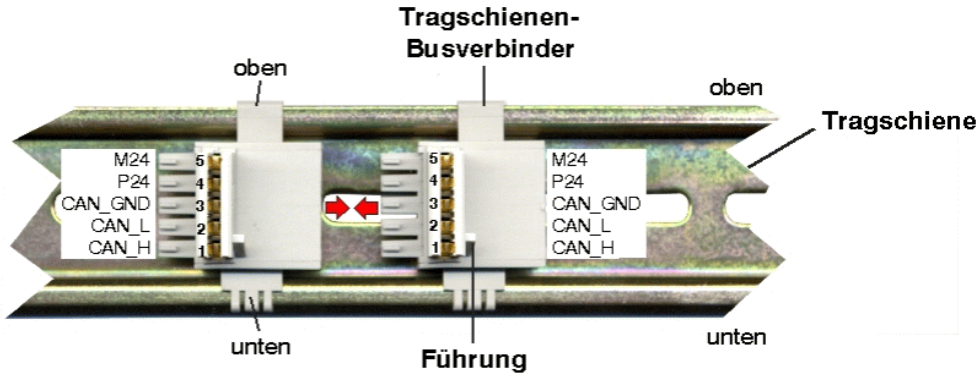


Abb. 5: Tragschiene mit Busverbinder

1. Der Tragschienen-Busverbinder des InRailBus wird an die Tragschiene angelegt und durch leichtes Andrücken auf der Tragschiene aufgerastet. Durch Zusammenstecken der einzelnen Busverbinder werden die Kommunikations- und Leistungssignale untereinander kontaktiert. Die Busverbinder können beliebig vor oder nach dem Aufstecken des CAN-CBX-Moduls verbunden werden.

2. Halten Sie das Modul leicht schräg nach hinten gekippt und setzen Sie das CBX-Modul auf den Busverbinder, so dass der obere Teil der Tragschiene dabei in die Einkerbung greift.

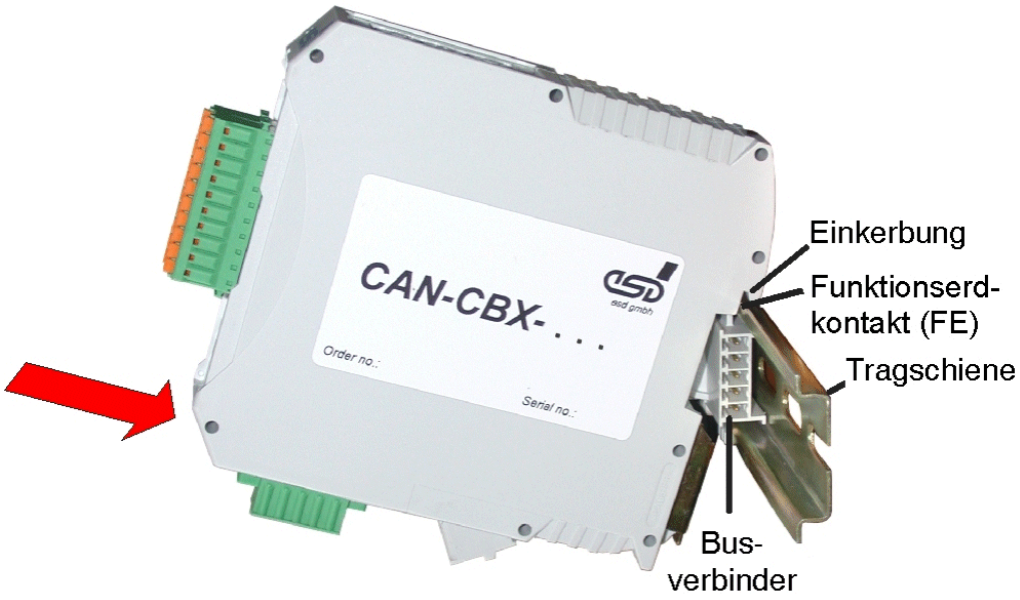


Abb. 6 : Einsetzen des CAN-CBX-Moduls

3. Schwenken Sie nun das CBX-Modul auf die Tragschiene auf, indem sie das Modul entsprechend der Pfeilrichtung in Abbildung 6 nach unten an die Tragschiene heran drücken. Dabei wird das Gehäuse durch die Führungsschiene des Busverbinders mechanisch geführt.

4. Beim Aufschwenken des CAN-CBX-Moduls rastet der untere Fußriegel auf dem unteren Teil der Tragschiene ein. Das Modul sitzt nun fest auf der Tragschiene und ist über den Busverbinder mit dem InRailBus verbunden.
   Verbinden Sie ggf. noch die Busverbinder untereinander und schließen Sie die +24 V Versorgungsspannung und das CAN-Interface an den InRailBus an.

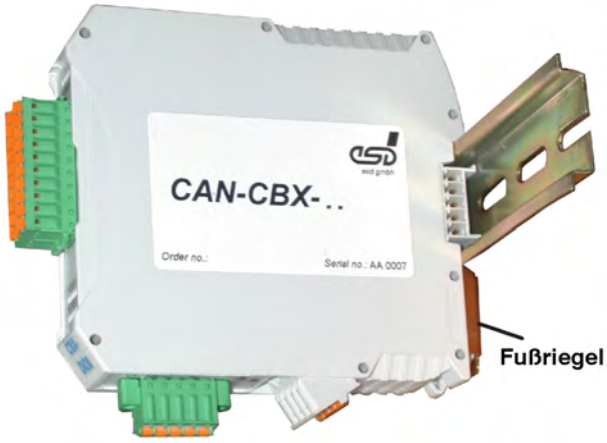


Abb. 7: Eingebautes CAN-CBX-Modul

4.6.1 Anschluss über den CBX-InRailBus

Um die Versorgungsspannung und die CAN-Signale über den InRailBus anschließen zu können, benötigen sie einen Anschlussstecker. Dieser Stecker ist nicht im Lieferumfang des CAN-CBX-Moduls enthalten. Er muss gesondert bestellt werden (Bestell-Nr. C.3000.02, siehe auch Bestellhinweise).

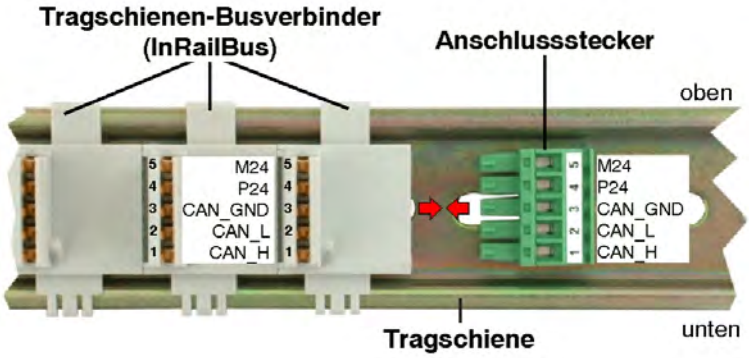


Abb. 8: Tragschiene mit InRailBus-Verbindern und Anschlussstecker

Stecken Sie den Anschlussstecker, wie in Abb. 8 beschrieben, von rechts in die Buchsenseite des äußeren Tragschienen-Busverbinders des InRailBus. Schließen sie nun das CAN-Interface und die Versorgungsspannung über den Anschlussstecker an.

4.6.2 Anschluss der Versorgungsspannung

Der Anschluss der Versorgungsspannung erfolgt über den +24V-Stecker oder über den InRailBus-Stecker.

Achtung!
Bitte beachten Sie die Sicherheitshinweise und Anforderungen bezüglich der Versorgungsstromkreise (siehe Seite 4)!

Achtung!
Die Anschlüsse der 24 V Spannungsversorgung sind intern verbunden und dürfen **nicht** von zwei unabhängigen Stromquellen gleichzeitig versorgt werden!

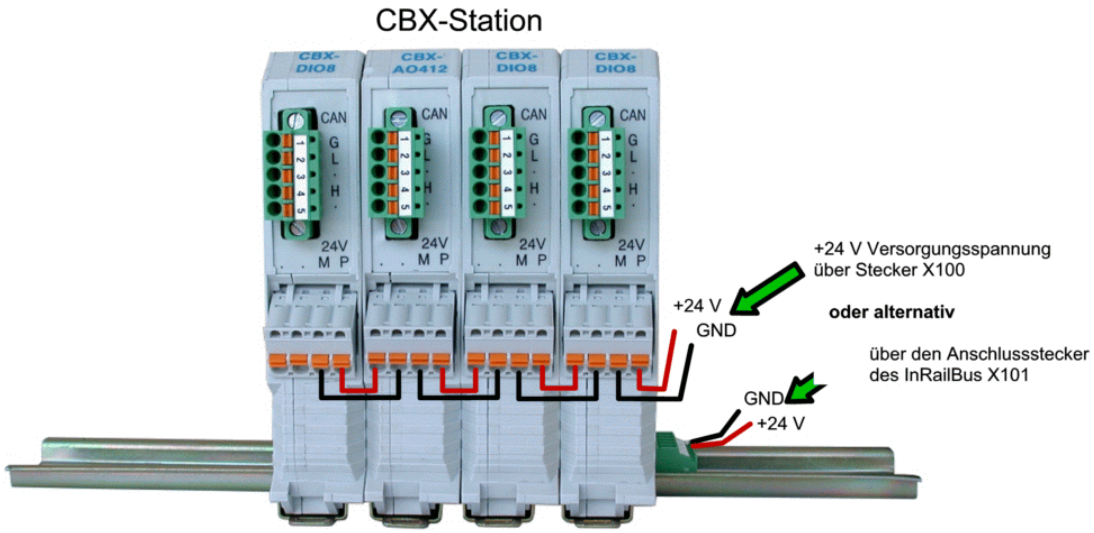


Abb. 9: CAN-CBX-Station mit Anschluss der Versorgungsspannung

Erdung der Tragschiene

Hinweis:
Das Modul ist über seinen Funktionserde-Kontakt mit der Tragschiene verbunden. Dies verbessert die Stabilität gegen elektromagnetische Störungen.
Die Tragschiene ist daher mit einer geeigneten Funktionserde in der Umgebung bzw. in der Anlage zu verbinden.
Dabei ist zu beachten, dass die Impedanz der Verbindung gering gehalten wird.
Der Funktionserde-Kontakt des Moduls dient nicht der elektrischen Sicherheit.

4.6.3 Anschluss von CAN

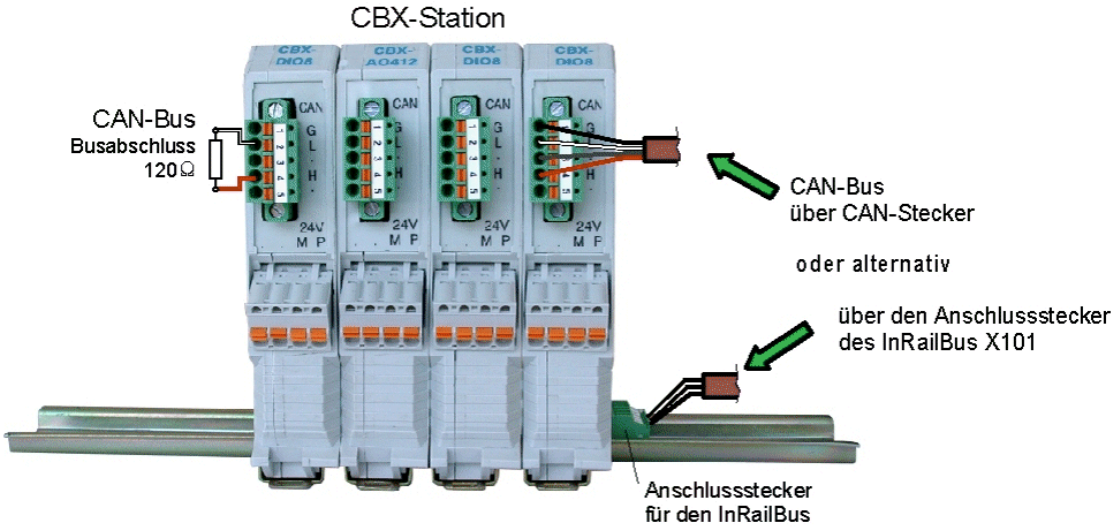


Abb. 10: Anschluss von CAN an die CBX-Station

Generell besteht die Möglichkeit die CAN-Signale über den InRailBus oder über den CAN-Stecker eines äußeren CAN-CBX-Moduls der CBX-Station einzuspeisen. Die Signale werden dann über den InRailBus an die folgenden CAN-CBX-Module der CAN-CBX-Station weitergeleitet. Die CAN-Signale dürfen über den CAN-Stecker des CAN-CBX-Moduls, das am anderen Ende der CBX-Station montiert ist, weitergeführt werden. Über die CAN-Stecker der mittleren CAN-CBX-Module der CBX-Station dürfen die CAN-Signale jedoch nicht weitergeführt werden, da dies zu unzulässigen Verzweigungen führt.

Bitte beachten Sie, dass an das CAN-CBX-Modul, das sich am Ende des InRailBus befindet ein Bus-Abschlusswiderstand angeschlossen werden muss, wenn der CAN-Bus dort endet (siehe Abb. 10, 11).

4.7 Ausbau des CAN-CBX-Moduls vom InRailBus

Ist das CAN-CBX-Modul über den InRailBus verbunden, gehen Sie beim Ausbau wie folgt vor:

Lösen Sie das Modul von der Tragschiene indem Sie den Fußriegel (siehe Abb. 7) nach unten ziehen (z.B. mit einem Schraubendreher). Dabei löst sich das Modul unten von der Tragschiene und kann abgezogen werden.

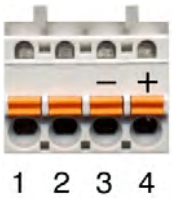
Hinweis:
Es ist möglich, einzelne Gehäuse aus dem Verbund zu lösen, ohne die InRailBus-Verbindung zu unterbrechen, da beim Ziehen einzelner Module aus dem Verbund die Signalkette nicht unterbrochen wird.

5. Schnittstellen und Steckerbelegungen

5.1 Spannungsversorgung 24V (X100)

Gerätebuchse: Phoenix Contact MSTBO 2,5/4-G1L-KMGY
Leitungsstecker: Phoenix Contact FKCT 2,5/4-ST KMGY, 5.0 mm Raster,
Federkraftanschluss-Kontakte,
Phoenix Contact Bestell-Nr.: 19 21 90 0 (im Lieferumfang enthalten)
Zu Leiteranschluss und Leiterquerschnitt siehe Seite 36.

Pin-Zuordnung:



Pin-Belegung:

Gehäusebezeichnung	24V	
	M	P
Stecker-Aufdruck	-	+

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

Siehe auch Anschlussplan Seite 12.

Hinweis:
Die Pins 1 und 4 sind intern miteinander verbunden.
Die Pins 2 und 3 sind intern miteinander verbunden.

Signalbeschreibung:

P24... Versorgungsspannung +24 V
M24... Bezugspotenzial

5.2 CAN

5.2.1 CAN-Interface-Schaltung

Das CAN Physical Layer ist gemäß ISO 11898-2 ausgelegt. Die differentiellen CAN-Bus-Signale sind von den anderen Potenzialen über einen magnetischen Datenkoppler und DC/DC-Wandler galvanisch getrennt.

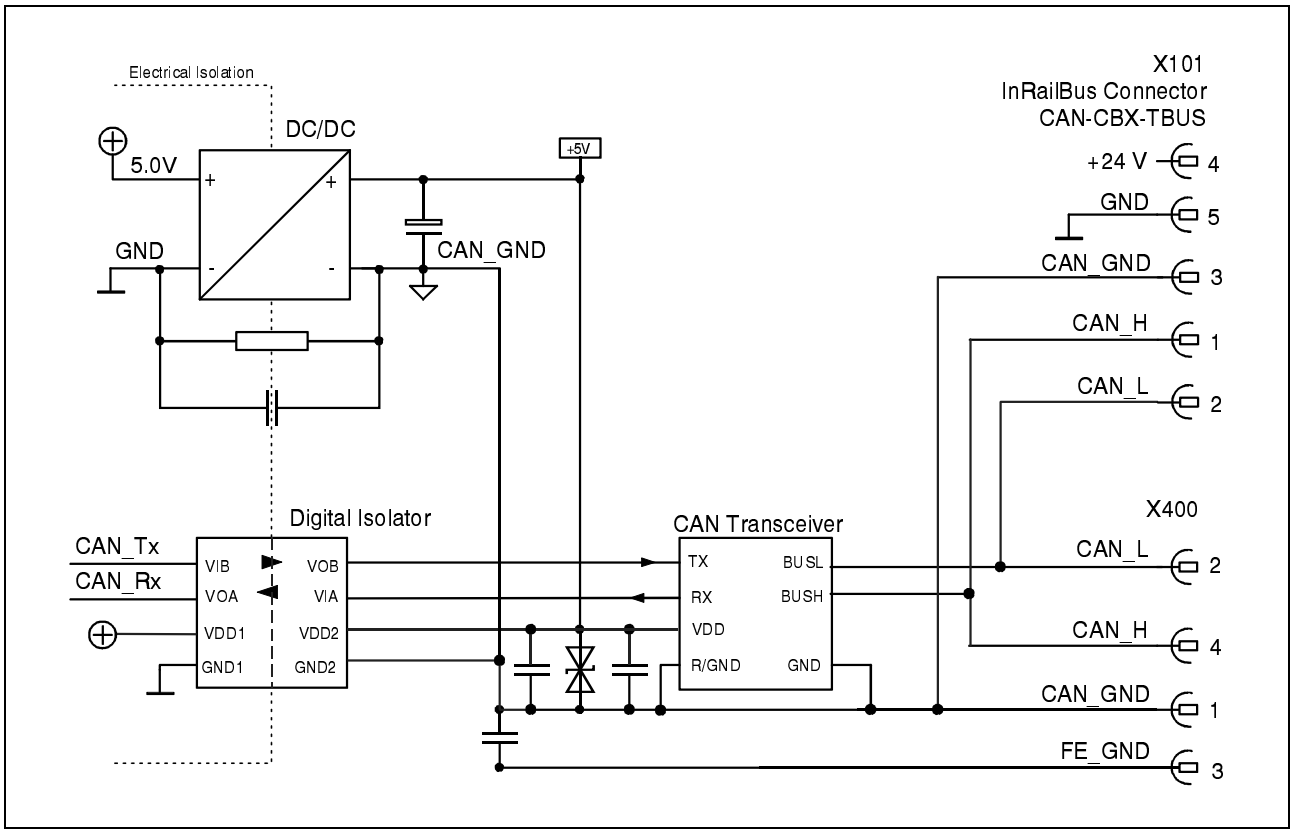


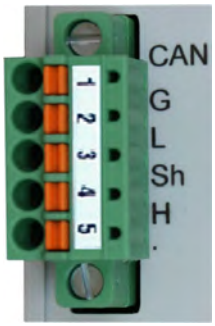
Abb. 11: Schaltung des CAN-Interface

Die CAN Schnittstelle kann entweder über den CAN-Stecker oder optional über den InRailBus angeschlossen werden. Benutzen Sie für den Anschluss über den InRailBus den Tragschienenbusverbinder des CBX-InRailBus (CAN-CBX-TBUS), siehe Bestellhinweise (Seite 124).

5.2.2 CAN-Stecker (X400)

Gerätebuchse: Phoenix Contact MC 1,5/5-GF-3,81
Leitungsstecker: Phoenix Contact FK-MCP 1,5/5-STF-3,81, Federkraftanschluss-Kontakte
Phoenix Contact Bestell-Nr.: 1851261 (im Lieferumfang enthalten)
Zu Leiteranschluss und Leiterquerschnitt siehe Seite 36.

Pin-Zuordnung:
(Ausschnitt Leitungsstecker mit Gehäuseaufdruck)



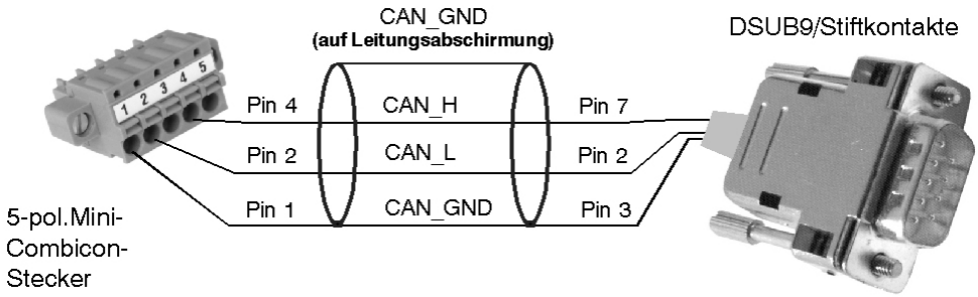
Pin-Belegung:

Gehäuse-aufdruck	Signal	Pin
G	CAN_GND	1
L	CAN_L	2
Sh	Shield	3
H	CAN_H	4
•	-	5

Signalbeschreibung:

CAN_L, CAN_H ... CAN-Signale
CAN_GND ... Bezugspotenzial des lokalen CAN-Physical Layers
Shield ... Anschluss für Leitungsabschirmung (bei Hutschienenmontage direkter Kontakt zu Hutschienenpotenzial)
- ... nicht angeschlossen

Empfehlung eines Adapterkabels 5-pol-Phoenix-Contact-Stecker (hier Leitungsstecker FK-MCP1,5/5-STF-3,81 mit Federkraftanschluss-Kontakten) auf 9-pol-DSUB:

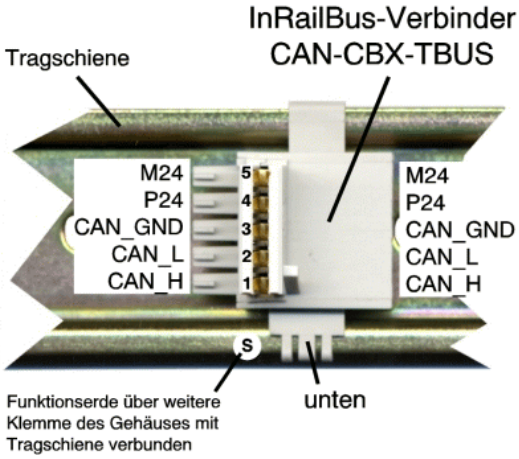


Die Belegung des 9-poligen DSUB-Steckers erfolgt nach CiA DS 102.

