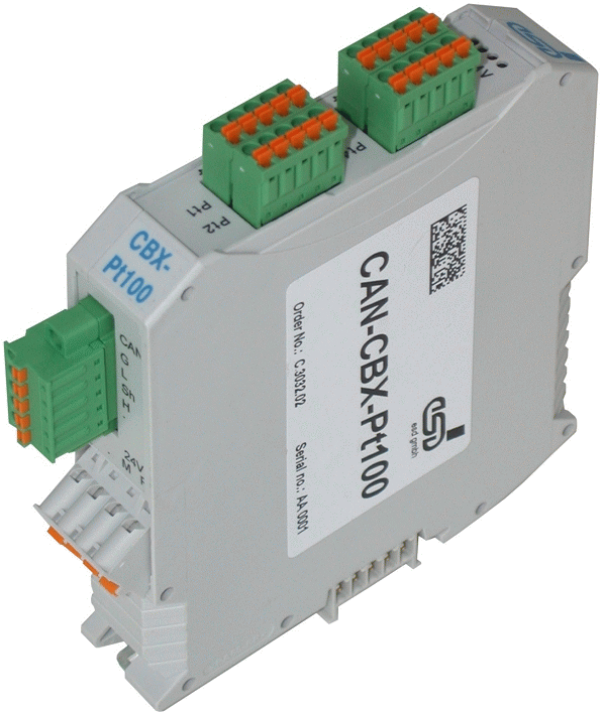
CAN-CBX-Pt100

4x RTD (Pt100, Pt1000) Eingänge



Handbuch

zu Artikel C.3032.02

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.
© 2014 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.

Dokument-Datei:	I:\Texte\Doku\MANUALS\CAN\CBX\PT100\Deutsch\CAN-CBX-PT100_Manual_de_11.wpd
Datum des Ausdrucks:	2014-12-16

Platinenversion:	ab CAN-CBX-Pt100 Rev. 1.0
Firmwareversion:	ab Rev. 2.03

Änderungen in den Kapiteln

Die hier aufgeführten Änderungen im Dokument betreffen sowohl Änderungen in der Hardware als auch reine Änderungen in der Beschreibung der Sachverhalte.

Version	Kapitel	Änderungen gegenüber Vorversion
1.1	-	Hinweise zur Sicherheit überarbeítet, Darstellungskonventionen eingefügt
	3.	Technische Daten überarbeitet
	4.1	Bild Anschlussplan aktualisiert
	4.2, 4.3	Kapitel überarbeitet
	5.	Hinweise zu Leiteranschluss und Leiterquerschnitt eingefügt, Abbildung Schaltung CAN Interface neu
	6., 7.	Kapitel aktualisiert
	8.	Kapitel umstrukturiert, Kapitel “Communication Profile Area” hinzugefügt
	9.	Referenzen überarbeitet
	11.	EU-Konformitätserklärung aktualisiert

Weitere technische Änderungen vorbehalten.

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.

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.

Konformität

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

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

Bestimmungsgemäßer Gebrauch

Die bestimmungsgemäße Verwendung ist der Einsatz des CAN-CBX-Moduls als CANopen-Slave mit Temperatur-Sensor-Messfühlern.
Der Hersteller haftet nicht für Schäden, die durch unsachgemäßen Gebrauch, nicht bestimmungsgemäße Verwendung oder in Folge von Nichtbeachtung der Sicherheitshinweise und Warnungen verursacht werden.
Jeder Eingriff in das CAN-CBX-Modul durch nicht von esd autorisierte Personen führt zum Verlust aller Garantieansprüche.

- Das CAN-CBX-Modul ist nur für die Anwendung innerhalb von Gebäuden vorgesehen.
- Das CAN-CBX-Modul 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-Modul 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. Technische Daten 11
3.1 Allgemeine technische Daten 11
3.2 Microcontroller 12
3.3 CAN-Schnittstelle 12
3.4 Messwiderstandseingänge 13
3.5 Software-Unterstützung 13

4. Hardware-Installation 14
4.1 Anschlussplan 14
4.1.1 Spannungsversorgung und CAN 14
4.1.2 Messwiderstand in 2-Leiter-Technik anschließen (Beispiel Pt2) 15
4.1.3 Messwiderstand in 4-Leiter-Technik anschließen (Beispiel Pt2) 16
4.2 LED-Anzeigen 17
4.2.1 Blinkzustände 17
4.2.2 Bedeutung der CAN-Error-LED 18
4.2.3 Bedeutung der CANopen-Status-LED 18
4.2.4 12Bedeutung der Error-LED 19
4.2.5 Bedeutung der Power-LED 19
4.2.6 Besondere Blinkzustände 20
4.2.7 Zuordnung LED-Bezeichnung zu Bezeichnung im Schaltplan 20
4.3 Kodierschalter 21
4.3.1 Einstellung der Node-ID über die Kodierschalter 21
4.3.2 Einstellung der CAN-Bitrate über den Kodierschalter 22
4.3.3 Zuordnung Kodierschalterbezeichnung zum Schaltplan 22
4.4 Einbau des Moduls bei Verwendung des InRailBus-Verbinders 23
4.4.1 Anschluss über den CBX-InRailBus 24
4.4.2 Anschluss der Versorgungsspannung 25
4.4.3 Anschluss von CAN 26
4.5 Ausbau des CAN-CBX-Moduls vom InRailBus 26

5. Steckerbelegungen 27
5.1 Spannungsversorgung 24V (X100) 27
5.2 CAN 28
5.2.1 CAN-Interface-Schaltung 28
5.2.2 CAN-Stecker 29
5.2.3 CAN und Versorgungsspannung über InRailBus X101 30
5.3 Messwiderstände Pt1...Pt4 31
5.4 Leiteranschluss/Leiterquerschnitt 33

6. Korrekte Verdrahtung galvanisch getrennter CAN-Netze 34
6.1 Standards zur CAN-Verdrahtung 34
6.2 Leicht störbehaftete Industrieumgebung (zweiadrig verdrillte Leitung) 35
6.2.1 Grundregeln 35
6.2.2 Verkabelung 36
6.2.3 Abschlusswiderstand 36

6.3 Stark störbehaftete Industrieumgebung (vieradrig verdrillte Leitung) 37
6.3.1 Grundregeln 37
6.3.2 Verkabelung 38
6.3.3 Abschlusswiderstand 38
6.4 Erdung 39
6.5 Buslänge 39
6.6 Beispiele für CAN-Kabel 40
6.6.1 Kabel für leicht störbehaftete Industrieumgebung (zweiadrig) 40
6.6.2 Kabel für stark störbehaftete Industrieumgebung (vieradrig) 40

7. CAN-Bus Troubleshooting Guide 41
7.1 Bus-Abschluss 41
7.2 Erdung 42
7.3 Kurzschluss in der CAN-Verdrahtung 42
7.4 CAN_H/CAN_L-Spannungen 42
7.5 CAN-Transceiver Widerstandstest 43
7.6 Support bei esd 43

8. CANopen-Firmware 44
8.1 Begriffsdefinition 44
8.2 NMT-Boot-up 45
8.3 Das CANopen-Objektverzeichnis 45
8.4 Communication Parameter 46
8.4.1 Zugriff auf das Objektverzeichnis über SDOs 46
8.5 Übersicht der verwendeten CANopen-Identifier 49
8.5.1 Einstellung der COB-ID 49
8.6 Default-PDO-Belegung 50
8.7 Lesen der Messwerte 51
8.7.1 Meldung der A/D-Wandler 51
8.7.2 Unterstützte Übertragungsarten nach [1] 51
8.8 Communication Profile Area 52
8.8.1 Verwendete Bezeichnungen und Abkürzungen 52
8.9 Implementierte CANopen-Objekte 53
8.9.1 Übersicht der 1000er-Objekt und Default-Werte 53
8.9.2 Device Type (1000_h) 55
8.9.3 Error Register (1001_h) 56
8.9.4 Pre-defined Error Field (1003_h) 57
8.9.5 COB-ID of SYNC-Message (1005_h) 59
8.9.6 Communication Cycle Period (1006_h) 60
8.9.7 Manufacturer Device Name (1008_h) 61
8.9.8 Manufacturer Hardware Version (1009_h) 62
8.9.9 Manufacturer Software Version ($100A_h$) 62
8.9.10 Guard Time ($100C_h$) und Life Time Factor ($100D_h$) 63
8.9.11 Node Guarding Identifier ($100E_h$) 64
8.9.12 Store Parameters (1010_h) 65
8.9.13 Restore Default Parameters (1011_h) 67
8.9.14 COB_ID Emergency Message (1014_h) 69
8.9.15 Inhibit Time EMCY (1015_h) 70
8.9.16 Consumer Heartbeat Time (1016_h) 71
8.9.17 Producer Heartbeat Time (1017_h) 73
8.9.18 Identity Object (1018_h) 74

8.9.19 Synchronous Counter Overflow Value (1019_h) 76
8.9.20 Verify Configuration (1020_h) 77
8.9.21 Error Behaviour Object (1029_h) 78
8.9.22 NMT Startup ($1F80_h$) 79
8.9.23 Self Starting Nodes Timing Parameters ($1F91_h$) 80
8.9.24 Objekt Transmit PDO Communication Parameter 1800_h ... 1803_h 81
8.9.25 Transmit PDO Mapping Parameter $1A00_h$ ... $1A03_h$ 82
8.10 Device Profile Area 83
8.10.1 Übersicht der implementierten Objekte 6110_h ...9135_h 83
8.10.2 Zusammenhang der implementierten Objekte für die analogen Eingänge 84
8.10.3 AI Sensor Type (6110_h) 85
8.10.4 AI Autocalibration (6111_h) 86
8.10.5 AI Operating Mode (6112_h) 87
8.10.6 AI ADC Sampling Rate (6114_h) 88
8.10.7 AI Physical Unit (6131_h) 89
8.10.8 AI Decimal Digits PV (6132_h) 91
8.10.9 AI Status (6150_h) 92
8.10.10 AI Input Field Value (9100_h) 93
8.10.11 AI Interrupt Delta Input FV (9103_h) 94
8.10.12 AI Input Process Value (9130_h) 95
8.10.13 AI Interrupt Delta Input PV (9133_h) 96
8.10.14 AI Interrupt Lower Limit PV (9134_h) 97
8.10.15 AI Interrupt Upper Limit PV (9135_h) 98
8.11 Manufacturer Specific Profile Area 99
8.11.1 Übersicht der Manufacturer Specifc Objects 2400_h ... 2512_h 99
8.11.2 ADC_PGA (2400_h) 100
8.11.3 Channel Enabled (2401_h) 101
8.11.4 Accu N (2402_h) 102
8.11.5 Average N (2403_h) 103
8.11.6 Current Source Value (2410_h) 104
8.11.7 Aktuelle Temperatur der Leiterkarte (2420_h) 105
8.11.8 Temperatur der Leiterkarte während der letzten Kalibrierung (2421_h) 106
8.11.9 Zulässige Temperaturänderung der Leiterkarte bis Neukalibrierung (2422_h) 107
8.11.10 Kalibrierdaten 108
8.11.10.1 Kalibrierung und Process/Field-Value Ermittlung 108
8.11.10.2 Referenztemperaturen bei der Kalibrierung (2500_h, 2501_h) 111
8.11.10.3 Verstärkungskorrektur bei I_{LOW} und I_{HIGH} (2502_h, 2503_h) 112
8.11.10.4 Verstärkungskorrektur für PGA=2/3/4 (2504_h, 2505_h, 2506_h) 113
8.11.10.5 Offset (2510_h) 114
8.11.10.6 Temperaturkoeffizient bei I_{LOW} und I_{HIGH} (2511_h, 2512_h) 115
8.12 Firmware-Verwaltung über DS-302-Objekte ($1F50_h$...$1F52_h$) 116
8.12.1 Download-Steuerung über Objekt ($1F51_h$) 117
8.12.2 Verify Application Software ($1F52_h$) 117

9. Referenzen 118

10. Glossar 118

11. EU-Konformitätserklärung 119

12. Bestellhinweise 120

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

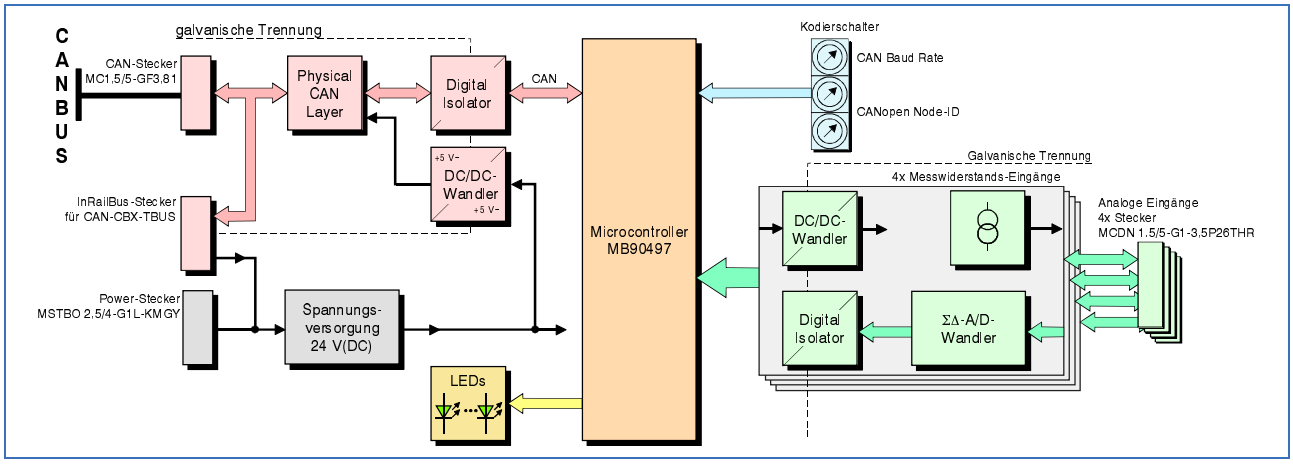


Abb. 1: Blockschaltbild des CAN-CBX-Pt100-Moduls

Die analoge Eingangsbaugruppe CAN-CBX-Pt100 erlaubt den direkten Anschluss von bis zu vier Platin- oder Nickel-Temperaturmessfühlern. Auch eine reine Widerstandsmessung mit Ausgabe der Ergebnisse in Ohm ist möglich.
Die Messwiderstände können in 2-Leiter- oder 4-Leiter-Technik angeschlossen werden. Der Anschluss der Eingänge erfolgt über vier 5-polige Klemmverbinder. Die Linearisierung gemäß NIST wird mit Hilfe des Microcontrollers erreicht.
Die Wandlung der vier Messwiderstandseingänge erfolgt über vier unabhängige ΣΔ-A/D-Wandler. Die in der Praxis erreichte Auflösung hängt wesentlich von der gewählten Sample-Rate und dem gewählten Messstrom ab.

Das CAN-CBX-Pt100-Modul verfügt über einen MB90F497 Microcontroller, der die CAN-Daten in seinem lokalen SRAM zwischen speichert. Die Firmware wird im Flash gehalten. Zur Speicherung von Parametern dient ein serielles EEPROM.

Die zu ISO 11898-2 kompatible CAN-Schnittstelle 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-Bit-Rate können über drei außen liegende Drehschalter eingestellt werden.

Die Versorgungsspannung und der CAN-Bus-Anschluss werden entweder über den in die Hutschiene integrierten InRailBus-Verbinder (TBUS-Connector) oder über separate Stecker zugeführt. Einzelne Module können vom InRailBus entnommen werden, ohne die Bussignale zu unterbrechen.

Das Modul unterstützt die CANopen®-Profile der CiA® nach [1] und [4].

2. Schnelle Inbetriebnahme

1. Modul anschließen (siehe Anschlussplan ab Seite 14)

2. In Objekt 9103_h (Sub-Index 1...4) eine “-1” schreiben ($FFFF\ FFFF_h$)
   (Mit jeder neuen A/D-Wandlung wird ein TPDO mit dem Messwert gesendet.)

3. NMT-Start Kommando an Modul senden

   Auf
   0180_h + Modul-Nummer
   0280_h + Modul-Nummer
   0380_h + Modul-Nummer
   0480_h + Modul-Nummer
   werden die PDOs gesendet.

   In den Daten-Bytes 0...3 wird der Process-Value (PV) gesendet.

   In der Default-Einstellung des Moduls wird als Process-Value der Widerstandswert in mOhm gesendet.

