Original-Systemhandbuch
Hintergrundwissen ecomat*mobile*

CODESYS® V2.3

Deutsch

7391020 / 01 04 / 2018

Inhaltsverzeichnis

1 Vorbemerkung

Inhalt

202

1.1 Copyright

6088

Alle auf unseren Seiten verwendeten Produktnamen, -Bilder, Unternehmen oder sonstige Marken sind Eigentum der jeweiligen Rechteinhaber:

- AS-i ist Eigentum der AS-International Association, (→ www.as-interface.net)
- CAN ist Eigentum der CiA (CAN in Automation e.V.), Deutschland (→ www.can-cia.org)
- CODESYS™ ist Eigentum der 3S – Smart Software Solutions GmbH, Deutschland (→ www.codesys.com)
- DeviceNet™ ist Eigentum der ODVA™ (Open DeviceNet Vendor Association), USA (→ www.odva.org)
- EtherNet/IP® ist Eigentum der →ODVA™
- EtherCAT® ist eine eingetragene Marke und patentierte Technologie, lizenziert durch die Beckhoff Automation GmbH, Deutschland
- IO-Link® (→ www.io-link.com) ist Eigentum der →PROFIBUS Nutzerorganisation e.V., Deutschland
- ISOBUS ist Eigentum der AEF – Agricultural Industry Electronics Foundation e.V., Deutschland (→ www.aef-online.org)
- Microsoft® ist Eigentum der Microsoft Corporation, USA (→ www.microsoft.com)
- Modbus® ist Eigentum der Schneider Electric SE, Frankreich (→ www.schneider-electric.com)
- PROFIBUS® ist Eigentum der PROFIBUS Nutzerorganisation e.V., Deutschland (→ www.profibus.com)
- PROFINET® ist Eigentum der →PROFIBUS Nutzerorganisation e.V., Deutschland
- Windows® ist Eigentum der →Microsoft Corporation, USA

1.2 Übersicht: Anwender-Dokumentation für CRnnnn

22853

Die Dokumentation für das Gerät besteht aus folgenden Modulen:
(Downloads von der Homepage → www.ifm.com)

Dokument	Inhalt / Beschreibung
Datenblatt	Technische Daten in Tabellenform
Montageanleitung (gehört zum Lieferumfang des Geräts)	• Anleitung für Montage, elektrische Installation und Inbetriebnahme • Technische Daten
Programmierhandbuch	• Funktionen des Setup-Menüs des Gerät • Erstellen eines CODESYS-Projekts mit diesem Gerät • Zielsystem einstellen mit CODESYS • Geräteinterne SPS mit CODESYS programmieren • Beschreibung der gerätespezifischen CODESYS-Funktionsbibliotheken
Systemhandbuch "Know-How ecomatmobile"	Hintergrundwissen zu folgenden Themen (Beispiele): • Übersicht Templates und Demo-Programme • CAN, CANopen • Ausgänge steuern • Visualisierungen • Übersicht Dateien und Bibliotheken

1.3 Für welche Geräte gilt diese Anleitung?

14403

Technik und Methoden können sich geräteabhängig unterscheiden.

Diese Anleitung gilt für folgende Geräte:
- alle ecomat*mobile*-Controller
- PDM: CR1nnn
- Platinensteuerung: CS0015

1.4 Was bedeuten die Symbole und Formatierungen?

203

Folgende Symbole oder Piktogramme verdeutlichen Ihnen unsere Hinweise in unseren Anleitungen:

⚠ WARNUNG

Tod oder schwere irreversible Verletzungen sind möglich.

⚠ VORSICHT

Leichte reversible Verletzungen sind möglich.

ACHTUNG

Sachschaden ist zu erwarten oder möglich.

	Wichtiger Hinweis Fehlfunktionen oder Störungen sind bei Nichtbeachtung möglich
	Information Ergänzender Hinweis
► ...	Handlungsaufforderung
> ...	Reaktion, Ergebnis
→ ...	"siehe"
abc	Querverweis
123 0x123 0b010	Dezimalzahl Hexadezimalzahl Binärzahl
[...]	Bezeichnung von Tasten, Schaltflächen oder Anzeigen

1.5 Wie ist diese Dokumentation aufgebaut?

204
1508

Diese Dokumentation ist eine Kombination aus verschiedenen Anleitungstypen. Sie ist eine Lernanleitung für den Einsteiger, aber gleichzeitig auch eine Nachschlageanleitung für den versierten Anwender. Dieses Dokument richtet sich an die Programmierer der Anwendungen.

Und so finden Sie sich zurecht:

- Um gezielt zu einem bestimmten Thema zu gelangen, benutzen Sie bitte das Inhaltsverzeichnis.
- Mit dem Stichwortregister "Index" gelangen Sie ebenfalls schnell zu einem gesuchten Begriff.
- Am Anfang eines Kapitels geben wir Ihnen eine kurze Übersicht über dessen Inhalt.
- Abkürzungen und Fachbegriffe → Anhang.

Bei Fehlfunktionen oder Unklarheiten setzen Sie sich bitte mit dem Hersteller in Verbindung:
Kontakt → www.ifm.com

Wir wollen immer besser werden! Jeder eigenständige Abschnitt enthält in der rechten oberen Ecke eine Identifikationsnummer. Wenn Sie uns über Unstimmigkeiten unterrichten wollen, dann nennen Sie uns bitte diese Nummer zusammen mit Titel und Sprache dieser Dokumentation. Vielen Dank für Ihre Unterstützung!

Im Übrigen behalten wir uns Änderungen vor, so dass sich Abweichungen vom Inhalt der vorliegenden Dokumentation ergeben können. Die aktuelle Version finden Sie auf der **ifm**-Homepage:
→ www.ifm.com

1.6 Historie der Anleitung (SEM)

14337

Was hat sich wann in dieser Anleitung geändert? Ein Überblick:

Datum	Thema	Änderung
2017-01-13	Software-Handbuch für CODESYS 2.3	Hinweis auf Download von ifm-Homepage entfernt
2017-12-18	Liste der ifm-Niederlassungen	aktualisiert
2018-04-16	Kapitel "CAN: Hardware"	wieder eingefügt

2 Templates und Demo-Programme

Inhalt

13747

2.1 Einführung

11646

2.1.1 Was sind ifm-Templates?

11647

Templates sind Vorlagen für CODESYS-Anwendungsprogramme.
Diese Vorlagen gibt es separat für alle programmierbaren **ecomat*mobile***-Geräte.

Struktur der Dateinamen:
`ifm_template_CRnnnn(CAN)_(V1)_(V2).pro`
Dabei bedeuten die Klammerausdrücke Folgendes:

(CAN)	CAN-Protokoll: • Layer2 • CANopen-Master • CANopen-Slave
(V1)	Version (Vxxyyzz) des Laufzeitsystems des Geräts CRnnnn
(V2)	Version (Vnn) des Templates

Die Artikelnummer im Template muss zwingend mit der Artikelnummer des zu programmierenden Geräts überreinstimmen! → Gerätehandbuch, Kapitel "Angaben zur Software"

Kurzanleitung: ifm-Templates

18057

So finden Sie die **ifm**-Templates:

► Im CODESYS-Menü [Datei] > [Neu aus Vorlage...] öffnen.
> Der Dialog [Öffnen] erscheint.
► Im Verzeichnisbaum folgenden Pfad wählen:
(*Programme-Laufwerk*) > [Programme] > [ifm electronic] > [CoDeSys (Version)] > [Projects] > (*aktuelle Template-DVD*) > (*gewünschte Vorlage*)
► Wahl mit [Öffnen] bestätigen.
> Ein neues CODESYS-Projekt wird angelegt.
Dieses Projekt enthält alle erforderlichen Elemente und Parametrierungen für ein auf dem gewählten Gerät lauffähiges Projekt.
► Dieses Projekt manuell an die Anwendung anpassen.
Bei Bedarf einzelne **ifm**-Demos integrieren (→ Kapitel **ifm-Demo-Programme** (→ S. 20)).

2.1.2 Was sind ifm-Demo-Programme?

11648

ifm-Demo-Programme sind CODESYS-Beispiele für einzelne Funktionen.
Meist gelten die Beispiele für kein bestimmtes **ifm**-Gerät, sofern nichts anderes angegeben ist.

Struktur der Dateinamen:
(Gerät)demo_(V1)_(V2).pro
Dabei bedeuten die Klammerausdrücke Folgendes:

(Gerät)	Artikelnummer des Beispielgeräts
(V1)	Art der Demonstration
(V2)	Version (Vnn) des Demo-Programms

Kurzanleitung: ifm-Demo-Programme

18058

So finden Sie die **ifm**-Demo-Programme:

► Im CODESYS-Menü [Projekt] > [öffnen] öffnen.
> Der Dialog [Öffnen] erscheint.
► Im Verzeichnisbaum folgenden Pfad wählen:
(*Programme-Laufwerk*) > [Programme] > [ifm electronic] > [CoDeSys (Version)] > [Projects] > (*gewünschtes Demo-Verzeichnis*) > (*gewünschtes Demo-Projekt*)
► Wahl mit [Öffnen] bestätigen.
> Das Fenster [Objekte kopieren] erscheint.
► Die Elemente markieren, die ausschließlich die gewünschte Funktion enthalten.
► Wahl mit [OK] bestätigen.
> Im aktuellen Projekt werden die markierten Elemente aus dem Demo-Projekt eingefügt.
► Die Elemente der Anwendung anpassen und z.B. dem Modul PLC_PRG hinzufügen.

2.2 Programmiersystem über Templates einrichten

Inhalt

18051

ifm bietet vorgefertigte Templates (Programm-Vorlagen), womit Sie das Programmiersystem schnell, einfach und vollständig einrichten können.

970

Beim Installieren der **ecomat*mobile***-DVD "Software, tools and documentation" wurden auch Projekte mit Vorlagen auf Ihrem Computer im Programmverzeichnis abgelegt:
`…\ifm electronic\CoDeSys V…\Projects\Template_DVD_V…`

► Die gewünschte dort gespeicherte Vorlage in CODESYS öffnen mit:
[Datei] > [Neu aus Vorlage…]

> CODESYS legt ein neues Projekt an, dem der prinzipielle Programmaufbau entnommen werden kann. Es wird dringend empfohlen, dem gezeigten Schema zu folgen.

2.2.1 Über die ifm-Templates

Inhalt

3981

In der Regel werden für jedes Gerät folgende Templates angeboten:

- `ifm_template_CRnnnnLayer2_Vxxyyzz.pro`
  für den Betrieb des Geräts mit CAN Layer 2
- `ifm_template_CRnnnnMaster_Vxxyyzz.pro`
  für den Betrieb des Geräts als CANopen-Master
- `ifm_template_CRnnnnSlave_Vxxyyzz.pro`
  für den Betrieb des Geräts als CANopen-Slave

Die hier beschriebenen Templates gelten für:
- CODESYS ab Version 2.3.9.6
- auf der **ecomat*mobile***-DVD "Software, tools and documentation" ab Version 020000

Die Templates enthalten alle die gleichen Strukturen.

Mit dieser Auswahl der Programm-Vorlage für den CAN-Betrieb ist bereits eine wichtige Grundlage für ein funktionsfähiges Programm geschaffen.

Ordner-Struktur, allgemein

3978

Die Bausteine sind sortiert in die folgenden Ordner:

Ordner	Beschreibung
CAN_OPEN	für Controller und PDM, CAN-Betrieb als Master oder Slave: Enthält die Bausteine für CANopen.
I_O_CONFIGURATION	für Controller, CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Bausteine zum Parametrieren der Betriebsarten der Ein- und Ausgänge.
PDM_COM_LAYER2	für Controller, CAN-Betrieb als Layer 2 oder Slave: Bausteine zur Basiskomunikation über Layer2 zwischen PLC und PDM.
CONTROL_CR10nn	für PDM, CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Enthält Bausteine zur Bild- und Tastensteuerung im laufenden Betrieb.
PDM_DISPLAY_SETTINGS	für PDM, CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Enthält Bausteine zum Einstellen des Monitors.

Programme und Funktionen in den Ordnern der Templates für Controller

18048

Die vorgenannten Ordner enthalten die folgenden Programme und Bausteine:

Bausteine im Ordner CAN_OPEN	**Beschreibung**
CANOPEN	CAN-Betrieb als Master: Enthält folgende parametrierte Bausteine: • CAN1_MASTER_EMCY_HANDLER, • CAN1_MASTER_STATUS, • SELECT_NODESTATE (→ unten).
CANOPEN	CAN-Betrieb als Slave: Enthält folgende parametrierte Bausteine: • CAN1_SLAVE_EMCY_HANDLER, • CAN1_SLAVE_STATUS, • SELECT_NODESTATE (→ unten).
Objekt1xxxh	CAN-Betrieb als Slave: Enthält die Werte [STRING] zu folgenden Parametern: • ManufacturerDeviceName, z.B.: 'CR1051' • ManufacturerHardwareVersion, z.B.: 'HW_Ver 1.0' • ManufacturerSoftwareVersion, z.B.: 'SW_Ver 1.0'

Bausteine im Ordner I_O_CONFIGURATION	**Beschreibung**
CONF_IO_CRnnnn	CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Parametriert die Betriebsarten der Ein- und Ausgänge.

Bausteine im Ordner PDM_COM_LAYER2	**Beschreibung**
PLC_TO_PDM	CAN-Betrieb mit Layer 2 oder als Slave: Organisiert die Kommunikation vom Controller zum PDM: • überwacht die Übertragungszeit, • überträgt Steuerdaten für Bildwechsel, LEDs, Eingabewerte usw.
TO_PDM	CAN-Betrieb mit Layer 2 oder als Slave: Organisiert die Signale für LEDs und Tasten zwischen Controller und PDM. Enthält folgende parametrierte Bausteine: • PACK (→ 3S), • PLC_TO_PDM (→ oben), • UNPACK (→ 3S).

Bausteine im Wurzel-Verzeichnis	**Beschreibung**
PLC_CYCLE	CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Ermittelt die Zykluszeit der SPS im Gerät.
PLC_PRG	CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Hauptprogramm; hier werden die weiteren Programm-Elemente eingebunden.

Programme und Funktionen in den Ordnern der Templates für PDM

18049

Die vorgenannten Ordner enthalten die folgenden Programme und Bausteine:

Bausteine im Ordner CAN_OPEN	**Beschreibung**
CANOPEN	CAN-Betrieb als Master: Enthält folgende parametrierte Bausteine: • CAN1_MASTER_EMCY_HANDLER, • CAN1_MASTER_STATUS, • SELECT_NODESTATE (→ unten).
CANOPEN	CAN-Betrieb als Slave: Enthält folgende parametrierte Bausteine: • CAN1_SLAVE_EMCY_HANDLER, • CAN1_SLAVE_STATUS, • SELECT_NODESTATE (→ unten).
Objekt1xxxh	CAN-Betrieb als Slave: Enthält die Werte [STRING] zu folgenden Parametern: • ManufacturerDeviceName, z.B.: 'CR1051' • ManufacturerHardwareVersion, z.B.: 'HW_Ver 1.0' • ManufacturerSoftwareVersion, z.B.: 'SW_Ver 1.0'
SELECT_NODESTATE	CAN-Betrieb als Master oder als Slave: Wandelt den Wert des Knoten-Status [BYTE] in den zugehörigen Text [STRING]: 4 ⇨ 'STOPPED' 5 ⇨ 'OPERATIONAL' 127 ⇨ 'PRE-OPERATIONAL'

Bausteine im Ordner CONTROL_CR10nn	**Beschreibung**
CONTROL_PDM	CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Organisiert die Bildsteuerung im PDM. Enthält folgende parametrierte Bausteine: • PACK (→ 3S), • PDM_MAIN_MAPPER, • PDM_PAGECONTROL, • PDM_TO_PLC (→ unten), • SELECT_PAGE (→ unten).
PDM_TO_PLC	CAN-Betrieb mit Layer 2: Organisiert die Kommunikation vom PDM zum Controller: • überwacht die Übertragungszeit, • überträgt Steuerdaten für Bildwechsel, LEDs, Eingabewerte usw. Enthält folgende parametrierte Bausteine: • CAN_1_TRANSMIT, • CAN_1_RECEIVE.
RT_SOFT_KEYS	CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Liefert von den (virtuellen) Tasten-Signalen im PDM die steigenden Flanken. Es können beliebige Variablen (als virtuelle Tasten) auf die globalen Variablen SoftKeyGlobal gemappt werden, wenn z.B. ein Programmteil von einem CR1050 in ein CR1055 kopiert werden soll. Dort gibt es nur die Tasten F1...F3: ► Für die virtuellen Tasten F4...F6 Variablen erzeugen. Diese selbst erzeugten Variablen hier auf die globalen Softkeys mappen. Im Programm nur mit den globalen Softkeys arbeiten. Vorteil: Anpassungsarbeiten sind nur an einer Stelle erforderlich.
SELECT_PAGE	CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Organisiert die Wahl der Visualisierungen. Enthält folgende parametrierte Bausteine: • RT_SOFT_KEYS (→ oben).