5.2.3 CAN und Versorgungsspannung über InRailBus X101

Steckertyp: Tragschienensteckverbinder CAN-CBX-TBUS
(Phoenix Contact ME 22,5 TBUS 1,5/5-ST-3,81 KMGY)

Steckeraufsicht:



Pin-Belegung:

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

S	FE (PE_GND)

Signalbeschreibung:

CAN_L,
CAN_H ... CAN-Signale
CAN_GND ... Bezugspotenzial des lokalen CAN-Physical Layers
P24... Versorgungsspannung +24 V
M24... Bezugspotenzial
FE... direkt am Gehäuse des Moduls: Funktionserde (mit Hutschienenpotenzial verbunden)

5.3 Serielle Schnittstellen COM A und COM B

Das CAN-CBX-COM2-Modul ist mit zwei seriellen Schnittstellen (COM A und COM B) ausgestattet. In der Grundversion CAN-CBX-COM2 2xRS232 (esd Bestell-Nr.: C.3055.02) sind zwei RS232-Treiber bestückt.
Die serielle Schnittstelle COM A ist immer als RS232-Schnittstelle ausgelegt.
Die serielle Schnittstelle COM B läßt sich über Add-On-Platinen (Piggybacks) als RS232-, RS422-, RS485- oder TTY-Interface konfigurieren.
Im Folgenden wird die Verdrahtung der seriellen Schnittstellen in Bezug auf die Datenrichtung dargestellt. Die Abbildungen sollen die im Kapitel *Steckerbelegung* verwendeten Kurzbezeichnungen der Signale erläutern.

5.3.1 RS232-Schnittstelle

Als Beispiel für das Anschlusskabel ist hier ein Adapterkabel vom 7-poligen Leiterplattensteckverbinder auf DSUB9/Buchse gewählt worden. Dieses Kabel ist für den direkten Anschluss der CAN-CBX-COM2 an einen PC ohne Null-Modem vorgesehen.
Für individuelle Steckerbelegungen empfehlen wir Adapterkabel mit frei konfigurierbaren DSUB-Steckern und Buchsen. Selbstverständlich können Sie sich Ihre Kabel auch von uns konfigurieren lassen.

Achtung !
Bitte beachten Sie, dass die Funktion der Steuersignale von den eingesetzten Software-Treibern der seriellen Schnittstellen abhängt.

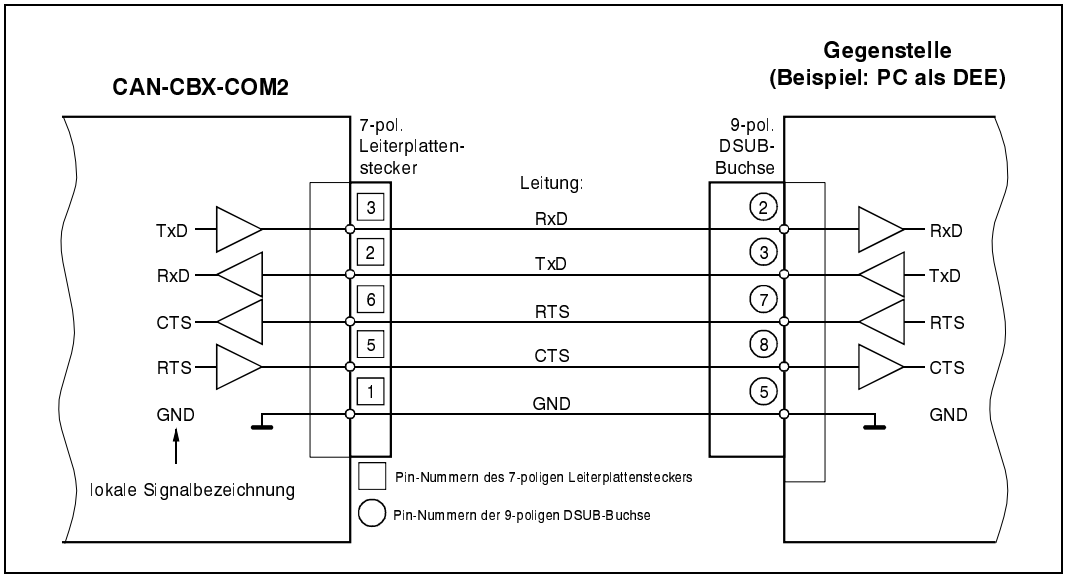


Abb. 12: Anschlussschema für RS232-Betrieb (Beispiel: CAN-CBX-COM2 <-> PC)

5.3.2 Option: RS422-Schnittstelle

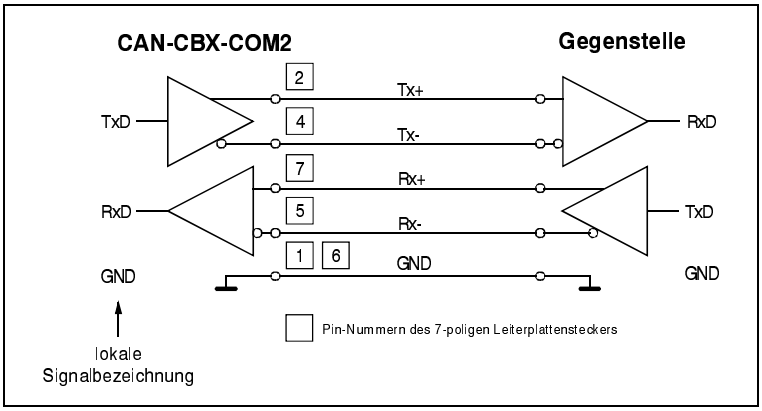


Abb. 13: Anschlussschema für RS422-Betrieb, optional

5.3.3 Option: RS485-Schnittstelle

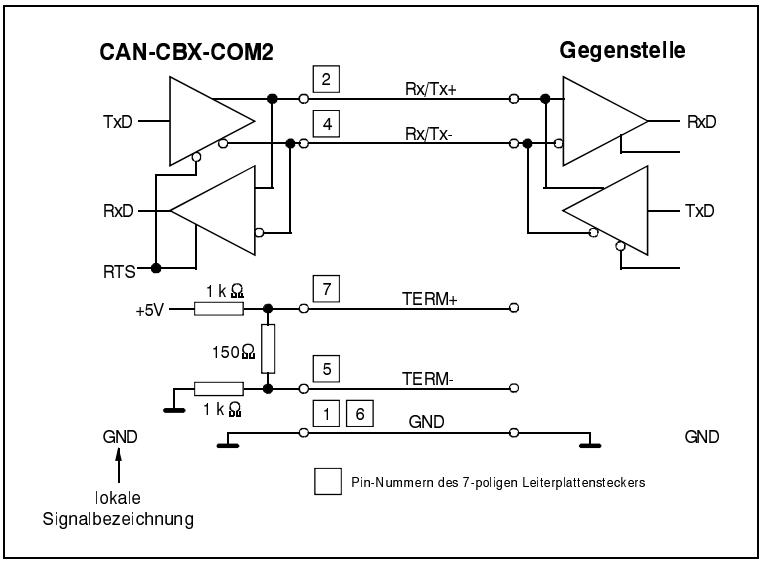


Abb. 14: Anschlussschema für RS485-Betrieb, optional

Die Pins 5 und 7 des Leiterplattensteckers führen beim RS485-Betrieb auf ein Abschlusswiderstandsnetzwerk, das sich auf dem Piggyback befindet. Um den Abschluss zu aktivieren, ist das Signal Rx/Tx+ mit TERM+ und das Signal Rx/Tx- mit TERM- zu verbinden.

5.3.4 Option: TTY(20 mA)-Schnittstelle

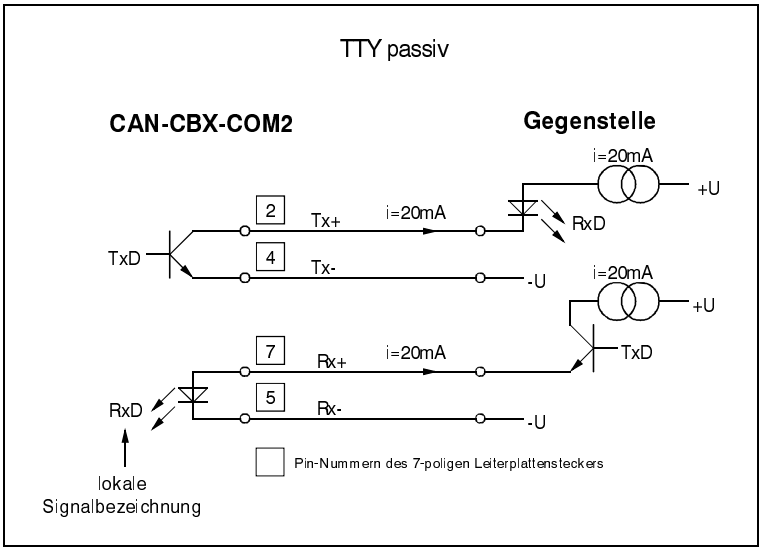


Abb. 15: Anschlussschema für TTY-Betrieb (passiv), optional

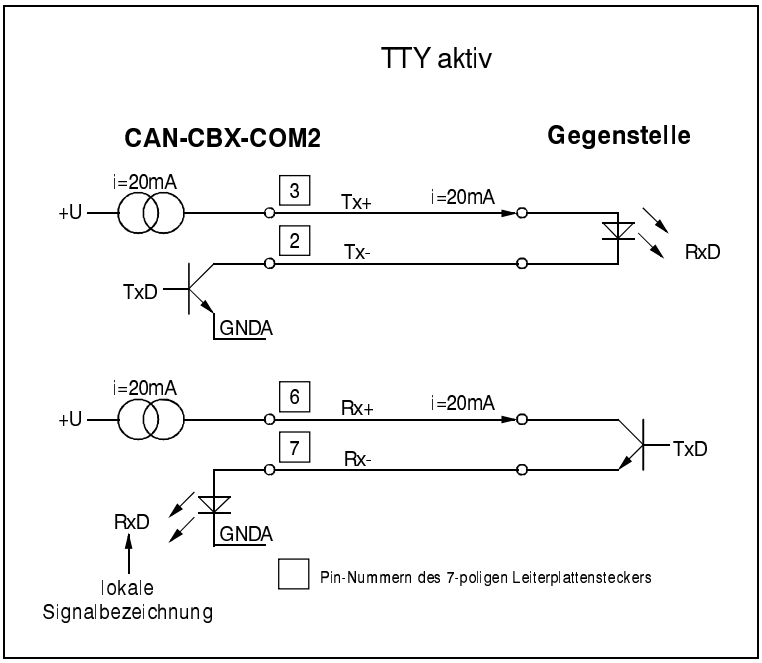
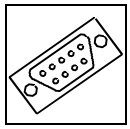


Abb. 16: Anschlussschema für TTY-Betrieb (aktiv), optional



5.3.5 Steckerbelegung

Gerätebuchse: 2x Phoenix Contact MC 1,5/7-G-3,5
Leitungsstecker: 2x Phoenix Contact FK-MCP 1,5/7-3,5 (Federkraftanschluss-Kontakte)
Phoenix Contact Bestell-Nr.: 1939960 (im Lieferumfang enthalten)
Zu Leiteranschluss und Leiterquerschnitt siehe Seite 36.

Pin-Zuordnung:

Serielle Schnittstelle **COM A**

Serielle Schnittstelle **COM B**

Die serielle Schnittstelle COM A ist als RS232 Interface ausgelegt.

Die serielle Schnittstelle COM B läßt sich über Add-On-Platinen (Piggybacks) als RS232-, RS422-, RS485- oder TTY-Interface konfigurieren.

In der Grundversion sind zwei RS232-Treiber bestückt.
CAN_CBX-COM2 2x RS232 (esd-Bestell-Nr.: C.3055.02)

Weitere Bestückungsoptionen auf Anfrage.

Pin-Belegung der Grundversion CAN-CBX-COM2 2x RS232 (esd Bestell Nr.: C.3055.02):

In dieser Version sind beide seriellen Schnittstellen als RS232-Interface ausgelegt.

Stecker-Pin	Signalbelegung COM A/COM B RS-232
1	GND
2	RxD
3	TxD
4	-
5	RTS
6	CTS
7	-

Pin-Zuordnung der Version CAN-CBX-COM2 1xRS232 1x RS485 (esd-Bestell-Nr.: C.3055.03):

In dieser Version sind die seriellen Schnittstellen COM A als RS232 Interface und COM B als RS485 Interface konfiguriert.

Connector Pin	Signalbelegung	
	COM A RS232	COM B RS485
1	GND	GND
2	RxD	Tx/Rx+
3	TxD	-
4	-	Tx/Rx-
5	RTS	Term-
6	CTS	GND
7	-	Term+

Option:
Auf Nachfrage kann die serielle Schnittstelle COM B optional über Piggybacks als RS422-, TTYp- oder TTYa-Interface konfiguriert werden.

Stecker-Pin	Optionale Signalbelegung COM B		
	RS422	TTYp	TTYa
1	GND	GND	GND
2	Tx+	Tx+	Tx-
3	-	I1	Tx+
4	Tx-	Tx-	-12 V
5	Rx-	Rx-	-12 V
6	GND	I2	Rx+
7	Rx+	Rx+	Rx-

5.4 Zuordnung der Stecker-Bezeichnung zur Bezeichnung im Schaltplan

Bezeichnung auf CAN-CBX-COM2	Bezeichnung im Schaltplan*[4]
COM A	X300
COM B	X310
CAN	X400
24V	X100
InRailBus	X101

*[4] Der Schaltplan ist nicht Bestandteil dieses Handbuchs.

5.5 Leiteranschluss/Leiterquerschnitt

Die folgende Tabelle enthält einen Auszug aus den technischen Daten der verwendeten Stecker.

Schnittstelle	**Spannungs-versorgung 24 V** [4]	**CAN-Stecker** [5]	**Serielle Schnittstellen** [6]
Steckertyp Leitungsstecker (Artikelfamilie)	**FKCT 2,5/..-ST KMGY**	**FK-MCP 1,5/..-STF-3,81**	**FK-MCP 1,5/..-ST-3,5**
Anschlussart	Zugfeder-anschluss	Federkraft-anschluss	Federkraft-anschluss
Abisolierlänge	10 mm	9 mm	9 mm
Leiterquerschnitt starr min	0,2 mm²	0,14 mm²	0,14 mm²
Leiterquerschnitt starr max	2,5 mm²	1,5 mm²	1,5 mm²
Leiterquerschnitt flexibel min	0,2 mm²	0,14 mm²	0,14 mm²
Leiterquerschnitt flexibel max	2,5 mm²	1,5 mm²	1,5 mm²
Leiterquerschnitt flexibel mit Aderendhülse ohne Kunststoffhülse min	0,25 mm²	0,25 mm²	0,25 mm²
Leiterquerschnitt flexibel mit Aderendhülse ohne Kunststoffhülse max	2,5 mm²	1,5 mm²	1,5 mm²
Leiterquerschnitt flexibel mit Aderendhülse mit Kunststoffhülse min	0,25 mm²	0,25 mm²	0,25 mm²
Leiterquerschnitt flexibel mit Aderendhülse mit Kunststoffhülse max	2,5 mm²	0,5 mm²	0,5 mm²
Leiterquerschnitt AWG/kcmil min	24	26	26
Leiterquerschnitt AWG/kcmil max	12	16	16
2 Leiter gleichen Querschnitts starr min	unzulässig	unzulässig	unzulässig
2 Leiter gleichen Querschnitts starr max	unzulässig	unzulässig	unzulässig
2 Leiter gleichen Querschnitts flexibel min	unzulässig	unzulässig	unzulässig
2 Leiter gleichen Querschnitts flexibel max	unzulässig	unzulässig	unzulässig
2 Leiter gleichen Querschnitts flexibel mit Aderendhülse ohne Kunststoffhülse min	unzulässig	unzulässig	unzulässig
2 Leiter gleichen Querschnitts flexibel mit Aderendhülse ohne Kunststoffhülse max	unzulässig	unzulässig	unzulässig
2 Leiter gleichen Querschnitts flexibel mit Zwillingsadererndhülsen (TWIN-AEH) mit Kunststoffhülse min	0,5 mm²	unzulässig	unzulässig
2 Leiter gleichen Querschnitts flexibel mit Zwillingsaderendhülsen (TWIN-AEH) mit Kunststoffhülse max	1,0 mm²	unzulässig	unzulässig
AWG nach UL/CUL min	26	28	28
AWG nach UL/CUL max	12	16	16

6. Korrekte Verdrahtung galvanisch getrennter CAN-Netze

Generell sind bei der CAN-Verdrahtung sämtliche gültigen Richtlinien (DIN, VDE) bzgl. EMV-gerechtem Aufbau, Leitungsführung, Leiterquerschnitt, zu verwendende Materialien und Mindest-abstand zu beachten.

6.1 Standards zur CAN-Verdrahtung

Die Flexibilität beim Entwurf von CAN-Netzwerken ist eine wesentliche Stärke der verschiedenen, auf dem CAN-Standard ISO11898-2 aufbauenden Erweiterungen wie z.B. CANopen, ARINC825, DeviceNet und NMEA2000. Die Nutzung dieser Flexibilität erfordert jedoch zwingend ein Netzwerk-Design, das die Wechselwirkungen aller Netzwerkparameter berücksichtigt.

Zum Teil haben die CAN-Organisationen in ihren Spezifikationen den Einsatzbereich von CAN angepasst, um Anwendungen außerhalb des ISO 11898-Standards zu ermöglichen. Sie haben Einschränkungen auf Systemebene bei der Datenrate, der Leitungslänge und bei parasitären Bus-Lasten vorgenommen. Beim Entwurf von CAN-Netzwerken muss jedoch immer ein Spielraum für Signalverluste über das Gesamtsystem und die Verdrahtung, parasitäre Lasten, Netzwerk-Unsymmetrien, Potenzialunterschiede gegen Erde und Signal-Integritäten eingeplant werden. **Daher kann die maximal erreichbare Anzahl an Knoten, Buslängen und Stichleitungslängen von der theoretisch möglichen Anzahl abweichen!**

esd hat sich bei seinen Empfehlungen zur CAN-Verdrahtung auf die Vorgaben der ISO 11898-2 beschränkt. Auf die Beschreibung der Besonderheiten der abgeleiteten Spezifikationen CANopen, ARINC825, DeviceNet und NMEA2000 wird an dieser Stelle verzichtet.

Die konsequente Einhaltung der ISO 118998-2-Vorgaben bietet wesentliche Vorteile:

- Zuverlässiger Betrieb durch bewährte Design-Vorgaben
- Minimieren von Fehlerquellen durch ausreichend Abstand zu den physikalischen Grenzwerten.
- Unproblematische Wartung, weil bei zukünftigen Anpassungen und bei Fehlersuche keine „Sonderfälle“ zu berücksichtigen sind.

Selbstverständlich lassen sich auch zuverlässige Netzwerke nach den Spezifikationen von CANopen, ARINC825, DeviceNet und NMEA2000 aufbauen. **Zu beachten ist jedoch, dass die Verdrahtungsvorgaben der verschiedenen Spezifikationen nicht bedenkenlos miteinander vermischt werden dürfen!**

6.2 Leicht störbehaftete Industrieumgebung (zweiadrig verdrillte Leitung)

6.2.1 Grundregeln

Hinweis:
esd garantiert die EU-Konformität des Produkts, wenn für die CAN-Verdrahtung mindestens Kabel mit einfach abgeschirmten **zweiadrig** verdrillten Leitungen verwendet werden, die die Anforderungen der Norm ISO 11898-2 erfüllen. Einfach abgeschirmte vieradrig verdrillte Leitungen, wie in Kapitel 6.3 beschrieben, stellen die EU-Konformität ebenfalls sicher.

Die folgenden Grundregeln für die CAN-Bus Verdrahtung mit einfach abgeschirmten zweiadrig verdrillten Leitungen sollten unbedingt beachtet werden:

1	Es ist ein geeigneter Leitungstyp (Wellenwiderstand ca. 120 Ω ±10%) mit ausreichendem Leiterquerschnitt (≥0.22 mm²) zu verwenden. Der Spannungsabfall auf der Leitung ist zu berücksichtigen.
2	Für den Einsatz in leicht störbehafteter Industrieumgebung ist mindestens ein zweiadriges CAN-Kabel zu verwenden, dessen Adern wie folgt zu belegen sind: ● Zwei verdrillte Adern sind mit den CAN-Signalleitungen (CAN_H, CAN_L) zu belegen. ● Die Kabel-Abschirmung ist mit dem Bezugspotenzial (CAN_GND) zu belegen.
3	Das Bezugspotenzial CAN_GND muss an genau **einem** Punkt mit Funktionserde (FE) verbunden sein.
4	Ein CAN-Netz darf sich nicht verzweigen (Ausnahme: kurze Stichleitungen) und muss an beiden Enden mit dem Wellenwiderstand der Leitung (in der Regel 120 Ω ±10%) abgeschlossen werden (zwischen den Signalen CAN_L und CAN_H und **nicht** an CAN_GND).
5	Stichleitungen sind so kurz wie möglich zu halten (l < 0,3 m).
6	Die Baudrate muss an die Leitungslänge angepasst werden.
7	Die CAN-Leitungen sollten nicht in unmittelbarer Nähe von Störquellen verlegt werden. Lässt sich dies nicht vermeiden, so sind doppelt abgeschirmte Leitungen vorzuziehen.