   Bsp.: 1234567 entspricht 1234,567 Ohm

3. Technische Daten

3.1 Allgemeine technische Daten

Versorgungsspannung	Nennspannung: 24 V/DC Eingangsspannungsbereich: 24 V ±20% Stromaufnahme (24 V, 20 °C): typisch: ca. 120 mA
Steckverbinder	24V (4-pol. Leiterplattenstecker mit Federkraftanschluss-Kontakten, X100) - 24V-Spannungsversorgung Pt1 (5-pol. Leiterplattenstecker mit Federkraftanschluss-Kontakten, X500A) - Messfühler-Eingang 1 Pt2 (5-pol. Leiterplattenstecker mit Federkraftanschluss-Kontakten, X500B) - Messfühler-Eingang 2 Pt3 (5-pol. Leiterplattenstecker mit Federkraftanschluss-Kontakten, X800A) - Messfühler-Eingang 3 Pt4 (5-pol. Leiterplattenstecker mit Federkraftanschluss-Kontakten, X800B) - Messfühler-Eingang 4 CAN (5-pol. Leiterplattenstecker mit Federkraftanschluss-Kontakten, X400) - CAN-Schnittstelle InRailBus (5-pol. CAN-CBX-TBUS Verbinder, Phoenix Contact, X101) - CAN-Schnittstelle und Spannungsversorgung über InRailBus Nur für Test und Programmierzwecke: X200 (6-pol. Buchsenleiste) der Stecker befindet sich innerhalb des Gehäuses
Temperaturbereich	-20 °C ... +70 °C Umgebungstemperatur
Luftfeuchtigkeit	max. 90%, nicht kondensierend
IP-Schutzklasse	IP20
Verschmutzungsgrad	max. Verschmutzungsgrad 2 (DIN EN 61131-2)
Gehäuse	Kunststoffgehäuse für Tragschienenmontage NS35/7,5 DIN EN 60715
Abmessungen	Breite: 22,5 mm, Höhe: 99 mm, Tiefe: 114,5 mm (Einschließlich Tragschienenhalterung und Steckerüberstand, ohne Gegenstecker)
Gewicht	ca. 125 g

Tabelle 1: Allgemeine Daten des Moduls

3.2 Microcontroller

CPU	16 Bit µC MB90F497
RAM	2 kByte integriert
Flash	64 kByte integriert
EEPROM	min. 256 Byte

Tabelle 2: Microcontroller

3.3 CAN-Schnittstelle

Anzahl	1x CAN
Anschluss	5-pol. Leiterplattenstecker mit Federkraftanschluss-Kontakten oder über InRailBus-Verbinder (CAN-CBX-TBUS)
CAN-Controller	im MB90F497 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 3: Daten der CAN-Schnittstelle

3.4 Messwiderstandseingänge

Anzahl	4 unabhängige ΣΔ-A/D-Wandler-Eingänge
Thermo-Widerstände	Temperaturfühler: Pt100, Pt200, Pt500, Pt1000 und Pt5000, Ni100, Ni200, Ni500, Ni1000 und Ni5000 und Widerstandsmessung allgemein
Messstrom	wählbar: ca. 400 µA oder ca. 40 µA
Anschlusstechnik	2-Leiter- oder 4-Leiter-Technik
Messbereich	Platin-Messfühler: -250 ˚C ... +850 ˚C Nickel-Messfühler: -200 ˚C ... +400 ˚C Widerstandsmessung: 0 ... 50 K Ω
Wandlungsrate	2,5 Hz ... 1000 Hz
Auflösung	<1 µV bei 25 Hz Wandlungsrate ≈ < 0.01˚C
Messfehler	Abhängig von Sensor-Typ und Messtemperatur: z.B. bei Pt100-Fühler mit 4-Leiter-Anschluss: Messstrom = ca. 400 µA: Messfehler < 0,1 Ω Messstrom = ca. 40 µA: Messfehler < 0,5 Ω Die Daten der Kennlinie können der Norm DIN EN 60751:2008 entnommen werden
Elektrische Auflösung	elektrische Auflösung = f (Messstrom, Sample-Rate) z.B. Messstrom = ca. 400 µA Sample-Rate = 400 msec resultierende Auflösung = ca. 1 mΩ
Galvanische Trennung zwischen den Messkanälen und gegenüber den anderen Baugruppen	Die Temperaturfühlereingänge sind gegen einander und gegenüber der Spannungsversorgung galvanisch getrennt.
Eingangsfilter	parametrierbar über CiA DS 404-Objekte
Steckverbinder	4x 5- Leiterplattenstecker mit Federkraftanschluss-Kontakten

Tabelle 4: Daten der Messwiderstandseingänge

3.5 Software-Unterstützung

Die Firmware des Moduls unterstützt CANopen® nach den CiA® CANopen Spezifikationen CiA 301 [1] und CiA DS-404 [4].
Die EDS-Datei des CAN-CBX-Pt100 kann von der esd-Webseite unter www.esd.eu heruntergeladen werden.

4. Hardware-Installation

4.1 Anschlussplan

4.1.1 Spannungsversorgung und CAN

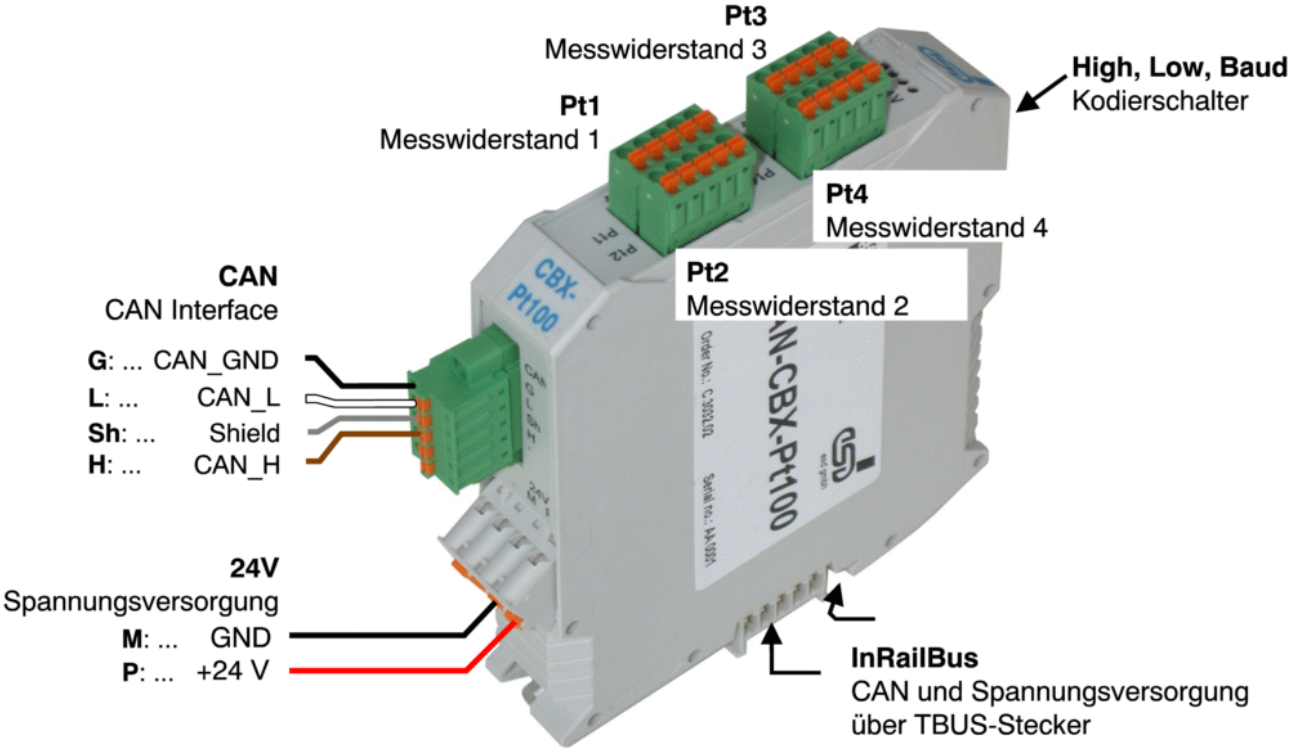


Abb. 2: Spannungsversorgungs- und CAN-Anschluss des CAN-CBX-Pt100-Moduls

Hinweis:
Hinweise zum Leiteranschluss/Leiterquerschnitt entnehmen Sie bitte Seite 33.
Die Signalbelegung der Steckverbinder ist in tabellarischer Form ab Seite 27 abgedruckt.

4.1.2 Messwiderstand in 2-Leiter-Technik anschließen (Beispiel Pt2)

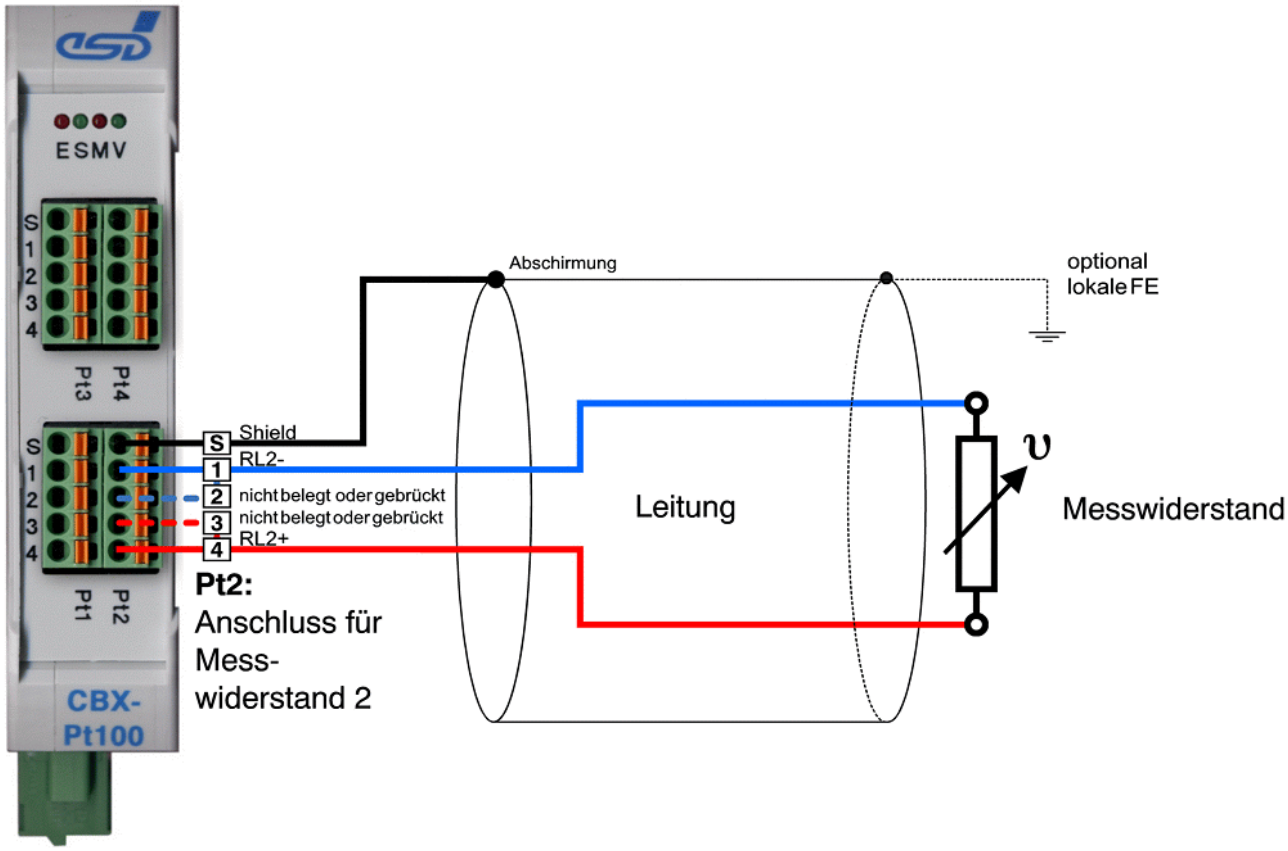


Abb. 3: Beispiel Messwiderstand-Anschluss in 2-Leiter-Technik an Pt2

Hinweis:
Hinweise zum Leiteranschluss/Leiterquerschnitt entnehmen Sie bitte Seite 33.
Die Signalbelegung der Steckverbinder ist in tabellarischer Form ab Seite 27 abgedruckt.

4.1.3 Messwiderstand in 4-Leiter-Technik anschließen (Beispiel Pt2)

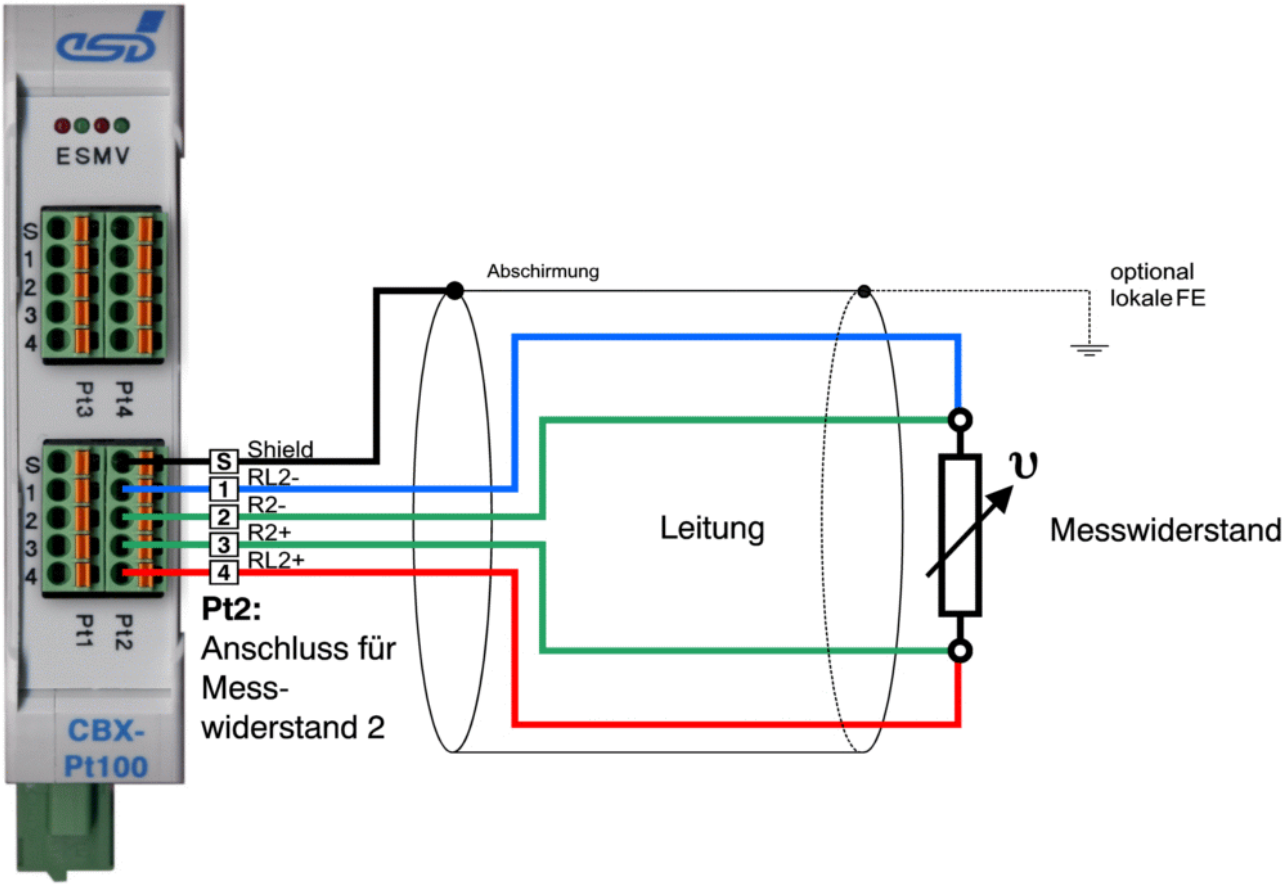


Abb. 4: Beispiel Messwiderstand-Anschluss in 4-Leiter-Technik an Pt2

Hinweis:
Hinweise zum Leiteranschluss/Leiterquerschnitt entnehmen Sie bitte Seite 33.
Die Signalbelegung der Steckverbinder ist in tabellarischer Form ab Seite 27 abgedruckt.

4.2 LED-Anzeigen

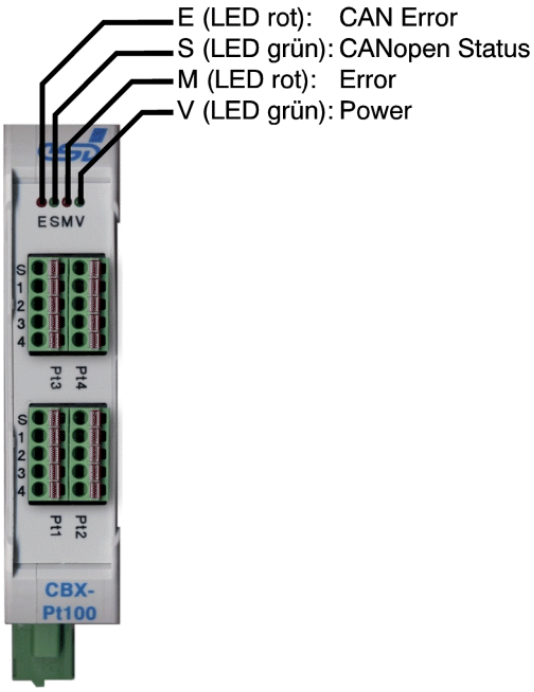


Abb. 5: Position der LEDs in der Frontplatte

Das CAN-CBX-Pt100-Modul verfügt über vier Status-LEDs. Die Bezeichnung der Blinkzustände der einzelnen LEDs ist in Anlehnung an die Bezeichnungen der CANopen®-Norm DS 303-3 [3] gewählt. Die Bedeutung der Blinkzustände ist in den folgenden Kapiteln beschrieben.

4.2.1 Blinkzustände

Es gibt prinzipiell 8 Blinkzustände, die die 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 5: Anzeigefunktion der 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.

4.2.2 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 beim Einschalten ist Kodierschalterstellung ID-Node > $7F_h$; siehe Besondere Blinkzustände Seite 20)
			2 Blitze	Heartbeat- oder Nodeguard-Fehler aufgetreten. Werden wieder Nodeguard/Heartbeat-Messages empfangen, wird die Fehlermeldung zurückgenommen und die LED erlischt.