Bausteine im Ordner PDM_DISPLAY_SETTINGS	Beschreibung
CHANGE_BRIGHTNESS	CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Organisiert Helligkeit / Kontrast des Monitors.
DISPLAY_SETTINGS	CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Stellt die Echtzeituhr, steuert Helligkeit / Kontrast des Monitors, zeigt die Software-Version. Enthält folgende parametrierte Bausteine: • CHANGE_BRIGHTNESS (→ oben), • CurTimeEx (→ 3S), • PDM_SET_RTC, • READ_SOFTWARE_VERS (→ unten), • TP (→ 3S).
READ_SOFTWARE_VERS	CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Zeigt die Software-Version. Enthält folgende parametrierte Bausteine: • DEVICE_KERNEL_VERSION1, • DEVICE_RUNTIME_VERSION, • LEFT (→ 3S).

Bausteine im Wurzel-Verzeichnis	Beschreibung
PDM_CYCLE_MS	CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Ermittelt die Zykluszeit der SPS im Gerät.
PLC_PRG	CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Hauptprogramm; hier werden die weiteren Programm-Elemente eingebunden.

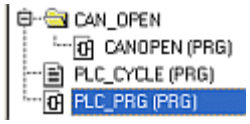
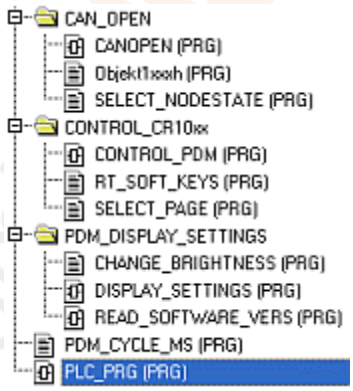
2.2.2 Wie richten Sie das Programmiersystem schnell und einfach ein? (z.B. CR2500)

18052

- ► Im CoDeSys-Menü wählen: [Datei] > [Neu aus Vorlage...].
- ► Verzeichnis der aktuellen DVD wählen, z.B. `...\Projects\TEMPLATE_DVD020000`.
- ► Artikelnummer des Geräts in der Liste suchen, z.B. CR2500 als CANopen-Master:

- ► Achten Sie auch auf die richtige Programm-Version!
- ► Wie ist das CAN-Netzwerk organisiert?
  Soll auf Layer2-Basis gearbeitet werden oder gibt es (mit CANopen) einen Master mit mehreren Slaves?
- ► Wahl mit [Öffnen] bestätigen.

> Neues CODESYS-Projekt wird angelegt mit zunächst folgender Ordnerstruktur (links):

Beispiel für CR2500 als CANopen-Master:	Anderes Beispiel für CR1051 als CANopen-Slave:
	

(Über die Ordnerstrukturen in Templates → Kapitel **Über die ifm-Templates** (→ S. 11)).

- ► Das neue Projekt speichern mit [Datei] > [Speichern unter...], dabei geeignetes Verzeichnis und Projektnamen festlegen.

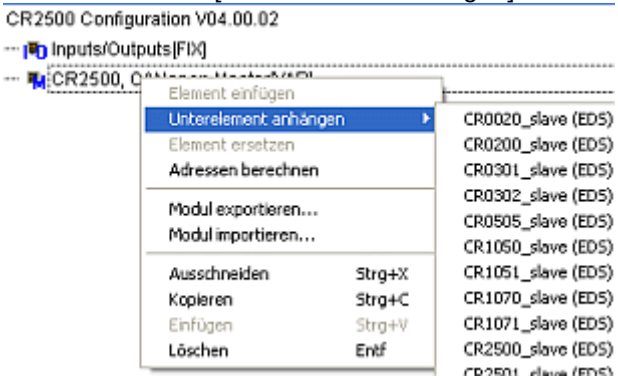
2.2.3 CANopen-Slave hinzufügen (Beispiel: CR2500 <-- CR2011)

18053

► Das CAN-Netzwerk im Projekt konfigurieren:
Im CODESYS-Projekt über dem Tabulator [Ressourcen] das Element [Steuerungskonfiguration] doppelklicken.

► Mit **rechter** Maustaste in den Eintrag [CR2500, CANopen Master] klicken.

► Im Kontext-Menü [Unterelement anhängen] klicken:



> Im ergänzten Kontextmenü erscheint eine Liste aller verfügbaren EDS-Dateien.

► Gewünschtes Element wählen, z.B. "System R360: I/O CompactModule CR2011 (EDS)".
Die EDS-Dateien liegen im Verzeichnis `C:\…\CoDeSys V…\Library\PLCConf\`.

> Das Fenster [Steuerungskonfiguration] ändert sich wie folgt:

► Für den eingetragenen Slave den Erfordernissen entsprechend die CAN-Parameter, das PDO-Mapping und die SDOs einstellen.
[!] [alle SDOs erzeugen] besser abwählen.

► Mit weiteren Slaves sinngemäß wie vorstehend verfahren.

► Projekt speichern!

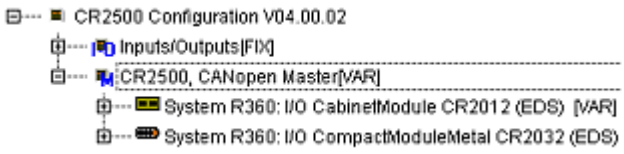
Damit ist das Netzwerk Ihres Projekts hinreichend beschrieben. Sie wollen dieses Projekt mit weiteren Elementen und Funktionen ergänzen?
→ Kapitel **Projekt mit weiteren Funktionen ergänzen** (→ S. 17)

2.2.4 Projekt mit weiteren Funktionen ergänzen

3987

Sie haben ein Projekt mittels eines ifm-Templates angelegt und das CAN-Netzwerk definiert. Nun wollen Sie diesem Projekt weitere Funktionen hinzufügen.

Für das Beispiel nehmen wir einen CabinetController CR2500 als CANopen-Master an, an den ein I/O-CabinetModul CR2011 und ein I/O-Compact-Modul CR2032 als Slaves angeschlossen sind:

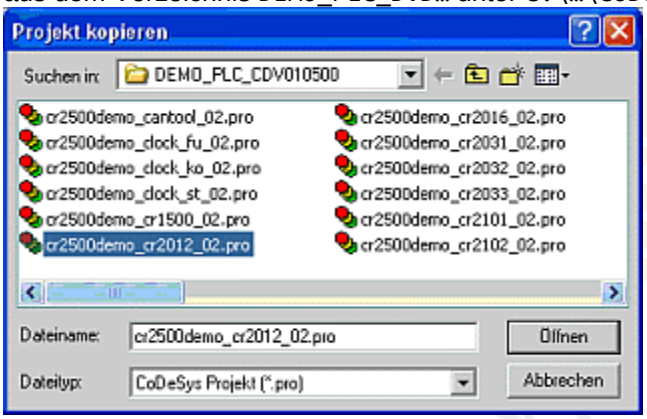


Beispiel: Steuerungskonfiguration

Am CR2012 sei ein Joystick angeschlossen, der am CR2032 einen PWM-Ausgang ansteuern soll. Wie geht das schnell und einfach?

► CODESYS-Projekt speichern!

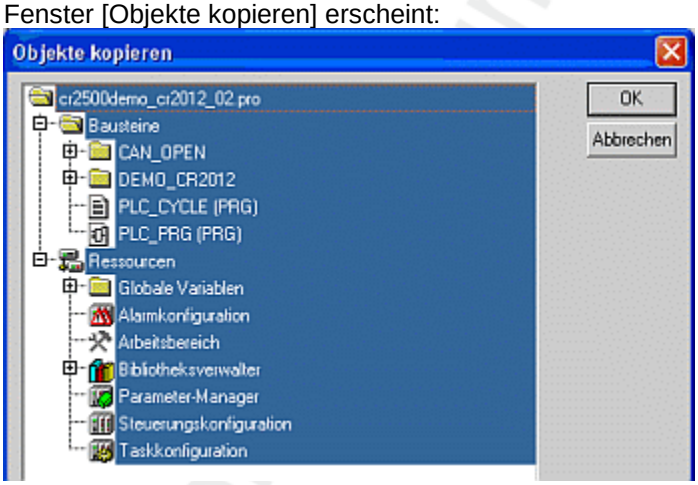
► In CODESYS mit [Projekt] > [kopieren...] das Projekt öffnen, das die gewünschte Funktion enthält:
z.B. `CR2500Demo_CR2012_02.pro`
aus dem Verzeichnis `DEMO_PLC_DVD…` unter `C:\…\CoDeSys V…\Projects\`:



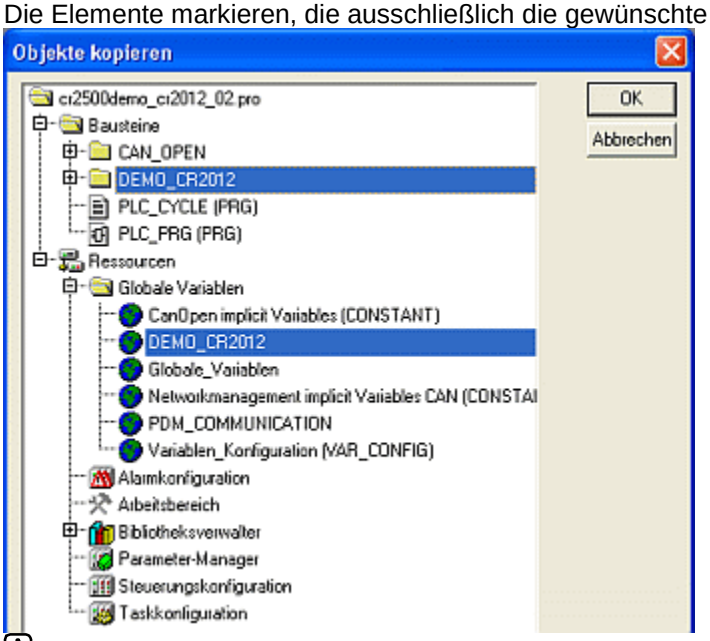
► Wahl mit [Öffnen] bestätigen.

► Die Meldung "Fehler beim Laden der Steuerungskonfiguration" kann ignoriert werden.

> Fenster [Objekte kopieren] erscheint:



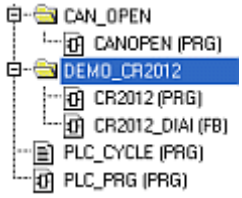
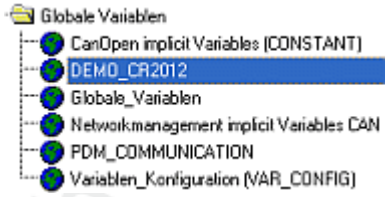
► Die Elemente markieren, die ausschließlich die gewünschte Funktion enthalten, hier z.B.:



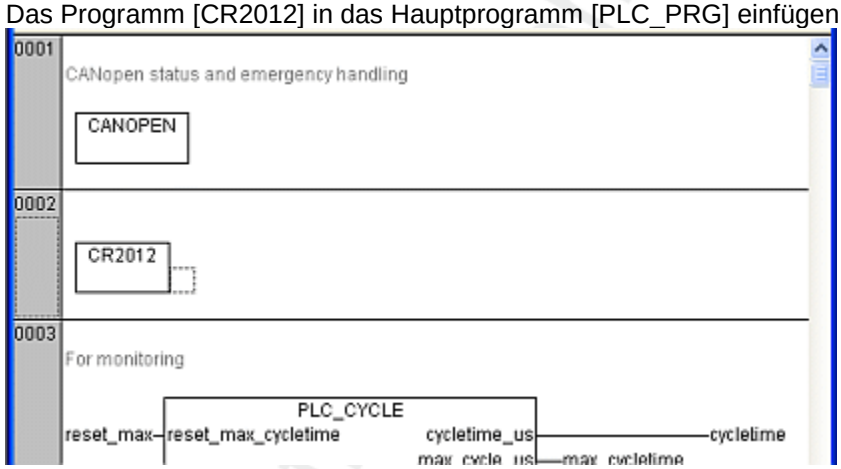
In anderen Fällen können auch Bibliotheken und / oder Visualisierungen erforderlich sein.

► Wahl mit [OK] bestätigen.

> In unserem Beispiel-Projekt sind die im Demo-Projekt gewählten Elemente hinzugekommen:

Bausteine:	Ressourcen:
	

► Das Programm [CR2012] in das Hauptprogramm [PLC_PRG] einfügen, z.B.:



► In den Kommentaren der Bausteine und Globalen Variablen stehen meist Hinweise, wie bei Bedarf einzelne Elemente daraus konfiguriert, eingeschlossen oder ausgeschlossen werden müssen. Diesen Hinweisen Folge leisten.

► Ein- und Ausgangsvariable sowie CAN-Parameter und ggf. Visualisierungen den eigenen Bedingungen anpassen.

► [Projekt] > [speichern] und
[Projekt] > [Alles übersetzen].

► Nach eventuell erforderlichen Korrekturen und Ergänzen von fehlenden Bibliotheken (→ Fehlermeldungen nach dem Übersetzen) das Projekt nochmals speichern.

► Nach diesem Prinzip schrittweise (!) mit weiteren Funktionen aus anderen Projekten ergänzen und jeweils die Ergebnisse prüfen.

► [Projekt] > [speichern] und
[Projekt] > [Alles übersetzen].

2.3 ifm-Demo-Programme

Inhalt

3982

Im Verzeichnis
- `DEMO_PLC_DVD…` (für Controller) oder
- `DEMO_PDM_DVD…` (für PDMs)

unter `C:\…\CoDeSys V…\Projects\`
erklären wir bestimmte Funktionen in getesteten Demo-Programmen. Bei Bedarf können diese Funktionen in eigene Projekte übernommen werden. Die Strukturen und Variablen der **ifm**-Demo-Programme passen zu denen in den **ifm**-Templates.

In jedem Demo-Programm wird nur genau **ein** Thema gezeigt. Auch für Controller werden dazu einige Visualisierungen gezeigt, die auf dem PC-Monitor die getestete Funktion anschaulich machen sollen.

Kommentare in den Bausteinen und in den Variablenlisten helfen beim Anpassen der Demo-Programme an Ihr Projekt.

Wenn nicht anders angegeben, gelten die Demo-Programme jeweils für alle Controller oder für alle PDMs.

Die hier beschriebenen Demo-Programme gelten für:
- CODESYS ab Version 2.3.9.6
- auf der **ecomat*mobile***-DVD "Software, tools and documentation" ab Version 020000

2.3.1 Demo-Programme für Controller

3995

Demo-Programm	Funktion
`CR2500Demo_CanTool_xx.pro`	getrennt für PDM360, PDM360compact, PDM360smart und Controller: Enthält Funktionen zum Einstellen und Analysieren der CAN-Schnittstelle.