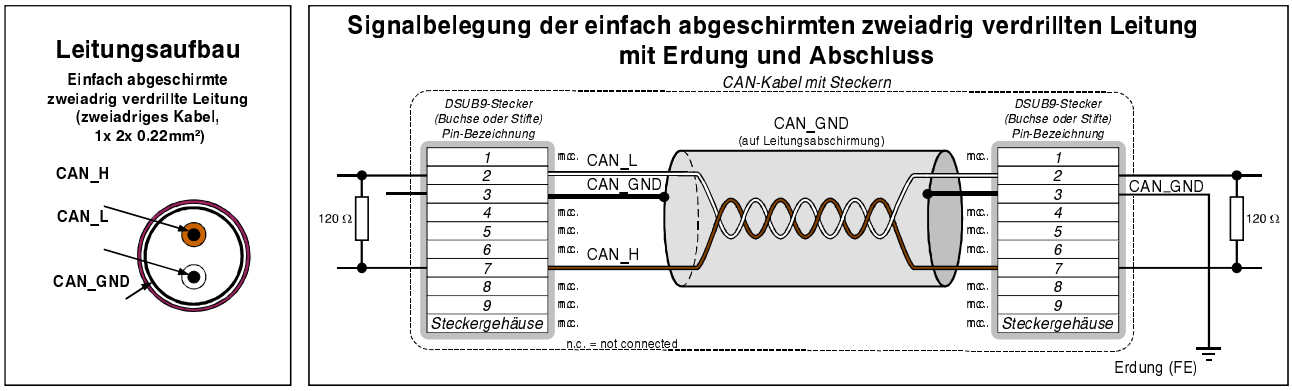


Abb. 17: CAN-Verdrahtung in leicht störbehafteter Industrieumgebung

6.2.2 Verkabelung

- Geräte, die pro CAN-Netz nur einen CAN-Stecker besitzen, sind über eine kurze Stichleitung (< 0,3 m) und ein T-Stück (als Zubehör lieferbar) anzuschließen. Werden sie am Ende des CAN-Netzes angeschlossen, kann auch der CAN-Abschlussstecker „CAN-Termination DSUB9“ verwendet werden.

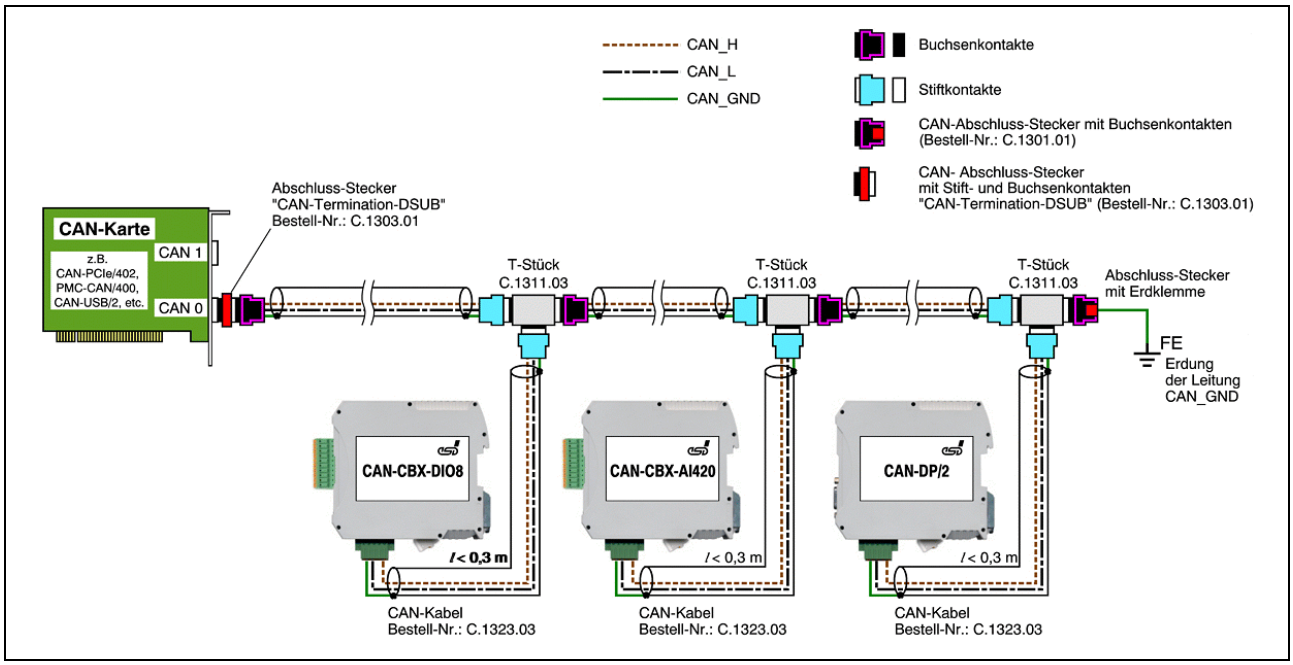


Abb. 18: Beispiel für korrekte Verdrahtung einfach abgeschirmter zweiadrig verdrillter Leitung

6.2.3 Abschlusswiderstand

- An beiden Enden des CAN-Bus ist ein Abschlusswiderstand anzuschließen.
  Ist auf dem CAN-Interface am Ende des CAN-Busses ein integrierter CAN-Busabschluss angeschlossen, so ist dieser an Stelle eines externen Abschlusswiderstands zu verwenden.

- 9-polige DSUB-Verbinder mit integriertem Abschlusswiderstand und Stift- und Buchsenkontakten (Gender Changer) sind unter der Artikelnummer C.1303.01 lieferbar.

- Für den Abschluss des CAN-Bus und Erdung des CAN_GND sind DSUB-Abschlussstecker mit Stiftkontakten (C.1302.01) oder Buchsenkontakten (C.1301.01) mit Erdungsklemme erhältlich.

6.3 Stark störbehaftete Industrieumgebung (vieradrig verdrillte Leitung)

6.3.1 Grundregeln

Die folgenden **Grundregeln** für die CAN-Bus Verdrahtung mit einfach abgeschirmten vieradrig verdrillten Leitungen sollten unbedingt beachtet werden:

1	Es ist ein geeigneter Leitungstyp (Wellenwiderstand ca. 120 Ω ±10%) mit ausreichendem Leiterquerschnitt (≥0.22 mm²) zu verwenden. Der Spannungsabfall auf der Leitung ist zu berücksichtigen!
2	Für den Einsatz in stark störbehafteter Industrieumgebung ist ein vieradriges CAN-Kabel zu verwenden, dessen Adern wie folgt zu belegen sind: ● Zwei verdrillte Adern sind mit den CAN-Signalleitungen (CAN_H, CAN_L) zu belegen. ● Die anderen beiden verdrillten Adern sind mit dem Bezugspotenzial (CAN_GND) zu belegen. ● Die Leitungsabschirmung ist an mindestens einem Punkt an Funktionserde (FE) anzuschließen!
3	Das Bezugspotenzial CAN_GND muss an genau **einem** Punkt mit Funktionserde (FE) verbunden sein.
4	Ein CAN-Netz darf sich nicht verzweigen (Ausnahme: kurze Stichleitungen) und muss an beiden Enden mit dem Wellenwiderstand der Leitung (in der Regel 120 Ω ±10%) abgeschlossen werden (zwischen den Signalen CAN_L und CAN_H und **nicht** an CAN_GND)!
5	Stichleitungen sind so kurz wie möglich zu halten (l < 0,3 m)!
6	Die Baudrate muss an die Leitungslänge angepasst werden.
7	Die CAN-Leitungen sollten nicht in unmittelbarer Nähe von Störquellen verlegt werden. Lässt sich dies nicht vermeiden, so sind doppelt abgeschirmte Leitungen vorzuziehen.

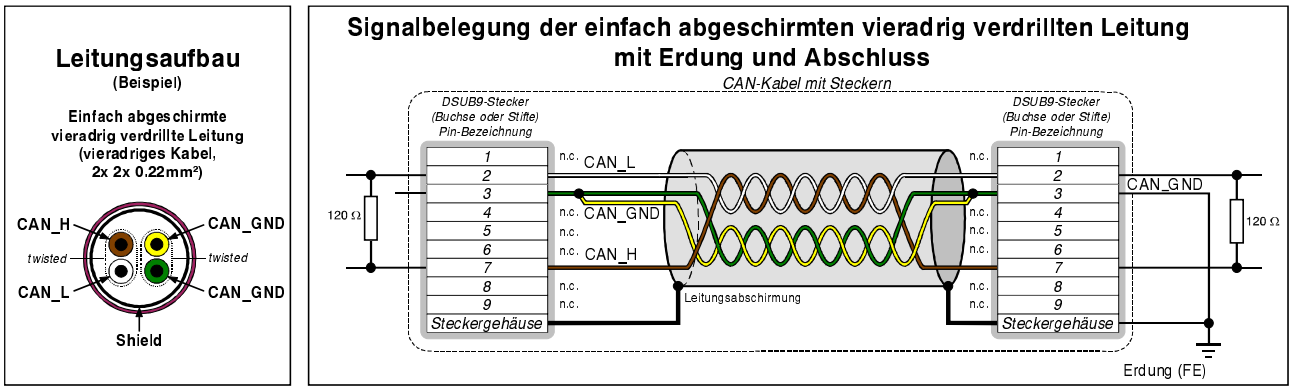


Abb. 19: CAN-Verdrahtung in stark störbehafteter Industrieumgebung

6.3.2 Verkabelung

Achtung:
Werden einfach abgeschirmte vieradrig verdrillte Leitungen verwendet, ist für den CAN-Bus Steckverbinder ein T-Verbindungsstecker zu verwenden, der den Anschluss zweier CAN-Kabel gestattet, und bei dem die Kabel-Abschirmung (Shield) durchgeführt wird, z.B. DSUB9-Stecker von ERNI (ERBIC CAN BUS MAX, Bestell-Nr.:154039).

Die Verwendung des esd T-Connectors (Bestell-Nr: C.1311.03) wird für einfach abgeschirmte vieradrig verdrillte Leitungen nicht empfohlen, da das Schirm-Potenzial des leitenden DSUB-Gehäuses nicht durch diesen T-Connector-Typ durchgeführt wird.

Ist eine gemischte Verwendung von zwei- und vieradrig verdrillten Leitungen nicht zu vermeiden, ist auf eine durchgehende CAN-GND-Verbindung zu achten!

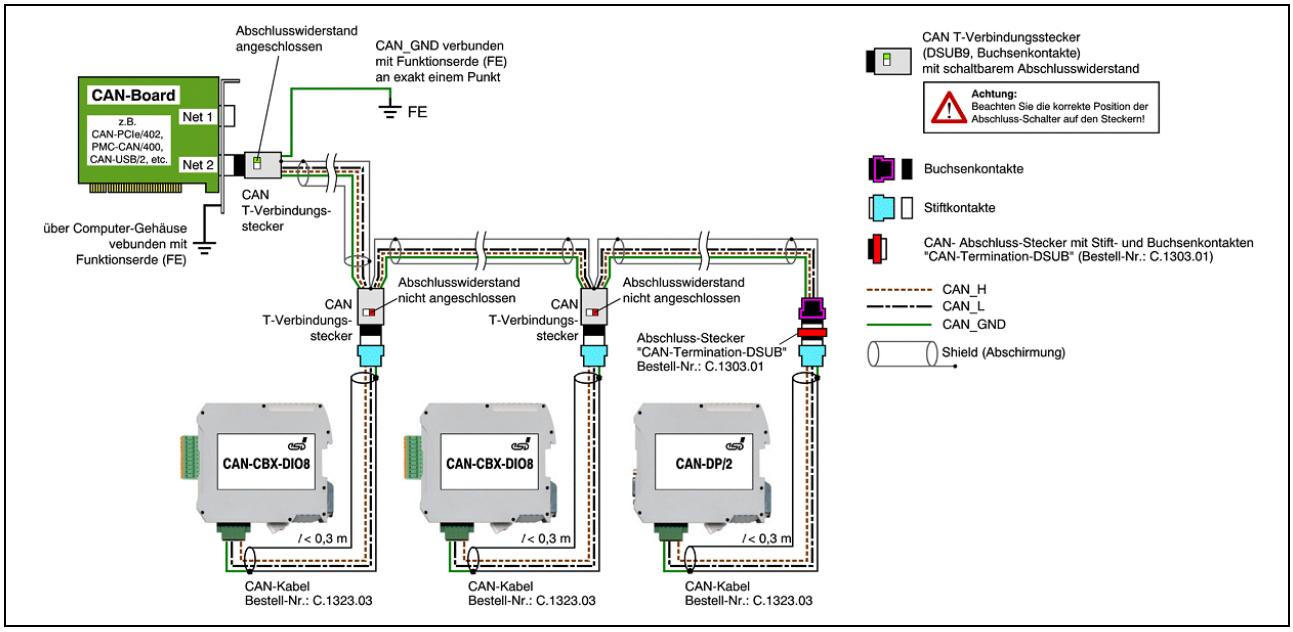


Abb. 20: Beispiel für korrekte Verdrahtung einfach abgeschirmter, vieradrig verdrillter Leitungen

6.3.3 Abschlusswiderstand

- An beiden Enden des CAN-Bus ist ein Abschlusswiderstand anzuschließen.
  Ist auf dem CAN-Interface am Ende des CAN-Busses ein integrierter CAN-Busabschluss angeschlossen, so ist dieser an Stelle eines externen Abschlusswiderstands zu verwenden.

- 9-polige DSUB-Verbinder mit integriertem Abschlusswiderstand und Stift- und Buchsenkontakten (Gender Changer) sind unter der Artikelnummer C.1303.01 lieferbar.

- 9-polige DSUB-Stecker mit integriertem, umschaltbarem Abschlusswiderstand können z.B. von ERNI (ERBIC CAN BUS MAX, Buchsenkontakte, Bestell-Nr.:154039) bezogen werden.

6.4 Erdung

- Bei CAN-Modulen mit galvanischer Trennung muss CAN_GND zwischen den CAN-Modulen verbunden werden.
- CAN_GND sollte an **exakt einem** Punkt im Netz mit dem Erdpotenzial (FE) verbunden sein.
- Jedes CAN-Modul mit galvanischer Verbindung zum Erdpotenzial wirkt wie eine Erdung. Aus diesem Grund sollte nicht mehr als ein CAN-Modul mit galvanischer Verbindung zum Erdpotenzial angeschlossen werden.
- Die Erdung kann z.B. an einem Abschlussstecker vorgenommen werden.

6.5 Buslänge

Bit-Rate [kBit/s]	Typische Werte der erreichbaren Leitungslänge **mit esd-Interface** l_{max} [m]	**CiA-Empfehlungen** (07/95) für erreichbare Leitungslängen l_{min} [m]
1000	37	25
800	59	50
$666.\overline{6}$	80	-
500	130	100
$333.\overline{3}$	180	-
250	270	250
166	420	-
125	570	500
100	710	650
$83.\overline{3}$	850	-
$66.\overline{6}$	1000	-
50	1400	1000
$33.\overline{3}$	2000	-
20	3600	2500
12.5	5400	-
10	7300	5000

Tabelle 14: Erreichbare Leitungslängen in Abhängigkeit von der Bitrate (mit esd-CAN-Interfaces)

- Optokoppler verzögern die CAN-Signale. esd-Module erreichen typischerweise eine Leitungslänge von 37 m bei 1 MBit/s. Voraussetzung hierfür ist ein abgeschlossenes Netz ohne Impedanzstörungen, wie z.B. längere Stichleitungen > 0.3 m.

Hinweis:
Bitte beachten Sie die Empfehlungen gemäß ISO 11898 für die Auswahl der Leiterquerschnitte in Abhängigkeit von der Kabellänge.

6.6 Beispiele für CAN-Kabel

Die folgenden zwei- und vieradrigen Kabel empfiehlt esd für den Aufbau von CAN-Netzwerken. Diese Kabeltypen werden auch für die bei esd erhältlichen CAN-Kabel verwendet.

6.6.1 Kabel für leicht störbehaftete Industrieumgebung (zweiadrig)

Hersteller	Leitungstyp
U.I. LAPP GmbH Schulze-Delitzsch-Straße 25 70565 Stuttgart Germany www.lappkabel.de	z.B. UNITRONIC ®-BUS CAN UL/CSA (1x 2x 0.22) (UL/CSA approved) Bestell-Nr.: 2170260 UNITRONIC ®-BUS-FD P CAN UL/CSA (1x 2x 0.25) (UL/CSA approved) Bestell-Nr.: 2170272
ConCab GmbH Äußerer Eichwald 74535 Mainhardt Germany www.concab.de	z.B. BUS-PVC-C (1x 2x 0,22 mm²) Bestell-Nr.: 93 022 016 (UL appr.) BUS-Schleppflex-PUR-C (1x 2x 0,25 mm²) Bestell-Nr.: 94 025 016 (UL appr.)

6.6.2 Kabel für stark störbehaftete Industrieumgebung (vieradrig)

Hersteller	Leitungstyp
U.I. LAPP GmbH Schulze-Delitzsch-Straße 25 70565 Stuttgart Germany www.lappkabel.de	z.B. UNITRONIC ®-BUS CAN UL/CSA (2x 2x 0.22) (UL/CSA approved) Bestell-Nr.: 2170261 UNITRONIC ®-BUS-FD P CAN UL/CSA (2x 2x 0.25) (UL/CSA approved) Bestell-Nr.: 2170273
ConCab GmbH Äußerer Eichwald 74535 Mainhardt Germany www.concab.de	z.B. BUS-PVC-C (2x 2x 0,22 mm²) Bestell-Nr.: 93 022 026 (UL appr.) BUS-Schleppflex-PUR-C (2x 2x 0,25 mm²) Bestell-Nr.: 94 025 026 (UL appr.)

Hinweis:
Fertig konfektionierte Kabel in Standard- und Sonderlängen können bei **esd** bezogen werden.

7. CAN-Bus Troubleshooting Guide

Der CAN-Bus Troubleshooting Guide ist eine Anleitung zum Auffinden und Beseitigen der häufigsten Hardware-Fehlerursachen in der CAN-Bus-Verdrahtung.

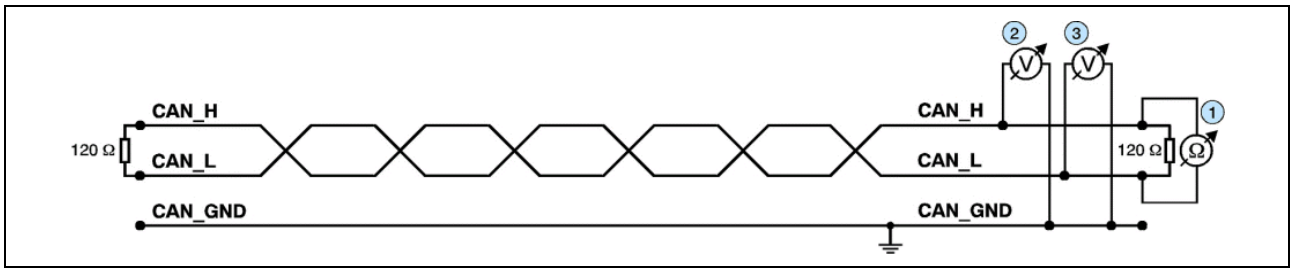


Abb. 21: Vereinfachtes Schaltbild eines CAN-Netzwerks

7.1 Bus-Abschluss

Der Bus-Abschluss wird verwendet, um den Widerstand eines Knotens an den Widerstand der verwendeten Busleitung anzupassen. Ist die Impedanz falsch angepasst, wird das gesendete Signal nicht ganz von der Last aufgenommen und zum Teil in die Übertragungsleitung zurück reflektiert. Sind die Quellen-, Übertragungsleitungs- und Last-Impedanz gleich groß, so werden die Reflexionen vermieden. Dieser Test misst den Gesamtwiderstand der beiden CAN-Datenleitungen und des angeschlossenen Abschlusswiderstandes.

Zum Testen verfahren Sie bitte wie folgt:

1. Schalten Sie die Versorgungsspannungen aller angeschlossenen CAN-Knoten aus.
2. Messen Sie den DC-Widerstand zwischen CAN_H und CAN_L an einem Ende des Netzwerks (1) (siehe obere Abbildung).

Der gemessene Wert sollte zwischen 50 Ω und 70 Ω liegen.

Liegt der ermittelte Wert unter 50 Ω, stellen Sie bitte sicher, dass:
- kein **Kurzschluss** zwischen den CAN_H- und CAN_L-Leitungen besteht
- **nicht mehr als zwei** Abschlusswiderstände angeschlossen sind
- die Transceiver der einzelnen Knoten nicht defekt sind.

Liegt der ermittelte Wert über 70 Ω, stellen Sie bitte sicher, dass:
- alle CAN_H- und CAN_L- Leitungen korrekt angeschlossen sind
- zwei Abschlusswiderstände von **je 120 Ω** an Ihr CAN-Netzwerk angeschlossen sind (einer an jedem Ende).

7.2 Erdung

CAN_GND des CAN-Netzwerks sollte nur an einer einzigen Stelle mit dem Funktionserde-Potenzial (FE) verbunden sein. Dieser Test zeigt an, ob die Abschirmung an mehreren Stellen geerdet ist. Zum Testen verfahren Sie bitte wie folgt:

1. Trennen Sie die CAN_GND vom Erdpotenzial (FE).
2. Messen Sie den DC-Widerstand zwischen CAN_GND und Erdpotenzial (siehe nebenstehende Abbildung).
3. Verbinden Sie CAN_GND wieder mit dem Erdpotenzial.