Tabelle 6: Anzeigefunktion der roten CANopen Error-LED

4.2.3 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, die Power LED ist dabei aus. (oder Kodierschalterstellung ID-Node > $7F_h$ beim Einschalten; siehe Seite 20)

Tabelle 7: Anzeigefunktion der CANopen Status-LED

4.2.4 12Bedeutung 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	Sensor-Fehler (emergency error code 5030_h) z.B.: - Sensor nicht angeschlossen - Sensor-Daten fehlerhaft

Tabelle 8: Anzeigefunktion der Error-LED

4.2.5 Bedeutung der Power-LED

LED-Kennung			Anzeigefunktion	
Auf-druck	Name	Farbe	Blinkzustand	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 9: Anzeigefunktion der Power-LED

4.2.6 Besondere Blinkzustände

Dieser Zustand wird von der CANopen-Status-LED und der CAN-Error-LED gemeinsam angezeigt:

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 10: Besondere Blinkzustände

4.2.7 Zuordnung LED-Bezeichnung zu Bezeichnung im Schaltplan

Bezeichnung auf CAN-CBX-Pt100	Bezeichnung im Schaltplan *1)
E	LED200A
S	LED200B
M	LED200C
V	LED200D

*1) Der Schaltplan ist nicht Bestandteil dieses Handbuchs.

4.3 Kodierschalter

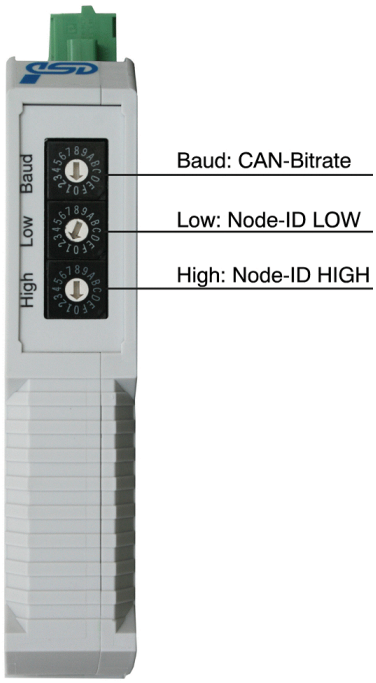


Abb. 6: Position der Kodierschalter

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 Reset (z.B. NMT-Reset) werden die Einstellungen erneut eingelesen.

4.3.1 Einstellung der Node-ID über die Kodierschalter

Der Adressbereich 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.

4.3.2 Einstellung der CAN-Bitrate über den Kodierschalter

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	$83,\overline{3}$

Tabelle 11: Index der Bitrate

4.3.3 Zuordnung Kodierschalterbezeichnung zum Schaltplan

Bezeichnung auf CAN-CBX-Pt100	Bezeichnung im Schaltplan *1)
Baud	SW301
Low	SW300
High	SW302

*1) Der Schaltplan ist nicht Bestandteil dieses Handbuchs.

4.4 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:

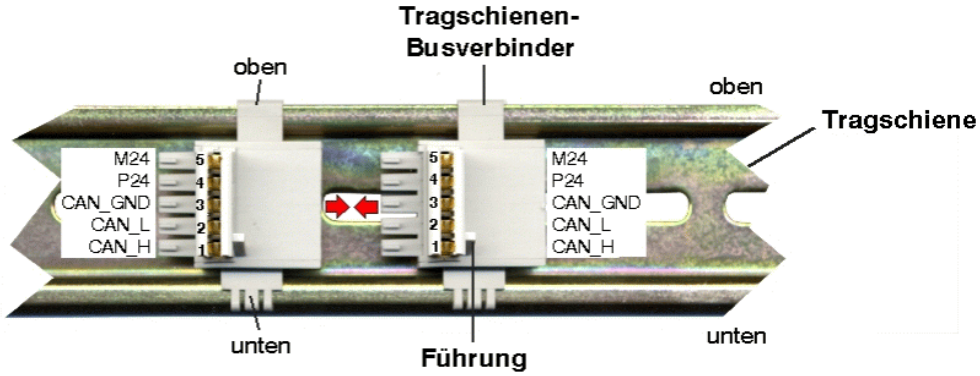


Abb. 7: Tragschiene mit Busverbinder

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.

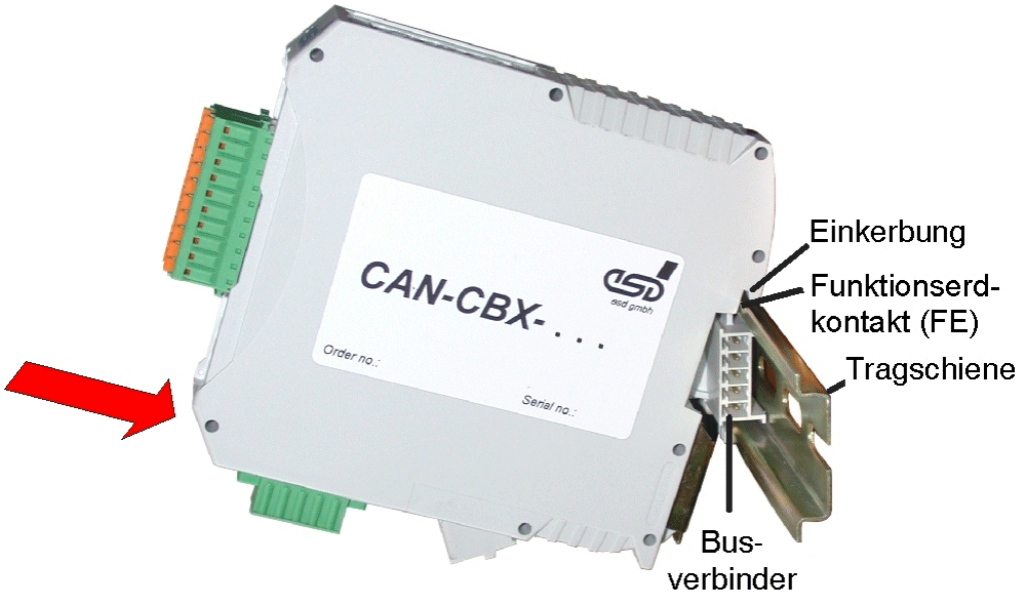


Abb. 8 : Einsetzen des CAN-CBX-Moduls

3. Schwenken Sie nun das CBX-Modul auf die Tragschiene auf, indem sie das Modul entsprechend der Pfeilrichtung in Abbildung 8 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.

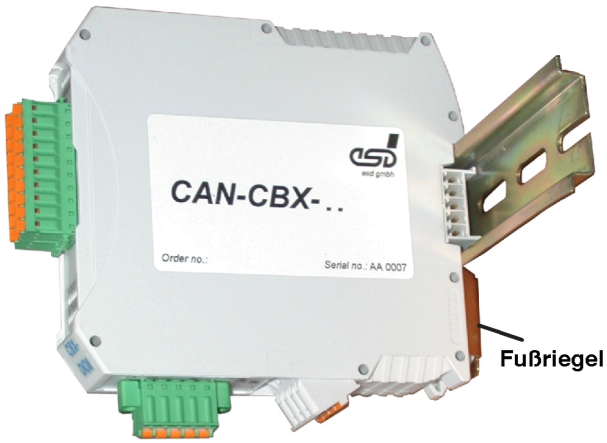


Abb. 9: Eingebautes CAN-CBX-Modul

4.4.1 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).

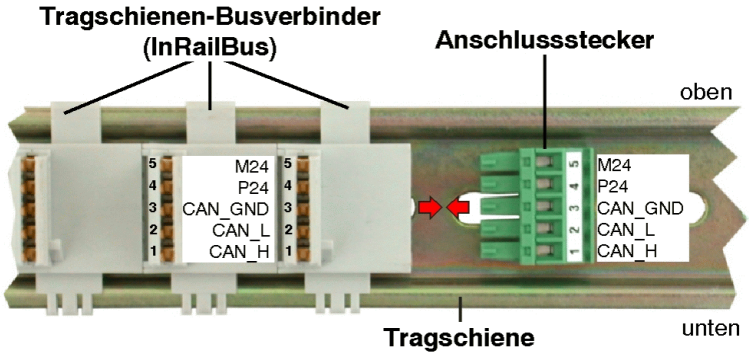


Abb. 10: Tragschiene mit InRailBus-Verbindern und Anschlussstecker

Stecken Sie den Anschlussstecker, wie in Abb. 10 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.

4.4.2 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!

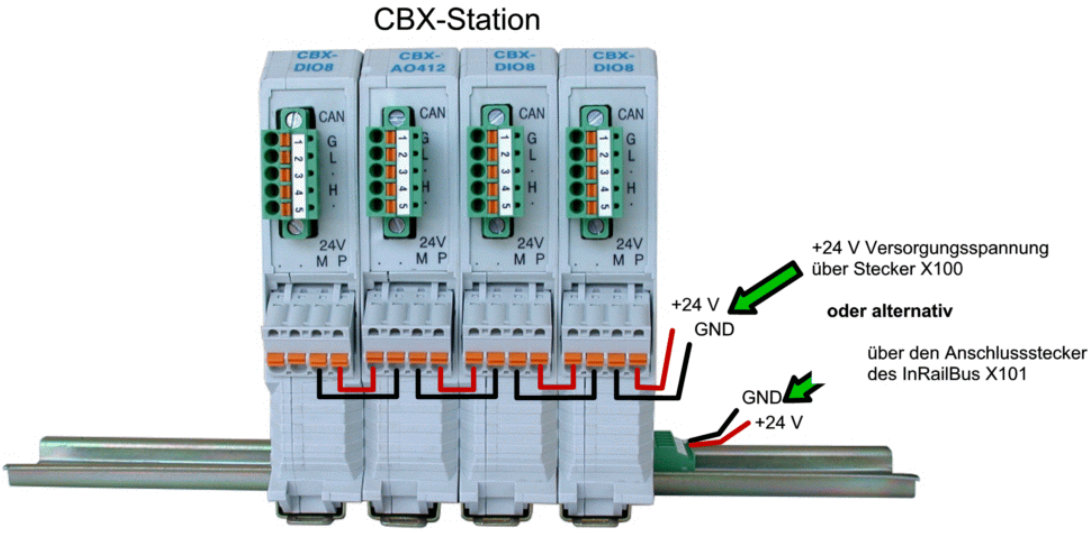


Abb. 11: CAN-CBX-Station mit Anschluss der Versorgungsspannung

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.

4.4.3 Anschluss von CAN

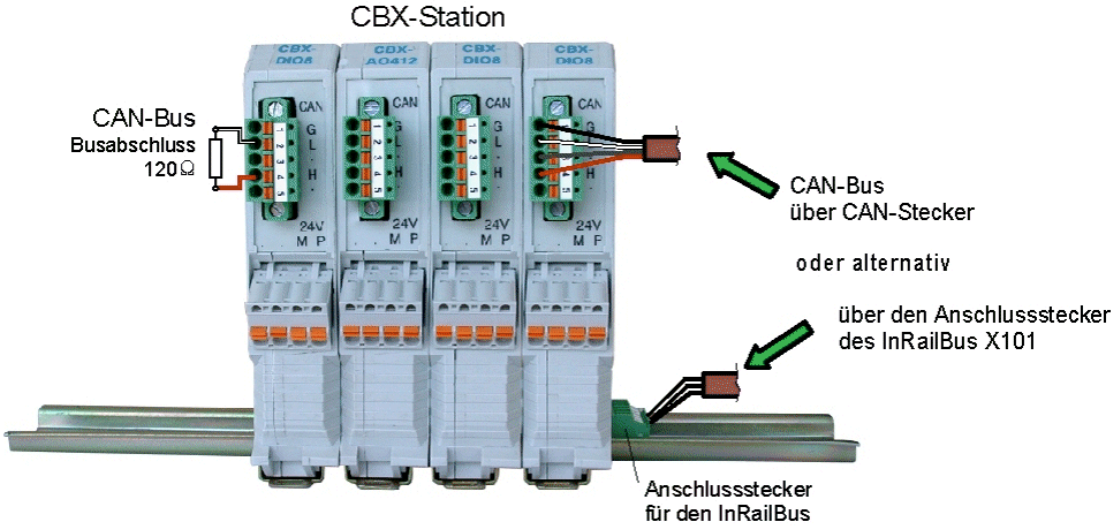


Abb. 12: Anschluss von CAN an die CBX-Station

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.

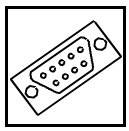
Bitte beachten Sie, dass an das CAN-CBX-Modul, das sich am Ende des InRailBus befindet ein Bus-Abschlusswiderstand angeschlossen werden muss, wenn der CAN-Bus dort endet (siehe Abb. 12).

4.5 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. 9) 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.

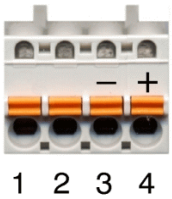


5. Steckerbelegungen

5.1 Spannungsversorgung 24V (X100)

Gerätebuchse: Phoenix Contact MSTBO 2,5/4-G1L-KMGY
Leitungsstecker: Phoenix Contact FKCT 2,5/4-ST, 5.0 mm Raster,
Federkraftanschluss-Kontakte,
Phoenix Contact Bestell-Nr.: 19 21 90 0 (im Lieferumfang enthalten)
Zu Leiteranschluss und Leiterquerschnitt siehe Seite 33.

Pin-Zuordnung:



Pin-Belegung:

Gehäusebezeichnung	24V			
	·	·	M	P
Stecker-Aufdruck	(frei)	(frei)	-	+

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

Siehe auch Anschlussplan Seite 14.

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 digitalen Isolator und DC/DC-Wandler galvanisch getrennt.

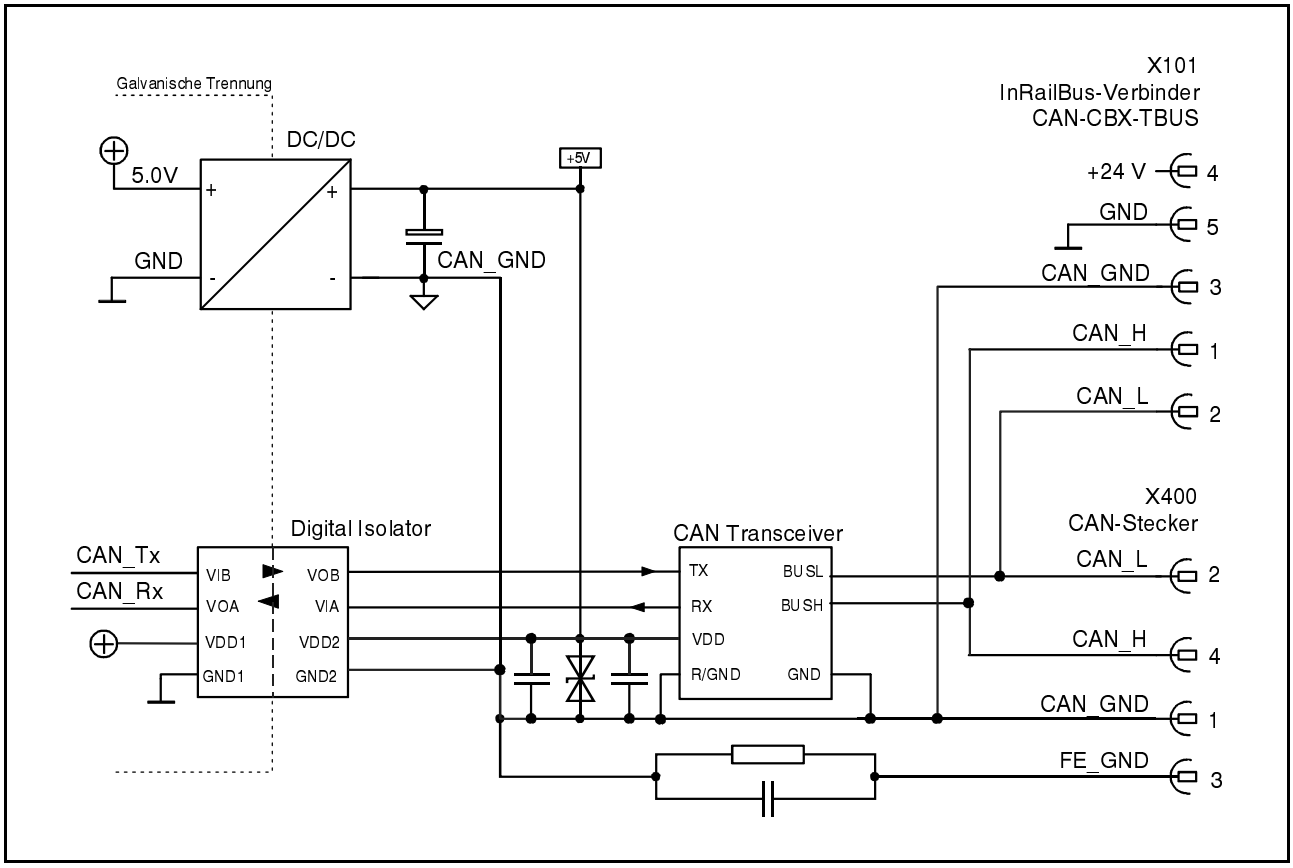
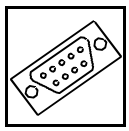


Abb. 13: Schaltung des CAN-Interface

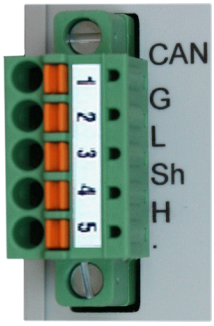
Die CAN Schnittstelle kann entweder über den CAN-Stecker oder optional über den InRailBus angeschlossen werden. Benutzen Sie für den Anschluss über den InRailBus den Tragschienenbusverbinder des CBX-InRailBus (CAN-CBX-TBUS), siehe Bestellhinweise (Seite 120).