`CR2500Demo_ClockFu_xx.pro`
`CR2500Demo_ClockKo_xx.pro`
`CR2500Demo_ClockSt_xx.pro`
	Taktgenerator für Controller als Funktion eines Wertes an einem Analog-Eingang:
	Fu = in Funktionsplan
	Ko = in Kontaktplan
	St = in Strukturiertem Text

`CR2500Demo_CR1500_xx.pro`	Anschluss eines Tastatur-Moduls CR1500 als Slave eines Controllers (CANopen-Master).
`CR2500Demo_CR2012_xx.pro`	I/O-Cabinet-Modul CR2012 als Slave eines Controllers (CANopen-Master), Anschluss eines Joysticks mit Richtungsschalter und Referenz-Mittelspannung.
`CR2500Demo_CR2016_xx.pro`	I/O-Cabinet-Modul CR2016 als Slave eines Controllers (CANopen-Master), 4x Frequenz-Eingang, 4x Digital-Eingang minus-schaltend, 4x Digital-Eingang plus-schaltend, 4x Analog-Eingang ratiometrisch, 4x PWM1000-Ausgang und 12x Digitalausgang.
`CR2500Demo_CR2031_xx.pro`	I/O-Compact-Modul CR2031 als Slave eines Controllers (CANopen-Master), Strommessung an den PWM-Ausgängen.

Demo-Programm	Funktion
CR2500Demo_CR2032_xx.pro	I/O-Compact-Modul CR2032 als Slave eines Controllers (CANopen-Master), 4x Digital-Eingang, 4x Digital-Eingang analog ausgewertet, 4x Digital-Ausgang, 4x PWM-Ausgang.
CR2500Demo_CR2033_xx.pro	I/O-Compact-Modul CR2033 als Slave eines Controllers (CANopen-Master), 4x Digital-Eingang, 4x Digital-Eingang analog ausgewertet, 4x Digital-Ausgang.
CR2500Demo_CR2101_xx.pro	Neigungssensor CR2101 als Slave eines Controllers (CANopen-Master).
CR2500Demo_CR2102_xx.pro	Neigungssensor CR2102 als Slave eines Controllers (CANopen-Master).
CR2500Demo_CR2511_xx.pro	I/O-Smart-Modul CR2511 als Slave eines Controllers (CANopen-Master), 8x PWM-Ausgang stromgeregelt.
CR2500Demo_CR2512_xx.pro	I/O-Smart-Modul CR2512 als Slave eines Controllers (CANopen-Master), 8x PWM-Ausgang. Anzeige des aktuellen Stroms für jedes Kanalpaar.
CR2500Demo_CR2513_xx.pro	I/O-Smart-Modul CR2513 als Slave eines Controllers (CANopen-Master), 4x Digital-Eingang, 4x Digital-Ausgang, 4x Analogeingang 0...10 V.
CR2500demo_input_from_pdm_CANopen_xx.pro	Systemvariablen via CANopen nutzen: • HANDLE, • INPUT_VALUE, • LENGHT
CR2500demo_input_from_pdm_Layer2_xx.pro	Systemvariablen via CAN-Layer2 nutzen: • HANDLE, • INPUT_VALUE, • LENGHT
CR2500Demo_Interrupt_xx.pro	Beispiel mit SET_INTERRUPT_XMS.
CR2500Demo_Operating_hours_xx.pro	Beispiel für einen Betriebsstundenzähler mit Schnittstelle zu einem PDM.
CR2500Demo_PWM_xx.pro	Wandelt einen Potentiometer-Wert an einem Eingang in einen normierten PWM-Wert an einem Ausgang mit folgenden Bausteinen: • INPUT_VOLTAGE, • NORM, • PWM100.
CR2500Demo_RS232_xx.pro	Beispiel für den Empfang von Daten auf der seriellen Schnittstelle mit Hilfe des Windows-Hyperterminal.
StartersetDemo.pro StartersetDemo2.pro StartersetDemo2_final.pro	Verschiedene Übungen zum E-Learning mit dem Starterset EC2074.

_xx = Angabe der Demo-Version

2.3.2 Demo-Programme für PDM und BasicDisplay

3996

Demo-Programm	Funktion
CR1051Demo_CanTool_xx.pro CR1053Demo_CanTool_xx.pro CR1071Demo_CanTool_xx.pro	getrennt für PDM360, PDM360compact, PDM360smart und Controller: Enthält Funktionen zum Einstellen und Analysieren der CAN-Schnittstelle.
CR1051_Camera_Reset_xx.pro	Erkennen einer Unterbrechung zwischen Kamera und Display. Automatisches Wiederherstellen der Verbindung. Für CR1051 ab Hardware-Stand AG, ab Software-Stand V4.3.2 Getestet mit einer Kamera Typ O2M100 / O2M102 / MO1580 mit IP-Adresse 192.168.82.15 Bild PAGE_001: leer Bild PAGE_002: Meldung "Camera connection lost!"
CR1051Demo_Input_Character_xx.pro	Ermöglicht beliebige Zeicheneingabe in eine Zeichenkette: • Großbuchstaben, • Kleinbuchstaben, • Sonderzeichen, • Ziffern. Auswahl der Zeichen mit dem Drehgeber. Beispiel ist auch z.B. für eine Passworteingabe geeignet. Bild P01000: Auswahl und Übernahme von Zeichen
CR1051Demo_Input_Lib_xx.pro	Demo von INPUT_INT aus der Bibliothek ifm_pdm_input_Vxxyyzz (mögliche Alternative zum 3S-Standard). Werte wählen und einstellen mittels Drehgeber. Bild P10000: 6 Werte INT Bild P10010: 2 Werte INT Bild P10020: 1 Wert REAL
CR1051Demo_Linear_logging_on_flash_intern_xx.pro	Schreibt einen CSV-Datensatz mit dem Inhalt einer CAN-Nachricht in den internen Flash-Speicher (/home/project/daten.csv), wenn [F3] gedrückt wird oder eine CAN- Nachricht auf dem ID 100 empfangen wurde. Wenn der definierte Speicherbereich gefüllt ist, wird die Aufzeichnung der Daten beendet. Verwendete Bausteine: • WRITE_CSV_8BYTE, • SYNC. Bild P35010: Anzeige Datei-Informationen Bild P35020: Anzeige aktueller Datensatz Bild P35030: Anzeige Liste von 10 Datensätzen
CR1051Demo_O2M_1Cam_xx.pro	Anschluss von 1 Kamera O2M100 am Monitor mit CAM_O2M. Umschalten zwischen Teil- und Vollbild. Bild 39000: Auswahlmenü Bild 39010: Kamerabild + Textbox Bild 39020: Kamerabild als Vollbild Bild 39030: nur Visualisierung
CR1051Demo_O2M_2Cam_xx.pro	Anschluss von 2 Kameras O2M100 am Monitor mit CAM_O2M. Umschalten zwischen den Kameras und zwischen Teil- und Vollbild. Bild 39000: Auswahlmenü Bild 39010: Kamerabild + Textbox Bild 39020: Kamerabild als Vollbild Bild 39030: nur Visualisierung
CR1051Demo_Powerdown_Retain_bin_xx.pro	Beispiel mit PDM_POWER_DOWN aus der Bibliothek ifm_CR1051_Vxxyyzz.Lib, um Retain-Variable in die Datei Retain.bin zu speichern. Simulation des ShutDown mit [F3].
CR1051Demo_Powerdown_Retain_bin2_xx.pro	Beispiel mit PDM_POWER_DOWN aus der Bibliothek ifm_CR1051_Vxxyyzz.Lib, um Retain-Variable in die Datei Retain.bin zu speichern. Simulation des ShutDown mit [F3].

Demo-Programm	Funktion
CR1051Demo_Powerdown_Retain_cust_xx.pro	Beispiel mit PDM_POWER_DOWN und PDM_READ_RETAIN aus der Bibliothek ifm_CR1051_Vxxyyzz.Lib, um Retain-Variable in die Datei /home/project/myretain.bin zu speichern. Simulation des ShutDown mit [F3].
CR1051Demo_Read_Textline_xx.pro	Das Beispiel-Programm liest jeweils 7 Textzeilen aus dem PDM-Dateisystem mit Hilfe von READ_TEXTLINE. Bild P01000: Anzeige gelesener Text
CR1051Demo_Real_in_xx.pro	Einfaches Beispiel für die Eingabe eines REAL-Werts in das PDM. Bild P01000: Eingabe und Anzeige des REAL-Werts
CR1051Demo_Ringlogging_on_flash_intern_xx.pro	Schreibt einen CSV-Datensatz in den internen Flash-Speicher, wenn [F3] gedrückt wird oder eine CAN-Nachricht auf dem ID 100 empfangen wurde. Die Dateinamen sind frei definierbar. Wenn der definierte Speicherbereich gefüllt ist, beginnt die Aufzeichnung der Daten von vorn. Verwendete Bausteine: • WRITE_CSV_8BYTE, • SYNC. Bild P35010: Anzeige Datei-Informationen Bild P35020: Anzeige aktueller Datensatz Bild P35030: Anzeige Liste von 8 Datensätzen
CR1051Demo_Ringlogging_on_flash_pcmcia_xx.pro	Schreibt einen CSV-Datensatz auf die PCMCIA-Karte, wenn [F3] gedrückt wird oder eine CAN-Nachricht auf dem ID 100 empfangen wurde. Die Dateinamen sind frei definierbar. Wenn der definierte Speicherbereich gefüllt ist, beginnt die Aufzeichnung der Daten von vorn. Verwendete Bausteine: • WRITE_CSV_8BYTE, • OPEN_PCMCIA, • SYNC. Bild P35010: Anzeige Datei-Informationen Bild P35020: Anzeige aktueller Datensatz Bild P35030: Anzeige Liste von 8 Datensätzen
CR1051Demo_RW-Parameter_xx.pro	In einer Liste können Parameter gewählt und geändert werden. Beispiel mit folgenden Bausteinen: • READ_PARAMETER_WORD, • WRITE_PARAMETER_WORD. Bild P35010: Liste von 20 Parametern
CR1071demo_Input_to_plc_xx.pro	Systemvariablen via CAN-Layer2 nutzen: • HANDLE, • INPUT_VALUE, • LENGHT Bild P01000: 3 Temperaturwerte (Byte, Word, DWord) Bild P01010: 3 Druckwerte (Byte, Word, DWord)
CR1071demo_Input_to_plc_CANopen_xx.pro	Systemvariablen via CANopen nutzen: • HANDLE, • INPUT_VALUE, • LENGHT Bild P01000: 3 Temperaturwerte (Byte, Word, DWord) Bild P01010: 3 Druckwerte (Byte, Word, DWord)
CR1071demo_Input_to_plc_Layer2_xx.pro	Systemvariablen via CAN-Layer2 nutzen: • HANDLE, • INPUT_VALUE, • LENGHT Bild P01000: 3 Temperaturwerte (Byte, Word, DWord) Bild P01010: 3 Druckwerte (Byte, Word, DWord)

_xx = Angabe der Demo-Version

3 CAN einsetzen – Beschreibung

Inhalt

13743

3.1 Allgemeines zu CAN

Inhalt

1164

Der CAN-Bus (**C**ontroller **A**rea **N**etwork) gehört zu den Feldbussen.

Es handelt sich dabei um ein asynchrones, serielles Bussystem, das 1983 von Bosch für die Vernetzung von Steuergeräten in Automobilen entwickelt und 1985 zusammen mit Intel vorgestellt wurde, um die Kabelbäume (bis zu 2 km pro Fahrzeug) zu reduzieren und dadurch Gewicht zu sparen.

3.1.1 CAN: Hardware

14103

Topologie

1244

Das CAN-Netzwerk wird als Linienstruktur aufgebaut. Stichleitungen sind in eingeschränktem Umfang zulässig.

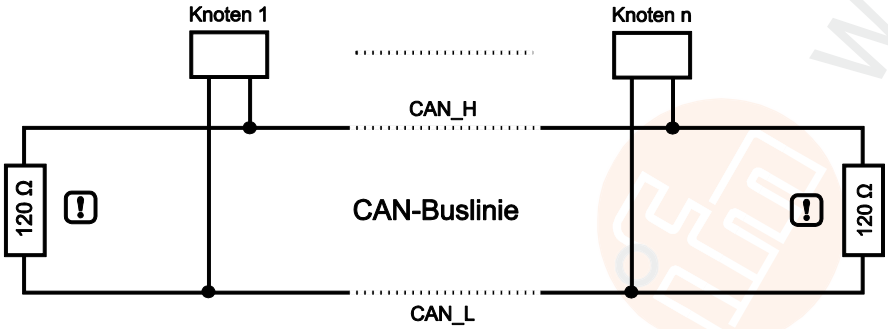
Weitere Möglichkeit:
- sternförmiger Bus (z.B. Zentralverrieglung).

Stichleitungen und sternförmiger Bus haben den Nachteil, dass der Wellenwiderstand schwer zu bestimmen ist. Im schlimmsten Fall funktioniert der Bus nicht mehr.

Netzaufbau

1178

Die Norm ISO 11898 setzt einen Aufbau des CAN-Netzes mit einer Linienstruktur voraus.



Grafik: CAN-Netzaufbau Linienstruktur

Der Innenwiderstand einer CAN-Schnittstelle beträgt etwa 40...45 kΩ. Bei 32 Geräten am CAN-Bus beträgt der resultierende Widerstand im Netzwerk nur noch 1,25...1,4 kΩ.

HINWEIS

Verfälschen der Signalqualität wegen Signal-Echos an den Leitungsenden verhindern:

► Die CAN-Buslinie an ihren beiden Enden jeweils mit einem Abschlusswiderstand von jeweils ≥ 120 Ω abschließen!

Die Geräte der **ifm electronic gmbh**, die mit einem CAN-Interface ausgestattet sind, haben grundsätzlich keine Abschlusswiderstände.

Zusammen mit den Abschlusswiderständen soll der Gesamtwiderstand (gemessen zwischen CAN_H und CAN_L) der spannungslosen CAN-Buslinie etwa 60±5 Ω betragen.

Stichleitungen

13013

In Abhängigkeit von der Gesamtleitungslänge und den zeitlichen Abläufen auf dem Bus können Signal-Reflektionen auftreten. Daher sollten zu den Busteilnehmern (Node 1...n) idealerweise keine Stichleitungen führen.

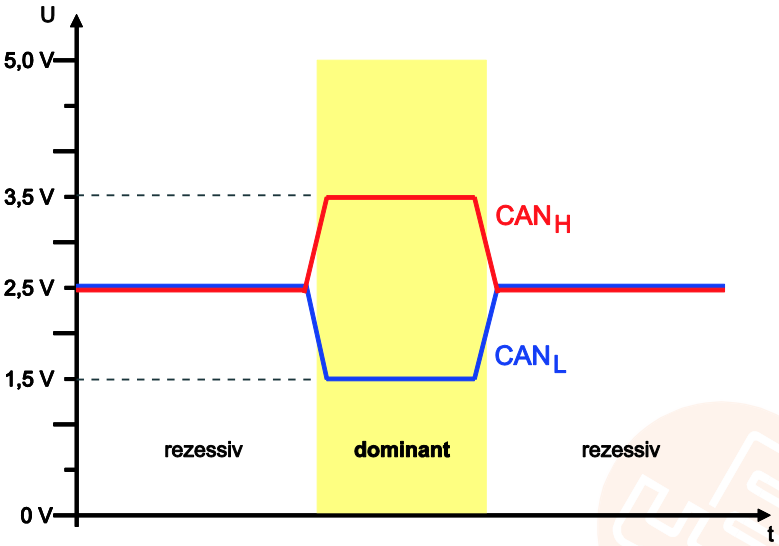
Falls Stichleitungen nicht vermeidbar sind:
- Einzelne Stichleitungen von bis zu 2 m Länge (bezogen auf 125 kBit/s) gelten als unkritisch.
- Die Summe aller Stichleitungen im Gesamtsystem sollte 30 m nicht übersteigen.

► In besonderen Fällen die Leitungslängen der Linie und der Stiche genau berechnen!