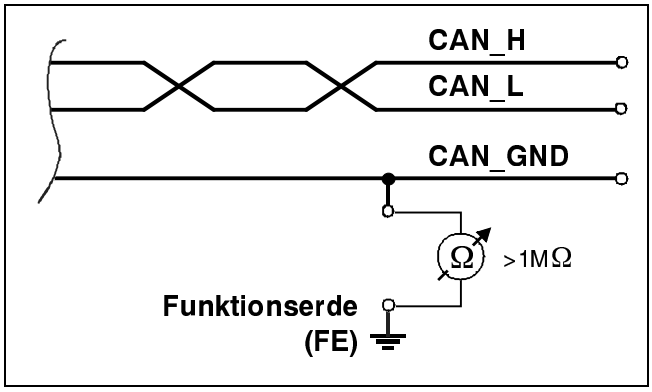


Abb. 22: Vereinfachtes Schaltbild Erdungsmessung

Der gemessene Widerstand sollte größer als ein 1 MΩ sein. Ist er kleiner, suchen Sie bitte nach zusätzlichen Erdungen der CAN_GND-Leitung.

7.3 Kurzschluss in der CAN-Verdrahtung

Ein CAN-Bus kann möglicherweise auch dann noch Daten übertragen, wenn CAN_GND und CAN_L kurzgeschlossen sind. Dadurch wird aber in der Regel die Fehlerrate stark ansteigen.
Stellen Sie sicher, dass zwischen CAN_GND und CAN_L kein Kurzschluss besteht!

7.4 CAN_H/CAN_L-Spannungen

Jeder Knoten verfügt über einen CAN-Transceiver, der differenzielle Signale auf den Datenleitungen generiert. Ruht die Netzwerk-Kommunikation, betragen die CAN_H- und CAN_L-Spannungen etwa 2,5 V zu CAN_GND. Defekte Transceiver können diese Ruhespannungen verändern und die Netzwerk-Kommunikation unterbrechen.

Um auf defekte Transceiver zu testen, verfahren Sie bitte wie folgt:

1. Schalten Sie alle Versorgungsspannungen an.
2. Beenden sie jegliche Netzwerk-Kommunikation.
3. Messen Sie die DC-Spannung zwischen CAN_H und GND ②
   (siehe Abbildung auf vorhergehender Seite).
4. Messen Sie die DC-Spannung zwischen CAN_L und GND ③
   (siehe Abbildung auf vorhergehender Seite).

Die gemessene Spannung sollte zwischen 2,0 V und 3,0 V liegen.

Ist die Spannung kleiner als 2,0 V oder größer als 3,0 V, ist es möglich, dass ein oder mehrere Knoten defekte Transceiver haben. Bei einer Spannung, die unter 2,0 V liegt, überprüfen Sie bitte den Anschluss der CAN_H- und CAN_L-Leitungen.

Um in einem Netzwerk einen Knoten mit einem defekten Transceiver zu finden, überprüfen Sie bitte einzeln die Widerstände der CAN-Transceiver der Knoten (siehe folgendes Kapitel).

7.5 CAN-Transceiver Widerstandstest

CAN-Transceiver verfügen über Schaltkreise, die CAN_H und CAN_L kontrollieren. Die Erfahrung zeigt, dass elektrische Beschädigungen den Leckstrom in diesen Schaltkreisen erhöhen können.
Um den Leckstrom durch die CAN-Schaltungen zu messen, benutzen Sie bitte ein Widerstands-messgerät und:

1. Schalten Sie den Koten aus und trennen Sie ihn vom Netzwerk (4) (siehe untere Abbildung).

2. Messen Sie den DC-Widerstand zwischen CAN_H und CAN_GND (5) (siehe untere Abbildung).

3. Messen Sie den DC-Widerstand zwischen CAN_L und CAN_GND (6) (siehe untere Abbildung).

Der gemessene Widerstand sollte bei jeder Messung etwa 500 kΩ betragen. Liegt der Widerstand deutlich niedriger, ist der CAN-Transceiver möglicherweise defekt.
Ein weiterer Hinweis auf einen fehlerhaften CAN-Transceiver ist eine sehr hohe Abweichung der beiden gemessenen Eingangswiderstände (>> 200%).

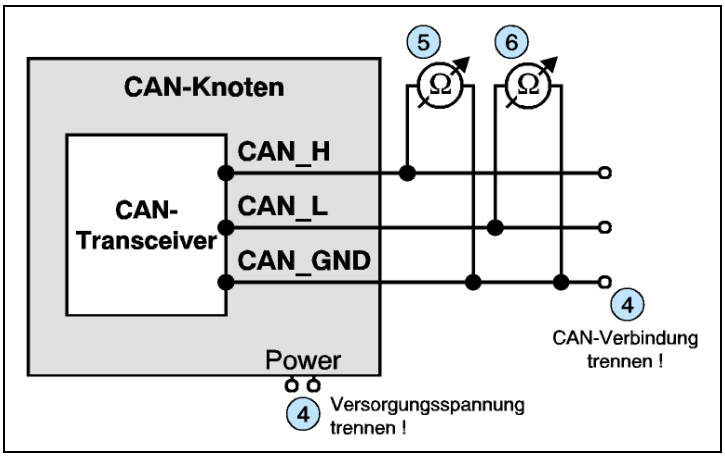


Abb. 23: Messung des Eingangswiderstandes des CAN-Transceivers

7.6 Support bei esd

Sollten Sie trotz Anwendung des CAN-Bus Troubleshooting Guides zu keiner Lösung kommen, wenden Sie sich bitte an unseren Support unter support@esd.eu oder Tel. **0511-37298-130**

8. CANopen-Firmware

Dieses Kapitel enthält neben allgemeinen Grundlagen zum Thema CANopen die wichtigsten Informationen über die implementierten Funktionen.

Eine komplette CANopen-Beschreibung läge außerhalb des Rahmens dieses Handbuchs. Weitergehende Informationen sind daher der CANopen - Dokumentation [1] zu entnehmen.

8.1 Begriffsdefinition

COB ...	Communication Object	
Emergency-Id...	Emergency Data Object	Notfalldaten
NMT...	Network Management (Master)	Netzwerkmanagement
SDO...	Service Data Object	Service-Daten (Parameter)
Sync...	Sync(frame) Telegramm	Synchronisations-Identifier

PDOs (Process Data Objects)

Die PDOs dienen zur Übertragung der Prozessdaten.
Im ‘Transmit’-PDO (TPDO) sendet das CAN-CBX-Modul Daten auf dem CANopen-Netz.
Im ‘Receive’-PDO (RPDO) empfängt das CAN-CBX-Modul Daten aus dem CANopen-Netz.

SDOs (Service Data Objects)

Die SDOs dienen zur Übertragung von modulinternen Konfigurations- und Parameterdaten. Im Gegensatz zu den PDOs werden die SDO-Nachrichten bestätigt. Einer Schreib- oder Leseanforderung auf ein Daten-Objekt folgt immer ein Bestätigungstelegramm mit einem Fehlerindex.

8.2 NMT-Boot-up

Das CAN-CBX-Modul kann mit dem in [1] beschriebenen ‘Minimal - Boot-up’ initialisiert werden. (Ab Firmware-Version 1.02 oder aktueller)

Im einfachsten Fall reicht nach dem Einschalten ein Telegramm zum Umschalten aus dem Zustand *Pre-Operational* in den Zustand *Operational*. Dazu ist an den CAN-Identifier ‘0000_h’ z. B. das 2-Byte-Telegramm ‘01_h’, ‘00_h’ zu senden (= Start Remote Node all Devices).

8.3 Das CANopen-Objektverzeichnis

Das Objektverzeichnis ist eine (sortierte) Gruppierung von Objekten, auf die über das CAN-Netzwerk zugegriffen werden kann. Jedes Objekt in diesem Verzeichnis wird über einen 16-Bit-Index adressiert. In den Objektverzeichnissen wird der Index in hexadezimaler Form angegeben.

Der Index kann 16-Bit-Parameter nach der CANopen-Spezifikation [1] oder ein herstellerspezifischer Code sein. Anhand der höherwertigen Bits des Index wird festgelegt, zu welcher Objektklasse der Parameter gehört.

Zum Objektverzeichnis gehören unter anderem:

Index	Objekt
0001_h ... $009F_h$	Definition von Datentypen
1000_h ... $1FFF_h$	Communication Profile Area
2000_h ... $5FFF_h$	Manufacturer Specific Profile Area
6000_h ... $9FFF_h$	Standardised Device Profile Area
$A000_h$ ... $FFFF_h$	reserviert

8.4 Communication Parameter

Die Communication Parameter der PDOs (gemäß [1]) werden als SDOs Service Data Objects) auf der ID '**600_h + NodeID**' übertragen (Request). Der Empfänger quittiert die Parameter auf der ID '**580_h + NodeID**' (Response).

Die **Node-ID** (Modul-Nr.) wird über die Kodierschalter Low und High konfiguriert. Die möglichen Einstellungen sind in Kapitel " Einstellung der Node-ID über die Kodierschalter" auf Seite 17 beschrieben.

8.4.1 Zugriff auf das Objektverzeichnis über SDOs

Die SDOs (Service Data Objects) dienen zum Zugriff auf das Objektverzeichnis eines Gerätes.
Ein SDO stellt daher einen 'Kanal' zum Zugriff auf die Parameter des Gerätes dar. Der Zugriff über diesen Kanal ist im Zustand *operational* und *pre-operational* möglich.

Dieses Kapitel beschreibt nicht alle möglichen sondern nur einige wichtige Arten des Zugriffs im CAN-CBX-Modul.
Definitionen der Zugriffsmodi können [1] entnommen werden.

Ein SDOCAN Frame ist wie folgt aufgebaut:

Identifier	Befehls-code	Index		Sub-Index	LSB	Datenfeld	MSB
		(low)	(high)				

Beispiel:

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

Beschreibung der SDOs:

Identifier
Die Parameter werden auf der ID '600_h + NodeID' übertragen (Request).
Der Empfänger quittiert die Parameter auf der ID '580_h + NodeID' (Response).

Befehlscode
Der gesendete Befehlscode setzt sich unter anderem aus dem Command Specifier und der Länge zusammen. Häufig benötigte Kombinationen sind z.B.:

$40_h = 64_{dez}$: Read Request, d.h. ein Parameter soll gelesen werden
$23_h = 35_{dez}$: Write Request mit 32 Bit Daten, d.h. ein Parameter soll gesetzt werden

Das CAN-CBX-Modul antworten auf jedes empfangene Telegramm mit einem Antworttelegramm. Das Antworttelegramm kann folgende Befehlscodes enthalten:

$43_h = 67_{dez}$: Read Response mit 32 Bit Daten, dieses Telegramm enthält den gewünschten Parameter
$60_h = 96_{dez}$: Write Response, d.h. ein Parameter wurde erfolgreich gesetzt
$80_h = 128_{dez}$: Error Response, d.h. das CAN-CBX-Modul meldet einen Fehler.

Häufig verwendete Befehlscodes
Die folgende Tabelle gibt eine Übersicht über häufig verwendete Befehlscodes. Die Kommando-Frames müssen immer 8 Bytes beinhalten. Hinweise zur Syntax und weitere Befehlscodes sind in [1] nachzulesen.

Kommando	für Anzahl Datenbytes	Befehlscode
Write Request (Initiate Domain Download)	1 2 3 4	$2F_h$ $2B_h$ 27_h 23_h
Write Response (Initiate Domain Download)	-	60_h
Read Request (Initiate Domain Upload)	-	40_h
Read Response (Initiate Domain Upload)	1 2 3 4	$4F_h$ $4B_h$ 47_h 43_h
Error Response (Abort Domain Transfer)	-	80_h

Index, Sub-Index
Der Index und der Sub-Index werden in den Kapiteln “Device Profile Area” und “Manufacturer Specific Profile Area” dieses Handbuches beschrieben.

Datenfeld
Das maximal 4 Byte lange Datenfeld ist grundsätzlich nach der Regel ‘niederwertiges Byte zuerst, höherwertiges Byte zuletzt’ aufgebaut. Dabei steht das niederwertige Byte immer in ‘Data 1’, bei 16-Bit-Werten steht das höchstwertige Byte (Bits 8...15) in ‘Data 2’, und bei 32-Bit-Werten steht das MSB (Bits 24...31) in ‘Data 4’.

Fehlercodes des SDO-Transfers

Die folgenden SDO Abort Codes (Fehlercodes) können zur Anwendung kommen (gemäß [1]):

SDO Abort Codes	Erläuterung
05040001_h	falscher Command Specifier
06010002_h	Schreibzugriff ist hier falsch
06020000_h	falscher Index
06040041_h	Objekt kann nicht auf PDO abgebildet werden
06060000_h	kein Zugriff wegen Hardware Fehler
06070010_h	falsche Anzahl Daten-Bytes
06070012_h	Länge des Service-Parameters ist zu groß
06070013_h	Länge des Service-Parameters ist zu klein
06090011_h	falscher Sub-Index
06090030_h	gesendeter Parameter außerhalb des zulässigen Wertebereiches
08000000_h	nicht definierte Fehlerursache
08000020_h	Daten können nicht übertragen oder gespeichert werden
08000022_h	wegen des aktuellen Device States können Daten nicht übertragen oder gespeichert werden
08000024_h	Zugriff auf Flash fehlgeschlagen

8.4.2 Nichtflüchtiges Speichern von Parametern im EEPROM

Die eingestellten Parameter sind sofort nach der Übergabe aktiv. Das ‘nichtflüchtige’ Speichern der Parameter erfolgt jedoch nicht automatisch. Es muss über einen Schreibzugriff auf das Objekt 1010_h veranlasst werden und sollte nur erfolgen, wenn sich das Modul im Zustand *pre-operational* befindet.

Der Speicher-Modus wird über den Inhalt des Objektes 1010_h angezeigt:
Bit 1 des Objekts 1010_h, Sub-Index 1 ist nicht gesetzt, d.h. das CAN-CBX-COM2-Modul speichert die Konfiguration nicht ‘automatisch’, sondern muss durch Schreiben der Zeichenkette ‘save’ (73_h 61_h 76_h 65_h, Reihenfolge aus CAN-Telegramm) in das Objekt 1010_h, Sub-Index 1 extra dazu veranlasst werden.

Read-Request zur Anzeige des aktuellen Speicher-Modus:

Identifier	Befehls-code	Index (low)	Index (high)	Sub-Index	Data 1	Data 2	Data 3	Data 4
600_h+ Node-ID	40_h (read request)	10_h	10_h	01_h	00_h	00_h	00_h	00_h
					Daten werden hier nicht ausgewertet			

Rückmeldung des CAN-CBX-COM2-Moduls (bei Default-Einstellung):

Identifier	Befehls-code	Index (low)	Index (high)	Sub-Index	Data 1	Data 2	Data 3	Data 4
600_h+ Node-ID	$4F_h$ (read response, 1 Byte)	10_h	10_h	01_h	01_h	00_h	00_h	00_h
					Speichern der Parameter nur bei Request			

Kommando zum Speichern der Parameter:

Identifier	Befehls-code	Index (low)	Index (high)	Sub-Index	Data 1	Data 2	Data 3	Data 4
600_h+ Node-ID	23_h (write, 4 Bytes)	10_h	10_h	01_h	73_h ‘s’	61_h ‘a’	76_h ‘v’	65_h ‘e’
					= Speichern der Parameter			

8.4.3 Wiederherstellen des Default-Zustandes der Parameter

Der Default-Zustand der Parameter wird in zwei Schritten wiederhergestellt:

1. Parameter ‘Restore Default-Parameter’ aufrufen : Index 1011_h, Sub-Index 1_d
2. Modul zurücksetzen über NMT-Service ‘Reset’: auf Identifier ‘0’ die Daten 81xx senden (xx steht für Node-ID)

8.5 Übersicht der verwendeten CANopen-Identifier

Funktion	Identifier	Bemerkungen
Netzwerkmanagement	0	NMT
SYNC	80_h	Sync an alle, (konfigurierbar über Objekt 1005_h)
Emergency Message	80_h + *NodeID*	konfigurierbar über Objekt 1014_h
RPDO1	200_h + *NodeID*	PDO1 zum CAN-CBX-COM2 (Empfangs-PDO1 COM A)
TPDO1	180_h + *NodeID*	PDO1 vom CAN-CBX-COM2 (Objekt 1801_h) (Sende-PDO1 COM A)
RPDO3	400_h + *NodeID*	PDO3 zum CAN-CBX-COM2 (Empfangs-PDO3 COM B)
TPDO3	380_h + *NodeID*	PDO3 vom CAN-CBX-COM2 (Objekt 1802_h) (Sende-PDO3 COM B)
Client-SDO	580_h + *Node-ID*	SDO vom CAN-CBX-COM2 (Tx)
Server-SDO	600_h + *Node-ID*	SDO zum CAN-CBX-COM2 (Rx)
Node Guarding	700_h + *NodeID*	nicht konfigurierbar

NodeID: Eingestellte CANopen-Adresse [1_h...$7F_h$]

8.5.1 Einstellung der COB-ID

Die COB-IDs die einstellbar sind (außer der von SYNC), werden zunächst von der Einstellung der Node-ID über die Kodierschalter (siehe Seite 17) abgeleitet. Wurden die COB-IDs über SDO beschrieben, gilt diese Einstellung, auch wenn der Schalter danach auf eine andere Node-ID eingestellt wird.
Um die Default-COB-ID wieder vom Schalter zu übernehmen, müssen die *Comm defaults* oder alle Defaults geladen werden (Objekt 1011_h).

8.6 Default PDO-Belegung

Die PDOs (Process Data Objects) dienen zur Übertragung der seriellen Daten. Die folgenden Tabellen zeigen die möglichen Inhalte der TPDOs und RPDOs:

PDO	CAN-Identifier	Länge	Übertragungs-richtung	Default-Belegung
TPDO1	180_h + Node-ID	bis zu 8 Byte	vom CAN-CBX-COM2 (Sende-PDO)	COM A Daten seriell -> CAN, Byte 0...7
TPDO3	380_h + Node-ID	bis zu 8 Byte		COM B Daten seriell -> CAN, Byte 0...7

PDO	CAN-Identifier	Länge	Übertragungs-richtung	Default-Belegung
RPDO1	200_h + Node-ID	bis zu 8 Byte	zum CAN-CBX-COM2 (Empfangs-PDO)	COM A Daten seriell <- CAN, Byte 0...7
RPDO3	400_h + Node-ID	bis zu 8 Byte		COM B Daten seriell <- CAN, Byte 0...7

8.7 Communication Profile Area

8.7.1 Verwendete Bezeichnungen und Abkürzungen

Folgende Bezeichnungen werden in den Tabellen zur Beschreibung der Kommunikationsparameter verwendet:

PDO-Mappable	für diesen Sub-Index des PDOs ist PDO-Mapping möglich
Save to EEPROM	der Wert dieses Parameters wird im lokalen EEPROM gespeichert, wenn das Kommando ‘save’ aufgerufen wird (siehe Seite 52)
Datentyp	Datentyp (z.B. unsigned 8, unsigned 32)
Zugriff	zugelassene Zugriffsarten auf diesen Parameter ro... read_only Dieser Parameter kann nur gelesen werden. Schreibzugriffe führen zu einer Fehlermeldung. const.... constant Dieser Parameter kann vom Anwender nicht verändert werden. Er ist Lesbar. Schreibzugriffe führen zu einer Fehlermeldung. rw... read&write Dieser Parameter kann gelesen oder gesetzt werden.
Wertebereich	Wertebereich des Parameters
Default-Wert	Grundeinstellung des Parameters
Name	Name und Kurzbeschreibung des Parameters

8.8 Implementierte CANopen-Objekte

Eine detaillierte Beschreibung der Objekte ist in [1] nachzulesen.

Index	Name	Sub-Index (max.)	Datentyp	Produktspezifische Eigenschaften
1000_h	Device Type	-	unsigned 32	$0080\ 0000_h$
1001_h	Error Register	-	unsigned 8	unterstützte Fehler-Bits: 0: generic 7: manufacturer
1002_h	Manufacturer Status	-	unsigned 32	Status
1003_h	Predefined Error	A_h	unsigned32	-
1005_h	COB-ID SYNC-Message	-	unsigned 32	default: 80_h
1006_h	Communication Cycle Period	-	unsigned 32	default: 0 µs
1008_h	Manufacturer's Device Name	-	visible string	default: 'CAN-CBX-COM2'
1009_h	Manufacturer's Hardware Version	-	visible string	default ist versionsabhängig
$100A_h$	Manufacturer's Software Version	-	visible string	default ist versionsabhängig
$100C_h$	Guard Time	-	unsigned 16	default: 0
$100D_h$	Life Time Factor	-	unsigned 8	default:0
1010_h	Store Parameters	4	unsigned 32	Parameter, die gepeichert oder geladen werden können: 1005h ... 1029h, 2800h, 2810h, 2880h, 2881h, 2900h, 2910h, 2980h, 2981h
1011_h	Restore Default Parameters	4	unsigned 32	
1014_h	COB-ID Emergency	-	unsigned 32	default: 80_h + Node-ID
1015_h	Inhibit Time Emergency	-	unsigned 16	default: 0
1016_h	Consumer Heartbeat Time	1	unsigned 32	default: 0
1017_h	Producer Heartbeat Time	0	unsigned 16	default: 0
1018_h	Identity Object	4	unsigned 32	Vendor Id: 00000017_h Prod. Code: 23055002_h (= C.3055.02)
1019_h	Synchronous counter overflow value	-	unsigned 8	default: 0
1020_h	Verify Configuration	2	unsigned 32	--
1029_h	Error Behaviour Object	4	unsigned 8	default: 0

Index	Sub-Index (max.)	Beschreibung	Datentyp
1400_h	6	1. Receive PDO Communication Parameter	PDO CommPar
1402_h	6	3. Receive PDO Communication Parameter	PDO CommPar
1600_h	8	1. Receive PDO Mapping	PDO Mapping
1602_h	8	3. Receive PDO Mapping	PDO Mapping
1800_h	6	1.Transmit PDO Communication Parameter	PDO CommPar
1802_h	6	3. Transmit PDO Communication Parameter	PDO CommPar
$1A00_h$	8	1. Transmit PDO Mapping	PDO Mapping
$1A02_h$	8	3. Transmit PDO Mapping	PDO Mapping

Index	**Sub-Index** (max.)	**Beschreibung**	**Datentyp**	**Zugriff**	**Produktspezifische Eigenschaften**
$1F80_h$	-	NMT startup	unsigned 32	rw	default: 2 (autostart disabled)
$1F91_h$	1	Self starting nodes timing parameters	unsigned 16	rw	default: 64_h (= 100 ms)

8.8.1 Device Type (1000_h)

INDEX	**1000_h**
Name	*device type*
Datentyp	unsigned 32
Zugriff	ro
Default-Wert	$0080\ 0000_h$

Der Wert des *device type* beträgt: 0080.0000_h (Additional Information: 0080_h
Device Profile Number: 0000_h)

Beispiel: Auslesen des Device Type

Der CANopen Master sendet unter dem Identifier '603_h' (600_h + Node-ID) den Read Request an das CAN-CBX-COM2-Modul mit der Modul-Nr. 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				

Das CAN-CBX-COM2-Modul Nr. 3 antwortet dem Master anhand der Read Response unter dem Identifier '583_h' (580_h + Node-ID) mit dem Wert des 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	00_h	00_h	80_h	00_h
			Read Response	Index=1000_h		Sub Index	Device Profile Nr. 0000_h		Additional Information 0080_h	

ausgegebener Wert des Device Type: 0080.0000_h

Das Datenfeld ist grundsätzlich nach der Regel 'niederwertiges Byte zuerst, höherwertiges Byte zuletzt' aufgebaut.