5.2.2 CAN-Stecker

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 33.

Pin-Zuordnung:
(Ausschnitt Leitungsstecker mit Gehäuseaufdruck)



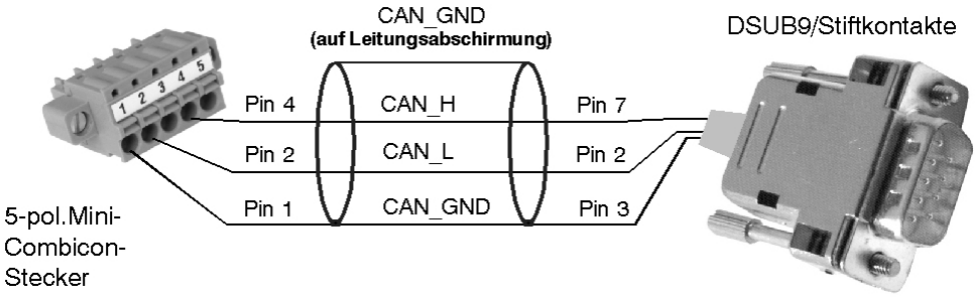
Pin-Belegung:

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-Stecker (hier Leitungsstecker FK-MCP 1,5/5-STF-3,81 mit Federkraftanschluss-Kontakten) auf 9-pol-DSUB:



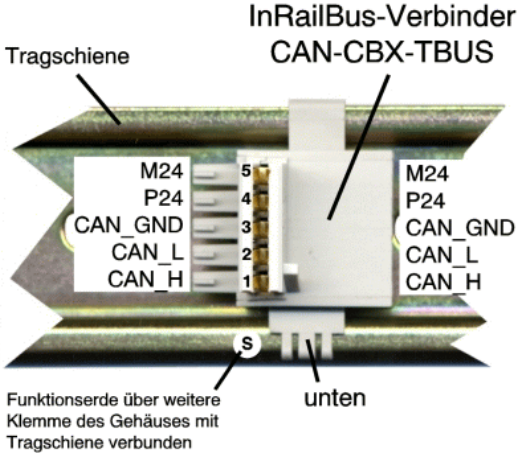
Die Belegung des 9-poligen DSUB-Steckers erfolgt nach CiA DS 102.

Steckerbelegung

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:



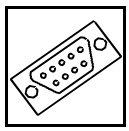
Pin-Belegung:

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

S	FE (PE_GND)

Signalbeschreibung:

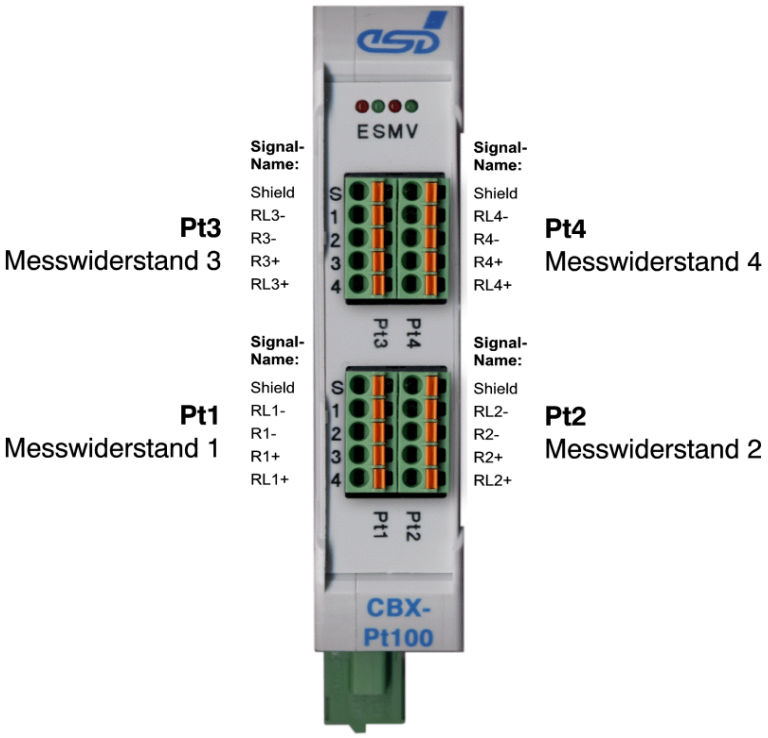
CAN_L,
CAN_H ... CAN-Signale
CAN_GND ... Bezugspotenzial des lokalen CAN-Physical Layers
P24... Versorgungsspannung +24 V
M24... Bezugspotenzial
FE... direkt am Gehäuse des Moduls: Funktionserde (mit Hutschienenpotenzial verbunden)



5.3 Messwiderstände Pt1...Pt4

Gerätebuchse: 2x Phoenix Contact MCDN 1,5/5-G1-3,5P26THR
Leitungsstecker: 4x Phoenix Contact FMC 1,5/5-ST-3,5 (Federkraftanschluss-Kontakte)
Phoenix Contact Bestell-Nr.: 19522296 (im Lieferumfang enthalten)
Zu Leiteranschluss und Leiterquerschnitt siehe Seite 33.

Pin-Zuordnung:

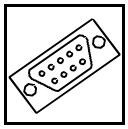


Signalbeschreibung:

Pin	Signalname	Funktion bei 4-Leiter-Anschluss	Funktion bei 2-Leiter-Anschluss
S	Shield	Anschluss für Leitungsabschirmung (bei Hutschienenmontage direkter Kontakt zu Hutschienenpotenzial)	
1	RLx-	Stromquelle -	Negativer Abgriff für Messwiderstand
2	Rx-	Negativer Abgriff für Messwiderstand	offen oder gebrückt auf RLx-
3	Rx+	Positiver Abgriff für Messwiderstand	offen oder gebrückt auf RLx+
4	RLx+	Stromquelle +	Positiver Abgriff für Messwiderstand

(x = 1, 2, 3, 4)

Hinweis:
Beispiele für den Anschluss des Messwiderstands in 2-Leiter- und 4-Leitertechnik finden Sie ab Seite 15, 16.

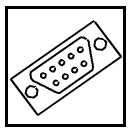


Steckerbelegung

Zuordnung der Steckerbezeichnung auf dem Gerät zu der Bezeichnung im Schaltplan

Bezeichnung auf Gerät	Bezeichnung im Schaltplan *1)
Pt1	X500:A
Pt2	X500:B
Pt3	X800:A
Pt4	X800:B

*1) Der Schaltplan ist nicht Bestandteil dieses Handbuchs.



5.4 Leiteranschluss/Leiterquerschnitt

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

Schnittstelle	**Spannungs-versorgung 24 V** [6]	**CAN Stecker** [7]	**Temperatur-sensor-Interface** [8]
Steckertyp Leitungsstecker (Artikelfamilie)	**FKCT 2,5/..-ST KMGY**	**FK-MCP 1,5/..-STF-3,81**	**FMC 1,5/..-ST-3,5**
Anschlussart	Federkraft-anschluss	Federkraft-anschluss	Federkraft-anschluss
Abisolierlänge	10 mm	9 mm	10 mm
Leiterquerschnitt starr min	0,2 mm²	0,14 mm²	0,2 mm²
Leiterquerschnitt starr max	2,5 mm²	1,5 mm²	1,5 mm²
Leiterquerschnitt flexibel min	0,2 mm²	0,14 mm²	0,2 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	0,75 mm²
Leiterquerschnitt AWG/kcmil min	24	26	24
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	24
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 Aderquerschnitt (≥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.

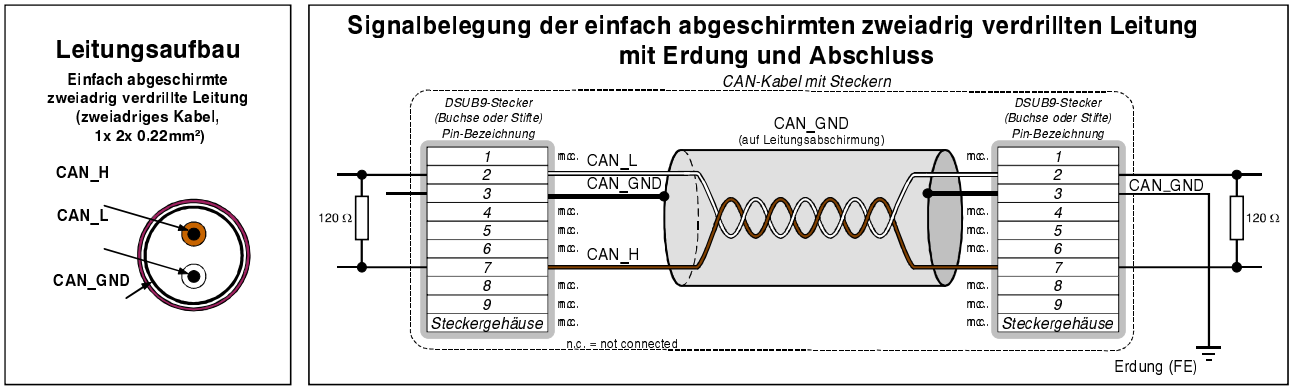


Abb. 14: CAN-Verdrahtung in leicht störbehafteter Industrieumgebung

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.

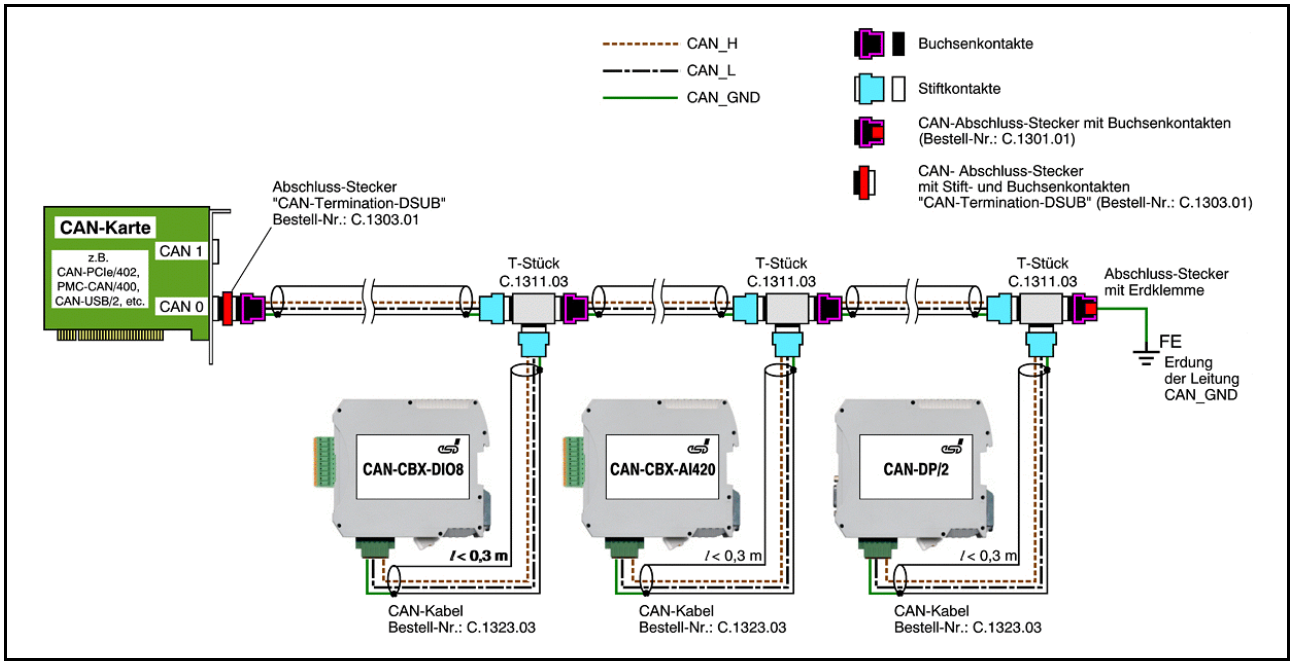


Abb. 15: Beispiel für korrekte Verdrahtung einfach abgeschirmter zweiadrig verdrillter Leitung

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 Aderquerschnitt (≥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.

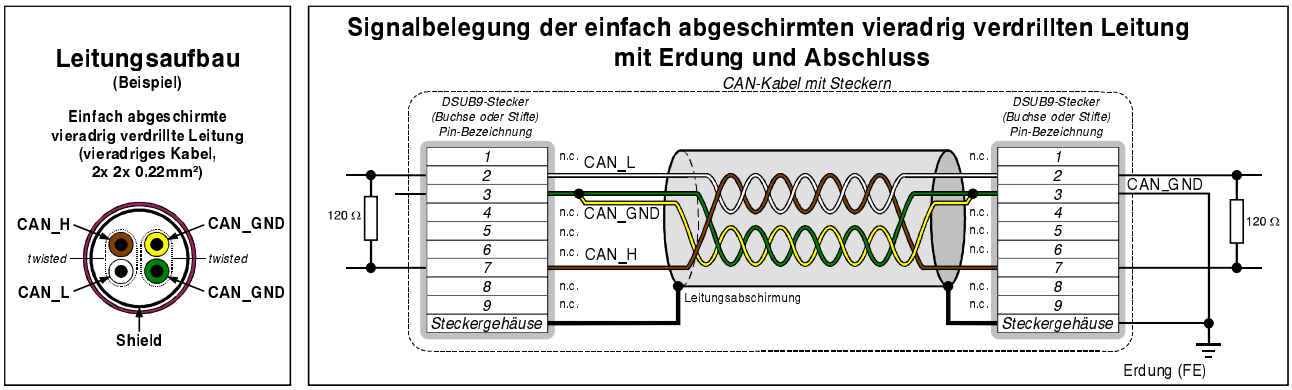


Abb. 16: CAN-Verdrahtung in stark störbehafteter Industrieumgebung

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!

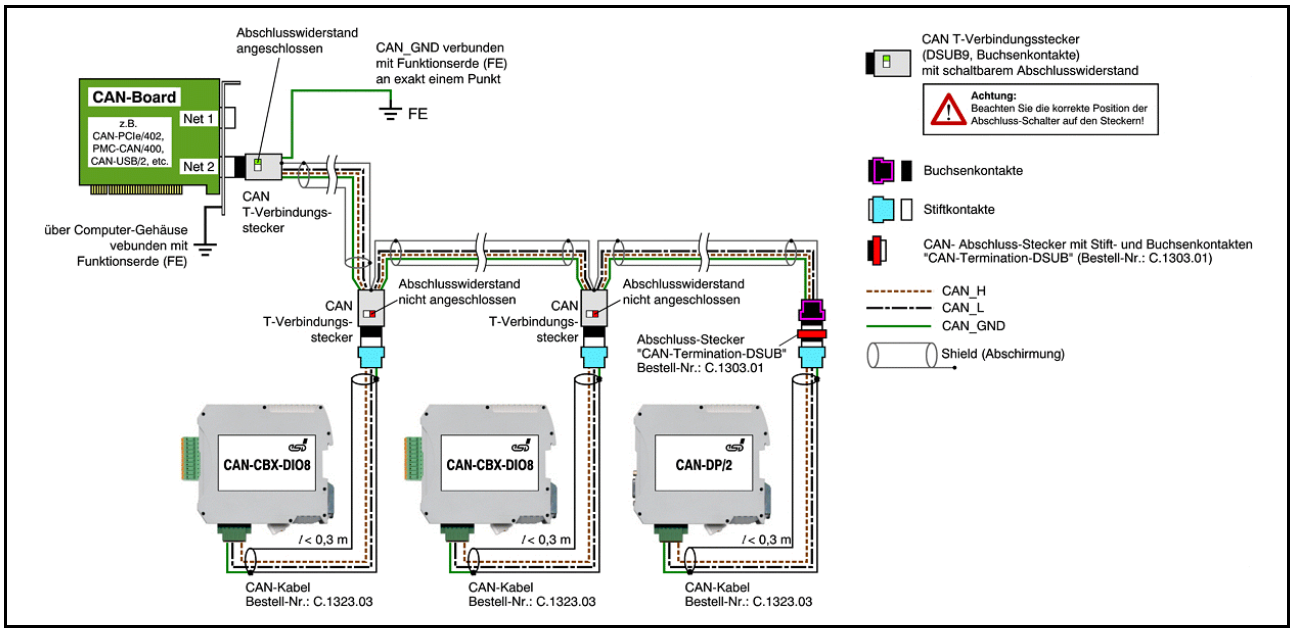


Abb. 17: Beispiel für korrekte Verdrahtung einfach abgeschirmter, vieradrig verdrillter Leitungen

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 12: 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 Leitungsquerschnitte 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.