CAN-Buspegel

1179

Der CAN-Bus befindet sich im inaktiven (rezessiven) Zustand, wenn die Ausgangstransistorpaare in allen Busteilnehmern ausgeschaltet sind. Wird mindestens ein Transistorpaar eingeschaltet, wird ein Bit auf den Bus gegeben. Der Bus wird dadurch aktiv (dominant). Es fließt ein Strom durch die Abschlusswiderstände und erzeugt eine Differenzspannung zwischen den beiden Busleitungen. Die rezessiven und dominanten Zustände werden in den Busknoten in entsprechende Spannungen umgewandelt und von den Empfängerschaltkreisen erkannt.



Grafik: Buspegel

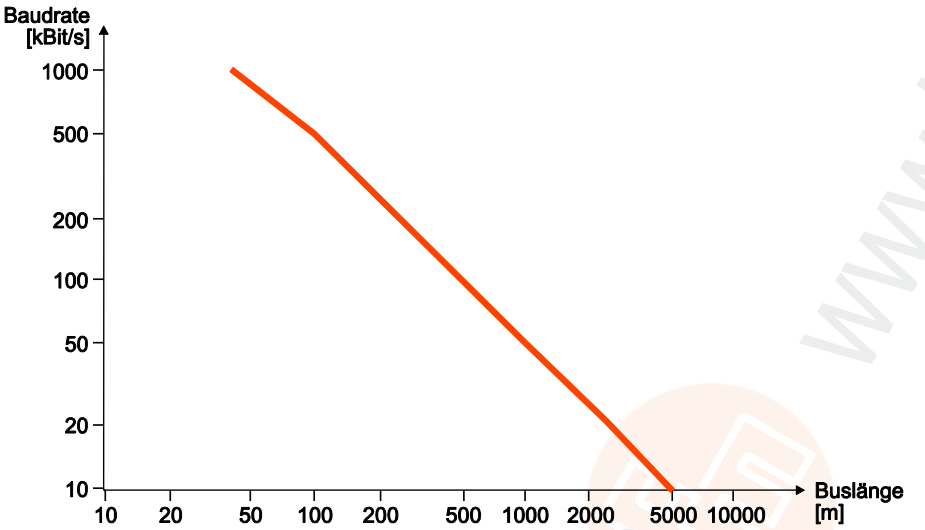
Durch diese differentielle Übertragung mit gemeinsamem Rückleiter wird die Übertragungssicherheit entscheidend verbessert. Störspannungen, die von außen auf das System einwirken, oder Massepotential-Verschiebungen beeinflussen beide Signalleitungen mit gleichen Störgrößen. Dadurch fallen die Störungen bei der Differenzbildung im Empfänger wieder heraus.

Busleitungslänge

1180

Die Länge der Busleitung ist abhängig von:
- Beschaffenheit der Busverbindung (Kabel, Steckverbinder),
- Leitungswiderstand,
- benötigte Übertragungsrate (Baudrate),
- Länge der Stichleitungen.

Vereinfachend kann man von folgender Abhängigkeit zwischen Buslänge und Baudrate ausgehen:



Grafik:
Busleitungslänge

Baudrate [kBit/s]	Buslänge [m]	nominelle Bit-Zeit [µs]
1 000	40	1
800	50	1,25
500	100	2
250	250	4
125	500	8
62,5	1 000	20
20	2 500	50
10	5 000	100

Tabelle: Abhängigkeiten Buslänge / Baudrate / Bit-Zeit

Leitungsquerschnitte

1181

► Für die Auslegung des CAN-Netzes auch den Leitungsquerschnitt der eingesetzten Busleitung beachten!

Die folgende Tabelle beschreibt die Abhängigkeit des Leitungsquerschnitts bezogen auf die Leitungslänge und der Anzahl der daran angeschlossenen Teilnehmer (Knoten):

Leitungslänge [m]	Leitungsquerschnitt [mm²]		
	bei 32 Knoten	bei 64 Knoten	bei 100 Knoten
≤ 100	0,25	0,25	0,25
≤ 250	0,34	0,50	0,50
≤ 500	0,75	0,75	1,00

Abhängig von den EMV-Anforderungen können Sie die Busleitungen wie folgt ausführen:
- parallel mit / ohne Abschirmung
- als Twisted-Pair mit / ohne Abschirmung

3.1.2 CAN: Software

Inhalt

14104

IDs (Adressen) in CAN

15448

In CAN werden diverse Arten von Kennungen (hier: IDs) unterschieden:

- COB-ID
  Der **C**ommunication-**Ob**ject-**Id**entifier adressiert die Nachricht (= das Kommunikationsobjekt). Ein Kommunikationsobjekt besteht aus einer netzwerkweiten CAN-Nachricht.
  Je niedriger die COB-ID, desto höher die Priorität der Nachricht.
- Download-ID
  Die Download-ID bezeichnet die Kennung für den Programm-Download und den Wartungszugriff in CODESYS.
  ifm nutzt hierfür den SDO-Mechanismus aus dem CANopen-Protokoll.
  Die Software addiert dazu die Download-ID zur Basisadresse.
- Node-ID
  Der **Node-Id**entifier ist ein eindeutiger Bezeichner für CANopen-Geräte (Devices) im CAN-Netzwerk. Die Node-ID ist auch Bestandteil einiger vordefinierter Verbindungssätze (→ **Funktions-Code / Predefined Connectionset** (→ S. 72)). Anhand der Node-ID werden die COB-IDs ermittelt, wenn die vordefinierten Verbindungseinstellungen verwendet werden.
  Die Node-ID und die Download-ID müssen sich im gleichen CAN-Netzwerk unterscheiden!

13182

Gegenüberstellung Download-ID vs. Node-ID

Programm-Download / Wartungszugriff		CANopen	
Download-ID	**COB-ID SDO**	**Node-ID**	**COB-ID SDO**
1…127	TX: 0x580 + Download-ID	1…127	TX: 0x580 + Node-ID
	RX: 0x600 + Download-ID		RX: 0x600 + Node-ID

TX = Slave sendet an Master
RX = Slave empfängt von Master

HINWEIS

Die CAN-Download-ID des Geräts muss mit der in CODESYS eingestellten CAN-Download-ID übereinstimmen!

Im selben CAN-Netzwerk müssen die CAN-Download-IDs einmalig sein!

Den verschiedenen CAN-Schnittstellen eines Geräts darf die gleiche CAN-Download-ID zugewiesen werden, vorausgesetzt, diese CAN-Schnittstellen sind an getrennten CAN-Netzwerken angeschlossen.

COB-ID

18384
18379

Je nach Typ sind folgende COB-Identifier frei verfügbar für den Datentransfer:

COB-ID (base)	COB-ID (extended)
11 Bit	29 Bit
COB-Identifier: 0...2 047	COB-Identifier: 0...536 870 911
Standard-Anwendungen	Motor-Management (SAE J1939), Truck & Trailer Interface (ISO 11992)

18382

Beispiel 11-Bit COB-ID (base):

SOF	COB-ID (base) Bit 28 ... Bit 18											RTR	IDE
0	0	0	0	0	1	1	1	1	1	1	1	0	0
	0			7				F					

Beispiel 29-Bit COB-ID (extended):

SOF	COB-ID (base) Bit 28 ... Bit 18										SRR	IDE	COB-ID (extended) Bit 17 ... Bit 0																RTR
0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1			F				C				0				0				0				0				

Legende:
SOF = Start of frame
Flanke von rezessiv zu dominant

RTR = **R**emote **t**ransmission **r**equest
dominant: Diese Nachricht liefert Daten
rezessiv: Diese Nachricht fordert Daten an

IDE = **Id**entifier **e**xtension flag
dominant: Hiernach folgen Steuerungs-Bits
rezessiv: Hiernach folgt der zweite Teil des 29-Bit-Identifier

SRR = **S**ubstitute **r**emote **r**equest
rezessiv: Extended CAN-ID: Ersetzt das RTR-Bit an dieser Stelle

3.2 CAN-Schnittstellen

Inhalt

14101

Anschlüsse und Daten → Datenblatt

3.2.1 CAN: Schnittstellen und Protokolle

18060

Die Geräte werden je nach Aufbau der Hardware mit mehreren CAN-Schnittstellen ausgerüstet. Grundsätzlich können alle Schnittstellen unabhängig voneinander mit folgenden Funktionen genutzt werden:

- Layer 2: CAN auf Ebene 2
- CANopen-Master
- CANopen-Slave
- CANopen-Netzwerkvariablen (via CODESYS)
- SAE J1939 (für Antriebsmanagement)
- Buslast-Erkennung
- Errorframe-Zähler
- Download-Schnittstelle
- 100 % Buslast ohne Paketverlust

Wenn für ein Gerät mehrere CANopen-fähige Schnittstellen verfügbar sind, dann gilt (abhängig vom Gerät) für die Zuordnung des CANopen-Protokolls zur CAN-Schnittstelle:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
die Reihenfolge, mit der Sie in der Steuerungskonfiguration die Unterelemente anhängen	die Wahl der CAN-Schnittstelle, an der Sie das Unterelement anhängen

3.3 CAN: Datenaustausch

19015

Der CAN-Datenaustausch erfolgt über das in der ISO 11898 international genormte CAN-Protokoll der Verbindungsschicht (Ebene 2) des siebenschichtigen ISO/OSI-Referenzmodells.

Jeder Bus-Teilnehmer kann Nachrichten senden (Multimaster-Fähigkeit). Der Datenaustausch arbeitet ähnlich dem Rundfunk. Daten werden ohne Absender und Adresse auf den Bus gesendet. Die Daten sind lediglich durch ihren Identifier gekennzeichnet. Es ist Aufgabe jedes Teilnehmers, die gesendeten Daten zu empfangen und an Hand des Identifiers zu prüfen, ob die Daten für diesen Teilnehmer relevant sind. Dieser Vorgang wird vom CAN-Controller in Verbindung mit dem Laufzeitsystem automatisch durchgeführt.

Für den normalen CAN-Datenaustausch muss der Programmierer lediglich bei der Software-Erstellung die Datenobjekte mit ihren Identifiern dem System bekannt machen. Dies erfolgt über die gerätespezifischen FBs für CAN-Transmit und CAN-Receive:

- CAN-Bausteine auf Schicht 2 (RAW-CAN):
  einfache Funktionen.
- CAN-Bausteine nach SAE J1939:
  hochwertige Funktionen für das Motor-Management.
- CAN-Bausteine nach CANopen:
  komplexe CAN-Funktionen.
- CANopen-Safety-Bausteine (optional):
  CAN-Funktionen für die Sicherheitsanwendungen in SafetyControllern.

Über diese FBs werden folgende Einheiten zu einem Datenobjekt verknüpft:
- die Arbeitsdaten,
- der Frame-Typ (optional),
- der gewählte Identifier (ID).

Diese Datenobjekte nehmen am Datenaustausch über den CAN-Bus teil. Die Sende- und Empfangsobjekte können aus allen gültigen IEC-Datentypen (z.B. BOOL, WORD, INT, ARRAY) definiert werden.

Die CAN-Nachricht besteht aus einem COB-Identifier (**COB-ID** (→ S. 30)) und maximal 8 Datenbytes. Die ID repräsentiert nicht das Absender- oder Empfängermodul, sondern kennzeichnet die Nachricht. Um Daten zu übertragen, ist es notwendig, dass im Sendemodul ein Sendeobjekt und in mindestens einem anderen Modul ein Empfangsobjekt deklariert ist. Beide Deklarationen müssen dem gleichen Identifier und dem gleichen Telegrammtyp (base oder extended) zugeordnet sein.

3.3.1 Daten empfangen

19016

Grundsätzlich werden die empfangenen Datenobjekte automatisch (also ohne Einfluss durch den Anwender) in einem Zwischenspeicher abgelegt.

Pro Identifier steht ein solcher Zwischenspeicher (Warteschlange) zur Verfügung. Dieser Zwischenspeicher wird in Abhängigkeit von der Anwendersoftware nach dem FiFo-Prinzip (**F**irst **I**n, **F**irst **O**ut) über den gerätespezifischen CAN-Receive-Baustein entleert.

3.3.2 Daten senden

19017

Durch den Aufruf des gerätespezifischen CAN-Transmit-Bausteins übergibt das Anwendungsprogramm genau eine CAN-Nachricht an den Zwischenspeicher (Warteschlange). Als Rückgabe erhält man die Information, ob noch Platz im Zwischenspeicher war und die Nachricht erfolgreich übergeben wurde. Der Zwischenspeicher übergibt eigenständig die Nachricht an den CAN-Controller, der die Nachricht auf dem Bus versendet.

Der Sendeauftrag wird abgewiesen, wenn der Zwischenspeicher schon voll ist. Der Sendeauftrag muss dann durch das Anwendungsprogramm wiederholt werden. Der Programmierer bekommt diese Information durch ein Bit angezeigt.

Im Zwischenspeicher findet keine Sende-Priorisierung der Telegramme anhand ihrer **COB-ID** (→ S. 30) statt. Der Programmierer muss daher die Reihenfolge, mit der er die Nachrichten übergibt, umsichtig wählen.

3.4 Technisches zu CANopen

Inhalt

13822

3.4.1 CANopen Netzwerk-Konfiguration, Status- und Fehlerbehandlung

13824

Die Netzwerkkonfiguration und die Parametrierung der angeschlossenen Geräte programmieren Sie über die Programmiersoftware CODESYS.

Bei einigen Geräten erreichen Sie die Fehlermeldungen nur über verschachtelte Variablenstrukturen im CANopen-Stack.

Die nachfolgende Dokumentation zeigt Ihnen den Aufbau und die Anwendung der Netzwerkkonfiguration.

Die folgenden Kapitel beschreiben die internen Bausteine des CODESYS-CANopen-Stacks und ihre Anwendung. Außerdem bekommen Sie einen Einblick über die Anwendung des Netzwerkkonfigurators.

HINWEIS

Es ist zwingend notwendig, dass Sie nur die jeweilige gerätespezifische Bibliothek einsetzen.
Den Zusammenhang erkennen Sie an der im Dateinamen integrierten Geräte-Artikelnummer.

Beispiel: CR0032 mit CANopen-Master für CAN-Schnittstelle 1:
→ `ifm_CR0032_CANopen1Master_Vxxyyzz.lib`

→ Gerätehandbuch, Kapitel 'Target einrichten'

Bei Verwendung anderer Bibliotheken kann das Gerät nicht mehr richtig funktionieren.

3.4.2 CANopen-Unterstützung durch CODESYS

1857

Allgemeines zu CANopen mit CODESYS

13826

CODESYS ist eines der führenden Systeme für die Programmierung von Steuerungssystemen nach dem internationalem Standard IEC 61131. Um CODESYS für den Anwender interessanter zu gestalten, wurden viele wichtige Funktionen in das Programmiersystem integriert, darunter auch ein Konfigurator für CANopen. Mit diesem CANopen-Konfigurator können Sie CANopen-Netzwerke (in einzelnen Punkten eingeschränkt) unter CODESYS konfigurieren.

Ein **ecomat*mobile***-Controller kann als CANopen-Master und als CANopen-Slave genutzt werden.

> **HINWEIS**
>
> Für alle **ecomat*mobile***-Controller und das PDM360smart nur die 3S-CANopen-Bibliotheken mit folgendem Zusatz im Dateinamen einsetzen: "**OptTableEx**"
>
> Wenn Sie ein Projekt neu anlegen, werden diese Bibliotheken im Allgemeinen automatisch geladen. Sollten Sie selbst die Bibliotheken über die Bibliotheksverwaltung einfügen, müssen Sie auf die korrekte Auswahl achten.
>
> Die CANopen-Bibliotheken ohne diesen Zusatz werden für andere programmierbare **ifm**-Geräte genutzt.

CANopen: Begriffe und Implementierung

1858

In Bezug auf die Übertragung von Prozessdaten in CANopen gibt es keine Master und Slaves in einem CAN-Netz. Jedoch gibt es bei CANopen eine Master-/Slave-Architektur in Bezug auf das Netzwerkmanagement (NMT) und bei der Konfiguration.

Das CAN-Protokoll (unterhalb des CANopen-Protokolls) kennt keine Master-/Slave-Beziehungen.