8.8.2 Error Register (1001_h)

Das CAN-CBX-Modul nutzt das Error-Register, um Fehlermeldungen anzuzeigen.

INDEX	**1001_h**
Name	*error register*
Datentyp	unsigned 8
Zugriff	ro
Default-Wert	0

Bit	Bedeutung	von diesem Modul unterstützt?
0	generic	✔
1	current	-
2	voltage	-
3	temperature	-
4	communication error (overrun, error, state)	-
5	device profile	-
6	reserved	-
7	manufacturer-specific error	✔

Die nicht unterstützten Bits (-) werden immer als ‘0’ zurückgegeben.

Folgende Meldungen sind möglich:

00_h - keine Fehler
81_h - eine der von esd definierten Fehlerbedingungen ist aufgetreten (any manufacturer-specific error)

8.8.3 Manufacturer Status Register 1002_h

INDEX	**1002_h**
Name	manufacturer status register
Datentyp	unsigned 32
Default-Wert	No

Die Bits des Status-Registers sind Hersteller-spezifisch und nur für interne Zwecke zu verwenden.

8.8.4 Pre-defined Error Field (1003_h)

INDEX	**1003_h**
Name	*pre-defined error field*
Datentyp	unsigned 32
Zugriff	ro
Default-Wert	No

Im *pre-defined error field* wird eine Liste der zuletzt aufgetretenen Fehler gespeichert. Der Sub-Index 0 enthält die aktuelle Anzahl der in der Liste gespeicherten Fehler. Unter Sub-Index 1 wird der zuletzt aufgetretene Fehler abgelegt. Tritt ein neuer Fehler auf, so wird der vorhergehende Fehler auf Sub-Index 2 gespeichert und der neue Fehler unter Sub-Index 1 usw.. So entsteht eine Liste mit der Fehler-Historie.

Der Fehlerspeicher ist wie ein Ringspeicher aufgebaut. Ist er gefüllt, wird der älteste Eintrag gelöscht um Platz für den aktuellen Eintrag zu schaffen.

Dieses Modul unterstützt maximal 10 Fehlereinträge. Beim Eintreten des 11. Fehlers wird der älteste Fehlereintrag gelöscht. Zum Löschen der gesamten Fehler-Liste ist der Sub-Index ‘0’ auf ‘0’ zu setzen. Dies ist der einzig zulässige Schreibzugriff auf das Objekt.

Mit jedem neuen Eintrag in die Liste sendet das Modul ein **Emergency-Frame**, um den Fehler mitzuteilen.

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
1003_h	0	*no_of_errors_in_list*	0, 1...A_h	-	unsigned 8	rw
	1	*error-code n*	0...$FFFFFFFF_h$	-	unsigned 32	ro
	2	*error-code (n-1)*	0...$FFFFFFFF_h$	-	unsigned 32	ro
	:	*:*	:	:	:	ro
	A_h	*error-code (n-9)*	0...$FFFFFFFF_h$	-	unsigned 32	ro

Bedeutung der Variablen:

no_of_errors_in_list - enthält die aktuelle Anzahl der in der Liste eingetragenen Fehler-Codes
n = Nummer des zuletzt aufgetretenen Fehlers
- zum Löschen der Fehlerliste ist diese Variable auf ‘0’ zu setzen
- ist *no_of_errors_in_list* $\neq 0$, so wird das Error-Register (Object 1001_h) gesetzt

error-code x Der 32-Bit lange Fehler-Code setzt sich aus dem in [1] aufgeführten CANopen-Emergency-Error-Code und den von esd definierten Fehler-Codes (Manufacturer-Specific Error Field) zusammen.

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

manufacturer-specific error field: immer ‘00’, es sei denn *emergency-error-code* = 2300_h (siehe unten)

emergency-error-code: Es werden die folgenden Fehler-Codes unterstützt:

8110_h - CAN Overrun Error
- Sample-Rate ist so hoch eingestellt, dass die Firmware nicht in der Lage ist, alle Daten auf dem CAN-Bus zu senden

8120_h - CAN in Error Passive Mode
8130_h - Lifeguard Error / Heartbeat Error
8140_h - Recovered from “Bus Off”
8240_h - Unexpected SYNC data lenght
6000_h - Software-Fehler:
-EEPROM Checksum Fehler (keine Übertragung dieser Message als Emergency)

6110_h - Internal Software Error
z.B.:
- gespeicherte Daten hatten ungültige Checksumme und Default-Werte wurden geladen
- interner Watchdog hat ausgelöst

$FF10_h$ - Data lost (A/D-Daten Overflow)
5000_h - Hardware Fehler (z.B. A/D-Wandler defekt)

Emergency Message

Die Daten des vom CAN-CBX Modul gesendeten Emergency-Frames sind wie folgt aufgebaut:

Byte:	0	1	2	3	4	5	6	7
Inhalt:	*emergency-error-code* (siehe oben)		*error-register* 1001_h	*no_of_errors_in_list* $1003,00_h$	-			

Eine Emergency Message wird beim ersten Auftreten eines Fehlers gesendet. Tritt der selbe Fehler ein weiteres Mal auf, wird keine weitere Emergency Message gesendet.
Sobald eine Fehlermeldung zurückgenommen wird, wird wiederum eine Emergency Message gesendet.

8.8.5 COB-ID of SYNC-Message (1005_h)

INDEX	**1005_h**
Name	*COB-ID SYNC message*
Datentyp	unsigned 32
Zugriff	rw
Default-Wert	80_h
Low Limit	01_h
High Limit	C000 $07FF_h$

Struktur des Parameters:

Bit-No.	Wert	Bedeutung
31 (MSB)	-	do not care
30	0/1	0: keine SYNC Erzeugung 1: Modul erzeugt SYNC
29	0	immer 0, da 11-Bit-ID
28...11	0	immer 0, da keine 29-Bit-ID unterstützt werden
10...0 (LSB)	x	Bit 0...10 der SYNC-COB-ID

Der Identifier kann Werte zwischen $0...7FF_h$ annehmen.

8.8.6 Communication Cycle Period (1006_h)

INDEX	**1006_h**
Name	*Communication Cylcle Period*
Datentyp	unsigned 32
Zugriffsart	rw
Default-Wert	0 µs

Wertebereich des Parameters:

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
1006_h	0	*communication cycle period*	0, $0000\ 01F4_h$ ... $FFFF\ FFFF_h$	0	unsigned 32	rw

communication cycle period Zykluszeit des Sync-Frames in [µs].
Ist hier ‘0’ eingetragen, so erfolgen keine SYNC-Sendungen.
(Beispiel: $4E20_h$ entspricht einer Zykluszeit von 20000µs = 20 ms)

8.8.7 Manufacturer Device Name (1008_h)

INDEX	**1008_h**
Name	*manufacturer device name*
Datentyp	visible string
Default-Wert	siehe Kapitel 8.8, Seite 56

Eine ausführliche Beschreibung des SDO Uploads ist [1] zu entnehmen.

8.8.8 Manufacturer Hardware Version (1009_h)

INDEX	**1009_h**
Name	*manufacturer hardware version*
Datentyp	visible string
Default-Wert	string: z.B. ‘1.0’ (Versionsabhängig)

Das Lesen der Hardware-Version erfolgt ähnlich wie das Lesen des Manufacturer Device Names über das SDO Upload Protokoll. Eine ausführliche Beschreibung des Uploads ist [1] zu entnehmen.

8.8.9 Manufacturer Software Version ($100A_h$)

INDEX	**$100A_h$**
Name	*manufacturer software version*
Datentyp	visible string
Default-Wert	string: z.B.: ‘1.2’ ’ (Versionsabhängig)

Das Lesen der Hardware-Version erfolgt ähnlich wie das Lesen des Manufacturer Device Names über das SDO Upload Protokoll. Eine ausführliche Beschreibung des Uploads ist [1] zu entnehmen.

8.8.10 Guard Time ($100C_h$) und Life Time Factor ($100D_h$)

Das CAN-CBX-Modul unterstützt das Node-Guarding oder alternativ die Heartbeat-Funktion (siehe Seite 76).

Hinweis:
Nach den CiA-Empfehlungen sollte bevorzugt die Heartbeat-Funktion verwendet werden. Das Node-Guarding sollte nur noch für bestehende Systeme eingesetzt werden!

Guard Time und Life Time Factor werden zusammen ausgewertet. Die Multiplikation beider Werte ergibt die Life Time. Die Guard Time wird in Millisekunden angegeben.

INDEX	**$100C_h$**
Name	*guard time*
Datentyp	unsigned 16
Zugriff	rw
Default-Wert	0 [ms]
Minimaler Wert	0
Maximaler Wert	$FFFF_h$ (65,535 s)

INDEX	**$100D_h$**
Name	*life time factor*
Datentyp	unsigned 8
Zugriff	rw
Default-Wert	0
Minimaler Wert	0
Maximaler Wert	FF_h

8.8.11 Store Parameters (1010_h)

INDEX	**1010_h**
Name	*store parameters*
Datentyp	unsigned 32

Mit diesem Kommando werden die Parameter im “nicht-flüchtigen” Speicher, hier im EEPROM, gespeichert. Dazu werden die unten aufgeführten Parameter Gruppen unterschieden.
Die eingestellten Parameter sind sofort nach der Übergabe aktiv.
Das ‘nichtflüchtige’ Speichern der Parameter erfolgt jedoch nicht automatisch. Es muss über einen Schreibzugriff auf das Objekt 1010_h veranlasst werden und sollte nur erfolgen, wenn sich das Modul im Zustand *pre-operational* befindet. Beim Schreiben des Index muss die unten angegebene Byte-Folge gesendet werden.

Das Lesen des Index gibt Informationen über die implementierten Speicher-Funktionen zurück (siehe [1]).

Index	Sub-Index	Beschreibung	Wertebereich	Datentyp	Zugriff
1010_h	0	*number_of_entries*	4	unsigned 8	ro
	1	*save_all_parameters* (Objekte 1000_h ... $9FFF_h$)	no default, write: $65\ 76\ 61\ 73_h$ (= ASCII: ‘e’ ‘v’ ‘a’ ‘s’)	unsigned 32	rw
	2	*save_communication_parameter* (Objekte 1000_h ... $1FFF_h$)		unsigned 32	rw
	3	*save_application_parameter* (Objekte 6000_h ... $9FFF_h$)		unsigned 32	rw
	4	*save_manufacturer_parameter* (Objekte 2000_h ... $5FFF_h$)		unsigned 32	rw

Belegung der Variablen

save all parameters

speichert die Parameter aller Objekte (sofern vorhanden), deren Zugriffsrecht Read/Write (rw) ist.

save_communication_parameter

speichert alle Communication Parameter der Objekte (1000_h ... $1FFF_h$, sofern vorhanden), deren Zugriffsrecht Read/Write (rw) ist (hier z.B. 1005_h ... 1029_h).

save_application_parameter

speichert alle Application Parameter der Objekte (Objekte 6000_h ... $9FFF_h$, sofern vorhanden), deren Zugriffsrecht Read/Write (rw) ist (hier z.B. $6xxx_h$).

save_manufacturer_parameter

speichert alle Manufacturer Parameter der Objekte (Objekte 2000_h ... $5FFF_h$, sofern vorhanden), deren Zugriffsrecht Read/Write (rw) ist (hier z.B. $2xxx_h$).

Der Speicher-Modus wird über den Inhalt dieses Objektes angezeigt:
Bit 1 des Objekts 1010_h, Sub-Index 1 ist nicht gesetzt, d.h. das CAN-CBX-Modul speichert die Konfiguration nicht ‘automatisch’, sondern muss durch Schreiben der Zeichenkette ‘save’ (73_h 61_h 76_h 65_h, Reihenfolge aus CAN-Telegramm) in das Objekt 1010_h, Sub-Index 1-4 extra dazu veranlasst werden.

Beim Lesen eines Sub-Index liefert das CANopen Device Informationen über die Speicher-Funktionalität im folgenden Format:

Bit:	31 … 2	1	0
Inhalt:	reserved	auto	cmd
	0	0	1
	MSB		LSB

Bit	Wert	Beschreibung
auto	0	das CAN-CBX-Modul speichert die Parameter **nicht** autonom
	1	das CAN-CBX-Modul speichert die Parameter autonom
cmd	0	das CAN-CBX-Modul speichert die Parameter **nicht** auf Befehl
	1	das CAN-CBX-Modul speichert die Parameter auf Befehl

Autonomes Speichern bedeutet, dass die Parameter “nicht-flüchtig” ohne weitere Abfrage gespeichert werden.

8.8.12 Restore Default Parameters (1011_h)

INDEX	**1011_h**
Name	*restore default parameters*
Datentyp	unsigned 32

Mit diesem Kommando werden die bei der Auslieferung aktiven Default-Parameter wieder geladen. Dazu werden die unten aufgeführten Parameter Gruppen unterschieden.
Alle individuellen Einstellungen, die im EEPROM gespeichert worden sind, gehen verloren, das betrifft auch Kalibrierungswerte.
Die Default-Parameter sind erst nach einem Reset aktiv. Das Aktivieren der Parameter muss über einen Schreibzugriff veranlasst werden.
Beim Schreiben des Index muss die unten angegebene Byte-Folge gesendet werden. Das Lesen des Index gibt Informationen über die implementierten Restore-Funktionen zurück (siehe [1])

Index	Sub-Index	Beschreibung	Wertebereich	Datentyp	Zugriff
1011_h	0	*number_of_entries*	4	unsigned 8	ro
	1	*restore_all_default_parameters* (Objekte 1000_h ... $9FFF_h$)	no default, write: 64 61 6F $6C_h$ (= ASCII: ‘d’ ‘a’ ‘o’ ‘l’)	unsigned 32	rw
	2	*restore_communication_default_parameter* (Objekte 1000_h ... $1FFF_h$)		unsigned 32	rw
	3	*restore_application_default_parameter* (Objekte 6000_h ... $9FFF_h$)		unsigned 32	rw
	4	*restore_manufacturer_default_parameter* (Objekte 2000_h ... $5FFF_h$)		unsigned 32	rw

Belegung der Variablen

restore_all_default_parameters
lädt die Default-Werte aller Parameter aller Objekte (sofern vorhanden).

restore_communication_default_parameter
lädt die Default-Werte aller Communication Parameter der Objekte (1000_h ... $1FFF_h$, sofern vorhanden, z.B. 1005_h ... 1029_h).

restore_application_default_parameter
lädt die Default-Werte der Application Parameter der Objekte (Objekte 6000_h ... $9FFF_h$, sofern vorhanden, z.B. $6xxx_h$).

restore_manufacturer_default_parameter
lädt die Default-Werte aller Manufacturer Parameter der Objekte (Objekte 2000_h ... $5FFF_h$, sofern vorhanden, hier z.B. $2xxx_h$).

Bit 0 des Objekts 1011_h, Sub-Index 1 ist gesetzt, d.h. das CAN-CBX-Modul lädt die Standardwerte durch Schreiben der Zeichenkette ‘load’ (64_h 61_h $6F_h$ $6C_h$, Reihenfolge aus CAN-Telegramm) in das Objekt 1011_h, Sub-Index 1-4.

Beim Lesen eines Sub-Index liefert das CANopen Device Informationen über die Restore-Funktionalität im folgenden Format:

Bit:	31 … 1	0
Inhalt:	reserved	cmd
	0	1
	MSB	LSB

Bit	Wert	Beschreibung
cmd	0	die Default-Parameter des CAN-CBX-Moduls können **nicht** geladen werden
	1	die Default-Parameter des CAN-CBX-Moduls können geladen werden

8.8.13 COB_ID Emergency Message (1014_h)

INDEX	**1014_h**
Name	*COB-ID emergency object*
Datentyp	unsigned 32
Zugriff	ro
Default-Wert	80_h + Node-ID

Dieses Objekt bestimmt die COB-ID der Emergency Message (EMCY), das Objekt kann jedoch nur gelesen werden.

Die Struktur des Objektes ist in der folgenden Tabelle beschrieben:

Bit-No.	Wert	Bedeutung
31 (MSB)	0/1	0: EMCY existiert / ist gültig 1: kein EMCY / EMCY ist nicht gültig
30	0	reserviert (immer 0)
29	0	immer 0, da 11-Bit ID
28...11	0	immer 0, da keine 29-Bit-ID unterstützt werden
10...0 (LSB)	x	Bit 0...10 des COB-ID

Der Identifier kann Werte zwischen $0...7FF_h$ annehmen.

8.8.14 Inhibit Time EMCY (1015_h)

INDEX	**1015_h**
Name	*inhibit_time_emergency*
Datentyp	unsigned 16
Zugriffsart	rw
Wertebereich	0...$FFFF_h$
Default-Wert	0

Mit diesem Objekt kann die *Inhibit Time* für die Emergency-Nachricht festgelegt werden. Die Zeit wird als ein Vielfaches von 100 µs angegeben.

8.8.15 Consumer Heartbeat Time (1016_h)

INDEX	**1016_h**
Name	*consumer heartbeat time*
Datentyp	unsigned 32
Default-Wert	No

Zur gegenseitigen Überwachung der CANopen-Module (insbesondere zum Erkennen von Verbindungsausfällen) kann die Heartbeat-Funktion genutzt werden. Im Gegensatz zum Node Guarding/Life Guarding kommt die Heartbeat-Funktion ohne RTR-Frames aus.

Funktionsweise:
Ein Modul, der sog. Heartbeat-Producer sendet auf dem Node-Guarding-Identifier (siehe Object $100E_h$) zyklisch eine Heartbeat-Nachricht auf dem CAN-Bus. Ein oder mehrere Heartbeat-Consumer empfangen die Nachricht. Die Nachricht muss innerhalb der auf dem Heartbeat-Consumer gespeicherten Heartbeat-Time empfangen werden, sonst wird auf dem Heartbeat-Consumer-Modul ein Heartbeat-Event ausgelöst. Auf dem CAN-CBX-Modul löst der Heartbeat-Event einen Heartbeat-Error aus.
Jedes Modul kann Heartbeat-Producer und Heartbeat-Consumer sein. Das CAN-CBX-Modul unterstützt in einem CAN-Netz maximal einen Heartbeat-Producer.

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
1016_h	0	*number_of_entries*	1	1	unsigned 8	ro
	1	*consumer-heartbeat_time*	$0...007FFFFF_h$	0	unsigned 32	rw

Bedeutung der Variablen *consumer-heartbeat_time_x*:

	consumer-heartbeat_time_x		
Bit	31 24	23 16	15 0
Belegung	reserved (immer ‘0’)	*Node-ID* (unsigned 8)	*heartbeat_time* (unsigned 16)

Node-ID Node-Id des zu überwachenden Heartbeat-Producers.

heartbeat_time Zeit in [ms], innerhalb der sich der zu überwachende Heartbeat-Producer auf dem Node-Guarding-ID melden muss, damit kein Heartbeat-Event ausgelöst wird. Diese Consumer-Heartbeat-Time muss immer größer sein als die Producer-Heartbeat-Time.

Beispiel:

consumer-heartbeat_time = 0031 $03E8_h$

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

8.8.16 Producer Heartbeat Time (1017_h)

INDEX	**1017_h**
Name	*producer heartbeat time*
Datentyp	unsigned 16
Default-Wert	0 ms

Hier wird die Zeit eingetragen, mit der das CAN-CBX-Modul zyklisch einen Heartbeat-Frame auf dem Node-Guarding-ID sendet. Die Heartbeat-Funktion wird auf Seite 67 beschrieben
.
Wird die Producer-Heartbeat-Time auf ‘0’ gesetzt, so wird das Senden des Heartbeats durch dieses Modul beendet.

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
1017_h	0	*producer-heartbeat_time*	0...$FFFF_h$	0 ms	unsigned 16	rw

producer-heartbeat_time Zykluszeit in [ms] des Heartbeat-Producers zum Senden des Heartbeats auf dem Node-Guarding-ID.
Die Consumer-Heartbeat-Time der überwachenden Module muss immer größer sein als die Producer-Heartbeat-Time dieses Heartbeat-sendenden Moduls.

8.8.17 Identity Object (1018_h)

INDEX	**1018_h**
Name	*identity object*
Datentyp	unsigned 32
Default-Wert	No

Dieses Objekt enthält allgemeine Informationen zum CAN-Modul.