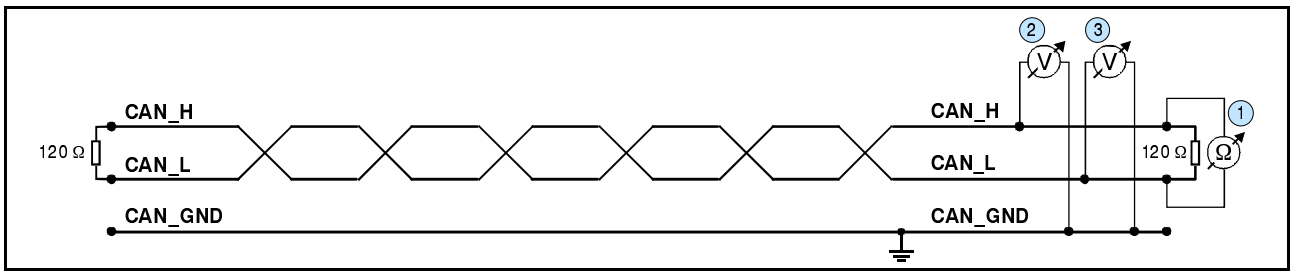


Abb. 18: Vereinfachtes Schaltbild eines CAN-Netzwerks

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.

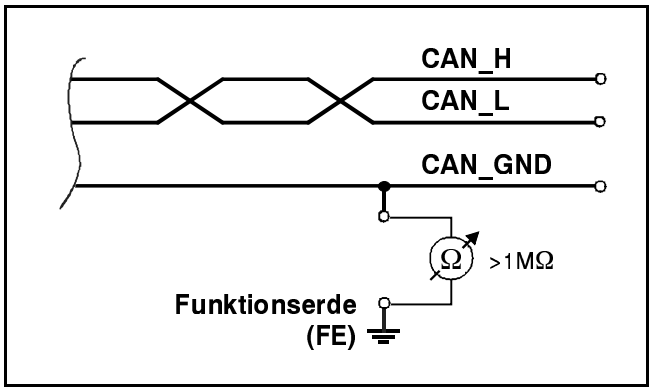


Abb. 19: Vereinfachtes Schaltbild Erdungsmessung

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 (4) aus und trennen Sie ihn vom Netzwerk (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%).

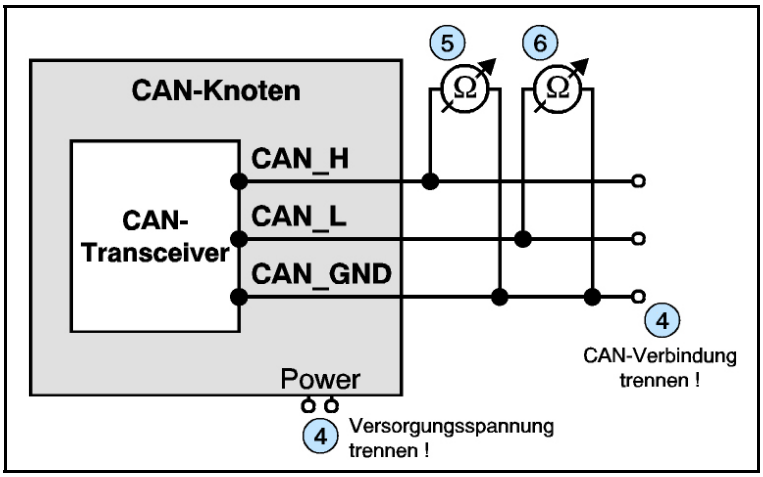


Abb. 20: Messung des Eingangswiderstandes des CAN-Transceivers

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.

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 21 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 SDO 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 (Index=1400_h) (Receive-PDO-Comm-Para)	01_h (COB-def.)	$7F_h$	04_h	00_h	00_h 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 Daten-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

8.5 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
TPDO1	180_h + *NodeID*	PDO1 vom CAN-CBX-Pt100 (Objekt 1800_h)
TPDO2	280_h + *NodeID*	PDO2 vom CAN-CBX-Pt100 (Objekt 1801_h)
TPDO3	380_h + *NodeID*	PDO3 vom CAN-CBX-Pt100 (Objekt 1802_h)
TPDO4	480_h + *NodeID*	PDO4 vom CAN-CBX-Pt100 (Objekt 1803_h)
Client SDO	580_h + *Node-ID*	SDO vom CAN-CBX-Pt100
Server SDO	600_h + *Node-ID*	SDO zum CAN-CBX-Pt100
Node Guarding	700_h + *NodeID*	konfigurierbar über Objekt $100E_h$

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 21) 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 Prozessdaten. Das PDO-Mapping ist veränderbar. Die folgenden Tabellen zeigen das Default-Mapping bei Auslieferung des Moduls:

PDO	CAN-Identifier	Länge	Übertragungsrichtung	Default-Belegung
TPDO1	180_h + Node-ID	4 Byte	vom CAN-CBX-Pt100 (Sende-PDO)	*Process Value 1* (9130_h,1)
TPDO2	280_h + Node-ID	4 Byte		*Process Value 2* (9130_h,2)
TPDO3	380_h + Node-ID	4 Byte		*Process Value 3* (9130_h,3)
TPDO4	480_h + Node-ID	4 Byte		*Process Value 4* (9130_h,4)

Per Default werden die *Process Values* 1 bis 4 (Objekt 9130_h, Sub-Index 1 - 4) gemappt.
Darüber hinaus können auch die *Field Values* 1 bis 4 (Objekt 9100_h, Sub-Index 1 - 4) gemappt werden (siehe dazu auch Seite 82).

TPDO1 (CAN-CBX-Pt100 ->)
CAN-Identifier: 180_h + Node-ID

Byte	0	1	2	3
Parameter	*Process_Value_1*			

TPDO2 (CAN-CBX-Pt100 ->)
CAN-Identifier: 280_h + Node-ID

Byte	0	1	2	3
Parameter	*Process_Value_2*			

TPDO3 (CAN-CBX-Pt100 ->)
CAN-Identifier: 380_h + Node-ID

Byte	0	1	2	3
Parameter	*Process_Value_3*			

TPDO4 (CAN-CBX-Pt100 ->)
CAN-Identifier: 480_h + Node-ID

Byte	0	1	2	3
Parameter	*Process_Value_4*			

8.7 Lesen der Messwerte

8.7.1 Meldung der A/D-Wandler

Die Übertragungsarten für die Messfühlereingänge können unterschieden werden in:

- *azyklisch, synchron*: Die Sendung erfolgt nach dem Empfang einer SYNC-Message (PDO - Übertragungstyp 0), wenn sich die Daten geändert haben.

- *zyklisch, synchron*: Die Sendung erfolgt, nachdem jeweils eine bestimmte Anzahl SYNC-Messages empfangen wurden (PDO-Übertragungstyp 1...240).

- *ereignisgesteuert, asynchron*: Die Sendung erfolgt, wenn sich der Zustand bestimmter Messfühler-eingänge ändert (PDO-Übertragungstyp 254, 255).

8.7.2 Unterstützte Übertragungsarten nach [1]

RTR-Transfers werden vom CAN-CBX-Pt100-Modul nicht unterstützt.

Transmission Type	PDO-Transmission					unterstützt von CAN-CBX-Pt100
	cyclic	acyclic	synchro-nous	asynchro-nous	RTR	
0		X	X			x
1...240	X		X			x
241...253	reserviert					-
254				X		x
255				X		x

8.8 Communication Profile Area

8.8.1 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 65)
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

8.9 Implementierte CANopen-Objekte

Eine detaillierte Beschreibung der Objekte ist in der CiA Standard 301 [1] nachzulesen.

8.9.1 Übersicht der 1000er-Objekt und Default-Werte

Index	Sub-Index (max.)	Beschreibung	Datentyp	Zugriff	Produktspezifische Eigenschaften
1000_h	-	Device Type	unsigned 32	ro	default: 00020194_h
1001_h	-	Error Register	unsigned 8	ro	unterstütze Fehler-Bits: 0: generic 4: communication error 5: device profile 7: manufacturer
1003_h	A_h	Pre-Defined-Error-Field	unsigned 32	rw	default: 0
1005_h	-	COB-ID-Sync	unsigned 32	rw	default: 80_h
1006_h	-	Communication Cycle Periode	unsigned 32	rw	default: 0
1008_h	-	Manufacturer Device Name	visible string	ro	default: “CAN-CBX-PT100”
1009_h	-	Manufacturer Hardware Version	visible string	ro	default ist Versionsabhängig
$100A_h$	-	Manufacturer Software Version	visible string	ro	default ist Versionsabhängig
$100C_h$	-	Guard Time	unsigned 16	rw	default: 0
$100D_h$	-	Life Time Factor	unsigned 8	rw	default: 0
$100E_h$	-	Node Guarding Identifier	unsigned 32	rw	Node-ID + 700_h
1010_h	4	Store Parameter	unsigned 32	rw	-
1011_h	4	Restore Parameter	unsigned 32	rw	-
1014_h	-	COB-ID Emergency Object	unsigned 32	rw	default: 80_h + Node-ID
1015_h	-	Inhibit Time EMCY	unsigned 16	rw	default: 0
1016_h	1	Consumer Heartbeat Time	unsigned 32	rw	default: 0
1017_h	-	Producer Heartbeat Time	unsigned 16	rw	default: 0
1018_h	4	Identity Object	unsigned 32	ro	Vendor Id: 00000017_h Prod. Code: 23032002_h (= C.3032.02)
1019_h	-	Synchronous Counter Overflow	unsigned 8	rw	default: 0
1020_h	2	Verify Configuration	unsigned 32	rw	default: 0
1029_h	1	Error Behaviour	unsigned 8	ro	00_h

Implementierte CANopen Objekte

Index	Sub-index (max.)	Beschreibung	Datentyp	Zugriff
1800_h	5	1. Transmit PDO-Parameter	PDO CommPar (20_h)	rw
1801_h	5	2. Transmit PDO-Parameter	PDO CommPar (20_h)	rw
1802_h	5	3. Transmit PDO-Parameter	PDO CommPar (20_h)	rw
1803_h	5	4. Transmit PDO-Parameter	PDO CommPar (20_h)	rw
$1A00_h$	2	1. Transmit PDO-Mapping	PDO Mappping (21_h)	rw
$1A01_h$	2	2. Transmit PDO-Mapping	PDO Mappping (21_h)	rw
$1A02_h$	2	3. Transmit PDO-Mapping	PDO Mappping (21_h)	rw
$1A03_h$	2	4. Transmit PDO-Mapping	PDO Mappping (21_h)	rw

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)

8.9.2 Device Type (1000_h)

INDEX	**1000_h**
Name	*device type*
Datentyp	unsigned 32
Zugriff	ro
Default-Wert	siehe Kapitel 8.9.1, Seite 53

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-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-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	**94_h**	**01_h**	**02_h**	**00_h**
			Read Response	Index=1000_h		Sub Index	Beispiel hier: Device Profile CiA DS 404		Beispiel hier: Input	

ausgegebener Wert des Device Types in diesem Beispiel: 0002.0194_h.
Der Wert des Device Types für dieses CBX-Modul ist in Kapitel 8.9.1 auf Seite 53 abgedruckt.

Das Datenfeld ist grundsätzlich nach der Regel 'niederwertiges Byte zuerst, höherwertiges Byte zuletzt' aufgebaut (siehe Seite 47, Datenfeld).

8.9.3 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

Bits des Error-Registers:

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

Eine Liste der von diesem Modul unterstützten Fehler-Bits ist in Kapitel 8.9.1 auf Seite 53 abgedruckt.

Die nicht unterstützten Bits werden immer als ‘0’ zurückgegeben. Liegt ein Fehler vor, so ist das entsprechende Fehler-Bit auf ‘1’ gesetzt.

8.9.4 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)
5030_h - Sensor-Fehler

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.

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

INDEX	**1005_h**
Name	*COB-ID SYNC message*
Datentyp	unsigned 32
Zugriff	rw
Default-Wert	siehe Kapitel 8.9.1, Seite 53

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.

8.9.6 Communication Cycle Period (1006_h)

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

Wertebereich des Parameters:

Wert	Bedeutung
0	Übertragung von SYNC-Nachrichten ausgesetzt
1...$FFFFFFFF_h$	Zyklus-Zeit in Mikrosekunden (Beispiel: 4E20h entspricht einer Zykluszeit von 20000µs = 20 ms)

8.9.7 Manufacturer Device Name (1008_h)

INDEX	**1008_h**
Name	*manufacturer device name*
Datentyp	visible string
Default-Wert	siehe Kapitel 8.9.1, Seite 53

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

8.9.8 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.

8.9.9 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.

8.9.10 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 73).

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

8.9.11 Node Guarding Identifier ($100E_h$)

Das Modul unterstützt nur 11-Bit-Identifier.

INDEX	**$100E_h$**
Name	*node guarding identifier*
Datentyp	unsigned 32
Zugriff	rw
Default-Wert	700_h + Node-ID

Struktur des Parameters *node guarding identifier* :

Bit-No.	Bedeutung
31 (MSB) 30	reserved
29...11	immer 0, da keine 29-Bit-ID unterstützt werden
10...0 (LSB)	Bit 0...10 des Node Guarding Identifiers

Der Identifier kann Werte zwischen 1...$7FF_h$ annehmen.

8.9.12 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.

8.9.13 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 werden **nicht** geladen
	1	die Default-Parameter des CAN-CBX-Moduls werden geladen

8.9.14 COB_ID Emergency Message (1014_h)

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

Dieses Objekt bestimmt die COB-ID der Emergency Message (EMCY).

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.

8.9.15 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.

8.9.16 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

8.9.17 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.

Wird für die Producer-Heartbeat-Time ein Wert größer Null eingesetzt, so ist sie aktiv und unterbindet das Node-/ Life-Guarding (siehe Seite 63).
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 (siehe Object $100E_h$).
Die Consumer-Heartbeat-Time der überwachenden Module muss immer größer sein als die Producer-Heartbeat-Time dieses Heartbeat-sendenden Moduls.

8.9.18 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	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
1018_h	0	*no_of_entries*	4	4	unsigned 8	ro
	1	*vendor_id*	0...$FFFFFFFF_h$	0000 0017_h	unsigned 32	ro
	2	*product_code*	0...$FFFFFFFF_h$	siehe Kapitel 8.9.1 ab Seite 53	unsigned 32	ro
	3	*revision_number*	0...$FFFFFFFF_h$	0	unsigned 32	ro
	4	*serial_number*	0...$FFFFFFFF_h$	-	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.
Die Nibble des Langwortes haben folgende Bedeutung:

product_code = `abcd efgh`$_h$

`a`: 1... Artikelnummer beginnt mit Buchstaben “K”
2....Artikelnummer beginnt mit Buchstaben “C”
`bcde`: 4-stellige Hex-Zahl, die als Dezimalzahl vor dem Dezimalpunkt interpretiert wird.
`f`: wird zur Zeit nicht ausgewertet
`gh`: 2-stellige Hex-Zahl, die als Dezimalzahl nach dem Dezimalpunkt interpretiert wird.

Beispiel: ‘2303 2002_h’ entspricht der Artikelnummer ‘C.3032.02’.

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.

8.9.19 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

8.9.20 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.

8.9.21 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	Datentyp	Zugriff
1029_h	0	*no_of_error_classes*	1	1	unsigned 8	ro
	1	*communication_error*	0...2	0	unsigned 8	rw

Bedeutung der Variablen:

Variable	Bedeutung
no_of_error_classes	Anzahl der Fehler-Klassen (hier immer ‘1’)
communication_error	Heartbeat/Lifeguard-Fehler und *Bus off*

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

8.9.22 NMT Startup ($1F80_h$)

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

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

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

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

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.

8.9.24 Objekt Transmit PDO Communication Parameter 1800_h ... 1803_h

Mit diesem Objekten werden die Eigenschaften der Sende-PDOs definiert.

INDEX	**1800_h ... 1803_h**
Name	*transmit PDO parameter*
Data Type	PDOCommPar

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
1800_h	0	*number_of_entries*	5	5	unsigned 8	ro
	1	*COB-ID used by PDO*	1...C00007FF$_h$	40000180$_h$ +Node-ID	unsigned 32	rw
	2	*transmission type*	0...FF$_h$	255	unsigned 8	rw
	3	*inhibit time*	0...FFFF$_h$	0	unsigned 16	rw
	4	*reserved*	0..FF$_h$	0	unsigned 8	const
	5	*event timer*	0...FFFF$_h$	0	unsigned 16	rw
1801_h	0	*number_of_entries*	5	5	unsigned 8	ro
	1	*COB-ID used by PDO*	1...C00007FF$_h$	40000280$_h$ +Node-ID	unsigned 32	rw
	2	*transmission type*	0...FF$_h$	255	unsigned 8	rw
	3	*inhibit time*	0...FFFF$_h$	0	unsigned 16	rw
	4	*reserved*	0..FF$_h$	0	unsigned 8	const
	5	*event timer*	0...FFFF$_h$	0	unsigned 16	rw
1802_h	0	*number_of_entries*	5	5	unsigned 8	ro
	1	*COB-ID used by PDO*	1...C00007FF$_h$	40000380$_h$ +Node-ID	unsigned 32	rw
	2	*transmission type*	0...FF$_h$	255	unsigned 8	rw
	3	*inhibit time*	0...FFFF$_h$	0	unsigned 16	rw
	4	*reserved*	0..FF$_h$	0	unsigned 8	const
	5	*event timer*	0...FFFF$_h$	0	unsigned 16	rw
1803_h	0	*number_of_entries*	5	5	unsigned 8	ro
	1	*COB-ID used by PDO*	1...C00007FF$_h$	40000480$_h$ +Node-ID	unsigned 32	rw
	2	*transmission type*	0...FF$_h$	255	unsigned 8	rw
	3	*inhibit time*	0...FFFF$_h$	0	unsigned 16	rw
	4	*reserved*	0..FF$_h$	0	unsigned 8	const
	5	*event timer*	0...FFFF$_h$	0	unsigned 16	rw

Value range refer [1].
Es werden die *transmission types* 0, 1...240, 254 und 255 unterstützt.