Die Implementierung geht davon aus, dass ein CAN-Netz als Peripherie einer mit CODESYS programmierbaren Steuerung dient.

Der CANopen-Master ist NMT-Master und Konfigurationsmaster. Im Normalfall wird der Master dafür sorgen, dass das Netz in Betrieb genommen werden kann. Er übernimmt die Initiative, die einzelnen Nodes (= Netzwerk-Knoten) zu starten, die ihm per Konfiguration bekannt sind. Diese Nodes werden als Slaves bezeichnet.

Um den Master ebenfalls dem Status eines CANopen-Slaves näherzubringen, wurde ein Objektverzeichnis für den Master eingeführt. Auch kann der Master als SDO-Server (SDO = Service Data Object) auftreten und nicht nur in der Konfigurationsphase der Slaves als SDO-Client.

3.4.3 CANopen-Master

Inhalt

1859

CANopen-Bibliotheken

Inhalt

18033

Bibliotheken: vom System für CANopen erforderlich

14356

Die folgenden Bibliotheken werden bei Verwendung der CANopen-Funktionalität automatisch in das CODESYS-Projekt eingebunden:

- die CODESYS-Bibliothek `3S_CanDrvOptTableEx.LIB`
- die CODESYS-Bibliothek `3S_CANopenMasterOptTableEx.LIB`
- die CODESYS-Bibliothek `3S_CANopenManagerOptTableEx.LIB`
- die CODESYS-Bibliothek `3S_CanOpenDeviceOptTableEx.LIB`
- die CODESYS-Bibliothek `3S_CanOpenNetVarOptTableEx.LIB`
- die CODESYS-Bibliothek `SysLibCallback.LIB`

Die darin enthaltenen Funktionsbausteine und Funktionen dürfen NICHT im Code des Anwendungsprogramms direkt aufgerufen werden!

Funktionen der CANopen-Bibliotheken

1990

Folgende in CANopen definierten Funktionen werden zurzeit von der CODESYS-CANopen-Bibliothek unterstützt:

- **PDOs Senden:** Master sendet zu den Slaves (Slave = Knoten, Device)
  • Senden ereignisgesteuert (d.h. bei Änderung),
  • Senden zeitgesteuert (RepeatTimer) oder
  • Senden als synchrone PDOs, d.h. immer wenn ein SYNC vom Master gesendet wurde.
  Auch eine externe SYNC-Quelle kann benutzt werden, um das Senden von synchronen PDOs zu initiieren.
- **PDOs Empfangen:** Master empfängt vom Slave
  je nach Slave: ereignisgesteuert, abfragegesteuert, azyklisch und zyklisch.
- **PDO-Mapping**
  Zuordnung zwischen lokalem Objektverzeichnis und PDOs vom/zum CANopen-Slave (wenn vom Slave unterstützt).
- **SDO Senden und Empfangen** (unsegmentiert, d.h. 4 Bytes pro Objektverzeichnis-Eintrag)
  • automatische Konfiguration aller Slaves über SDOs beim Systemstart
  • anwendungsgesteuertes Senden und Empfangen von SDOs zu konfigurierten Slaves
- **Synchronisation**
  automatisches Senden von SYNC-Nachrichten durch den CANopen-Master.
- **Nodeguarding**
  automatisches Senden von Guarding-Nachrichten und Überwachung der Lifetime für jeden entsprechend konfigurierten Slave.
  ⓘ Wir empfehlen: Für aktuelle Geräte besser mit Heartbeat arbeiten, weil dann die Buslast niedriger ist.
- **Heartbeat**
  automatisches Senden und Überwachen von Heartbeat-Nachrichten.

Folgende in CANopen definierten Funktionen werden von der CODESYS-CANopen-Bibliothek derzeit **nicht** unterstützt:

- dynamische Identifier-Zuordnung
- dynamische SDO-Verbindungen
- blockweiser SDO-Transfer (kann für einige **ifm**-Geräte mit Funktionsbausteinen in der jeweiligen **ifm**-Gerätebibliothek realisiert werden)
- segmentierter SDO-Transfer (kann mit Funktionsbausteinen in der jeweiligen **ifm**-Gerätebibliothek realisiert werden)
- alle oben nicht genannten Möglichkeiten des CANopen Protokolls.

Die folgenden Funktionen werden durch die ifm-CANopen-Bibliothek bereitgestellt:

- **Emergency**
  Empfangen und Speichern von Emergency-Nachrichten von den konfigurierten Slaves.
- **Node-ID** und **Baudrate** in den Slaves setzen
  Durch Aufruf einer einfachen Funktion können Node-ID und Baudrate eines Slaves zur Laufzeit der Anwendung gesetzt werden.

Ein CANopen-Projekt erstellen

19021

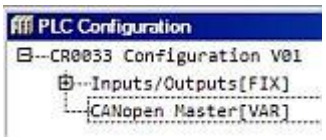
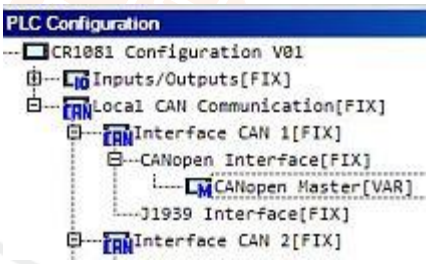
Die Erstellung eines neuen Projektes mit einem CANopen-Master wird nachfolgend schrittweise beschrieben. Dabei gehen wir davon aus, dass Sie CODESYS auf dem Rechner bereits fertig installiert haben und die Target- und EDS-Dateien ebenfalls richtig installiert oder kopiert wurden.

Eine weitergehende detaillierte Beschreibung zur Einstellung und Anwendung des Dialogs Steuerungs- und CANopen-Konfiguration finden Sie hier:
- im CODESYS-Handbuch unter [Ressourcen] > [Steuerungskonfiguration]
- in der CODESYS-Online-Hilfe.

► Nach der Neuanlage eines Projektes (→ Gerätehandbuch, Kapitel 'Target einrichten') in der Steuerungskonfiguration den CANopen-Master einfügen:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
► Rechtsklick auf die erste Zeile ("CRnnnn Configuration Vnn") ► [Unterelement anhängen] > [CANopen Master]	► An der gewünschten CAN-Schnittstelle: Rechtsklick auf [CANopen Interface] ► [Unterelement anhängen] > [CANopen Master]

Wenn für ein Gerät mehrere CANopen-fähige Schnittstellen verfügbar sind, dann gilt (abhängig vom Gerät) für die Zuordnung des CANopen-Protokolls zur CAN-Schnittstelle:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
die Reihenfolge, mit der Sie in der Steuerungskonfiguration die Unterelemente anhängen	die Wahl der CAN-Schnittstelle, an der Sie das Unterelement anhängen
Beispiel mit CR0033: 	Beispiel mit CR1081: 

> Die folgenden Bibliotheken und Software-Module werden automatisch eingebunden:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
die `STANDARD.LIB`, welche die in der IEC-61131 definierten Standardfunktionen für die Steuerung zu Verfügung stellt,	die `STANDARD.LIB`, welche die in der IEC-61131 definierten Standardfunktionen für die Steuerung zu Verfügung stellt,
die `3S_CanOpenManager.LIB`, welche die CANopen-Basisfunktionalitäten zur Verfügung stellt (ggf. `3S_CanOpenManagerOptTable.LIB` für C167-Controller),	–
eine oder mehrere der Bibliotheken `3S_CANopenNetVar.LIB`, `3S_CANopenDevice.LIB` und `3S_CANopenMaster.LIB` (ggf. `3S_...OptTable.LIB` für C167-Controller), je nach gewünschter Funktionalität,	–
die Systembibliotheken, z.B.: `SysLibSem.LIB` und `SysLibCallback.LIB`.	die Systembibliotheken, z.B.: `SysLibSem.LIB` und `SysLibCallback.LIB`.

► Um die vorbereiteten Netzwerkdiagnose-, Status- und EMCY-Funktion zu nutzen, die ifm-CANopen-Master-Bibliothek (oder die ifm-CANopen_NT-Bibliothek) manuell im Bibliotheksverwalter einfügen. Ohne diese Bibliothek müssen Sie die Netzwerkinformationen direkt aus den verschachtelten Strukturen der 3S-CANopen-Bibliotheken auslesen.

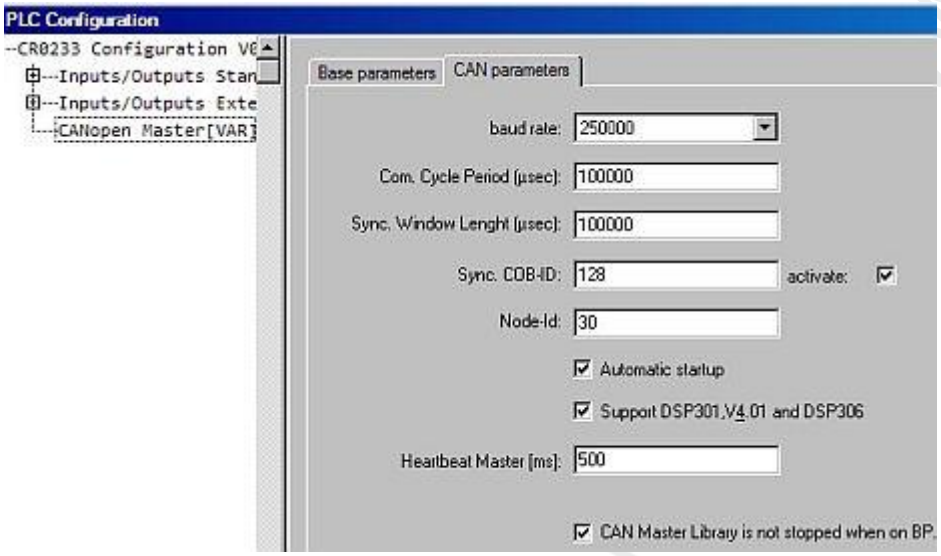
► Zusätzlich die folgenden Bibliotheken und Software-Module einbinden:

- die Gerätebibliothek für die jeweilige Hardware, z.B. `ifm_CR0032_Vxxyyzz.LIB`. Diese Bibliothek stellt alle gerätespezifischen Funktionen zur Verfügung.
- EDS-Dateien für alle Slaves, die am Netzwerk betrieben werden sollen. Die EDS-Dateien für alle ifm-CANopen-Slaves stellt die **ifm electronic gmbh** zur Verfügung.
  Für die EDS-Dateien von Fremd-Knoten ist der jeweilige Hersteller verantwortlich.

CANopen-Master: Register [CAN-Parameter]

1967

► In diesem Dialogfenster für den Master die wichtigsten Parameter einstellen.



Beispiel: Steuerungskonfiguration für CR0233 CANopen-Master an CAN-Schnittstelle 1

Legende:

CAN Parameter: Baudrate

10028

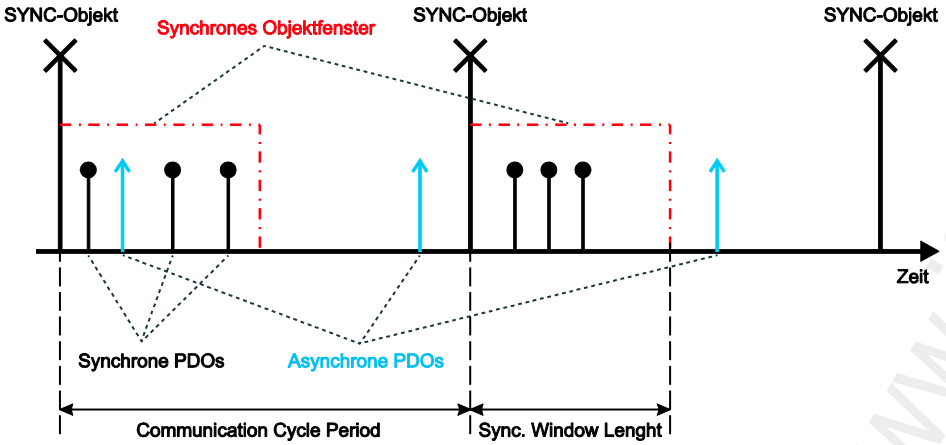
► An dieser Stelle die Baudrate für den Master wählen.

- Die Baudrate muss der Übertragungsgeschwindigkeit der anderen Netzwerkteilnehmer entsprechen.

CAN Parameter: Communication Cycle Period / Sync. Window Length

10029

Nach Ablauf der [Communication Cycle Period] wird eine SYNC-Nachricht vom Master verschickt:



Die [Sync. Window Length] gibt die Zeit an, in der synchrone PDOs von den anderen Netzwerkteilnehmern verschickt und vom Master empfangen werden müssen.

Da in den meisten Anwendungen keine besonderen Anforderungen an das SYNC-Objekt gestellt werden, können Sie für die [Communication Cycle Period] und die [Sync. Window Length] die gleiche Zeit einstellen.
Bitte beachten Sie, dass die Zeit in [µsec] eingegeben wird (der Wert 50000 entspricht 50 ms).

CAN Parameter: Sync. COB-ID

10030

In diesem Feld kann der Identifier für die SYNC-Nachricht einstellt werden. Diese wird immer nach Ablauf der Communication Cycle Period verschickt. Der Defaultwert ist 128 und sollte im Normalfall nicht geändert werden. Um das Versenden der SYNC-Nachricht zu aktivieren, muss das Kontrollfeld [aktivieren] gesetzt sein.

HINWEIS

Die SYNC-Nachricht wird immer am Anfang eines Programmzyklus erzeugt. Danach werden die Eingänge gelesen, das Programm abgearbeitet, die Ausgänge geschrieben und zuletzt alle synchronen PDOs gesendet.

Bitte beachten Sie, dass sich die SYNC-Zeit verlängert, wenn die eingestellte SYNC-Zeit kürzer als die Programmzykluszeit ist.

Beispiel: Communication Cycle Period = 10 ms und Programmzykluszeit = 30 ms.
Die SYNC-Nachricht wird erst nach 30 ms versendet.

CAN Parameter: Node-ID

10031

► Setzen Sie in diesem Feld die Knotennummer (nicht die Download-ID!) des Masters ein.

Die Knotennummer darf im Netzwerk nur einmal vorkommen, andernfalls kommt es zu Kommunikationsstörungen.

CAN Parameter: Automatisch starten

10032

Option [Automatisch starten] = aktiviert:
Nach einer erfolgreichen Konfiguration:

> Das Netzwerk und die angeschlossenen Knoten werden in den Zustand OPERATIONAL gesetzt und damit gestartet.

Option [Automatisch starten] = deaktiviert:

► Das Netzwerk manuell starten!

CAN Parameter: Heartbeat

10033

Wenn die anderen Teilnehmer im Netzwerk Heartbeat unterstützen, kann die Option [DSP301, V4.01... unterstützen] selektiert werden.

Bei Bedarf kann der Master noch ein eigenes Heartbeat-Signal nach Ablauf der eingestellten Zeit erzeugen.

CAN Parameter: CAN-Master läuft weiter

18040

Option [CAN-Master läuft weiter] = aktiviert UND
wurde in CODESYS ein "Break Point" gesetzt:
> der CAN-Master läuft weiter, aber
> das Anwendungsprogramm bleibt am Break Point stehen.

⚠ WARNUNG

Gefahr aus unkontrollierten Maschinenbewegungen!

Der Controller kann eventuell Slave-Signale nicht mehr verarbeiten, die zum Beenden einer (vor dem Break Point) begonnenen Bewegung führen sollten.

► Diese Option nur mit großer Umsicht einsetzen!