Index	Subindex	Beschreibung	Wertebereich	Default-Wert	Datentyp	Zugriff
1018_h	0	*no_of_entries*	4	4	unsigned 8	ro
	1	*vendor_id*	0...FFFF $FFFF_h$	00000017_h	unsigned 32	ro
	2	*product_code*	0...FFFF $FFFF_h$	23055002_h	:	ro
	3	*revision_no*	0...FFFF $FFFF_h$	*aktuelle SW-Rev.*	:	ro
	4	*serial_no*	0...FFFF $FFFF_h$	*HW-Seriennummer*	unsigned 32	ro

Bedeutung der Variablen:

vendor_id — Diese Variable enthält die esd-Vendor-ID. Hier ist immer der Wert 00000017_h eingetragen.

product_code — Hier ist die esd-Artikelnummer des Produkts abgelegt.
Beispiel:
Der Wert ‘2**305 5002**$_h$’ entspricht der Artikelnummer ‘C.3055.02’.

revision_number — Hier ist die Software-Version abgelegt. In den oberen zwei Bytes sind gemäß [1] die Revisionsnummern der wesentlichen (major) Änderungen aufgeführt und in den unteren zwei Bytes die Revisionsnummern einfacher Korrekturen oder Änderungen (minor).

revision_no	
major_revision_no 31 … 16	*minor_revision_no* 15 … 0

MSB LSB

serial_number

Hier wird die Seriennummer-Kennung der Hardware gelesen. Die ersten beiden Zeichen der Seriennummer sind Buchstaben, die das Fertigungslos kennzeichnen. Die folgenden Zeichen geben die eigentliche Seriennummer wieder.

In den beiden höherwertigen Bytes von *serial_no* sind die Buchstaben des Fertigungsloses kodiert. Sie enthalten jeweils den ASCII-Code des Buchstaben mit dem hochwertigsten Bit auf ‘1’ gesetzt, um Buchstaben und Zahlen unterscheiden zu können:
(ASCII-Code) + 80_h = gelesenes_Byte

In den beiden niederwertigen Bytes ist die Nummer des Moduls als BCD-Wert enthalten.

Beispiel:
Wird der Wert ‘C1C2 0105_h’ gelesen, so entspricht dies der Hardware-Seriennummer-Kennung ‘AB 0105’. Dieser Wert muss der aufgeklebten Seriennummer des Moduls entsprechen.

8.8.18 Synchronous Counter Overflow Value (1019_h)

INDEX	**1019_h**
Name	*Synchron_Counter_Overflow*
Datentyp	unsigned 8
Default-Wert	0

Dieses Objekt bestimmt den höchsten Wert, den der SYNC-Counter erreichen kann.

Der Wertebereich des Objektes ist in der folgenden Tabelle beschrieben:

Wert	Bedeutung
0	Die SYNC-Nachricht wird als CAN-Nachricht mit der Länge ‘0’ gesendet.
1	reserviert
2...240	Die SYNC-Nachricht wird als CAN-Nachricht mit der Länge ‘1’ gesendet. Das erste Byte der SYNC-Nachricht enthält den aktuellen Wert des SYNC-Counters.
241...255	reserviert

8.8.19 Verify Configuration (1020_h)

INDEX	**1020_h**
Name	*verify configuration*
Datentyp	unsigned 32
Default-Wert	No

In diesem Objekt können Datum und Tageszeit der letzten Konfiguration abgelegt werden, um zu einem späteren Zeitpunkt prüfen zu können, ob die gespeicherte Konfiguration der erwarteten entspricht.

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
1020_h	0	*no_of_entries*	2	2	unsigned 8	ro
	1	*configuration_date*	$0...FFFFFFFF_h$	0	unsigned 32	rw
	2	*configuration_time*	$0...FFFFFFFF_h$	0	unsigned 32	rw

Bedeutung der Variablen:

configuration_date Datum der letzten Konfiguration des Moduls, angegeben in Tagen seit dem 01.01.1984.

configuration_time Zeit in ms seit Mitternacht am Tag der letzten Konfiguration.

8.8.20 Error Behaviour Object (1029_h)

INDEX	**1029_h**
Name	*error behaviour object*
Datentyp	unsigned 8
Default-Wert	No

Tritt ein Error-Event ein (z.B. Heartbeat-Error), so wechselt das Modul in den Zustand, der in der Variablen *communication_error* oder *output_error* definiert ist.

Index	Sub-Index	Beschreibung	Wertebereich	Default-Wert	Datentyp	Zugriff
1029_h	0	*no_of_error_classes*	4	4	unsigned 8	ro
	1	*communication_error*	0...2	0	unsigned 8	rw
	2	*output_error*	0...2	0	unsigned 8	rw
	3	*input_error*	0...2	0	unsigned 8	rw
	4	*manufacturer_error*	0...2	0	unsigned 8	rw

Bedeutung der Variablen:

Variable	Bedeutung
no_of_error_classes	Anzahl der Fehler-Klassen (hier immer ‘4’)
communication_error	Heartbeat/Lifeguard-Fehler und *Bus off*
output_error	wird hier nicht verwendet
input_error	wird hier nicht verwendet
manufacturer_error	wird hier nicht verwendet

Das Modul wechselt beim Auftreten eines Fehlers in den jeweils angegebenen Modus.

Wert der Variablen	Modus, in den das Modul im Fehlerfall wechselt
0	pre-operational (nur wenn der aktuelle Zustand = operational)
1	no state change
2	stopped

8.8.21 NMT Startup ($1F80_h$)

INDEX	**$1F80_h$**
Name	*NMT startup*
Datentyp	unsigned 32
Default-Wert	2

Dieses Objekt ist ab Firmware Version 1.02 implementiert

Um in Umgebungen, in denen kein NMT-Master verfügbar ist, CANopen-Nodes starten zu können, ist der NMT Startup implementiert worden.

Über den NMT-Startup kann der Autostart eines CANopen-Nodes eingeschaltet oder ausgeschaltet werden. Weitere Funktionen des Parameters *NMT startup* werden zur Zeit nicht unterstützt.

Der Wertebereich des Objektes ist in der folgenden Tabelle beschrieben:

Wert	Bedeutung
$0000\ 0002_h$	Autostart disabled (default)
$0000\ 0008_h$	Autostart enabled
alle anderen Werte	reserviert

8.8.22 Self Starting Nodes Timing Parameters ($1F91_h$)

INDEX	**$1F91_h$**
Name	*Self starting nodes timing parameters*
Datentyp	unsigned 16

Dieses Objekt ist ab Firmware Version 1.02 implementiert

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
$1F91_h$	0	*number_of_entries*	1	1	unsigned 8	ro
	1	*NMT master detection timeout*	0...$FFFF_h$	64_h	unsigned 16	rw

Sub-Index 1 dieses Objektes enthält die Verzögerungszeit in [ms] zwischen dem Übergang von “preoperational” > “operational”. In der Default-Einstellung sind dies 100 ms.

Die Sub-Indizes 2 und 3 dieses Objektes werden nicht unterstützt.

8.8.23 RPDO Communication Parameter 1400_h, 1402_h

Mit dem Objekt '**Receive PDO Communication Parameter 1400_h, 1402_h**' werden die Eigenschaften eines Empfangs-PDOs (RPDO) definiert. Das Modul nutzt maximal **zwei** RPDOs.

Die Anzahl der RPDOs ist fest. Es wird immer die maximale Anzahl der RPDOs 'gemapped'. Über die Anzahl der gesendeten Bytes legt der Anwender fest, wie viele Bytes ausgewertet werden sollen.

INDEX	**1400_h**	**1402_h**
Name	receive PDO1 communication parameter	receive PDO3 communication parameter
Datentyp	PDOCommPar	PDOCommPar

Index	Sub-Index	PDO-mappable	Save to EEPROM	Zugriff	Datentyp	Default-Wert	Beschreibung
1400_h	0	no	no	ro	unsigned 8	6	number of entries
	1	no	yes	rw	unsigned 32	200_h + Node-ID	COB-ID used by PDO1
	2	no	no	ro	unsigned 8	FF_h	transmission type
	3	no	no	ro	unsigned 16	0	inhibit time
	4	no	no	rw	usigned 8	0	compatibility entry
	5	no	no	rw	unsigned 16	0	event-timer
	6	no	no	rw	unsigned 8	0	SYNC start value
1402_h	0	no	no	ro	unsigned 8	6	number of entries
	1	no	yes	rw	unsigned 32	400_h + Node-ID	COB-ID used by PDO3
	2	no	no	ro	unsigned 8	FF_h	transmission type
	3	no	no	ro	unsigned 16	0	inhibit time
	4	no	no	rw	unsigned 8	0	compatibility entry
	5	no	no	rw	unsigned 16	0	event-timer
	6	no	no	rw	unsigned 8	0	SYNC start value

Wertebereich siehe [1], Tabelle 10, 11.

Beispiel: Definition eines neuen Receive PDOs mit Index ‘1400_h’ auf dem CAN-Identifier’ $035A_h$’

Zu übertragen ist die hexadezimale Bytefolge auf ID = 600 + Node-ID.

Byte	Name	Wert	Inhalt
0	Command	23_h	Write Request
1	Index Low	0	Neues PDO 1400_h in Intel-Notation
2	Index High	14_h	
3	Sub-Index	1	COB-ID definieren
4	Data 1	$5A_h$	COB-ID $035A_h$ in Intel-Notation
5	Data 2	3	
6	Data 3	0	
7	Data 4	0	

8.8.24 RPDO Mapping Parameter 1600_h, 1602_h

Mit dem Objekt '**Receive PDO Mapping Parameter 1600_h, 1602_h**' wird die Zuordnung der Empfangsdaten zu den RPDOs festgelegt.

INDEX	**1600_h**	**1602_h**
Name	receive PDO1 mapping	receive PDO3 mapping
Datentyp	PDO Mapping	PDO Mapping

Die folgende Tabelle zeigt die Belegung der Receive PDO Mapping Parameter für die Default-Konfiguration:

Index	Sub-Index	PDO-mappable	Save to EEPROM	Zugriff	Datentyp	Default-Wert	Beschreibung
1600_h	0	no	no	ro	unsigned 8	8	number of entries
	1	no	no	ro	CANopen Map Struct	$2880\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	2	no	no	ro		$2880\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	3	no	no	ro		$2880\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	4	no	no	ro		$2880\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	5	no	no	ro		$2880\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	6	no	no	ro		$2880\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	7	no	no	ro		$2880\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	8	no	no	ro		$2880\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
1602_h	0	no	no	ro	unsigned 8	8	number of entries
	1	no	no	ro	CANopen Map Struct	$2980\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	2	no	no	ro		$2980\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	3	no	no	ro		$2980\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	4	no	no	ro		$2980\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	5	no	no	ro		$2980\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	6	no	no	ro		$2980\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	7	no	no	ro		$2980\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8
	8	no	no	ro		$2980\ 00\ 08_h$	Daten CAN -> seriell, Byte 1...8

Wertebereich siehe [1], Tabelle 10, 11.

Hinweis:
Die im PDO-Mapping eingetragenen $288x_h$- bzw. $298x_h$-Objekte sind nicht über SDO-Zugriffe erreichbar. Sie dienen hier nur als 'Platzhalter' um der Struktur der Objekte 1600_h, 1602_h, $1A00_h$ und $1A02_h$ zu genügen.

8.8.25 TPDO Communication Parameter 1800_h, 1802_h

Mit diesem Objekt werden die Eigenschaften eines Sende-PDO definiert.

INDEX	**1800_h**	**1802_h**
Name	transmit PDO1 communication parameter	transmit PDO3 communication parameter
Datentyp	PDOCommPar	PDOCommPar

Index	Sub Index	PDO-mappable	Save to EEPROM	Zugriff	Datentyp	Default-Wert	Beschreibung
1800_h	0	no	no	ro	unsigned 8	6	number of entries
	1	no	yes	rw	unsigned 32	180_h + Node-ID	COB-ID used by PDO1
	2	no	yes	rw	unsigned 8	FF_h	transmission type
	3	no	no	rw	unsigned 16	0	inhibit time
	4	no	no	rw	unsigned 8	0	reserved
	5	no	yes	rw	unsigned 16	0	event timer
	6	no	no	rw	unsigned 8	0	SYNC start value
1802_h	0	no	no	ro	unsigned 8	6	number of entries
	1	no	yes	rw	unsigned 32	380_h + Node-ID	COB-ID used by PDO3
	2	no	yes	rw	unsigned 8	FF_h	transmission type
	3	no	no	rw	unsigned 16	0	inhibit time
	4	no	no	rw	unsigned 8	0	reserved
	5	no	yes	rw	unsigned 16	0	event timer
	6	no	no	rw	unsigned 8	0	SYNC start value

Wertebereich siehe [1], Tabelle 10, 11.

Beispiel: Definition eines neuen Transmit PDOs mit Index ‘1800_h’ auf dem CAN-Identifier’ $047F_h$’

Zu übertragen ist die hexadezimale Bytefolge auf ID = 600_h + Node-ID.

Byte	Name	Wert	Inhalt
0	Command	23_h	Write Request
1	Index Low	0	Neues PDO 1800_h in Intel-Notation
2	Index High	18_h	
3	Sub-Index	1	COB definieren
4	Data 1	$7F_h$	COB $047F_h$ in Intel-Notation
5	Data 2	4	
6	Data 3	0	
7	Data 4	0	

8.8.26 TPDO Mapping Parameter $1A00_h$, $1A02_h$

Mit dem Objekt '**Transmit PDO Mapping Parameter $1A00_h$, $1A02_h$**' kann die Zuordnung der Sendedaten zu den TPDOs verändert werden.

INDEX	**$1A00_h$**	**$1A02_h$**
Name	transmit PDO1 mapping	transmit PDO3 mapping
Datentyp	PDO Mapping	PDO Mapping

Die folgende Tabelle zeigt die Belegung der Transmit PDO Mapping Parameter für die Default-Konfiguration:

Index	Sub-Index	PDO-mappable	Save to EEPROM	Zugriff	Datentyp	Default-Wert	Bescheibung
$1A00_h$	0	no	no	ro	unsigned 8	8	number of entries
	1	no	no	ro	CANopen Map Struct	$2881\ 00\ 08_h$	Daten seriell -> CAN, Byte 0...7
	2	no	no	ro		$2881\ 00\ 08_h$	Daten seriell -> CAN, Byte 0...7
	3	no	no	ro		$2881\ 00\ 08_h$	Daten seriell -> CAN, Byte 0...7
	4	no	no	ro		$2881\ 00\ 08_h$	Daten seriell -> CAN, Byte 0...7
	5	no	no	ro		$2881\ 00\ 08_h$	Daten seriell -> CAN, Byte 0...7
	6	no	no	ro		$2881\ 00\ 08_h$	Daten seriell -> CAN, Byte 0...7
	7	no	no	ro		$2881\ 00\ 08_h$	Daten seriell -> CAN, Byte 0...7
	8	no	no	ro		$2881\ 00\ 08_h$	Daten seriell -> CAN, Byte 0...7
$1A02_h$	0	no	no	ro	unsigned 8	8	number of entries
	1	no	no	ro	CANopen Map Struct	$2981\ 00\ 08_h$	Daten seriell -> CAN, Byte 0...7
	2	no	no	ro		$2981\ 00\ 08_h$	Daten seriell -> CAN, Byte 0...7
	3	no	no	ro		$2981\ 00\ 08_h$	Daten seriell -> CAN, Byte 0...7
	4	no	no	ro		$2981\ 00\ 08_h$	Daten seriell -> CAN, Byte 0...7
	5	no	no	ro		2981 00 08	Daten seriell -> CAN, Byte 0...7
	6	no	no	ro		2981 00 08	Daten seriell -> CAN, Byte 0...7
	7	no	no	ro		2981 00 08	Daten seriell -> CAN, Byte 0...7
	8	no	no	ro		2981 00 08	Daten seriell -> CAN, Byte 0...7

Wertebereich siehe [1], Tabelle 10, 11.

Hinweis:
Die im PDO-Mapping eingetragenen $288x_h$- bzw. $298x_h$-Objekte sind nicht über SDO-Zugriffe erreichbar. Sie dienen hier nur als 'Platzhalter' um der Struktur der Objekte 1600_h, 1602_h, $1A00_h$ und $1A02_h$ zu genügen.

8.9 Übertragungsarten

8.9.1 Unterstützte Übertragungsarten nach DSP-301, Tabelle 55

Transmission Type	PDO transmission					Supported by CAN-CBX-COM2
	cyclic	acyclic	synchronous	asynchro-nous	RTR	
0		X	X			NO
1...240	X		X			NO
241...251	reserved					NO
252			X		X	NO
253				X	X	NO
254				X	X	NO
255				X	X	**YES**

Das CAN-CBX-COM2-Modul unterstützt nur den Transmission-Type 255. Es sendet Tx-Frames in Abhängigkeit von den Parametern *MinChar* und *MaxChar*, *TxByRxTimeout* und nach dem Empfang von RTRs.

8.10 Datentransfer und PDO-Belegung

8.10.1 Funktionsweise der lokalen Firmware

Die Pufferung der seriellen Daten zwischen dem CAN-Bus und den seriellen Schnittstellen erfolgt in beiden Datenrichtungen über einen 256 Byte großen Ringspeicher. Die Daten, die zuerst eintreffen, werden auch zuerst wieder gesendet (FIFO).

Ist der Ringspeicher gefüllt und treffen noch weitere Daten ein, gehen Daten verloren. Aus diesem Grund müssen die Übertragungsraten des CAN-Bus und der seriellen Schnittstellen aufeinander abgestimmt werden.

8.10.1.1 Datentransfer CAN -> serielle Schnittstelle

Die Menge der Daten des CAN-Bus, die pro Zeit im Ringspeicher abgelegt werden, muss kleiner sein als die über die serielle Schnittstelle gesendete Datenmenge, da sonst der Ringspeicher überlaufen kann.

Die Anzahl der eintreffenden Daten pro Zeit hängt von der Anzahl der übertragenen Daten-Bytes, der Häufigkeit der Sendungen, der Bitrate des CAN-Bus und der Auslastung des CAN-Bus ab.

Treten während des Betriebs Datenverluste auf, dann sollten die Pausen zwischen den Sendungen vergrößert, oder/und die Anzahl der übertragenen Bytes verringert werden.

Die Daten werden vom CAN-Bus über den RPDO1 zum Modul übertragen. Die Anzahl der übertragenen Daten-Bytes kann bis zu 8 Bytes betragen und frei gewählt werden.

Der Datentransfer vom CAN-Bus zur seriellen Schnittstelle hat prinzipiell folgenden zeitlichen Ablauf:

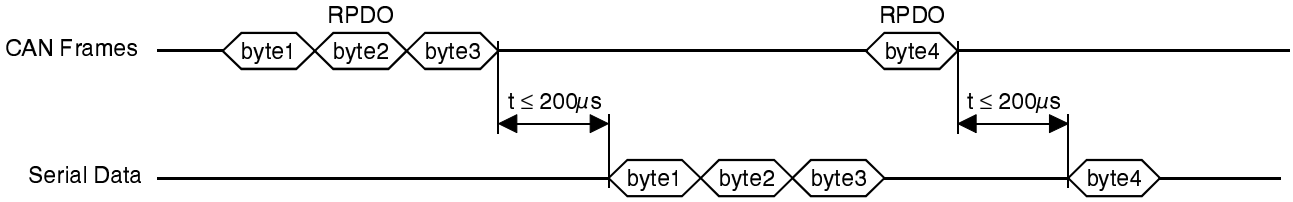


Abb. 24: Datentransfer CAN-Bus -> serielle Schnittstelle (Parameter *TxMinCharStart* = 0)

Das Modul bietet außerdem einen RTR-Handshake, bei dem das CAN-CBX-COM2-Modul per RTR dem CAN-Partner mitteilt, das sein lokaler Ringspeicher noch Daten aufnehmen kann. Wird kein RTR gesandt, ist der Ringspeicher gefüllt (siehe Seite 106).

Die acht Datenbytes der RPDOs sind wie folgt belegt:

RPDO	Daten-Bytes							
	0	1	2	3	4	5	6	7
RPDO1	1 ... 8 Byte Daten CAN -> seriell							

Tabelle 15: Belegung der Datenbytes des RPDOs mit seriellen Daten

8.10.1.2 Datentransfer serielle Schnittstelle -> CAN

Die Menge der Daten der seriellen Schnittstelle, die pro Zeit im Ringspeicher abgelegt werden, muss kleiner sein als die über den CAN-Bus in der aktuellen Anwendung übertragbare Datenmenge.

Der CAN-Bus schränkt den Datenfluss über die Belastung des Busses (Priorität), die CAN-Bus-Bitrate, die Häufigkeit der Sendungen und die Anzahl der übertragenen Bytes ein.

Das Modul bietet über die Parameter *Inhibit-Time*, *MaxChar* und *MinChar* (siehe Objekte 2800_h, 2900_h) Möglichkeiten, die letzten beiden Faktoren zu beeinflussen.

Mit *Inhibit-Time* wird die Wartezeit zwischen zwei Sendungen auf dem CAN-Bus festgelegt.

MinChar und *MaxChar* legen die minimale und die maximale Anzahl der Daten-Bytes fest, die innerhalb eines CAN-Frames auf dem CAN-Bus gesendet werden sollen (siehe Seite 97).

Die Daten werden vom Modul über den TPDO1 auf dem CAN-Bus gesendet.

Die acht Datenbytes der TPDOs sind wie folgt belegt:

TPDO	Datenbytes							
	0	1	2	3	4	5	6	7
TPDO1	1 ... 8 Byte Daten seriell -> CAN							

Tabelle 16: Belegung der Datenbytes des TPDOs mit seriellen Daten

8.11 Device Profile Area

8.11.1 Parameter-Übersicht 6000_h...$9FFF_h$

Das Modul unterstützt keine Objekte im Bereich 6000_h...$9FFF_h$

8.12 Manufacturer Specific Profile Area

8.12.1 Parameter-Übersicht 2800_h - 2981_h

Index	Objekt	Inhalt von Sub-Index 0	Bemerkung
2800_h	Serielle Kommunikationsparameter Schnittstelle COM A	10_h	-
2810_h	Cyclic_String COM A	$0...F_h$	zyklisch sendbare Zeichenketten
2880_h	Dummy Rx Objekt COM A	-	Platzhalter für serielle Daten COM A
2881_h	Dummy Tx Objekt COM A	-	Platzhalter für serielle Daten COM A
2900_h	Serielle Kommunikationsparameter Schnittstelle COM B	10_h	-
2910_h	Cyclic_String COM B	$0...F_h$	zyklisch sendbare Zeichenketten
2980_h	Dummy Rx Objekt COM B	-	Platzhalter für serielle Daten COM B
2981_h	Dummy Tx Objekt COM B	-	Platzhalter für serielle Daten COM B

Tabelle 17: Übersicht der Manufacturer-Specific Parameter

8.12.2 Serielle Kommunikationsparameter 2800_h, 2900_h

Die Seriellen Kommunikationsparameter von Objekt 2800_h und Objekt 2900_h werden in diesem Kapitel gemeinsam beschrieben, da sie identisch aufgebaut sind. Die Kommunikationsparameter von Objekt 2800_h werden dabei für die Schnittstelle COM A genutzt und die Kommunikationsparameter von Objekt 2900_h für die Schnittstelle COM B.