Achtung: Immer RTR-disabled 40000xxx $_h$!

8.9.25 Transmit PDO Mapping Parameter $1A00_h$ ... $1A03_h$

Mit dem Objekten wird die Zuordnung der Sendedaten zu den Tx-PDOs definiert.

INDEX	**$1A00_h$ ... $1A03_h$**
Name	*transmit PDO mapping*
Data Type	PDO Mapping

Die folgende Tabelle zeigt die Belegung der Transmit PDO Mapping Parameter:

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
$1A00_h$	0	*number of entries*	2	**1**	unsigned 8	rw
	1	*Process_Value_1*	0 ... $FFFFFFFF_h$	$9130\ 0120_h$	unsigned 32	rw
	2	*(Field_Value_1)*	0 ... $FFFFFFFF_h$	($9100\ 0120_h$)	unsigned 32	rw
$1A01_h$	0	*number of entries*	2	**1**	unsigned 8	rw
	1	*Process_Value_2*	0 ... $FFFFFFFF_h$	$9130\ 0220_h$	unsigned 32	rw
	2	*(Field_Value_2)*	0 ... $FFFFFFFF_h$	($9100\ 0220_h$)	unsigned 32	rw
$1A02_h$	0	*number of entries*	2	**1**	unsigned 8	rw
	1	*Process_Value_3*	0 ... $FFFFFFFF_h$	$9130\ 0320_h$	unsigned 32	rw
	2	*(Field_Value_3)*	0 ... $FFFFFFFF_h$	($9100\ 0320_h$)	unsigned 32	rw
$1A03_h$	0	*number of entries*	2	**1**	unsigned 8	rw
	1	*Process_Value_4*	0 ... $FFFFFFFF_h$	$9130\ 0420_h$	unsigned 32	rw
	2	*(Field_Value_4)*	0 ... $FFFFFFFF_h$	($9100\ 0420_h$)	unsigned 32	rw

Value range refer [1], table 10, 11.

Es können nur die in der Spalte “Default” angegebenen Objekte gemappt werden.

Struktur des PDO Mapping am Beispiel Objekt $1A00_h$, Sub-Index 01_h:

31 … 16	15 … 8	7 … 0
Index z.B.: 9130_h	Sub-Index z.B.: 01_h	Length z.B.: 20_h

MSB LSB

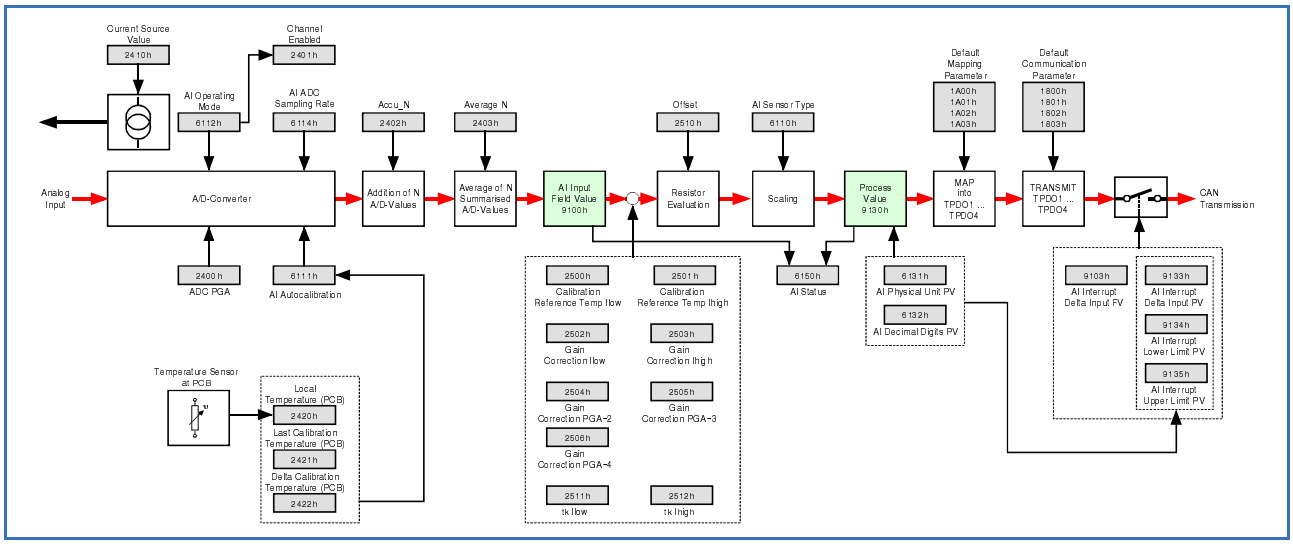
8.10 Device Profile Area

8.10.1 Übersicht der implementierten Objekte 6110_h ...9135_h

Index	Name	Datentyp
6110_h	*AI_sensor_type*	unsigned 16
6111_h	*AI_autocalibration*	unsigned 32
6112_h	*AI_operating_mode*	unsigned 8
6114_h	*AI_ADC_sampling_rate*	unsigned 32
6131_h	*AI_physical_unit_PV*	unsigned 32
6132_h	*AI_decimal_digits_PV*	unsigned 8
6150_h	*AI_status*	unsigned 8
9100_h	*AI_FV*	integer 32
9103_h	*AI_interrupt_delta_input_FV*	integer 32
9130_h	*AI_PV*	integer 32
9133_h	*AI_interrupt_delta_input_PV*	integer 32
9134_h	*AI_interrupt_lower_limit_PV*	integer 32
9135_h	*AI_interrupt_upper_limit_PV*	integer 32

Eine Übersicht des Zusammenwirkens der Objekte der “Device Profile Area” ($61xx_h$ und $91xx_h$) sowie der “Manufacturer Specific Profile Area” ($24xx_h$) zeigt das Diagramm auf der nachfolgenden Seite.

8.10.2 Zusammenhang der implementierten Objekte für die analogen Eingänge



Beispiel hier: Modul befindet sich im Zustand ‘Operational’

8.10.3 AI Sensor Type (6110_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
6110_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_sensor_type_1*	$1E_h$...2732_h	2710_h	unsigned 16	rw
	2	*AI_sensor_type_2*	$1E_h$...2732_h	2710_h	unsigned 16	rw
	3	*AI_sensor_type_3*	$1E_h$...2732_h	2710_h	unsigned 16	rw
	4	*AI_sensor_type_4*	$1E_h$...2732_h	2710_h	unsigned 16	rw

Belegung des Parameters *AI_sensor_type_x* (x = 1...4):

Der Parameter wählt den angeschlossenen Messwiderstand aus. Die Endziffer des Variablen-Namens gibt die Nummer des jeweiligen Messkanals an.

Wertebereich:

Wert des Parameters	Messwiderstandstyp
30 ($1E_h$)	Pt100-Temperaturfühler
31	Pt200-Temperaturfühler
32	Pt500-Temperaturfühler
33	Pt1000-Temperaturfühler
34	Pt5000-Temperaturfühler
10000 (2710_h)	Widerstandsmessung
10030	Ni100-Temperaturfühler
10031	Ni200-Temperaturfühler
10032	Ni500-Temperaturfühler
10033	Ni1000-Temperaturfühler
10034 (2732_h)	Ni5000-Temperaturfühler

Hinweis:
Bitte beachten Sie beim Anschluss der Temperaturfühler das der genaue Typ des Temperaturfühlers in diesem Parameter eingetragen ist.
Standardmäßig enthält der Parameter *AI_sensor_type_x* den Wert für die Widerstandsmessung (10000).

8.10.4 AI Autocalibration (6111_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
6111_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_autocalibration_1*	no default, write: 69 6C 61 63_h (= ASCII: ‘i’ ‘l’ ‘a’ ‘c’)		unsigned 32	wo
	2	*AI_autocalibration_2*			unsigned 32	wo
	3	*AI_autocalibration_3*			unsigned 32	wo
	4	*AI_autocalibration_4*			unsigned 32	wo

Funktion:
Schreiben des ASCII-Strings “cali” erzwingt die Selbstkalibrierung des entsprechenden Messkanals.

Die Selbstkalibrierung wird ansonsten, wie der Name sagt, eingenständig gestartet:

- nach Erkennung neuer Sensoren

- nach Änderung der Paramter
  - Sampling-Rate *AI_ADC_sampling_rate* (Objekt 6114_h) oder
  - Gain *ADC_PGA* (Objekt 2400_h) oder
  - Messstrom *current_source_value* (Objekt 2410_h)

- nach Überschreitung einer definierten Temperaturänderung der Leiterkarte seit der letzten Kalibrierung (sofern aktiviert)
  - letzte Temperatur bei der eine Kalibrierung stattgefunden hat
    last_calibration_temp (Objekt 2421_h)
  - max. zulässige Temperaturdifferenz seit letzter Kalibrierung
    delta_calibration_temp (Objekt 2422_h)

8.10.5 AI Operating Mode (6112_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
6112_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_operating_mode_1*	0, 1	1	unsigned 8	rw
	2	*AI_operating_mode_2*	0, 1	1	unsigned 8	rw
	3	*AI_operating_mode_3*	0, 1	1	unsigned 8	rw
	4	*AI_operating_mode_4*	0, 1	1	unsigned 8	rw

Belegung des Parameters *AI_operating_mode_x* (x = 1...4):

Der Parameter wählt die gewünschte Betriebsart aus. Er gibt den “Soll-Wert” des Operating Modes vor. Der “Istwert” ist über Objekt 2401_h lesbar.

Wertebereich:

Wert des Parameters	Betriebsart des Messkanals
0	Kanal ausgeschaltet
1	normaler Betrieb

8.10.6 AI ADC Sampling Rate (6114_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
6114_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_ADC_sampling_rate_1*	$03E8_h$...$61A80_h$	$61A80_h$	unsigned 32	rw
	2	*AI_ADC_sampling_rate_2*	$03E8_h$...$61A80_h$	$61A80_h$	unsigned 32	rw
	3	*AI_ADC_sampling_rate_3*	$03E8_h$...$61A80_h$	$61A80_h$	unsigned 32	rw
	4	*AI_ADC_sampling_rate_4*	$03E8_h$...$61A80_h$	$61A80_h$	unsigned 32	rw

Belegung des Parameters *AI_ADC_sampling_rate_x* (x = 1...4):

Mit diesem Parameter wird die Sampling-Rate kanalweise eingestellt. Die Auflösung beträgt 1 µs. Folgende Werte werden unterstützt:

Wertebereich:

Zulässige Parameterwerte *2)		Sample-Time	Sampling-Rate
1000	($03E8_h$)	1 ms	1000 Hz
2000		2 ms	500 Hz
10000		10 ms	100 Hz
16667		16,667 ms	60 Hz
20000	($4E20_h$)	20 ms	50 Hz
33333		33,333 ms	30 Hz
40000		40 ms	25 Hz
66667		66,667 ms	15Hz
100000		100 ms	10 Hz
200000		200 ms	5 Hz
400000	($61A80_h$)	400 ms	2,5 Hz

*2) Wird ein Wert übertragen, der nicht zu den zulässigen Paramterwerten gehört, so wird auf den nächstgelegenen zulässigen Wert gerundet.

Hinweis:
Empfehlenswert für eine hohe Auflösung sind Sample-Times, die ein Vielfaches von 20 ms (50 Hz) betragen, da dadurch Störungen durch die Netzfrequenz minimiert werden!

8.10.7 AI Physical Unit (6131_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
6131_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_physical_unit_1*	280000_h, $2D0000_h$	$unit_{SensorType}$_1	unsigned 32	ro
	2	*AI_physical_unit_2*	280000_h, $2D0000_h$	$unit_{SensorType}$_2	unsigned 32	ro
	3	*AI_physical_unit_3*	280000_h, $2D0000_h$	$unit_{SensorType}$_3	unsigned 32	ro
	4	*AI_physical_unit_4*	280000_h, $2D0000_h$	$unit_{SensorType}$_4	unsigned 32	ro

Die Default-Werte $unit_{SensorType}$_x (x = 1 - 4) sind abhängig vom eingestellten Sensor-Typ.:
Mit der Default-Einstellung von Object 6110_h = 10000_d (= Widerstandsmessung) wäre dies 2621440_d = Widerstand in Ohm.

Belegung der Variablen *AI_physical_unit_x* (x = 1...4):

Diese Variablen geben die verwendete physikalische Einheit der Messkanäle und die Präfixe wieder. Der Wert der Variablen wird von dem über Objekt "*AI_sensor_type*" (6110_h) gewählten Sensor-Typ bestimmt.

Die Struktur der Variablen ist in CiA404 festgelegt. Die Kodierung der Physikalischen Einheiten und Präfixe erfolgt gemäß CiA303-2 [5].
Die Struktur der Variablen ist auf der folgenden Seite beschrieben.

Struktur der Variablen:

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

MSB LSB

Kodierung der Präfixe für physikalische Einheiten:

Prefix	Faktor	Notation Index
-	10^0	00_h

SI Numerator und SI Denominator enthalten die physikalische Einheiten.

Kodierung der physikalischen Einheiten :

Internationales Symbol	Name der Einheit	Notation Index
Ω	Ohm	28_h
°C	Grad Celsius	$2D_h$
dimensionslos	keine	0

Wertebereich:

Sensor-Typ	Physikalische Einheit
Pt100 ... Pt5000	Grad Celsius (Variablen-Wert=$2D0000_h$)
Ni100 ... Ni5000	
Widerstandsmessung	Ohm (Variablen-Wert=280000_h)

8.10.8 AI Decimal Digits PV (6132_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
6132_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_decimal_digits_PV_1*	0...3	3	unsigned 8	rw
	2	*AI_decimal_digits_PV_2*	0...3	3	unsigned 8	rw
	3	*AI_decimal_digits_PV_3*	0...3	3	unsigned 8	rw
	4	*AI_decimal_digits_PV_4*	0...3	3	unsigned 8	rw

Belegung des Parameters *AI_decimal_digits_PV_x* (x = 1...4):

Dieser Parameter gibt die Anzahl der Nachkommastellen des PV im Objekt "*AI_PV*" (9130_h) vor. Der Wert des Parameters entspricht der Anzahl der Stellen.

Beispiel:

AI_decimal_digits_PV_1 = 3
Wert in Objekt *AI_PV_1* = 123456_d

Messwert: 123,456 Ω (bei der Standardeinstellung Widerstandsmessung, siehe Objekt 6110_h)

8.10.9 AI Status (6150_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
6150_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_status_1*	0...5	-	unsigned 8	ro
	2	*AI_status_2*	0...5	-	unsigned 8	ro
	3	*AI_status_3*	0...5	-	unsigned 8	ro
	4	*AI_status_4*	0...5	-	unsigned 8	ro

Belegung der Variablen *AI_status_x* (x = 1...4):

Diese Variablen geben die Stati der analogen Eingangskanäle zurück.

Bit:	7	6	5	4	3	2	1	0
Bedeutung:	reserviert *3)	nicht unterstützt*3)			reserviert *3)	Negative Overload	Positive Overload	Not Valid

*3) Diese Bits werden immer als ‘0’ zurückgegeben.

Wertebereich:

Sensor-Typ	Negative Overload-Grenze	Positive Overload-Grenze
PT-Sensoren	-200,715 ˚C	853,498 ˚C
NI-Sensoren	-200,000 ˚C	399,138 ˚C
Elektr.Widerstandsmessung bei Default-Belegung	-	-

Ein Bit ist aktiv (auf ‘1’ gesetzt), wenn der untere oder obere Grenzwert unter, bzw. überschritten ist. Die Bits 1 und 2 können nie gleichzeit gesetzt sein.

Ist kein Sensor angeschlossen, wird nur das Bit “Not Valid” gesetzt.
Bei Temperaturen außerhalb des Bereichs werden die Bits “Not Valid” UND das entsprechende “Overload”-Bit gesetzt.

Gültige Rückmeldungen sind z.B.:
AI_Status_1 = 00_h : kein Fehler
AI_Status_1 = 03_h : Positive Overload und Sensor-Wert ungültig (z.B. kein Sensor erkannt)

8.10.10 AI Input Field Value (9100_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
9100_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_FV_1*	80000000_h ... $7FFFFFFF_h$	-	integer 32	ro
	2	*AI_FV_2*	80000000_h ... $7FFFFFFF_h$	-	integer 32	ro
	3	*AI_FV_3*	80000000_h ... $7FFFFFFF_h$	-	integer 32	ro
	4	*AI_FV_4*	80000000_h ... $7FFFFFFF_h$	-	integer 32	ro

Belegung der Variablen *AI_FV_x* (x = 1...4):

Diese Variablen geben die unkorrigierten “Rohdaten” der A/D-Wandler zurück.

Wertebereich:

AI_FV_x = $8000\ 0000_h$ ... $7FFF\ FFFF_h$

8.10.11 AI Interrupt Delta Input FV (9103_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
9103_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_interrupt_delta_input_FV_1*	80000000_h ... $7FFFFFFF_h$	0	integer 32	rw
	2	*AI_interrupt_delta_input_FV_2*	80000000_h ... $7FFFFFFF_h$	0	integer 32	rw
	3	*AI_interrupt_delta_input_FV_3*	80000000_h ... $7FFFFFFF_h$	0	integer 32	rw
	4	*AI_interrupt_delta_input_FV_4*	80000000_h ... $7FFFFFFF_h$	0	integer 32	rw

Belegung des Parameters *AI_interrupt_delta_input_FV_x* (x = 1...4):

Ist ein Field-Value auf ein PDO gemappt und ändert sich der Wert des Field-Values um mehr als in *AI_interrupt_delta_input_FV*_x angegeben, so wird dieses PDO gesendet.