CANopen-Slaves einfügen und konfigurieren

Inhalt

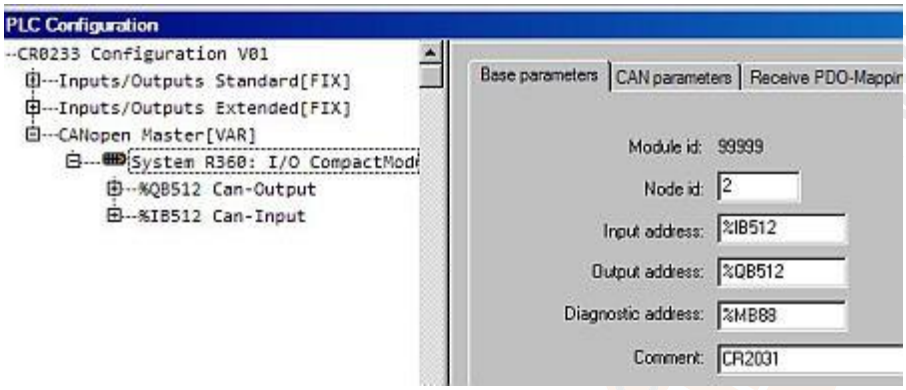
1861

Als nächstes können Sie nun die CANopen-Slaves einfügen.

► Dazu den Dialog in der Steuerungskonfiguration [Einfügen] > [Unterelement anhängen] aufrufen.

> Eine Liste der im Verzeichnis PLC_CONF gespeicherten CANopen-Gerätebeschreibungen (EDS-Dateien) erscheint.

► Durch Auswahl des entsprechenden Gerätes wird dieses direkt in den Baum der Steuerungskonfiguration eingefügt.



Beispiel: Steuerungskonfiguration für CR0233 CANopen-Master mit angeschlossenem I/O CompactModul CR2031

HINWEIS

Wird ein Slave über den Konfigurationsdialog in CODESYS hinzugefügt, wird für jeden Knoten dynamisch Quellcode in das Anwendungsprogramm integriert.

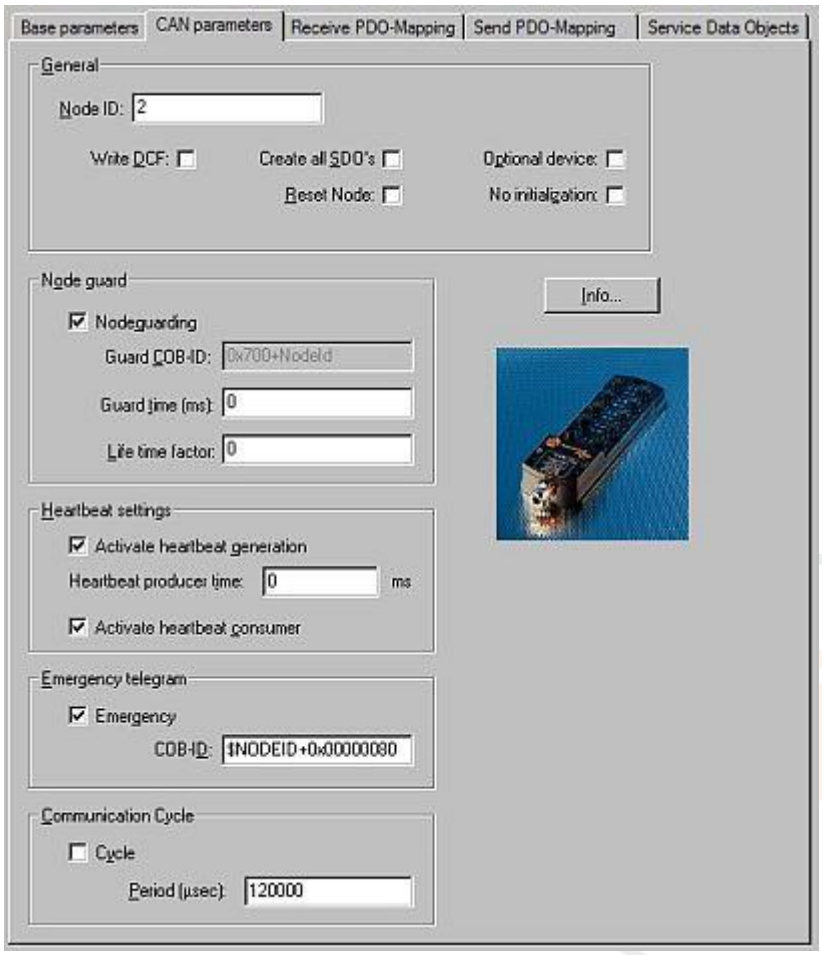
Gleichzeitig verlängert jeder zusätzlich hinzugefügte Slave die Zykluszeit des Anwendungsprogramms. Das bedeutet: in einem Netzwerk mit vielen Slaves kann der Master keine weiteren zeitkritischen Aufgaben (z.B. den FB OCC_TASK) abarbeiten.

Ein Netzwerk mit 27 Slaves hat eine Grund-Zykluszeit von 30 ms.

WICHTIG: die maximale Zeit für einen SPS-Zyklus sollte 50 ms nicht überschreiten! (Watchdog-Zeit = 100 ms)

CANopen-Slave: Register [CAN Parameter]

1968



Legende:

CAN Parameter: Node-ID

10036

Die Node-ID dient zur eindeutigen Identifizierung des CAN-Moduls im Netzwerk.

- Die ID soll der am Modul eingestellten Nummer (1...127) entsprechen.
- Die ID als Dezimalwert eingeben!
- Die ID wird automatisch um eins erhöht, wenn Sie ein weiteres Modul hinzufügen.

CAN Parameter: DCF schreiben

10037

Option [DCF schreiben] = aktiviert:
Nach dem Einfügen einer EDS-Datei wird im eingestellten Verzeichnis für Übersetzungsdateien eine DCF-Datei erstellt.
Der Name der DCF-Datei setzt sich zusammen aus:
• Name der EDS-Datei und
• angehängte Node-ID.

CAN Parameter: Alle SDOs erzeugen

10038

Option [Alle SDOs erzeugen] = aktiviert:
Für alle Kommunikationsobjekte werden SDOs erzeugt.
Default-Werte werden nicht erneut geschrieben!

CAN Parameter: Knoten zurücksetzen

10039

Option [Knoten zurücksetzen] = aktiviert:
Nach einem Neustart des Masters
> beim Initialisieren des CANopen-Netzwerks
> wird der Slave zurückgesetzt (via "load" und NMT-Kommando "Reset Node").

Anschließend: der Slave wird konfiguriert.

CAN Parameter: Optionales Gerät

10040

Option [Optionales Gerät] = aktiviert:
Der Master versucht nur einmal, von diesem Knoten zu lesen.
Bei fehlender Antwort:
- der Knoten wird ignoriert und
- der Master geht in den normalen Betriebszustand über.

Wenn in den CAN-Parametern des Masters die Option [Automatisch starten] = aktiviert:
Der Slave wird automatisch gestartet, sobald er zu einem späteren Zeitpunkt an das Netzwerk angeschlossen und erkannt wird.

CAN Parameter: Nicht initialisieren

10041

Option [Nicht initialisieren] = aktiviert:
Der Master nimmt den Knoten sofort in Betrieb, ohne ihm Konfigurations-SDOs zu schicken.
Die SDO-Daten werden aber dennoch erzeugt und auf der Steuerung gespeichert.

CAN Parameter: Nodeguarding- / Heartbeat-Einstellungen

10042

Je nach Gerät haben Sie die Wahl:
- [Nodeguarding] und [Life Time Factor] einstellen ODER
- [Heartbeat] einstellen.

Wird beides eingestellt, wird nur Heartbeat ausgeführt.

Wir empfehlen: Für aktuelle Geräte besser mit Heartbeat arbeiten, weil dann die Buslast niedriger ist.

CAN Parameter: Emergency Telegram

10043

Option [Emergency] = aktiviert:
Die EMCY-Nachrichten werden mit der angegebenen COB-ID übertragen.
Die Option ist im Normalfall aktiviert.

CAN Parameter: Communication Cycle

10044

In ganz speziellen Anwendungsfällen können Sie an dieser Stelle eine Überwachungszeit für die vom Master erzeugten SYNC-Nachrichten einstellen.

Bitte beachten Sie, dass diese Zeit länger als die SYNC-Zeit des Masters sein muss. Der optimale Wert muss experimentell ermittelt werden.

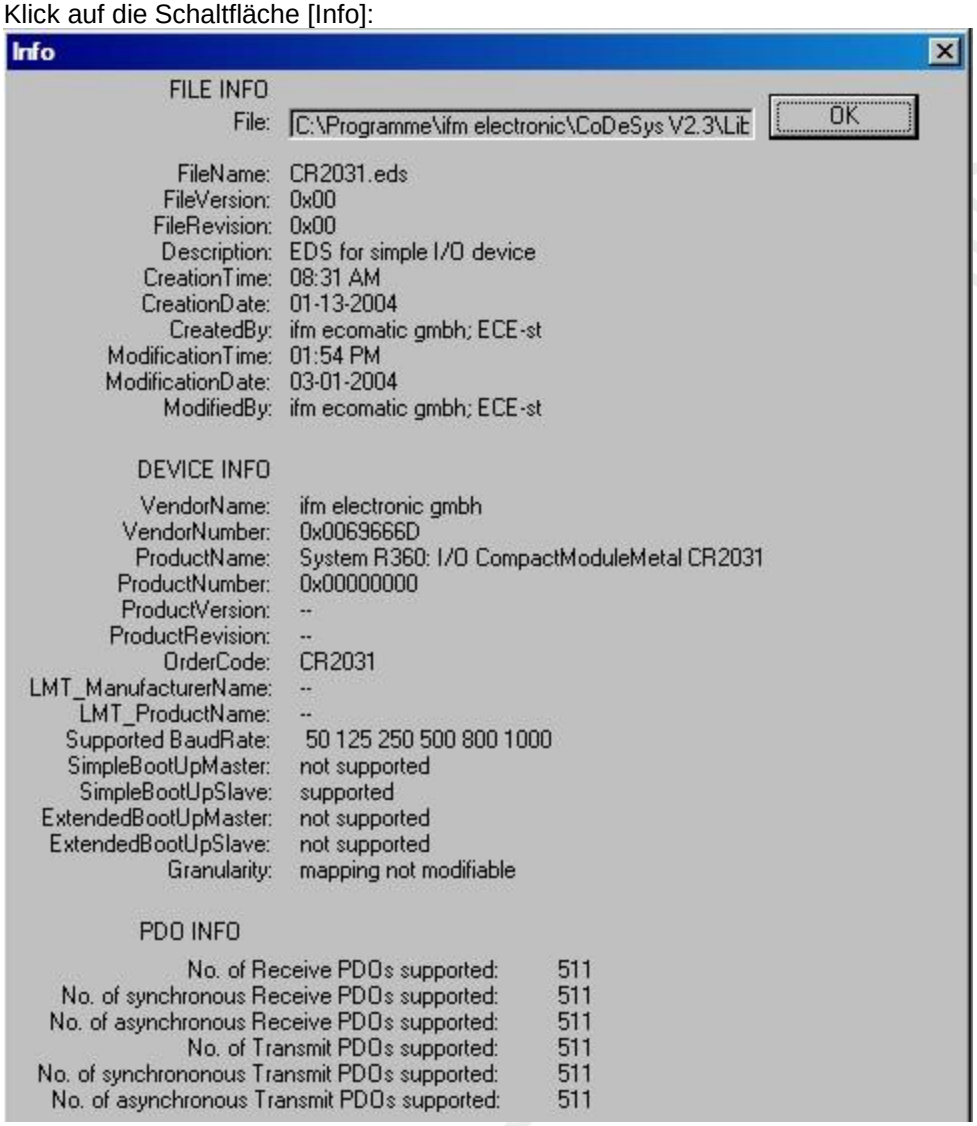
Nodeguarding oder Heartbeat reichen in den meisten Fällen zur Knotenüberwachung aus.

CAN Parameter: Info

18062

Für diesen Slave die Beschreibung in der EDS-Datei zeigen:

► Klick auf die Schaltfläche [Info]:



Beispiel: Info zum Slave CR2031

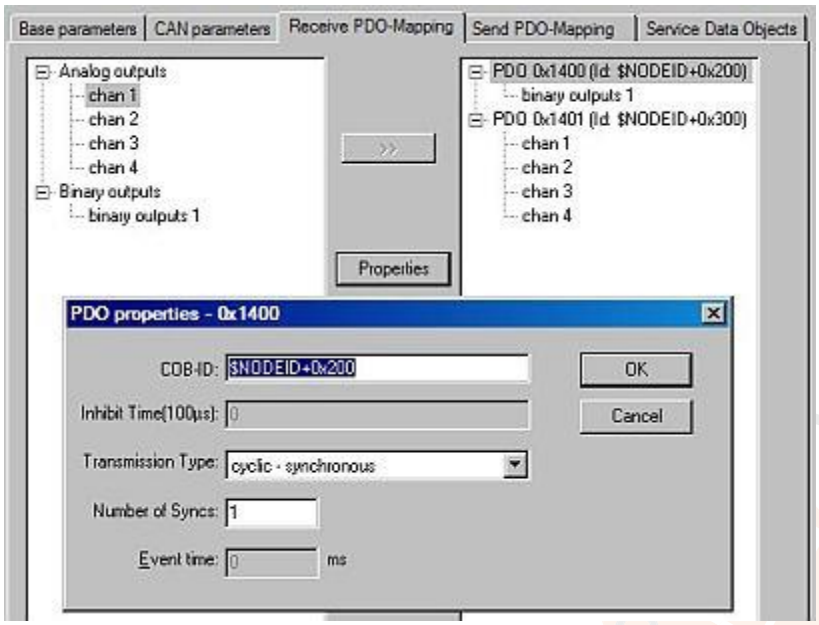
► Die Anzeige wieder schließen mit [OK].

CANopen-Slave: Register [PDO-Mapping empfangen] / [PDO-Mapping senden]

1969

In der EDS-Datei des Moduls ist die Zuordnung zwischen lokalem Objektverzeichnis und PDOs vom/zum CANopen-Slave beschrieben: das sogenannte "Mapping".

In den Registerkarten [PDO-Mapping empfangen] und [PDO-Mapping senden] kann dieses Mapping bei Bedarf geändert werden, sofern dies vom CAN-Modul unterstützt wird.



Auf der linken Seite stehen alle "mapbaren" Objekte der EDS-Datei zur Verfügung. Diese Objekte können zu den PDOs (Process Data Objects) der rechten Seite hinzugefügt oder dort wieder entfernt werden.

Die [StandardDataTypes] können eingefügt werden, um im PDO leere Zwischenräume zu erzeugen.

PDO-Mapping: Einfügen

10046

Mit der Schaltfläche [Einfügen] können Sie weitere PDOs erzeugen und mit entsprechenden Objekten belegen. Über die eingefügten PDOs erfolgt die Zuordnung der Ein- und Ausgänge zu den IEC-Adressen.

In der Steuerungskonfiguration werden die vorgenommenen Einstellungen nach Verlassen des Dialoges sichtbar. Die einzelnen Objekte können dort mit symbolischen Namen belegt werden.