8.12.2.1 Übersicht

COM A: 2800_h
COM B: 2900_h

Index	Sub-index	Beschreibung	Wertebereich	Default-Wert	Datentyp	Zugriff	Save to EEPROM	siehe Seite
2800_h/ 2900_h	0	*Anzahl der Einträge*	10_h	10_h	unsigned 8	ro	no	-
	1	*Inhibit-Time*	0 ...FF_h	14_h	unsigned 8	rw	yes	96
	2	*MaxChar_MinChar*	0...FF_h	11_h	unsigned 8	rw	yes	97
	3	*Serial_Baudrate*	0...FF_h	22_h (9600 Baud)	unsigned 8	rw	yes	99
	4	*Serial_Mode*	0...FF_h	73_h	unsigned 8	rw	yes	101
	5	*TxMinCharStart*	0...FF_h	0	unsigned 8	rw	yes	105
	6	*Protocol*	0...63_h	0	unsigned 8	rw	yes	106
	7	*Handshake_Off*	0...FF_h	$F6_h$	unsigned 8	rw	yes	108
	8	*Handshake_On*	10...FF_h	A_h	unsigned 8	rw	yes	108
	9	*TxByRxTimeout*	0...FF_h	0	unsigned 8	rw	yes	110
	A_h	*End_Char_1*	0...FF_h	D_h	unsigned 8	rw	yes	111
	B_h	*End_Char_2*	0...FF_h	A_h	unsigned 8	rw	yes	111
	C_h	*Special_Baudrates*	1... $07FF_h$	34_h	unsigned 16	rw	yes	112
	D_h	*RTR_T_Cycle_Time*	0... $FFFF_h$	1388_h	unsigned 16	rw	yes	113
	E_h	*Tx_Cycle_Time*	0... $FFFF_h$	0	unsigned 16	rw	yes	114
	F_h	*Break_Code*	0... $FFFF_h$	0	unsigned 16	rw	yes	115
	10_h	*Break_Time_Factor*	0...FF_h	B_h	unsigned 8	rw	yes	116

Tabelle 18: Übersicht der Parameter von Objekt 2800_h und Objekt 2900_h

8.12.2.2 *Inhibit-Time*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***Inhibit-Time***
Object-Index, Sub-Index	2800_h, 01_h (COM A)/ 2900_h, 01_h (COM B)
Zugriff	rw
Datentyp	unsigned 8
Wertebereich	$00...FF_h$
Einheit	ms
Default-Wert	14_h = 20 ms

Beschreibung: Mit *Inhibit-Time* wird die Wartezeit zwischen zwei Sendungen von seriellen Daten auf dem CAN-Bus festgelegt.

Der Parameter *Inhibit-Time* gibt die Zeit an, die der CAN-Controller nach der letzten erfolgreichen Sendung von CAN-Bus Daten wartet, bevor er eine neue Sendung initiiert. Die Eingabe erfolgt in [ms].

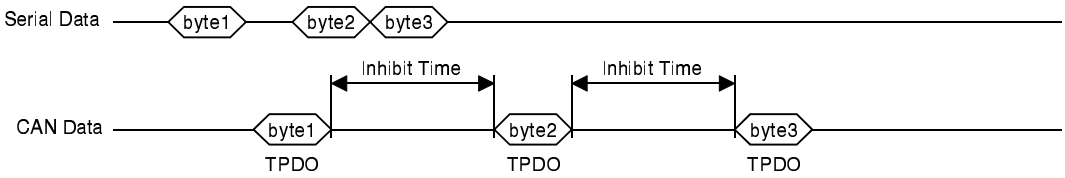


Abb. 25: Funktion des Parameters *Inhibit-Time* (Parameter *MaxChar* = 1)

8.12.2.3 *MaxChar_MinCar*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***MaxChar_MinChar***
Object-Index, Sub-Index	2800_h, 02_h (COM A)/ 2900_h, 02_h (COM B)
Zugriff	rw
Datentyp	unsigned 8
Wertebereich	$00...FF_h$
Einheit	n.a.
Default-Wert	11_h => jedes Byte wird gesendet

Beschreibung: In den Nibbles dieses Parameters wird die Anzahl der Bytes angegeben, ab der eine Sendung auf den CAN-Bus gestartet werden soll und wieviele Daten-Bytes maximal in einem CAN-Frame gesendet werden sollen. Das oberste Nibble (*MaxChar*) gibt die maximale Anzahl und das unterste Nibble (*MinChar*) die minimale Anzahl an.
Die Default-Einstellung beträgt 11_h, d.h. es wird jedes eintreffende Byte einzeln in einem Frame gesendet.

	MaxChar_MinChar	
Bit:	7 4	3 0
Parameter:	*MaxChar*	*MinChar*

Tabelle 19: Aufteilung des Parameters *MaxChar_MinChar*

Sendebedingung:
MinChar wird nur dann wie oben beschrieben ausgewertet, wenn Werte zwischen 1...8 eingegeben werden.

Bei Werten zwischen 9_h und F_h ändert sich die Funktionalität dieses Parameters: In diesem Fall wertet die lokale Software den Eintrag der Character, die über die Parameter *End_Char_1* (2800_h, 10_d / 2900_h, 10_d) oder *End_Char_2* (2800_h, 11_d / 2900_h, 11_d) festgelegt worden sind, als Sendebedingung aus.

Bei der Default-Einstellung der Parameters *End_Char_1, 2* sind dies die Steuerzeichen 'Carriage Return' ($0D_h$) oder 'Line Feed' ($0A_h$). Wird eines oder beide Zeichen erkannt, so werden die bisher von der seriellen Schnittstelle empfangenen Daten auf den CAN-Bus gesendet. Sind 8 Bytes empfangen worden, bevor die Steuerzeichen eingetroffen sind, so erfolgt die Sendung automatisch.

Für die Sendung kann außerdem noch ausgewählt werden, ob die Steuerzeichen auf dem CAN-Bus zusammen mit den Daten übertragen werden sollen oder nicht.
Die folgende Tabelle zeigt die verschiedenen Optionen:

MinChar	Sende-bedingung	Steuerzeichen -Übertragung auf CAN	Kommentar
1 ... 8	-	-	Es werden immer mindestens 1 bis 8 Bytes gesendet. Der Parameter *MaxChar* wird ebenfalls ausgewertet.
9	<*End_Char_1*>	mit <*End_Char_1*>	Daten werden gesendet, wenn <*End_Char_1*> empfangen wurde. <*End_Char_1*> wird mit übertragen.
A_h	<*End_Char_2*>	mit <*End_Char_2*>	Wie unter ‘9’, aber mit <*End_Char_2*>.
B_h	<*End_Char_1*>	ohne <*End_Char_1*>	Wie unter ‘9’, aber <*End_Char_1*> wird nicht mit übertragen.
C_h	<*End_Char_2*>	ohne <*End_Char_2*>	Wie unter ‘B’, aber mit <*End_Char_2*>.
D_h	<*End_Char_1*>	ohne <*End_Char_1*> und ohne <*End_Char_2*>	Am Ende der Nachricht kann <*End_Char_1*> und <*End_Char_2*> stehen. Die Daten werden gesendet, wenn <*End_Char_1*> empfangen wurde. Weder <*End_Char_1*> noch <*End_Char_2*> werden mit übertragen.
E_h	<*End_Char_2*>	ohne <*End_Char_1*> und ohne <*End_Char_2*>	Wie unter ‘D’, aber mit <*End_Char_2*>.
F_h	-	-	reserviert

Tabelle 20: Auswahl der Sendebedingung über den Parameter *MinChar*

Beispiele: Die aufgeführten Beispiele gelten für den Betrieb der CAN-CBX-COM2 mit den Parametern in der Default-Einstellung, d.h. *End_Char_1* = <CR>, *End_Char_2* = <LF>.

1. Für *MinChar* wurde der Wert B_h ausgewählt. Über die serielle Schnittstelle treffen die Daten ‘abcd<CR>’ ein. Auf dem CAN-Bus würden nach dem Empfang von <CR> die Daten ‘abcd’ gesendet werden.

2. Für *MinChar* wurde der Wert E_h ausgewählt. Über die serielle Schnittstelle treffen die Daten ‘abcd<CR><LF>’ ein. Auf dem CAN-Bus würden nach dem Empfang von <LF> die Daten ‘abcd’ gesendet werden.

8.12.2.4 *Serial_Baudrate*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***Serial_Baudrate***
Object-Index, Sub-Index	2800_h, 03_h (COM A)/ 2900_h, 03_h (COM B)
Zugriff	rw
Datentyp	unsigned 8
Wertebereich	$00...FF_h$
Einheit	n.a.
Default-Wert	22_h => 9600 bit/s

Beschreibung: Über den Parameter *Serial_Baudrate* ist die Rx- und die Tx-Baudrate der seriellen Schnittstellen in 14 Stufen einstellbar.

	Serial_Baudrate	
Bit:	7 4	3 0
Parameter:	*Rx-Baudrate*	*Tx-Baudrate*

Tabelle 21: Aufteilung des Parameters *Serial_Baudrate*

In 4 Bit wird jeweils festgelegt, mit welcher Baudrate auf den seriellen Schnittstellen gesendet (Tx), bzw. empfangen (Rx) werden soll. Die Standardeinstellung ist für Rx- und Tx-Baudrate 9600 KBit/s.
Die physikalisch erreichbare Baudrate wird durch die Hardware begrenzt (siehe Seite 112).

Hinweis:
Die eingestellten Rx- und Tx-Werte müssen aufgrund der verwendeten Hardware gleich sein!

Hinweis:
Über den User-Parameter *Special_Baudrates* (2800_h, 12_d / 2900_h, 12_d) lassen sich weitere Baudraten einstellen (siehe Seite 112)! Ist dies gewünscht, muss für die Parameter *Rx-Baudrate* und *Tx-Baudrate* der Wert E_h eingetragen werden.

Parameter *Rx-(Tx) Baudrate*	Standard-Baudrate [Bit/s]
00_h	38400
01_h	19200
$\underline{02}_h$	**<u>9600</u>**
03_h	4800
04_h	2400
05_h	1200
06_h	600
07_h, 08_h	300
09_h	7200
$0A_h$	14400
$0B_h$	28800
$0C_h$	57600
$0D_h$	115200
$0E_h$	*Special_ Baudrate*
$0F_h$	76800

Tabelle 22: Einstellung der Baudrate der seriellen Schnittstelle in 14 Stufen

8.12.2.5 *Serial_Mode*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***Serial_Mode***
Object-Index, Sub-Index	2800_h, 04_h (COM A)/ 2900_h, 04_h (COM B)
Zugriff	rw
Datentyp	unsigned 8
Wertebereich	$00...FF_h$
Einheit	n.a.
Default-Wert	73_h

Beschreibung: Die Bits des Parameters *Serial_Mode* sind mit folgenden Funktionen belegt:

Bit	7	6	5	4	3	2	1	0
Belegung	*RTS-Mode*	*CTS-enable*	*Stop-Bit*	*Parity-Mode*		*Parity-Type*	*Bits per Character*	
Default	0	1	1	1	0	0	1	1

Tabelle 23: Belegung des Parameters *Serial_Mode*

Die serielle Schnittstelle kann ausschließlich 8 Bits per Character verarbeiten. Außerdem wird die maximale Anzahl der seriellen Bits durch den seriellen Schnittstellen-Controller auf 11 limitiert. Dadurch ergeben sich für die Bits 4...0 die folgenden zulässigen Kombinationen:

Bit						Funktion
7, 6, 5	4	3	2	1	0	
RTS,CTS, Stop: siehe folgende Tabellen	*Parity-Mode*		*Parity-Type*	*Bits per Character*		
	0	0	$\overline{O/E}$	1	1	Tx Parität gemäß Bit ‘Parity Type’, keine Rx-Parität, 1 Stop-Bit
	0	1	$\overline{O/E}$	1	1	Tx-Parität und Rx-Parität gemäß Bit ‘Parity Type’, 1 Stop-Bit
	1	0	0	1	1	Keine Parität, evaluiere 1 oder 2 Stop-Bits
	1	0	1	1	1	Sender in Multidrop-Mode, wenn RTS = ‘1’ (siehe Seite 104)
	1	1	a	1	1	Erzwingen der Tx-Parität auf Wert ‘a’, 1 Stop-Bit

Tabelle 24: Zulässige Kombinationen der Bits 4...0

Erläuterung der Bits des Parameters *Serial_Mode*:

RTS-Mode... Für den Betrieb als RS485-Schnittstelle muss mit diesem Bit der RTS-Modem-Mode für RS485-Schnittstellen gewählt werden.

RTS-Mode-Bit	Auswertung
0	RTS on Rx (Default-Einstellung) Hardware-Handshake-Signal
1	RTS-Modem (RS485)

Tabelle 25: Auswertung des *RTS-Mode*-Bits

CTS enable... Dieses Bit ist nur von Bedeutung, wenn der RTS-Mode auf ‘0’ gesetzt ist!
Über dieses Bit wird die CTS-Funktion der seriellen Controller freigegeben.
Ist das Bit ‘0’, so wird das CTS-Signal nicht ausgewertet und der Controller sendet die vorhandenen Daten sofort (sofern der Transmitter nicht durch andere Bedingungen blockiert ist; z.B. ‘suspended’, <Xoff> empfangen, Anzahl Daten < *MinChar*).

CTS enable- Bit	CTS -Auswertung
0	CTS -Eingang wird ignoriert
1	CTS -Eingang wird ausgewertet: Hardware-Handshake (Default-Einstellung)

Tabelle 26: Auswertung des *CTS enable*-Bits

Stop Bit... Hier wird die Anzahl der Stop-Bits der seriellen Schnittstelle festgelegt:

Stop Bit	Anzahl der Stop Bits
0	1 Stop Bit
1	2 Stop Bits (Default-Einstellung)

Tabelle 27: Anzahl der Stop-Bits

Parity Mode... Mit diesen beiden Bits wird die Auswertung des Parity-Bits festgelegt:

Parity Mode-Bits		Auswertung
Bit 4	Bit 3	
0	0	Parity-Auswertung beim Empfangen von Daten und Senden des Parity-Bits
0	1	Parity-Bit wird nur gesendet
1	0	keine Parity-Auswertung, keine Parity-Sendung (Default-Einstellung)
1	1	Wert des zu sendenden Parity-Bits ist im Bit *Parity Type* eingetragen (‘forced parity’)

Tabelle 28: Parity-Auswertung

Parity Type... Die Polarität des Parity-Bits wird mit dem Parameter-Bit *Parity Type* festgelegt.

Parity Type-Bit	Polarität
0	‘gerade’ (even) (Default-Einstellung)
1	‘ungerade’ (odd)

Tabelle 29: Einstellung der Polarität

Bits per Character... Die Anzahl der Bits per Character ist auf ‘8’ festgelegt und kann nicht verändert werden. Daher müssen diese beiden Bits immer auf ‘1’ gesetzt werden!

Erläuterungen zum Multidrop-Mode:

Über die Bits des Parameters Serial Mode kann der Multidrop-Mode gewählt werden (siehe Tabelle 24).
Der Multidrop-Mode kann bei RS485-Netzwerken mit Multipoint-Struktur verwendet werden. Dazu ist das *RTS-Mode* Bit auf ‘1' zu setzen und die folgenden Bits wie in der folgenden Tabelle beschrieben.

Bit	7	6	5	4	3	2	1	0
Belegung	*RTS-Mode*	*CTS-enable*	*Stop-Bit*	*Parity-Mode*		*Parity-Type*	*Bits per Character*	
Default	1	-	-	1	0	1	1	1

Tabelle 30: Belegung des Parameters *Serial_Mode* für den Multidrop-Mode

Bei dem Netzaufbau mit Multipoit-Struktur befinden sich an einem seriellen Bus ein Master und mehrere Slaves:

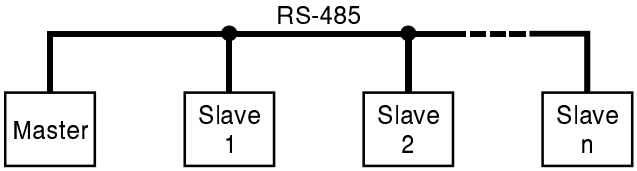


Abb. 26: Struktur eines Multipoint-Bus

Zur Unterscheidung der Slaves ist eine Adressierung notwendig. Um die Adress-information von den Daten unterscheiden zu können, wird das Paritiy-Bit im Multidrop-Mode zum ‘Adress-Bit’ umfunktioniert. Ist es ‘1’, so sind die empfangenen Daten als Adresse zu interpretieren. Ist es ‘0’ so handelt es sich um Nutzdaten.

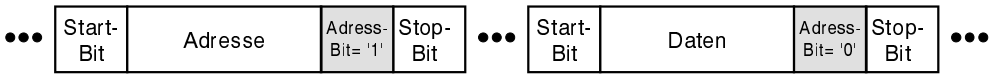


Abb. 27: Aufbau der Tx-Daten im Multidrop-Mode

8.12.2.6 *TxMinCharStart*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***TxMinCharStart***
Object-Index, Sub-Index	2800_h, 05_h (COM A)/ 2900_h, 05_h (COM B)
Zugriff	rw
Datentyp	unsigned 8
Wertebereich	$00...FF_h$
Einheit	n.a.
Default-Wert	0

Beschreibung: In diesem Parameter wird angegeben, wie viele Datenbytes mindestens im seriellen Tx-Buffer abgelegt sein müssen, bevor die Sendung der Daten auf der seriellen Schnittstelle gestartet wird (Daten CAN -> seriell).

Diese Funktion ist notwendig, wenn auf einer RS485-Schnittstelle Strings von mehr als 8 Bytes zusammenhängend gesendet werden sollen (z.B. bei Master/Slave-Protokollen).

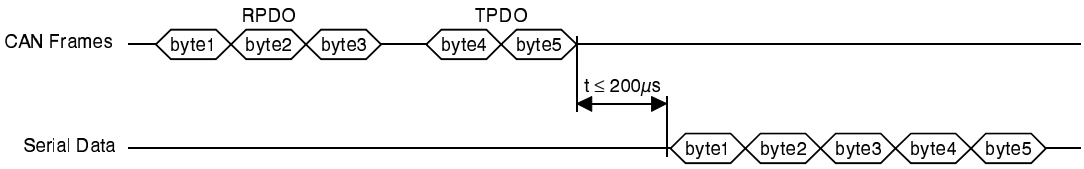


Abb. 28: Funktion des Parameters *TxMinCharStart*

8.12.2.7 *Protocol*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***Protocol***
Object-Index, Sub-Index	2800_h, 06_h (COM A)/ 2900_h, 06_h (COM B)
Zugriff	rw
Datentyp	unsigned 8
Wertebereich	$00...63_h$
Einheit	n.a.
Default-Wert	00_h

Beschreibung: Über den Parameter *Protocol* können Handshake-Funktionen des CAN-Bus und der seriellen Schnittstelle ausgewählt werden.

Bit	Name	Bedeutung
0	*RTR*	0 - kein RTR-Handshake (Default-Einstellung) 1 - Remote-Mode (RTR-Handshake)
1	*XON/XOFF*	0 - kein XON/XOFF 1 - XON/XOFF-Handshake enabled
2 ... 4	-	reserviert (immer auf ‘0’ setzen)
5	*Rx_Break*	0 - Nutzung des *Rx_Break* deaktiviert 1 - Nutzung des *Rx_Break* aktiviert
6	*Tx_Break*	0 - Nutzung des *Tx_Break* deaktiviert 1 - Nutzung des *Tx_Break* aktiviert
7	-	reserviert

Tabelle 31: Bits des Parameters *Protocol*

Bit 0: *RTR* (RTR-Handshake)

Bei diesem Handshake-Mode antwortet das CAN-CBX-COM2-Modul mit einem RTR-Frame auf die von einem CAN-Master empfangenen Rx-Daten (Daten CAN -> seriell), wenn der Platz im Buffer noch für mindestens einen weiteren CAN-Frame ausreicht. Das RTR-Frame wird auf dem Identifier gesendet, auf dem die Daten empfangen wurden (RPDO1).

Bleiben die Daten vom CAN-Master aus, so kann das Modul zyklisch RTR-

Frames senden, wenn der Parameter *RTR_T_Cycle* (2800_h, 13_d, siehe Seite 113) entsprechend gesetzt ist und der Buffer noch nicht gefüllt ist.

Empfängt die CAN-CBX-COM2 ein RTR auf TPDO1, so werden die Daten aus dem Rx-Buffer gesendet, auch wenn die Anzahl der im Buffer vorhandenen Daten geringer als die in *MinChar* definiert ist.

Bit 1: *XON/XOFF*

Dieser Mode kann alternativ zum Hardware-Handshake bei der seriellen Schnittstelle gewählt werden, wenn der Handshake über die Parameter *Handshake_On/Off* aktiviert ist.

Bit 5: *Rx_Break*

Über dieses Bit kann die Nutzung des “Rx_Break” aktiviert oder deaktiviert werden.
Das Break ist ein Nullsignal von mindestens einer Bytedauer, so dass Start- und Stop-Bit den gleichen Level haben.