Wertebereich:

*AI_interrupt_delta_input_FV*_x	Bedeutung
$8000\ 0000_h$ ... $FFFF\ FFFF_h$	bei negativen Werten des Parameters wird ein PDO-Versand mit jeder Änderung des Field Values initiiert
0	kein Vergleich des Field Values und somit kein Versand
$0000\ 0001_h$ .... $7FFF\ FFFF_h$	es wird nur PDO-Versand initiiert, wenn die Änderung des Field Values größer ist als der hier vorgegebene Wert

8.10.12 AI Input Process Value (9130_h)

Hinweis:
Dieses Objekt enthält die wichtigen Daten (Widerstands oder Temperatur-werte)!

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
9130_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_PV_1*	80000000_h ... $7FFFFFFF_h$	-	integer 32	ro
	2	*AI_PV_2*	80000000_h ... $7FFFFFFF_h$	-	integer 32	ro
	3	*AI_PV_3*	80000000_h ... $7FFFFFFF_h$	-	integer 32	ro
	4	*AI_PV_4*	80000000_h ... $7FFFFFFF_h$	-	integer 32	ro

Belegung der Variablen *AI_PV_x* (x = 1...4):

Diese Variablen geben die korrigierten, einheitenbehafteten Messwerte der vier Kanäle zurück.
Die physikalischen Einheiten der Werte sind in "*AI_physical_unit_x*" enthalten (Objekt $6131_{h,}$, siehe Seite 89).

Die Anzahl der Nachkommastellen der Variablen ist in Objekt "*AI_decimal_digits_PV_x*" (Objekt $6132_{h,}$ siehe Seite 91) definiert.

Wertebereich:

AI_PV_x = $8000\ 0000_h$ ... $7FFF\ FFFF_h$

8.10.13 AI Interrupt Delta Input PV (9133_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
9133_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_interrupt_delta_input_PV_1*	80000000_h ... $7FFFFFFF_h$	0	integer 32	rw
	2	*AI_interrupt_delta_input_PV_2*	80000000_h ... $7FFFFFFF_h$	0	integer 32	rw
	3	*AI_interrupt_delta_input_PV_3*	80000000_h ... $7FFFFFFF_h$	0	integer 32	rw
	4	*AI_interrupt_delta_input_PV_4*	80000000_h ... $7FFFFFFF_h$	0	integer 32	rw

Belegung des Parameters *AI_interrupt_delta_input_PV_x* (x = 1...4):

Ist eine Prozess-Variable auf ein PDO gemappt und ändert sich der Wert der Prozess-Variablen um mehr als in *AI_interrupt_delta_input_PV*_x angegeben, so wird dieses PDO gesendet.

Die Anzahl der Nachkommastellen der Variablen ist in Objekt "*AI_decimal_digits_PV_x*" (Objekt $6132_{h,}$ siehe Seite 91) definiert.

Wertebereich:

*AI_interrupt_delta_input_PV*_x	Bedeutung
$8000\ 0000_h$ ... $FFFF\ FFFF_h$	bei negativen Werten des Parameters wird ein PDO-Versand mit jeder Änderung des Process Values initiiert
0	kein Vergleich des Process Values und somit kein Versand
$0000\ 0001_h$ .... $7FFF\ FFFF_h$	es wird nur PDO-Versand initiiert, wenn die Änderung des Process Values größer ist als der hier vorgegebene Wert

8.10.14 AI Interrupt Lower Limit PV (9134_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
9134_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_interrupt_lower_limit_PV_1*	80000000_h ... $7FFFFFFF_h$	80000000_h	integer 32	rw
	2	*AI_interrupt_lower_limit_PV_2*	80000000_h ... $7FFFFFFF_h$	80000000_h	integer 32	rw
	3	*AI_interrupt_lower_limit_PV_3*	80000000_h ... $7FFFFFFF_h$	80000000_h	integer 32	rw
	4	*AI_interrupt_lower_limit_PV_4*	80000000_h ... $7FFFFFFF_h$	80000000_h	integer 32	rw

Belegung des Parameters *AI_interrupt_lower_limit_PV_x* (x = 1...4):

Ist eine Prozess-Variable auf ein PDO gemappt und sinkt der Wert der Prozess-Variablen unter den in *AI_interrupt_lower_limit_PV*_x angegeben Wert, so wird dieses PDO gesendet.

Die Anzahl der Nachkommastellen der Variablen ist in Objekt "*AI_decimal_digits_PV_x*" (Objekt $6132_{h,}$ siehe Seite 91) definiert.

Wertebereich:

*AI_interrupt_lower_limit_PV*_x = $8000\ 0000_h$ ... $7FFF\ FFFF_h$

8.10.15 AI Interrupt Upper Limit PV (9135_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
9135	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*AI_interrupt_upper_limit_PV_1*	80000000_h ... $7FFFFFFF_h$	$7FFFFFFF_h$	integer 32	rw
	2	*AI_interrupt_upper_limit_PV_2*	80000000_h ... $7FFFFFFF_h$	$7FFFFFFF_h$	integer 32	rw
	3	*AI_interrupt_upper_limit_PV_3*	80000000_h ... $7FFFFFFF_h$	$7FFFFFFF_h$	integer 32	rw
	4	*AI_interrupt_upper_limit_PV_4*	80000000_h ... $7FFFFFFF_h$	$7FFFFFFF_h$	integer 32	rw

Belegung des Parameters *AI_interrupt_upper_limit_PV_x* (x = 1...4):

Ist eine Prozess-Variable auf ein PDO gemappt und steigt der Wert der Prozess-Variablen über den in *AI_interrupt_upper_limit_PV*_x angegeben Wert, so wird dieses PDO gesendet.

Die Anzahl der Nachkommastellen der Variablen ist in Objekt "*AI_decimal_digits_PV_x*" (Objekt $6132_{h,}$ siehe Seite 91) definiert.

Wertebereich:

*AI_interrupt_upper_limit_PV*_x = $8000\ 0000_h$ ... $7FFF\ FFFF_h$

8.11 Manufacturer Specific Profile Area

8.11.1 Übersicht der Manufacturer Specifc Objects 2400_h ... 2512_h

Index	Name	Datentyp
2400_h	*ADC_PGA*	unsigned 8
2401_h	*Channel_enabled*	unsigned 8
2402_h	*Accu_N*	unsigned 8
2403_h	*Average_N*	unsigned 8
2410_h	*Measuring_current*	unsigned 8
2420_h	*Local_temperature (PCB)*	integer 16
2421_h	*Last_calibration_temperature (PCB)*	integer 16
2422_h	*Delta_calibration_temperature (PCB)*	integer 16
2500_h	*Ref_temp_Ilow*	integer 16
2501_h	*Ref_temp_Ihigh*	integer16
2502_h	*Gain_correction_Ilow*	integer 16
2503_h	*Gain_correction_Ihigh*	integer 16
2504_h	*Gain_correction_PGA=2*	integer 16
2505_h	*Gain_correction_PGA=3*	integer 16
2506_h	*Gain_correction_PGA=4*	integer 16
2510_h	*Offset*	integer 16
2511_h	*tk_Ilow*	integer 16
2512_h	*tk_Ihigh*	integer 16

Siehe hierzu auch das Diagramm “Zusammenhang der implementierten Objekte für die analogen Eingänge” auf Seite 84.

8.11.2 ADC_PGA (2400_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2400_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*ADC_PGA_1*	1...4	3	unsigned 8	rw
	2	*ADC_PGA_2*	1...4	3	unsigned 8	rw
	3	*ADC_PGA_3*	1...4	3	unsigned 8	rw
	4	*ADC_PGA_4*	1...4	3	unsigned 8	rw

Belegung des Parameters *ADC_PGA_x* (x = 1...4):
Dieser Parameter stellt den Verstärkungsfaktor (PGA) der A/D-Wandler (ADS1255) ein.

Wertebereich:
Der Verstärkungsfaktor V_x ergibt sich wie folgt:

$$V_x = 2^{ADC_PGA_x}$$

Es sind also die Verstärkungsfaktoren 2, 4, 8 und 16 einstellbar. Der Verstärkungsfaktor ist für den Anwender transparent und schon in die Firmware eingerechnet.

Achtung:
Beim Anschluss der Messwiderstände sind folgende Spannungsgrenzen zu berücksichtigen: Spannungsbegrenzung durch Eingangsbeschaltung: U_{maxph} = 1,25 V

Daraus ergeben sich maximal zulässige Widerstandswerte in Abhängigkeit von *ADC_PGA_x*.

Gain (*ADC_PGA_x*)	Eingangs-spannung U_{FS} [V]	max. zulässiger Messwiderstand bei einem Messstrom von	
		ca. 400 µA	ca. 40 µA
1	± 2,5	3 kΩ	34 kΩ
2	± 1,25	3 kΩ	34 kΩ
3	± 0,6125	1,5 kΩ	17 kΩ
4	± 0,306	773 Ω	8,5 kΩ

Hinweis:
Es wird empfohlen die Default-Einstellung *ADC_PGA_x* = 3 beizubehalten!

8.11.3 Channel Enabled (2401_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2401_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*channel_enabled_1*	0, 1	-	boolean	ro
	2	*channel_enabled_2*	0, 1	-	boolean	ro
	3	*channel_enabled_3*	0, 1	-	boolean	ro
	4	*channel_enabled_4*	0, 1	-	boolean	ro

Belegung der Variablen *channel_enabled_x* (x = 1...4):

Wurde für einen Messkanal ein Sensor erkannt, dann gibt diese Variable den “Ist-Wert”des Objektes “*AI_Operating_mode_x*” (6112_h) zurück. Sie zeigt an, ob der jeweilige Kanal aktiv ist.

Ob ein Sensor erkannt wurde, kann über das Objekt 6150_h “*AI_status_x*” ermittelt werden.

Wertebereich:

channel_enabled_x	Binär-Wert	Bedeutung
false	0	A/D-Wandler-Kanal ist ausgeschaltet (*AI_operation_mode_x* = 0)
true	1	A/D-Wandler-Kanal ist an (*AI_operation_mode_x* = 1)

8.11.4 Accu N (2402_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2402_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*accu_count_1*	0...8	0	unsigned 8	rw
	2	*accu_count_2*	0...8	0	unsigned 8	rw
	3	*accu_count_3*	0...8	0	unsigned 8	rw
	4	*accu_count_4*	0...8	0	unsigned 8	rw

Belegung des Parameters *accu_count_x* (x = 1...4):

Mit diesem Parameter wird definiert, wie viele Analogwerte aufaddiert werden sollen. Der Parameter *accu_count_x* reduziert die Datenrate und erhöht die Auflösung.
Die Anzahl der Additionen errechnet sich wie folgt:

n ... Anzahl der Additionen

$$n = 2^{(accu_count_x)}$$

Es können also bis zu 256 Werte aufaddiert werden.

Features:
- Filter mit Decimation
- Verbesserung der Auflösung
- Verringerung der Datenrate

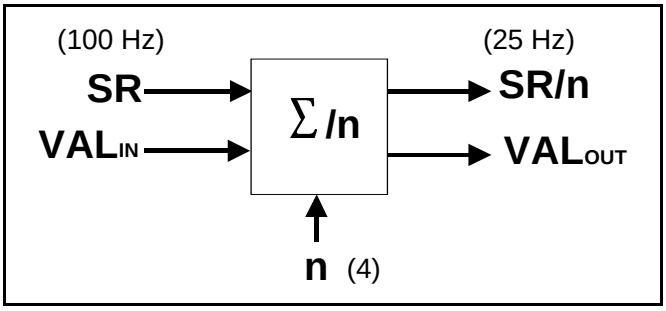
Die Datenrate des Analogwerts berechnet sich demnach zu:

$$T_{Data} = 2^n \cdot T_sampling_ADC \text{ (Objekt } 6114_h\text{)}$$

Beschreibung der Filter:

SR ... Sample Rate (z.B.:100 Hz)
n ... Anzahl der Additionen (z.B.:4)
SR/n... Sample Rate geteilt durch Anzahl der Summationen (e.g.:25 Hz)

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



8.11.5 Average N (2403_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2403_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*average_count_1*	0...4	0	unsigned 8	rw
	2	*average_count_2*	0...4	0	unsigned 8	rw
	3	*average_count_3*	0...4	0	unsigned 8	rw
	4	*average_count_4*	0...4	0	unsigned 8	rw

Belegung des Parameters *average_count_x* (x = 1...4):

Mit diesem Parameter wird definiert, aus wie vielen Analogwerten ein gleitender Mittelwert (Floating Average) gebildet wird.
Da die A/D-Werte in einem Ringpuffer zwischengespeichert werden, steht nach jeder Wandlung ein neuer Mittelwert (Average) zur Verfügung.

Die Anzahl der gemittelten Werte errechnet sich wie folgt:

m ... Anzahl gemittelter Werte

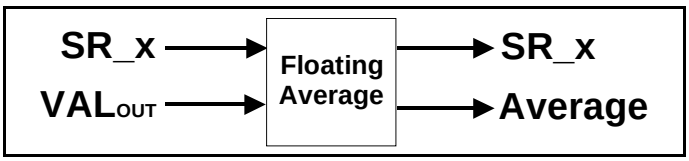
$$m = 2^{(average_count_x)}$$

Es kann also über die letzten 1, 2, 4, 8 oder 16 Werte gemittelt werden.

Features:
- Filter mit Decimation
- Verbesserung der Auflösung
- nach jeder Wandlung steht ein neuer Mittelwert zur Verfügung
- Einschwingzeit nach einem analogen Sprung beträgt $2^m \cdot T_{Data}$ (siehe Objekt 2402_h)

Hinweis:
"Average" (2403_h) setzt auf die Daten von "Accu" (2402_h) auf!

Beschreibung der Filter:
SR_x ... Sample-Rate



8.11.6 Current Source Value (2410_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2410_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*current_source_value_1*	0, 1	0	unsigned 8	rw
	2	*current_source_value_2*	0, 1	0	unsigned 8	rw
	3	*current_source_value_3*	0, 1	0	unsigned 8	rw
	4	*current_source_value_4*	0, 1	0	unsigned 8	rw

Belegung des Parameters *current_source_value_x* (x = 1...4):

Mit diesem Parameter wird der Messstrom ausgewählt.

Für Messwiderstände bis etwa 500 Ω kann der Messstrom 400 µA betragen. Darüber hinaus sollte die Eigenerwärmung des Sensors in Betracht gezogen werden und ggf. der niedrige Messstrom gewählt werden.
Wird der niedrige Messstrom gewählt, sollte der Parameter *ADC_PGA_x* (Objekt 2400_h) auf “4” gesetzt werden (zur Auswahl von Messstrom und Messwiderstand siehe auch Seite 100).

Wertebereich:

current_source_value_x = 0 : Messstrom = ca. 400 µA
current_source_value_x = 1 : Messstrom = ca. 40 µA

8.11.7 Aktuelle Temperatur der Leiterkarte (2420_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2420_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*local_temperature*	8000_h...$7FFF_h$	-	integer 16	ro

Belegung der Variablen *local_temperature*:

Diese Variable enthält den Wert der Temperatur auf der Leiterkarte in Schritten zu 1/256 ˚C .

Der Wert ist wie folgt kodiert:

$$Temperature[°C] = \frac{Int16_Value}{256_d}$$

Beispiel:

local_temperature =2640_h => Temperatur = 38,25 ˚C

8.11.8 Temperatur der Leiterkarte während der letzten Kalibrierung (2421_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2421_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*last_calibration_temp_1*	8000_h...$7FFF_h$	$temp_{_last_cal_}1$	integer 16	ro
	2	*last_calibration_temp_2*	8000_h...$7FFF_h$	$temp_{_last_cal_}2$	integer 16	ro
	3	*last_calibration_temp_3*	8000_h...$7FFF_h$	$temp_{_last_cal_}3$	integer 16	ro
	4	*last_calibration_temp_4*	8000_h...$7FFF_h$	$temp_{_last_cal_}4$	integer 16	ro

Als Default-Wert $temp_{_last_cal_}x$ (x = 1 - 4) ist jeweils der beim letzten Kalibrieren ermittelte Temperaturwert eingetragen.

Belegung der Variablen *last_calibration_temp_x*:

Diese Variable enthält den Wert der Temperatur auf der Leiterkarte in Schritten zu 1/256 ˚C, bei dem die letzte Kalibrierung des betreffenden Eingangs stattfand.

Der Wert ist wie folgt kodiert:

$$Temperature[°C] = \frac{Int16_Value}{256_d}$$

Beispiel:

last_calibration_temp_1 = 2640_h => Temperatur = 38,25 ˚C

8.11.9 Zulässige Temperaturänderung der Leiterkarte bis Neukalibrierung (2422_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2422_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*delta_calibration_temp_1*	0...$FFFF_h$	0	unsigned 16	ro
	2	*delta_calibration_temp_2*	0...$FFFF_h$	0	unsigned 16	ro
	3	*delta_calibration_temp_3*	0...$FFFF_h$	0	unsigned 16	ro
	4	*delta_calibration_temp_4*	0...$FFFF_h$	0	unsigned 16	ro

Belegung der Variablen *delta_calibration_temp_x*:

Ändert sich die Temperatur auf der Leiterkarte (2420_h) um mehr als den in *delta_calibration_temp_x* angegebenen Wert, wird eine Self Calibration des ADS1255 durchgeführt.