PDO-Mapping: Eigenschaften

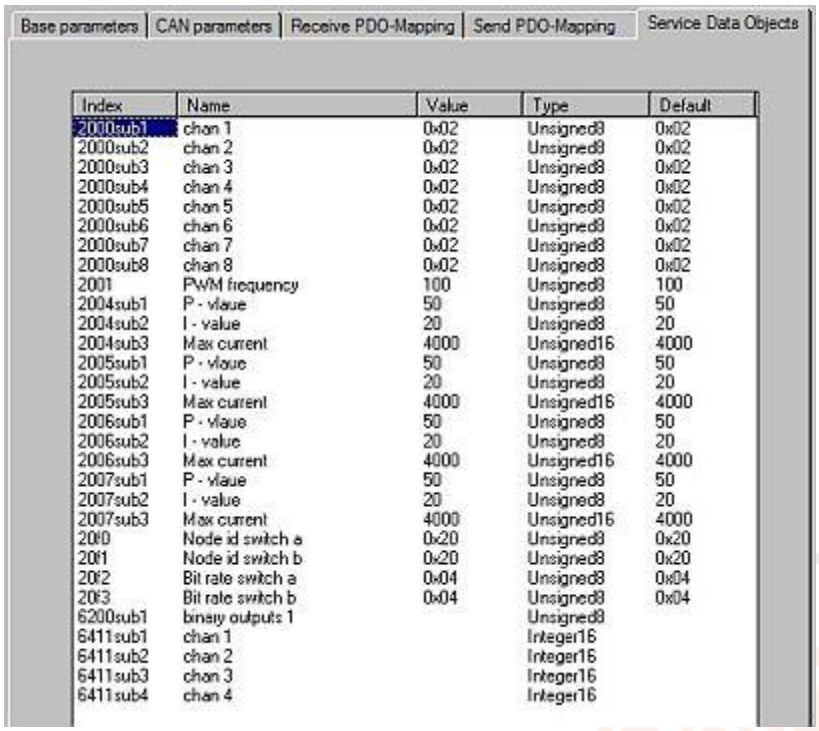
10047

Über Eigenschaften lassen sich die in der Norm definierten Eigenschaften der PDOs in einem Dialog editieren:

COB-ID	Jede PDO-Nachricht benötigt eine eindeutige COB-ID (Communication Object Identifier). Wird eine Option von dem Modul nicht unterstützt oder darf der Wert nicht verändert werden, so erscheint das Feld grau und kann nicht editiert werden.
Inhibit Time	Die Inhibit Time (100 µs) ist die minimale Zeit zwischen zwei Nachrichten dieses PDOs, damit die Nachrichten, die bei Änderung des Wertes übertragen werden, nicht zu häufig versendet werden. Die Einheit ist 100 µs.
Transmission Type	Bei Transmission Type erhalten Sie eine Auswahl von möglichen Übertragungmodi für dieses Modul: **acyclic – synchronous** Das PDO wird nach einer Änderung mit dem nächsten SYNC übertragen. **cyclic – synchronous** Das PDO wird synchron übertragen, wobei [Number of SYNCs] die Anzahl der Synchronisationsnachrichten angibt, die zwischen zwei Übertragungen dieses PDOs liegen. **asynchronous – device specific** Das PDO wird ereignisgesteuert, d.h. wenn sich der Wert ändert, übertragen. Welche Daten auf diese Weise übertragen werden können, ist im Geräteprofil festgelegt. **asynchronous – manufacturer specific** Das PDO wird ereignisgesteuert, d.h. wenn sich der Wert ändert, übertragen. Welche Daten auf diese Weise übertragen werden, wird vom Gerätehersteller festgelegt. **(a)synchronous – RTR only** Diese Dienste sind nicht implementiert. **Number of SYNCs** Abhängig vom Transmission Type ist dieses Feld editierbar zur Eingabe der Anzahl der Synchronisationsnachrichten (Definition in [CAN-Parameter-Dialog], [Com. Cycle Period], [Sync Window Length], [Sync. COB-Id]), nach denen das PDO wieder versendet werden soll. **Event-Time** Abhängig vom Transmission Type wird hier die Zeitspanne in Millisekunden [ms] angegeben, die maximal zwischen zwei Übertragungen des PDOs liegen soll.

CANopen-Slave: Register [Service Data Objects]

18036



Index	Name	Value	Type	Default
2000sub1	chan 1	0x02	Unsigned8	0x02
2000sub2	chan 2	0x02	Unsigned8	0x02
2000sub3	chan 3	0x02	Unsigned8	0x02
2000sub4	chan 4	0x02	Unsigned8	0x02
2000sub5	chan 5	0x02	Unsigned8	0x02
2000sub6	chan 6	0x02	Unsigned8	0x02
2000sub7	chan 7	0x02	Unsigned8	0x02
2000sub8	chan 8	0x02	Unsigned8	0x02
2001	PWM frequency	100	Unsigned8	100
2004sub1	P - value	50	Unsigned8	50
2004sub2	I - value	20	Unsigned8	20
2004sub3	Max current	4000	Unsigned16	4000
2005sub1	P - value	50	Unsigned8	50
2005sub2	I - value	20	Unsigned8	20
2005sub3	Max current	4000	Unsigned16	4000
2006sub1	P - value	50	Unsigned8	50
2006sub2	I - value	20	Unsigned8	20
2006sub3	Max current	4000	Unsigned16	4000
2007sub1	P - value	50	Unsigned8	50
2007sub2	I - value	20	Unsigned8	20
2007sub3	Max current	4000	Unsigned16	4000
20f0	Node id switch a	0x20	Unsigned8	0x20
20f1	Node id switch b	0x20	Unsigned8	0x20
20f2	Bit rate switch a	0x04	Unsigned8	0x04
20f3	Bit rate switch b	0x04	Unsigned8	0x04
6200sub1	binary outputs 1		Unsigned8	
6411sub1	chan 1		Integer16	
6411sub2	chan 2		Integer16	
6411sub3	chan 3		Integer16	
6411sub4	chan 4		Integer16	

Hier werden alle Objekte der EDS- oder DCF-Datei aufgelistet, die im Bereich von Index 0x2000...0x9FFF liegen und als beschreibbar definiert sind.

Zu jedem Objekt werden Index, Name, aktueller Wert, Typ und Default-Wert angegeben. Nur der Eintrag in [Wert] kann verändert werden.

SDOs: Wert ändern

18039

Der Eintrag in [Wert] kann verändert werden:

► Doppelklicken Sie auf den gewünschten Eintrag.

► Den neuen Wert eintragen.

► Die Änderung mit [Eingabe] bestätigen oder mit [ESC] verwerfen.

Bei der Initialisierung des CAN-Busses:

> Die Werte, die sich von den Default-Werten unterscheiden, werden in Form von SDOs (Service Data Objects) an die CAN-Module übertragen.

> Diese Werte haben damit direkten Einfluss auf das Objektverzeichnis des CANopen-Slaves.

> Diese Werte werden im Normalfall bei jedem Start des Anwendungsprogramms neu geschrieben – unabhängig davon, ob sie im CANopen-Slave dauerhaft gespeichert werden.

> Wurde der Wert gelöscht, ohne einen neuen Wert einzutragen, wird bei der Initialisierung der Default-Wert übertragen.

Der Master zur Laufzeit

Inhalt

8569

Hier lesen Sie über Funktionalität der CANopen-Master-Bibliotheken zur Laufzeit.

Die CANopen-Master-Bibliothek stellt dem CODESYS-Anwendungsprogramm implizite Dienste zur Verfügung, die für die meisten Anwendungen ausreichend sind. Diese Dienste werden für den Programmierer transparent integriert und stehen im Anwendungsprogramm ohne zusätzliche Aufrufe zur Verfügung.

Annahme: Sie haben die CANopen-Master-Bibliothek (oder die CANopen_NT-Bibliothek) manuell im Bibliotheksverwalter eingefügt.
Wunsch: Nutzung der Netzwerkdiagnose-, Status- und EMCY-Funktionen.

Zu den Diensten der CANopen-Master-Bibliothek zählen:

Reset aller konfigurierten Slaves am Bus beim Systemstart

19027

Die einzelnen NMT-Kommandos sind im CAN-Dokument DSP301 beschrieben.
NMT steht nach CANopen für **N**etwork **M**anagmen**t**.

Slaves einzeln nacheinander zurücksetzen

19029

Um die Slaves zurückzusetzen, wird standardmäßig das NMT-Kommando "Reset Node" benutzt, explizit für jeden Slave einzeln.

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
mit FB CANx_MASTER_STATUS: • GLOBAL_START = FALSE • RESET_ALL_NODES = FALSE	mit FB CANOPEN_NMTSERVICES: • NODE = Node-ID des Slaves • NMTSERVICE = 3

Alle Slaves gemeinsam zurücksetzen

19031

Um Slaves mit weniger leistungsstarken CAN-Controllern nicht zu überlasten, ist es sinnvoll, alle angeschlossenen Slaves gemeinsam mit einem Kommando "Reset All Nodes" zurückzusetzen.

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
mit FB CANx_MASTER_STATUS: • GLOBAL_START = TRUE • RESET_ALL_NODES = TRUE	mit FB CANOPEN_NMTSERVICES: • NODE = 0 • NMTSERVICE = 3

Abfrage des Slave-Gerätetyps

8021

Abfrage des Slave-Gerätetyps mittels SDO (Abfrage des Objekts 0x1000) und Vergleich mit der konfigurierten Slave-ID:

> Die Anfrage wird nach 0,5 s wiederholt, wenn:
- kein Gerätetyp wurde empfangen
- UND Slave wurde in der Konfiguration **nicht** als optional markiert
- UND Timeout ist **nicht** abgelaufen.

Fehlerstatus-Ausgabe für die Slaves, von denen ein falscher Gerätetyp empfangen wurde.

Konfiguration aller fehlerfrei detektierten Geräte

8022

Jedes SDO wird auf Antwort überwacht und wiederholt, wenn sich innerhalb der Überwachungszeit der Slave nicht meldet.

Automatische Konfiguration von Slaves

8023

Automatische Konfiguration von Slaves mittels SDOs bei laufendem Busbetrieb:
Voraussetzung: Der Slave hat sich mittels Bootup-Message beim Master angemeldet.

Start aller fehlerfrei konfigurierten Slaves

19032

Start aller fehlerfrei konfigurierten Slaves nach dem Ende der Konfiguration des betreffenden Slaves:

Slaves einzeln nacheinander starten

19034

Zum Starten der Slaves wird normalerweise das NMT-Kommando "Start Node" benutzt.

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
mit FB CANx_MASTER_STATUS: • GLOBAL_START = FALSE • START_ALL_NODES = FALSE	mit FB CANOPEN_NMTSERVICES: • NODE = Node-ID des Slaves • NMTSERVICE = 2

Alle Slaves gemeinsam starten

19036

Wie beim "Reset" kann dieses Kommando durch "Start All Nodes" ersetzt werden.

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
mit FB CANx_MASTER_STATUS: • GLOBAL_START = TRUE • START_ALL_NODES = TRUE	mit FB CANOPEN_NMTSERVICES: • NODE = 0 • NMTSERVICE = 2

Zyklisches Senden der SYNC-Message

8025

Dieser Wert ist nur bei der Konfiguration einstellbar.

Nodeguarding mit Lifetime-Überwachung

19038

Wir empfehlen: Für aktuelle Geräte besser mit Heartbeat arbeiten, weil dann die Buslast niedriger ist.
Nodeguarding mit Lifetime-Überwachung für jeden Slave einstellbar:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
Den Fehlerstatus für max. 8 Slaves zeigt der FB CANx_MASTER_STATUS: > ERROR_CONTROL = Liste der fehlenden Netzwerkknoten (Guard- oder Heartbeat-Fehler)	Mit FB CANOPEN_GETGUARDHBERRLIST in einem Array alle Knoten auflisten, für die der Master einen Fehler erkannt hat: Guarding-Fehler, Heartbeat-Fehler > N_NODES = Anzahl der Knoten mit Heartbeat- oder Guarding-Fehlern > NODEID = Liste der Knoten-IDs mit Heartbeat- oder Guarding-Fehler

Heartbeat vom Master an die Slaves

19039

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
Den Fehlerstatus für max. 8 Slaves zeigt der FB CANx_MASTER_STATUS: > ERROR_CONTROL = Liste der fehlenden Netzwerkknoten (Guard- oder Heartbeat-Fehler)	Mit FB CANOPEN_GETGUARDHBERRLIST in einem Array alle Knoten auflisten, für die der Master einen Fehler erkannt hat: Guarding-Fehler, Heartbeat-Fehler > N_NODES = Anzahl der Knoten mit Heartbeat- oder Guarding-Fehlern > NODEID = Liste der Knoten-IDs mit Heartbeat- oder Guarding-Fehler

Empfangen von Emergency-Nachrichten

19042

Empfangen von Emergency-Nachrichten für jeden Slave mit Speicherung der zuletzt empfangenen Emergency-Nachrichten:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
Fehlernachrichten auslesen mit FB CANx_MASTER_STATUS: > EMERGENCY_OBJECT_SLAVES = Liste der aktuellen EMCY-Nachrichten > GET_EMERGENCY = zuletzt erzeugte EMCY-Nachricht	Fehlernachrichten auslesen mit FB CANOPEN_GETEMCYMESSAGES: > N_MSGS = Anzahl der aufgelaufenen Nachrichten > EMCY = Liste der Emergency-Nachrichten Der jüngste Eintrag steht im Index 0

CANopen-Netzwerk starten

19048

Hier lesen Sie über das Starten des CANopen-Netzwerks.

Nach einem Download des Projekts auf die Steuerung oder einem Reset der Anwendung wird das CAN-Netz vom Master neu hochgefahren. Das geschieht immer in der gleichen Reihenfolge von Aktionen:

- Alle Slaves werden zurückgesetzt, außer wenn sie als [nicht initialisieren] im Konfigurator markiert sind. Das Zurücksetzen geschieht einzeln mit dem NMT-Kommando "Reset Node" (0x81), jeweils mit der Node-ID des Slaves.
  → Kapitel **Start aller fehlerfrei konfigurierten Slaves** (→ S. 50)
- Alle Slaves werden konfiguriert. Dazu wird zunächst das Objekt 0x1000 des Slaves abgefragt.
  - Wenn der Slave innerhalb der Überwachungszeit von 0,5 Sekunden antwortet:
    > das jeweils nächste Konfigurations-SDO wird gesendet.
  - Ist ein Slave als [optional] konfiguriert und antwortet nicht innerhalb der Überwachungszeit auf die Abfrage des Objekts 0x1000:
    > er wird als 'nicht vorhanden' markiert und
    > keine weiteren SDOs werden an ihn geschickt.
  - Wenn ein Slave auf die Abfrage des Objekts 0x1000 mit einem anderen Typ als dem konfigurierten (in den unteren 16 Bit) antwortet:
    > er wird nicht konfiguriert und
    > er wird als 'falscher Typ' markiert.
- Alle SDOs werden jeweils solange wiederholt, bis innerhalb einer Überwachungszeit eine Antwort des Slaves gesehen wurde. Das Anwendungsprogramm kann den Hochlauf der einzelnen Slaves überwachen.

► Bei Bedarf die Initialisierung eines Slaves abbrechen, falls...
  • Slave ist nicht vorhanden **und**
  • Slave ist nicht als [optional] konfiguriert:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
Vom FB CANx_MASTER_STATUS im Array NODE_STATE_SLAVE das Flag SET_TIMEOUT_STATE = TRUE setzen	Nach Zeitablauf von Timeout bricht der FB das Warten ab.

► Bei Bedarf die Initialisierung eines Slaves überspringen, der auf die Abfrage des Objekts 0x1000 mit einem anderen Typ als dem konfigurierten geantwortet hat:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
Vom FB CANx_MASTER_STATUS im Array NODE_STATE_SLAVE das Flag SET_NODE_STATE = TRUE setzen	mit FB CANOPEN_NMTSERVICES: • NODE = Node-ID des Slaves • NMTSERVICE = 1

> Somit den Slave in den Zustand PRE-OPERATIONAL versetzen.

- Wird der Slave später in den Zustand OPERATIONAL versetzt:
  > vom Master werden keine PDOs an den Slave gesendet
  > die vom Slave gesendeten PDOs werden ignoriert.
- Wenn der Master eine Heartbeat-Zeit ungleich 0 konfiguriert hat:
  > die Erzeugung des Hearbeats beginnt sofort nach dem Starten der Mastersteuerung.
- Nachdem alle Slaves ihre Konfigurations-SDOs erhalten haben:
  > für Slaves mit konfiguriertem Nodeguarding beginnt das Guarding.
- Wenn der Master auf [automatisch starten] konfiguriert wurde:
  > das NMT-Kommando "Start Remote Node" (0x01) wird genutzt
  > alle Slaves werden einzeln vom Master gestartet.
- Wurde das Flag GLOBAL_START gesetzt:
  > das NMT-Kommando wird mit Node-ID 0 genutzt
  > alle Slaves werden mit einem "Start all Nodes" gestartet.
- Es werden mindestens einmal alle konfigurierten TX-PDOs gesendet (für die Slaves sind das RX-PDOs).