Bit 6: *Tx_Break*

Über dieses Bit kann die Nutzung des “Tx_Break” aktiviert oder deaktiviert werden.
Das Break ist ein Nullsignal von mindestens einer Bytedauer, so dass Start- und Stop-Bit den gleichen Level haben.

8.12.2.8 *Handshake_On, Handshake_Off*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***Handshake_Off***	***Handshake_On***
Object-Index, Sub-Index	2800_h, 07_h (COM A)/ 2900_h, 07_h (COM B)	2800_h, 08_h (COM A)/ 2900_h, 08_h (COM B)
Zugriff	rw	rw
Datentyp	unsigned 8	unsigned 8
Wertebereich	00...FF_h	10...FF_h
Einheit	n.a.	n.a.
Default-Wert	$F6_h$	$0A_h$

Beschreibung: Über diese Parameter kann die Handshake-Funktion der seriellen Schnittstelle für die Datenrichtung seriell -> CAN beeinflusst werden (*im RTS-Modem-Mode wird diese Funktion nicht unterstützt!* Siehe Seite 102).

Der Buffer für die empfangenen seriellen Daten ist 256 Byte groß. Um zu verhindern, dass der Buffer überläuft und somit Daten verloren gehen, kann über die Handshake-Leitung RTS oder per Protokoll-Byte <Xoff> dem Sender der seriellen Daten mitgeteilt werden, daß er die Sendung unterbricht, bis wieder Raum für neue Daten im Buffer verfügbar ist.

Im Parameter ***Handshake_On*** wird die Anzahl der im Rx-Buffer empfangenen Bytes eingetragen, ab der der serielle Datenstrom unterbrochen werden soll. Dies ist in der Default-Einstellung $F6_h$, d.h. befinden sich 250 Bytes im Rx-Buffer, wird das RTS-Signal der seriellen Schnittstelle aktiviert. Ist über den Parmeter *Protocol XON/XOFF* freigegeben, so wird außerdem noch über den Software-Handshake <Xoff> die fehlende Empfangsbereitschaft signalisiert.

Im Parameter ***Handshake_Off*** wird die Anzahl der im Rx-Buffer befindlichen Bytes eingetragen, ab der der Empfang serieller Daten wieder zugelassen wird. Dies ist in der Default-Einstellung $0A_h$, d.h. befinden sich nur noch 10 Bytes im Rx-Buffer, so wird das RTS-Signal der seriellen Schnittstelle deaktiviert. Ist über den Parameter *Protocol, XON/XOFF* freigegeben, so wird außerdem noch über den Software-Handshake <Xon> die wieder vorhandene Empfangsbereitschaft signalisiert.

Der Wert des Parameters *Handshake_Off* muss immer kleiner sein als der Wert des Parameters *Handshake_On*!
Ist das Bit *RTS_Mode* = ‘0’, so führt das RTS-Signal immer den aktuellen Handshake, unabhängig von dem Bit *XON/XOFF_EN*.

Hinweis:
Der User-Paramter *Handshake_On/Off* hat keinen Einfluss auf das Handshake-Verhalten in der anderen Datenrichtung, also CAN -> seriell.

Der Sender der seriellen Schnittstelle wird abgeschaltet,
- wenn das Bit *CTS-Mode* = ‘1’ ist und das CTS-Signal inaktiv ist oder
- wenn *XON/XOFF* = ‘1’ ist und <Xoff> empfangen wurde.

8.12.2.9 *TxByRxTimeout*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***TxByRxTimeout***
Object-Index, Sub-Index	2800_h, 09_h (COM A)/ 2900_h, 09_h(COM B)
Zugriff	rw
Datentyp	unsigned 8
Wertebereich	00...FF_h
Einheit	ms
Default-Wert	0

Beschreibung: Das Senden der empfangenen seriellen Daten auf dem CAN-Bus wird normalerweise erst dann initiiert, wenn die in *MinChar* eingestellte Anzahl Bytes auf der seriellen Schnittstelle empfangen worden ist.

Bei großen Datenraten ist es sinnvoll, den Parameter *Minchar* = ‘8’ zu setzen, um möglichst viele Daten in einem CAN-Frame unterbringen zu können. Hier kann aber z.B. der Fall eintreten, daß mit der letzten Übertragung weniger Bytes als in *MinChar* definiert im Buffer der CAN-CBX-COM2 abgelegt werden. Diese Daten würden erst mit der Wiederaufnahme von Übertragungen gesendet werden. Damit diese Daten nicht für unbestimmte Zeit im Buffer verbleiben, ist der Parameter *TxByRxTimeout (Transfer_By_Rx-Timeout)* eingeführt worden.

Wenn sich mindestens ein Datenbyte im seriellen Rx-Buffer befindet und für die übergebene Timeout-Zeit keine weiteren Daten auf der seriellen Schnittstelle eintreffen, dann werden die Daten des Rx-Buffers auf dem CAN-Bus gesendet, auch wenn die Anzahl der Bytes noch kleiner ist als in *MinChar* definiert.

In der Default-Einstellung ist *TxByRxTimeout* = ‘0’ und somit diese Sendefunktion ausgeschaltet. Der Wert für *TxByRxTimeout* wird in [ms] angegeben. Zulässig sind Werte bis max. 255 ms.

8.12.2.10 *End_Char_1, End_Char_2*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***End_Char_1***	***End_Char_2***
Object-Index, Sub-Index	2800_h, $0A_h$ (COM A)/ 2900_h, $0A_h$ (COM B)	2800_h, $0B_h$ (COM A)/ 2900_h, $0B_h$ (COM B)
Zugriff	rw	rw
Datentyp	unsigned 8	unsigned 8
Wertebereich	$00...FF_h$	$00...FF_h$
Einheit	n.a.	n.a.
Default-Wert	$0D_h$	$0A_h$

Beschreibung: Über den bereits beschriebenen User-Parameter *MinChar* kann das Senden der empfangenen seriellen Daten auf dem CAN-Bus nach dem Empfang der Endebedingung *End_Char_1* oder *End_Char_2* gestartet werden.

Die Zeichen, deren Empfang als Endebedingung ausgewertet wird, lassen sich über die Parameter *End_Char_1* und *End_Char_2* vorgeben, d.h. es können andere Zeichen als <CR> oder <LF> gewählt werden. In der Default-Einstellung ist *End_Char_1* mit <CR>, also $0D_h$ und *End_Char_2* mit <LF>, also $0A_h$ belegt.

8.12.2.11 *Special_Baudrates*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***Special_Baudrates***
Object-Index, Sub-Index	2800_h, $0C_h$ (COM A)/ 2900_h, $0C_h$ (COM B)
Zugriff	rw
Datentyp	unsigned 16
Wertebereich	$0001...07FF_h$
Einheit	n.a.
Default-Wert	0034_h

Beschreibung: Die Baudrate der seriellen Schnittstelle läßt sich über den Parameter *Serial-Baudrate* in 14 Stufen einstellen. Weitere Baudraten können Sie über den Parameter *Special_Baudrates* einstellen, wenn die *Serial Baudrate* auf EE_h steht.

In *Special_Baudrates* wird eine ganze Zahl ‘*N* ‘ übergeben, deren Wertebereich von

$N = 1....07FF_h$, also $1....2047_d$ liegt.

Die möglichen einstellbaren Baudraten errechnen sich nach der folgenden Formel:

$$\text{Baudrate [Baud]} = \frac{8 \cdot 10^6}{16 \cdot N} = \frac{500\,000}{N}$$

Es können also Baudraten zwischen 244 Baud und 500 000 Baud eingestellt werden. Da für *N* nur ganze Zahlen einzusetzen sind, können nicht alle Standard-Baudraten exakt erreicht werden !

8.12.2.12 *RTR_T_Cycle_Time*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***RTR_T_Cycle_Time***
Object-Index, Sub-Index	2800_h, $0D_h$ (COM A)/ 2900_h, $0D_h$ (COM B)
Zugriff	rw
Datentyp	unsigned 16
Wertebereich	0000...$FFFF_h$
Einheit	ms
Default-Wert	1388_h => 5000 ms

Beschreibung: Die *RTR_T_Cycle_Time* ist nur von Bedeutung, wenn das CAN-CBX-COM2-Modul im RTR-Handshake-Mode betrieben wird. Der RTR-Handshake-Mode wird über das Bit *RTR* im Parameter *Protocol* (siehe Seite 106) aktiviert.

Bei dem RTR-Handshake-Mode antwortet das CAN-CBX-COM2-Modul mit einem RTR-Frame auf die von einem CAN-Master empfangenen Rx-Daten (Daten CAN -> seriell), wenn der Platz im Buffer noch für mindestens einen weiteren CAN-Frame ausreicht. Das RTR-Frame wird auf dem Identifier gesendet, auf dem die Daten empfangen wurden (RPDO1).

Bleiben die Daten vom CAN-Master aus und ist der Buffer des CAN-CBX-COM2-Moduls leer, so sendet das Modul zyklisch RTR-Frames, wenn der Parameter *RTR_T_Cycle* auf Werte größer ‘0’ gesetzt ist.

8.12.2.13 *Tx_Cycle_Time*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***Tx_Cycle_Time***
Object-Index, Sub-Index	2800_h, $0E_h$ (COM A) / 2900_h, $0E_h$ (COM B)
Zugriff	rw
Datentyp	unsigned 16
Wertebereich	$0000...FFFF_h$
Einheit	ms
Default-Wert	0

Beschreibung: Für wiederkehrende Sendeaufträge gleichen Inhalts, wie sie z.B. für die Initialisierung von Messgeräten am seriellen Bus benötigt werden, bietet die CAN-CBX-COM2 eine Funktion zum zyklischen Senden einer programmierbaren Zeichenkette. Die Funktion soll den CAN-Bus entlasten, da die Zeichenkette nur einmal an die CAN-CBX-COM2 gesandt werden muss.

Im Parameter *Tx_Cycle_Time* wird die Zykluszeit übergeben, mit der die Zeichenkette auf der seriellen Schnittstelle gesendet wird. Ist *Tx_Cycle_Time* = 0, so erfolgt keine Sendung.
Der Inhalt der Zeichenkette wird über den Parameter *Tx_Cycle_String* (Objekt 2810_h / 2910_h, siehe Seite 117) übergeben.

8.12.2.14 *Break_Code*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***Break_Code***
Object-Index, Sub-Index	2800_h, $0F_h$ (COM A)/ 2900_h, $0F_h$ (COM B)
Zugriff	rw
Datentyp	unsigned 16
Wertebereich	$0000...FFFF_h$
Default-Wert	0

Beschreibung: Der Parameter *Break_Code* enthält die Kodierung für Break.

Break_Code	
Tx_Break_Code 15 8	*Rx_Break_Code* 7 0

MSB LSB

8.12.2.15 *Break_Time_Factor*

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Parameter-Name	***Break_Time_Factor***
Object-Index, Sub-Index	2800_h, 10_h (COM A)/ 2900_h, 10_h (COM B)
Zugriff	rw
Datentyp	unsigned 8
Wertebereich	$00...FF_h$
Einheit	Bitzeiten
Default-Wert	11

Beschreibung: Der Parameter *Break_Time_Factor* gibt die Dauer des Break-Signals in Bit-Zeiten.

Beispiel: Baudrate: 9600 Baud
Break_Time_Factor: 11

$$\text{Break Time} = 11 \times \frac{1}{9600\,Baud} = 1{,}146\,ms$$

8.12.3 *Tx_Cycle_String* 2810_h, 2910_h

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Die Parameter von Objekt 2810_h und Objekt 2910_h werden in diesem Kapitel gemeinsam beschrieben, da sie identisch aufgebaut sind. Die *Tx_Cycle_String* Parameter von Objekt 2810_h werden dabei für die Schnittstelle COM A genutzt und die *Tx_Cycle_String* Parameter von Objekt 2910_h für die Schnittstelle COM B.

Index	Sub-Index	Beschreibung	Wertebereich	Default-Wert	Datentyp	Zugriff	Save to EEPROM
2810_h / 2910_h	0	*Länge des Strings*	0...F_h	-	unsigned 8	rw	yes
	1	*string_1*	0...FF_h	0	unsigned 8	rw	yes
	:	:	:	:	:	:	:
	F_h	*string_15*	0...FF_h	0	unsigned 8	rw	yes

In diese Parameter können Zeichenketten programmiert werden, die mit der Zykluszeit von *Tx_Cycle_Time* (Object 2800_h, 14_d, COM A/ 2900_h, 14_d, COM B) zyklisch auf dem seriellen Bus gesendet werden (siehe auch Seite 114).

Jede Zeichenkette kann bis zu 15 Bytes enthalten.

8.12.4 *Dummy_Rx_Object* 2880_h, 2980_h

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Die Parameter von Objekt 2880_h und Objekt 2980_h werden in diesem Kapitel gemeinsam beschrieben, da sie identisch aufgebaut sind. Die Dummy-Rx-Objekte dienen als Platzhalter für die seriellen Daten in den RPDOs.
Das *dummy_rx_object* (Objekt 2880_h) wird dabei für die Schnittstelle COM A genutzt und das *dummy_rx_object* (Objekt 2980_h) für die Schnittstelle COM B.

Index	Sub-Index	Beschreibung	Wertebereich	Default-Wert	Data Typ	Zugriff	Save to EEPROM
2880_h / 2980_h	0	*dummy_rx_object*	-	-	unsigned 8	rw	yes

8.12.5 *Dummy_Tx_Object* 2881_h, 2981_h

Eine Übersicht der Kommunikationsparameter 2800_h und 2900_h erhalten Sie auf Seite 95.

Die Parameter von Objekt 2881_h und Objekt 2981_h werden in diesem Kapitel gemeinsam beschrieben, da sie identisch aufgebaut sind.
Die Dummy-Tx-Objekte dienen als Platzhalter für die seriellen Daten in den TPDOs.
Das *dummy_tx_object* (Objekt 2881_h) wird dabei für die Schnittstelle COM A genutzt und das *dummy_rx_object* (Objekt 2981_h) für die Schnittstelle COM B.

Index	Sub-Index	Beschreibung	Wertebereich	Default-Wert	Datentyp	Zugriff	Save to EEPROM
2881_h / 2981_h	0	*dummy_tx_object*	-	-	unsigned 8	rw	yes

8.13 Programm Download über CANopen-Objekte

Die im folgenden beschriebenen Objekte werden benötigt, wenn die Firmware des CAN-CBX-COM2 ausgetauscht werden soll.

Achtung:
Das Firmware-Update sollte nur durch erfahrene Anwender durchgeführt werden!
Unsachgemäße Durchführung des Updates kann zum ungewollten Löschen des Speichers und zum Verlust der Firmware führen. Das Modul kann dann nicht weiter betrieben werden!

Hinweis:
Für ein Firmware-Update bietet esd das Programm CANfirmdown. Bitte wenden Sie sich dazu an unseren Support.

Im normalen Betrieb kann auf die Objekte $1F50_h$ und $1F52_h$ nicht zugegriffen werden.
Das Objekt $1F51_h$ steht auch im normalen Betrieb zur Verfügung.
Weitere Informationen zu den Objekten und zum Firmware-Update entnehmen Sie bitte [2].

Index	Sub-index	Beschreibung	Datentyp	Zugriff
$1F51_h$	1	Boot-Loader: FLASH-Kommando	unsigned 8	rw
$1F52_h$	0,1,2	Boot-Loader: Firmware-Datum	unsigned 32	ro

8.13.1 Download-Steuerung über Objekt $1F51_h$

INDEX	**$1F51_h$**
Name	Program Control

Index	Sub-Index	Beschreibung	Wertebereich	Default-Wert	Datentyp	Zugriff
$1F51_h$	0	*number of programs*	1	1	unsigned 8	ro
	1	*program_number_1*	0...2	1	unsigned 8	rw

Hinweis:
Der Wertebereich dieses Objektes in der Implementierung für die CAN-CBX-COM2 weicht vom Wertebereich, der in [2] angegeben ist, ab.
Weitere Informationen zum Objekt $1F51_h$ und zum Firmware-Update entnehmen Sie bitte [2].

8.13.2 Download-Steuerung über Objekt $1F52_h$

INDEX	**$1F52_h$**
Name	Verify Application Software

In diesem Objekt werden Datum und Tageszeit der Erstellung der Applikationssoftware abgelegt, um zu einem späteren Zeitpunkt prüfen zu können, ob die gespeicherte Software der erwarteten entspricht.

Index	Sub-index	Beschreibung	Wertebereich	Default-Wert	Datentyp	Zugriff
$1F52_h$	0	*number of entries*	2	2	unsigned 8	ro
	1	*application_software_date*	0...FFFF $FFFF_h$	-	unsigned 32	ro
	2	*application_software_time*	0...FFFF $FFFF_h$	-	unsigned 32	ro

Bedeutung der Variablen:

application_software_date Datum der letzten Applikationssoftware des Moduls, angegeben in Tagen seit dem 01.01.1984.
Beispiel: 0000 $196D_h$ -> 26.10.2001

application_software_time Zeit in ms seit Mitternacht am Tag der Erstellung der letzten Applikationssoftware.
Beispiel: $02A0AB8E_h$ -> ca. 12:14 Uhr

9. Referenzen

[1] CiA 301 CANopen Application Layer and Communication Profile V4.2.0 (02.2011)
CAN in Automation (CiA) e. V.

[2] CiA Draft Standard Proposal 302 CANopen Additional Application Layer Functions
Part 3: Configuration and program download V4.1.0 (04.2010)
CAN in Automation (CiA) e. V.

[3] CiA 303 Recommendation
Part 3: Indicator specification V1.4.0 (04.2012)
CAN in Automation (CiA) e. V.

[4] Phoenix Contact GmbH & Co. KG, Blomberg.
Technische Daten entnommen von der Phoenix Contact Web-Seite:
https://www.phoenixcontact.com/online/portal/de;
Leiterplattensteckverbinder - FKCT-2,5/4-ST KMGY - 1921900, Abgerufen am 30.05.2013

[5] Phoenix Contact GmbH & Co. KG, Blomberg.,
Technische Daten entnommen von der Phoenix Contact Web-Seite:
https://www.phoenixcontact.com/online/portal/de;
Leiterplattensteckverbinder - FK-MCP 1,5/ 5-STF-3,81 - 1851261, Abgerufen am 30.05.2013

[6] Phoenix Contact GmbH & Co. KG, Blomberg.,
Technische Daten entnommen von der Phoenix Contact Web-Seite:
https://www.phoenixcontact.com/online/portal/de;
Leiterplattensteckverbinder -FK-MCP- 1,5/7-ST-3,5 - 1939960, Abgerufen am 30.05.2013

10. Konformitätserklärung

EU-KONFORMITÄTSERKLÄRUNG
EU DECLARATION OF CONFORMITY

Adresse / *Address*
esd electronic system design 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-COM2 2xRS232	**C.3055.02**
CAN-CBX-COM2 1xRS232 1xRS485	**C.3055.03**

die Anforderungen der Normen *fulfills the requirements of the standards*	**EN 61000-6-2:2005,** **EN 61000-6-4:2007+A1:2011**
gemäß folgendem Prüfbericht erfüllt. *according to test certificate.*	**H-K00-0473-13**
Das Produkt entspricht damit der EU-Richtlinie „EMV" *Therefore the product corresponds to the EU Directive 'EMC'*	**2014/30/EU**
Das Produkt entspricht der EU-Richtlinie „RoHS" *The product corresponds to the EU Directive 'RoHS'*	**2011/65/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. Ramm
Funktion / *Title*	CE-Koordinator / *CE Coordinator*
Datum / *Date*	Hannover, 2014-12-09

Thorsten Ramm

Rechtsgültige Unterschrift / *authorized signature*

I:\Texte\Doku\MANUALS\CAN\CBX\CAN-CBX-COM2\Konformitaet\CAN-CBX-COM2_Konformitaetserklaerung_2014_12_09.odt

11. Bestellhinweise

Typ	Eigenschaften	Bestell-Nr.
CAN-CBX-COM2 2xRS232	CAN-CBX-COM2-Modul mit 2 seriellen RS232-Schnittstellen, incl. 1x CAN-CBX-TBUS (C.3000.01)	C.3055.02
CAN-CBX-COM2 1xRS232 1x RS485	CAN-CBX-COM2-Modul mit 1xRS232- und 1x RS485-Schnittstellen, incl. 1x CAN-CBX-TBUS (C.3000.01)	C.3055.03
Zubehör InRailBus		
CAN-CBX-TBUS 	Tragschienen-Busverbinder des CBX-InRailBus für CAN-CBX-Module, (ein Busverbinder ist im Lieferumfang des CAN-CBX-Moduls enthalten)	C.3000.01
CAN-CBX-TBUS-Connector 	Anschlussstecker des CBX-InRailBus zum Anschluss der +24V Versorgungsspannung und des CAN-Interface, Buchsenkontakte	C.3000.02
CAN-CBX-TBUS-Connection adapter 	Anschlussstecker des CBX-InRailBus zum Anschluss der +24V Versorgungsspannung und des CAN-Interface, Stiftkontakte	C.3000.03

Tabelle 32: Bestellhinweise

PDF-Handbücher
Handbücher sind in Englisch und üblicherweise auch in Deutsch erhältlich. Die Verfügbarkeit der Handbücher entnehmen Sie bitte der unteren Tabelle.
Die Handbücher im PDF-Format können Sie kostenlos von unserer Webseite www.esd.eu herunterladen.

Handbücher		Bestell-Nr.
CAN-CBX-COM2-MD	Anwenderhandbuch in Deutsch	C.3055.20
CAN-CBX-COM2-ME	Anwenderhandbuch in Englisch	C.3055.21

Tabelle 33: Verfügbare Handbücher

Gedruckte Handbücher
Benötigen Sie zusätzlich einen Ausdruck des Handbuches, kontaktieren Sie bitte unser Sales-Team sales@esd.eu für ein Angebot. Gedruckte Handbücher können gegen eine Gebühr bestellt werden.