Wert *delta_calibration_temp_x*	Funktion
0	keine automatische Selbstkalibrierung bei Temperaturänderung
0001_h...$FFFF_h$	automatische Selbstkalibrierung bei Überschreitung der vorgegebenen Temperaturänderung

Der Wert ist wie folgt kodiert:

$$Temperature[°C] = \frac{Int16_Value}{256_d}$$

Beispiel:

delta_calibration_temp_1 = $0A80_h$ => Temperatur-Änderung = 10,50 °C

8.11.10 Kalibrierdaten

Die analogen Eingänge des CAN-CBX-Pt100-Moduls sind werksseitig vor der Auslieferung kalibriert worden. Diese Kalibrierung wurde für

- den hohen Messstrom (ca. 400 µA)
- den niedrigen Messstrom (ca. 40µA)
- die ADC-Verstärkung PGA=2
- die ADC-Verstärkung PGA=3
- die ADC-Verstärkung PGA=4

durchgeführt.

Die dabei ermittelten Parameter sind in den folgenden Objekten lesbar und veränderbar. Eine Anpassung der Objekte durch den Anwender ist in der Regel jedoch nicht notwendig.
Lediglich beim Objekt “offset” ist eine Anpassung durch den Anwender sinnvoll, um die Leitungswiderstände zu kompensieren.

Wird das Kommando *Restore Default Parameters* (1011_h) aufgerufen, werden die ursprünglich bei esd ermittelten Kalibrierdaten wieder aktiviert.

8.11.10.1 Kalibrierung und Process/Field-Value Ermittlung

In den folgenden Formeln wurde aus Gründen der Übersichtlichkeit auf die Kanalkennung am Ende der Parameternamen verzichtet.

Die im folgenden dargestellten Berechnungen werden von der lokalen Firmware automatisch durchgeführt. Dem Anwender stehen die korrigierten Messwerte über die Prozess-Variablen PV zur Verfügung.

Field-Value Ermittlung

$$AI_FV = \frac{adc_raw_value}{2^{ADC_PGA}}$$

mit

adc_rah_value:	A/D-Wandler-Wert nach Addition und Mittelwertbildung
AI_FV [Objekt 9100_h]:	Field Value
ADC_PGA [Objekt 2400_h]:	ADC-Verstärkung

Process-Value Ermittlung

1) Korrektur FV

$$FV_{corr} = AI_FV\left(1 + \frac{corr}{2^{22}}\right)$$

mit

FV_{corr}:	interner korrigierter Field-Value in ADC-Einheiten
`AIinFV` [Objekt 9100_h]:	analoger Input Field Value
`corr`:	interner Korrekturfaktor

$$corr = gc_Ih + gc_PGAy + \left(local_temp - ref_temp_Ih\right)\frac{tk_Ih}{256}$$

mit

`gc_lh` [Objekte 2502_h, 2503_h]:	Gain-Korrekturfaktoren (**g**ain **c**orrection I**l**ow/I**h**igh) für den niedrigen und den hohen Messstrom
`gc_PGAy` [Objekte 2504_h, 2505_h, 2506_h]: (y= 2,3,4)	Gain-Korrektur für die Verstärkungen PGA=2 [2504_h], PGA=3 [2505_h] und PGA=4 [2506_h] (keine Korrektur bei PGA=1)
`local_temp` [Objekt 2420_h]:	bei der Kalibrierung gemessene Temperatur der Bauteile dieses Kanals auf der Leiterplatte
`ref_temp_lh` [Objekte 2500_h, 2501_h]:	bei der Kalibrierung aufgenommene Referenz-temperaturen für den niedrigen und den hohen Messstrom
`tk_lh` [Objekte 2511_h, 2512_h]:	bei der Kalibrierung ermittelter Temperaturkoeffizient für den niedrigen und den hohen Messstrom

2) Berechnung PV_R (Widerstandswert)

$$PV_R = \left(k_Ih \times FV_{corr}\right) + offset$$

`PV(R)`:	interner Widerstandswert (nur bei Widerstandsmessung (siehe Objekt 6110_h) für den Anwender zugänglich)
`k_lh`:	interner Umrechnungsfaktor für den niedrigen und den hohen Messstrom (nicht zugänglich für den Anwender); *k_Ilow/Ihigh* = f(*current_source_value* [Objekt 2410_h])
`offset` [Objekt 2510_h]:	Offset-Wert x zur Kompensation der angeschlossenen Leitungswiderstände

3) Temperaturberechnung

$$PV_T = Tab\left(PV_R\right)$$

mit

PV(T) [Objekt 9130_h]:	Process-Value (gemappt auf PDOs)
Tab:	Anwendung der Sensor-Typ-Umrechnungstabelle *Tab* = f (*AI_sensor_type* [Objekt 6110_h])

8.11.10.2 Referenztemperaturen bei der Kalibrierung (2500_h, 2501_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2500_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*ref_temp_Ilow_1*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	2	*ref_temp_Ilow_2*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	3	*ref_temp_Ilow_3*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	4	*ref_temp_Ilow_4*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
2501_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*ref_temp_Ihigh_1*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	2	*ref_temp_Ihigh_2*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	3	*ref_temp_Ihigh_3*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	4	*ref_temp_Ihigh_4*	8000_h...$7FFF_h$	*[6)]	integer 16	rw

*[6)] Die Default-Werte sind bei der Kalibrierung für jedes Moduls individuell ermittelt worden und werden daher in dieser allgemeinen Dokumentation nicht abgedruckt.

Belegung der Parameter *ref_temp_Ilow_x* und *ref_temp_Ihigh_x* (x = 1...4):

Diese Objekte dienen zu Speicherung der Bezugstemperaturen, bei denen die Kanäle kalibriert worden sind. Die Temperaturwerte sind in in Schritten zu 1/256 ˚C gespeichert.

Der Wert ist wie folgt kodiert:

$$Temperature[°C] = \frac{Int16_Value}{256_d}$$

Beispiel:

ref_temp_Ilow =2640_h => Temperatur = 38,25 ˚C

8.11.10.3 Verstärkungskorrektur bei I_{LOW} und I_{HIGH} (2502_h, 2503_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2502_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*gain_correction_Ilow_1*	8000_h...$7FFF_h$	*6)	integer 16	rw
	2	*gain_correction_Ilow_2*	8000_h...$7FFF_h$	*6)	integer 16	rw
	3	*gain_correction_Ilow_3*	8000_h...$7FFF_h$	*6)	integer 16	rw
	4	*gain_correction_Ilow_4*	8000_h...$7FFF_h$	*6)	integer 16	rw
2503_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*gain_correction_Ihigh_1*	8000_h...$7FFF_h$	*6)	integer 16	rw
	2	*gain_correction_Ihigh_2*	8000_h...$7FFF_h$	*6)	integer 16	rw
	3	*gain_correction_Ihigh_3*	8000_h...$7FFF_h$	*6)	integer 16	rw
	4	*gain_correction_Ihigh_4*	8000_h...$7FFF_h$	*6)	integer 16	rw

*6) Die Default-Werte sind bei der Kalibrierung für jedes Moduls individuell ermittelt worden und werden daher in dieser allgemeinen Dokumentation nicht abgedruckt.

Belegung der Parameter *gain_correction_Ilow_x und gain_correction_Ihigh_x* (x = 1...4):

Diese Objekte beinhalten die Gain-Korrekturfaktoren, die bei der Kalibrierung der Kanäle bei I_{low} und I_{high} bei PGA=1 ermittelt worden sind.

Wertebereich: 8000_h ... $7FFF_h$

8.11.10.4 Verstärkungskorrektur für PGA=2/3/4 (2504_h, 2505_h, 2506_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2504_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*gain_correction_PGA=2_1*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	2	*gain_correction_PGA=2_2*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	3	*gain_correction_PGA=2_3*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	4	*gain_correction_PGA=2_4*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
2505_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*gain_correction_PGA=3_1*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	2	*gain_correction_PGA=3_2*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	3	*gain_correction_PGA=3_3*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	4	*gain_correction_PGA=3_4*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
2506_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*gain_correction_PGA=4_1*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	2	*gain_correction_PGA=4_2*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	3	*gain_correction_PGA=4_3*	8000_h...$7FFF_h$	*[6)]	integer 16	rw
	4	*gain_correction_PGA=4_4*	8000_h...$7FFF_h$	*[6)]	integer 16	rw

*[6)] Die Default-Werte sind bei der Kalibrierung für jedes Moduls individuell ermittelt worden und werden daher in dieser allgemeinen Dokumentation nicht abgedruckt.

Belegung der Parameter *gain_correction_PGA=y_x* (y = 2, 3, 4 ; x = 1...4):

Diese Objekte beinhalten die Gain-Korrekturfaktoren für die PGA-Werte 2, 3 und 4, die bei der Kalibrierung ermittelt worden sind.

Wertebereich: 8000_h ... $7FFF_h$

8.11.10.5 Offset (2510_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2510_h	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*offset_1*	8000_h...$7FFF_h$	0	integer 16	rw
	2	*offset_2*	8000_h...$7FFF_h$	0	integer 16	rw
	3	*offset_3*	8000_h...$7FFF_h$	0	integer 16	rw
	4	*offset_4*	8000_h...$7FFF_h$	0	integer 16	rw

Belegung der Parameter *offset_x* (x = 1...4):

Mit Hilfe dieses Objekts kann der Anwender den Leitungswiderstand kompensieren. Es beinhaltet den Offset in Schritten von 0,1 mOhm.

Wenn die Kompensation eines Widerstands notwendig ist, so muss der negative Wert (siehe PV(R) (Objekt 9130_h)) des Widerstands eingetragen werden.

Beispiel:

$R_{Leitung_x}$ = 123,4 mΩ => 1234_d => *offset_x* = -1234_d
=> *offset_x* = $FB2E_h$

8.11.10.6 Temperaturkoeffizient bei I_{LOW} und I_{HIGH} (2511_h, 2512_h)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
2511	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*tk_Ilow_1*	8000_h...$7FFF_h$	0	integer 16	rw
	2	*tk_Ilow_2*	8000_h...$7FFF_h$	0	integer 16	rw
	3	*tk_Ilow_3*	8000_h...$7FFF_h$	0	integer 16	rw
	4	*tk_Ilow_4*	8000_h...$7FFF_h$	0	integer 16	rw
2512	0	*Anzahl der Einträge*	4	4	unsigned 8	ro
	1	*tk_Ihigh_1*	8000_h...$7FFF_h$	0	integer 16	rw
	2	*tk_Ihigh_2*	8000_h...$7FFF_h$	0	integer 16	rw
	3	*tk_Ihigh_3*	8000_h...$7FFF_h$	0	integer 16	rw
	4	*tk_Ihigh_4*	8000_h...$7FFF_h$	0	integer 16	rw

Belegung der Parameter *tk_Ilow_x* und *tk_Ihigh_x* (x = 1...4):

Über diese Parameter können die Temperaturkoeffizienten gesetzt werden. Der Default-Wert der Temperaturkoeffizienten ist auf ‘0’ eingestellt. Der Anwender kann jedoch die Temperaturkompensation nutzen. Die Formeln für die Berücksichtigung der Temperaturkompensation sind auf Seite 109 abgedruckt.

Wertebereich: 8000_h ... $7FFF_h$

8.12 Firmware-Verwaltung über DS-302-Objekte ($1F50_h$...$1F52_h$)

Die im folgenden beschriebenen Objekte werden benötigt, wenn die Firmware des Gerätes 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 DS-301-Modus kann auf das Objekt $1F50_h$ nicht zugegriffen werden.
Die Objekte $1F51_h$ und $1F52_h$ stehen auch im normalen DS-301-Modus zur Verfügung.
Weitere Informationen zu den Objekten und zum Firmware-Update entnehmen Sie bitte [2]

Index	Sub-Index	Beschreibung	Datentyp	Zugriff
$1F50_h$	0	Boot-Loader: Firmware Download	domain	rw
$1F51_h$	1	Boot-Loader: FLASH-Kommando	unsigned 8	rw
$1F52_h$	0,1,2	Boot-Loader: Firmware-Datum	unsigned 32	ro

8.12.1 Download-Steuerung über Objekt ($1F51_h$)

INDEX	**$1F51_h$**
Name	Program Control
Datentyp	unsigned 8
Zugriff	rw
Wertebereich	0...FE_h
Default-Wert	0

Hinweis:
Der Wertebereich dieses Objektes in der Implementierung für die CAN-CBX-Pt100 weicht vom Wertebereich, der in [2] angegeben ist, ab.
Weitere Informationen zum Objekt $1F51_h$ und zum Firmware-Update entnehmen Sie bitte [2]

8.12.2 Verify Application Software ($1F52_h$)

Index	Sub-Index	Beschreibung	Wertebereich	Default	Datentyp	Zugriff
$1F52_h$	0	*Anzahl der Einträge*	2	2	unsigned 8	ro
	1	*Application_Software_Date*	0...FFFF $FFFF_h$	-	unsigned 32	ro
	2	*Application_Software_Time*	0...0526 $5C00_h$	-	unsigned 32	ro

Beschreibung der Variablen:

Application_Software_Date — Datum der Erstellung der verwendeten Firmware, angegeben in Anzahl der Tage seit dem 1. Januar 1984

Application_Software_Time — Zeit der Erstellung der verwendeten Firmware, angegeben in Anzahl der Millisekunden ab Mitternacht.

9. Referenzen

[1] CiA Standard 301
CANopen Application Layer and Communication Profile V4.2.0 (12.2011)
CAN in Automation (CiA) e.V., Nürnberg, Deutschland

[2] CiA Draft Standard Proposal 302 V4.1 (02.2009)
Additional Application Layer Functions, Part 3: Configuration and program download

[3] CiA Draft Recommendation 303 V1.3 (08.2006)
CANopen Additional Specification, Part 3: Indicator specification

[4] CiA Draft Standard Proposal 404 V1.3.0 (03.2009)
Device Profile for Measuring Devices and Closed-Loop Controllers

[5] CiA Draft Recommendation 303 V1.4 (08.2006)
CANopen Additional Specification, Part 2: Representation of SI units and prefix

[6] Phoenix Contact GmbH & Co. KG, Blomberg.
Technische Daten entnommen aus Phoenix Contact Online-Katalog,
https://www.phoenixcontact.com/online/portal/de
Leiterplattensteckverbinder - FKCT-2,5/4-ST KMGY - 1921900, Abgerufen am 09.10.2013

[7] Phoenix Contact GmbH & Co. KG, Blomberg.,
Technische Daten entnommen aus Phoenix Contact Online-Katalog,
https://www.phoenixcontact.com/online/portal/de
Leiterplattensteckverbinder - FK-MCP 1,5/...-STF-3,81 - 1851261, Abgerufen am 09.10.2013

[8] Phoenix Contact GmbH & Co. KG, Blomberg.,
Technische Daten entnommen aus Phoenix Contact Online-Katalog,
https://www.phoenixcontact.com/online/portal/de
Leiterplattensteckverbinder - FMC 1,5/...-STF-3,5 - 1952296, Abgerufen am 09.10.2013

10. Glossar

FV . . . . . Field Value (ADC-Rohdaten)

PGA . . . . . Programmable Gain Amplifier (einstellbarer Verstärker in den A/D-Wandlers)

PV . . . . . Process Value (korrigierte Nutzdaten der Messwiderstandseingänge)

RTD . . . . . Resistance Temperature Detector (Widerstandstemperaturfühler)

R/W . . . . . Read/Write (Lese- und Schreibzugriff)

11. EU-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
CAN-CBX-PT100

Typ, Modell, Artikel-Nr.
Type, Model, Article No.
C.3032.02

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-K00-0433-11

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, 2014-11-17

Thorsten Ramm

Rechtsgültige Unterschrift / *authorized signature*

I:\Texte\Doku\MANUALS\CAN\CBX\PT100\Konformitätserklärungen\CAN-CBX-Pt100_EG-Konformitaetserklaerung_2014-11-17.odt

12. Bestellhinweise

Typ	Eigenschaften	Bestell-Nr.
CAN-CBX-Pt100	CAN-CBX-Pt100 4 Messwiderstandseingänge, incl. 1x CAN-CBX-TBUS (C.3000.01)	C.3032.02

Handbuch		
CAN-CBX-Pt100-MD	Handbuch in deutsch	C.3032.20

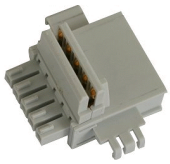
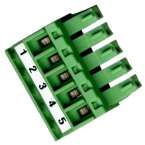
Zubehör		
CAN-CBX-TBUS 	Tragschienen-Busverbinder des CBX-InRailBus für CAN-CBX-Module, (ein Busverbinder ist im Lieferumfang des CAN-CBX-Moduls enthalten)	C.3000.01
CAN-CBX-TBUS-Connector 	Anschlussstecker des CBX-InRailBus zum Anschluss der +24V Versorgungsspannung und des CAN-Interface, Buchsenkontakte	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 13: 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-Pt100-MD	Anwenderhandbuch in Deutsch	C.3032.20
CAN-CBX-Pt100-EN	Anwenderhandbuch in Englisch	C.3032.21

Tabelle 14: 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.