► Wenn [automatisch starten] deaktiviert ist, die Slaves einzeln starten:
  • über das Flag START_NODE im NODE_STATE_SLAVE-Array oder
  • → Kapitel **Start aller fehlerfrei konfigurierten Slaves** (→ S. 50).

Netzwerkzustände

Inhalt

19050

Hier lesen Sie, wie Sie die Zustände des CANopen-Netzwerks interpretieren und darauf reagieren können.

Beim Hochlauf des CANopen Netzwerks (→ Kapitel **CANopen-Netzwerk starten** (→ S. 52)) und während des Betriebs durchlaufen die einzelnen Funktionsbausteine der Bibliothek verschiedene Zustände.

HINWEIS

Im Monitorbetrieb (Online-Modus) von CODESYS können Sie die Zustände des CAN-Netzwerkes in der globalen Variablenliste "Can Open implicit variables" einsehen. Dazu sind genaue Kenntnisse von CANopen und der Struktur der CODESYS-CANopen-Bibliotheken notwendig.

Um den Zugriff zu erleichtern, steht Ihnen aus der CANopen-Master-Bibliothek (spezifisch für Gerät und CAN-Kanal) folgender FB zur Verfügung:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
CANx_MASTER_STATUS	CANOPEN_GETSTATE

Hochlauf des CANopen-Masters

19056

Während des Hochlaufs des CAN-Netzwerks durchläuft der Master verschiedene Zustände, die Sie hier ablesen können:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
FB CANx_MASTER_STATUS: > NODE_STATE = aktueller Status des CANopen-Masters	FB CANOPEN_GETSTATE: > MASTERSTATE = interner Zustand des Masters > CANSTATE = Status des CANopen-Netzwerks

Details → Kapitel **NMT-Status für CANopen-Master** (→ S. 77)

Immer, wenn ein Slave auf eine SDO-Anfrage (Upload oder Download) nicht antwortet, dann wird die Anfrage wiederholt. Der Master verlässt den Status 3, wie oben beschrieben, aber erst, wenn alle SDOs erfolgreich übertragen wurden. So kann also erkannt werden, ob ein Slave fehlt oder ob der Master nicht alle SDOs richtig empfangen kann. Dabei ist es für den Master unerheblich, ob ein Slave mit einer Bestätigung oder einem Abort antwortet. Für den Master ist nur von Interesse, ob er überhaupt eine Antwort empfangen hat.

Eine Ausnahme stellt ein als [optional] markierter Slave dar. Optionale Slaves werden nur einmal nach ihrem Objekt 0x1000 gefragt. Wenn sie nicht innerhalb von 0,5 s antworten, wird der Slave vom Master zunächst ignoriert und der Master geht auch ohne weitere Reaktion dieses Slaves in Status 5.

Hochlauf der CANopen-Slaves

19057

Die Status eines Slaves (aus Sicht des Masters) sehen Sie hier:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
FB CANx_SLAVE_STATUS: > NODE_STATE = aktueller Status des CANopen-Slaves	FB CANOPEN_GETNMTSTATESLAVE: > NMTSTATE = Netzwerk-Betriebszustand des Knotens

Details → Kapitel **NMT-Status für CANopen-Slave** (→ S. 78)

Hochlauf des Netzwerks ohne [Automatisch starten]

8583

Manchmal ist es notwendig, dass die Anwendung den Zeitpunkt bestimmt, wann die CANopen-Slaves gestartet werden. Dazu müssen Sie die Option [Automatisch starten] des CANopen-Masters in der Konfiguration deaktivieren. Dann ist das Anwendungsprogramm für das Starten der Slaves zuständig.

Starten des Netzwerks mit GLOBAL_START

19073

In einem CAN-Netz mit vielen Teilnehmern (meist mehr als 8) kommt es häufig dazu, dass schnell aufeinanderfolgende NMT-Nachrichten nicht von allen (meist langsamen) IO-Knoten (z.B. CompactModule CR2013) erkannt werden. Das liegt daran, dass diese Knoten alle Nachrichten mit dem ID 0 mithören müssen. In zu schneller Folge gesendete NMT-Nachrichten überlasten den Empfangspuffer solcher Knoten.

Eine Abhilfe können Sie schaffen, wenn die Anzahl schnell aufeinanderfolgender NMT-Nachrichten reduziert wird:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
mit FB CANx_MASTER_STATUS: ► GLOBAL_START = TRUE	mit FB CANOPEN_SETSTATE: ► GlobalStart = TRUE

> Die CANopen-Master-Bibliothek benutzt den Befehl "Start All Nodes", anstatt alle Knoten einzeln mit dem Kommando "Start Node" zu starten.

> GLOBAL_START wird nur einmalig bei der Netzwerk-Initialisierung ausgeführt.

> Wenn dieser Eingang gesetzt wird, startet die Steuerung auch Knoten mit dem Status 98 (siehe oben). Die PDOs für diese Nodes bleiben jedoch weiterhin deaktiviert.

Starten des Netzwerks mit START_ALL_NODES

19074

Wird das Netzwerk nicht automatisch mit GLOBAL_START gestartet, kann jederzeit jeder Knoten einzeln nacheinander gestartet werden.

Ist das nicht gewünscht, besteht folgende Möglichkeit:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
mit FB CANx_MASTER_STATUS: • GLOBAL_START = FALSE • START_ALL_NODES = FALSE	mit FB CANOPEN_NMTSERVICES: • NODE = Node-ID des Slaves • NMTSERVICE = 2

- START_ALL_NODES wird typisch zur Laufzeit durch das Anwendungsprogramm gesetzt.

> Wenn dieser Eingang gesetzt wird, werden auch Knoten mit dem Status 98 (siehe oben) gestartet. Die PDOs für diese Nodes bleiben jedoch weiterhin deaktiviert.

Initialisieren des Netzwerks mit RESET_ALL_NODES

19075

Aus denselben Gründen, die für den Befehl "Start All Nodes" sprechen, gibt es Fälle, in denen Sie besser das NMT-Kommando "Reset All Nodes" einsetzen (anstelle "Reset Nodes" für jeden einzelnen Knoten).

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
mit FB CANx_MASTER_STATUS: • GLOBAL_START = TRUE • RESET_ALL_NODES = TRUE	mit FB CANOPEN_NMTSERVICES: • NODE = 0 • NMTSERVICE = 3

> Dadurch werden einmalig alle Knoten gleichzeitig zurückgesetzt.

Zugriff auf den Status des CANopen-Masters

19076

Damit der Anwendungs-Code erst abgearbeitet wird, wenn das IO-Netzwerk bereit ist, sollten Sie den Status des Masters abfragen. Das folgende Code-Fragment-Beispiel zeigt eine Möglichkeit:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:

Variablendeklaration
`VAR`
`  FB_MasterStatus := CR0020_MASTER_STATUS;`
`  :`
`END_VAR`
	Variablendeklaration
	`VAR`
	`  FB_MasterStatus : CANOPEN_GETSTATE;`
	`  ...`
	`END_VAR`

Programmcode
`IF  FB_MasterStatus.NODE_STATE = 5 THEN`
`  <application code>`
`END_IF`
	Programmcode
	`IF  FB_MasterStatus.MASTERSTATE = 5 THEN`
	`  <application code>`
	`END_IF`

Durch Setzen des Flags TIME_OUT_STATE im Array NODE_STATE_SLAVE des FB CANx_MASTER_STATUS kann die Anwendung reagieren und zum Beispiel den nicht konfigurierbaren Knoten überspringen.	Durch Setzen eines Wertes für den Eingang NODE des FB CANOPEN_GETSTATE kann die Anwendung reagieren und zum Beispiel den nicht konfigurierbaren Knoten überspringen.

Das Objektverzeichnis des CANopen-Masters

Inhalt

19156

In manchen Fällen ist es hilfreich, wenn der CANopen-Master über ein eigenes Objektverzeichnis verfügt. Das ermöglicht z.B. den Datenaustausch des Anwendungsprogramms mit anderen CAN-Knoten.

Das Objektverzeichnis des Masters wird über eine EDS-Datei mit dem Namen `CRnnnnMasterODEntry.EDS` während der Übersetzungszeit erstellt und mit Werten vorbelegt. Diese EDS-Datei ist im Verzeichnis `CoDeSys Vn\Library\PLCconf` abgelegt. Der Inhalt der EDS-Datei kann über die Schaltfläche [EDS...] im Konfigurations-Fenster [CAN-Parameter] angesehen werden.

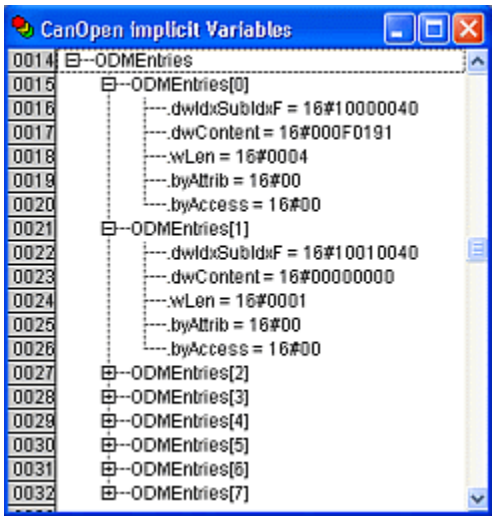
Auch, wenn das Objektverzeichnis nicht vorhanden ist, kann der Master ohne Einschränkungen genutzt werden.

Zugriff auf das Objektverzeichnis (Controller)

19157

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:

Der Zugriff auf das Objektverzeichnis durch das Anwendungsprogramm erfolgt über ein Array, das die folgende Struktur hat:



Strukturelement	Beschreibung
.dwIdxSubIdxF	Die Struktur der Komponente 0xiiiissff ist: iiii - Index (2 Byte, Bits 16...31), Idx ss - Subindex (1 Byte, Bits 8...15), SubIdx ff - Flags (1 Byte, Bits 0...7), F Die Flag-Bits haben folgende Bedeutung: Bit 0 = Schreiben (Write) Bit 1 = Inhalt ist ein Zeiger auf eine Adresse (Content is pointer) Bit 2 = mapbar (mappable) Bit 3 = swap Bit 4 = Vorzeichen behafteter Wert (signed) Bit 5 = Fließkomma (float) Bit 6 = Weitere Subindizes enthalten (has more elements)
.dwContent	Inhalt des Eintrags
.wLen	Länge der Daten
.byAttrib	Ursprünglich als Zugriffsberechtigung gedacht. Kann vom Anwendungsprogramm des Masters beliebig genutzt werden.
.byAccess	Früher Zugriffsberechtigung. Kann vom Anwendungsprogramm des Masters beliebig genutzt werden.

An der Oberfläche verfügt CODESYS über keinen Editor für dieses Objektverzeichnis.

Die EDS-Datei gibt nur vor, mit welchen Objekten das Objektverzeichnis angelegt wird. Dabei werden die Einträge immer mit der Länge 4 erzeugt und die Flags (niederwertigstes Byte der Komponente eines Objektverzeichniseintrags `.dwIdxSubIdxF`) immer mit 1 belegt. D.h. beide Bytes werden mit 0x41 belegt.

Wenn ein Objektverzeichnis im Master vorhanden ist, kann der Master als SDO-Server im Netz auftreten. Immer wenn ein Client auf einen Objektverzeichnis-Eintrag schreibend zugreift, wird das dem Anwendungsprogramm über das Flag OD_CHANGED in CANx_MASTER_STATUS angezeigt. Nach der Auswertung müssen Sie dieses Flag wieder zurücksetzen.

Das Anwendungsprogramm kann das Objektverzeichnis nutzen, indem die Einträge direkt beschrieben oder gelesen werden, oder indem die Einträge auf IEC-Variablen zeigen. D.h.: beim Lesen/Schreiben eines anderen Knotens wird direkt auf diese IEC-Variablen zugegriffen.

Wenn Index und Subindex des Objektverzeichnisses bekannt sind, kann ein Eintrag wie folgt angesprochen werden:

```
I := GetODMEntryValue(16#iiiiss00, pCanOpenMaster[0].wODMFirstIdx,
pCanOpenMaster[0].wODMFirstIdx + pCanOpenMaster[0].wODMCount;
```

Wobei für "iiii" der Index und für "ss" der Subindex (als Hex-Werte) eingesetzt werden müssen.

Damit steht die Nummer des Array-Eintrags in I zur Verfügung. Nun können Sie direkt auf die Komponenten des Eintrags zugreifen.
Damit Sie diesen Eintrag direkt auf einer IEC-Variable ausgeben können, genügt es, Adresse, Länge und Flags einzutragen:

```
ODMEntries[I].dwContent := ADR(<Variablenname>);
ODMEntries[I].wLen := sizeof(<Variablenname>);
ODMEntries[I].dwIdxSubIdxF := ODMEntries[I].dwIdxSubIdxF OR OD_ENTRYFLG_WRITE OR
OD_ENTRYFLG_ISPOINTER;
```

Um nur den Inhalt des Eintrags zu ändern, genügt es, den Inhalt von "`.dwContent`" zu ändern.

Zugriff auf das Objektverzeichnis (andere)

19158

Für alle CR04nn, CR1nnn, CR253n gilt:

Der Zugriff auf das Objektverzeichnis durch das Anwendungsprogramm erfolgt über Funktionsbausteine:
- CANOPEN_GETODCHANGEDFLAG,
- CANOPEN_READOBJECTDICT,
- CANOPEN_WRITEOBJECTDICT.

An der Oberfläche verfügt CODESYS über keinen Editor für dieses Objektverzeichnis.

Die EDS-Datei gibt nur vor, mit welchen Objekten das Objektverzeichnis angelegt wird.

Wenn ein Objektverzeichnis im Master vorhanden ist, kann der Master als SDO-Server im Netz auftreten. Immer wenn ein Client auf einen Objektverzeichnis-Eintrag schreibend zugreift, wird das dem Anwendungsprogramm über den Ausgang ODCHANGED des FB CANOPEN_GETODCHANGEDFLAG angezeigt. Nach der Auswertung müssen Sie dieses Flag durch Setzen des Eingangs RESETFLAG wieder zurücksetzen.

Das Anwendungsprogramm kann das Objektverzeichnis nutzen, indem die Einträge direkt beschrieben oder gelesen werden.

3.4.4 CANopen-Slave

Inhalt

1865

Eine mit CODESYS programmierbare Steuerung kann in einem CAN-Netzwerk auch als CANopen-Slave erscheinen.

Funktionalität der CANopen-Slave-Bibliothek

19161

Die CANopen-Slave-Bibliothek zusammen mit dem CANopen-Konfigurator stellt dem Anwender folgende Möglichkeiten zur Verfügung:

- In CODESYS: Konfiguration der Eigenschaften NodeGuarding/Heartbeat, Emergency, Node-ID und Baudrate, auf der das Device arbeiten soll.
- Zusammen mit dem Parametermanager in CODESYS kann ein Default-PDO-Mapping erstellt werden, das zur Laufzeit vom Master geändert werden kann. Die Konfiguration des PDO-Mappings erfolgt während der Konfigurationsphase durch den Master. Durch das Mapping können IEC-Variablen des Anwendungsprogramms in PDOs gemappt werden. D.h. den PDOs werden IEC-Variable zugeordnet, um sie im Anwendungsprogramm einfach auswerten zu können.
- Die CANopen-Slave-Bibliothek stellt ein Objektverzeichnis zur Verfügung. Die Größe dieses Objektverzeichnisses wird zur Übersetzungszeit von CODESYS festgelegt. In diesem Verzeichnis befinden sich alle Objekte, die den CANopen-Slave beschreiben und zusätzlich die, die vom Parametermanager definiert sind. Im Parametermanager können zusammen mit dem CANopen-Slave nur die Listenarten Parameter und Variablen verwendet werden.
- Der CANopen-Slave verwaltet die Zugriffe auf das Objektverzeichnis, tritt also am Bus als SDO-Server auf.
- Der CANopen-Slave überwacht das Nodeguarding und die Heartbeat-Consumer-Zeit (immer nur von einem Producer) und setzt entsprechende Fehlerflags für das Anwendungsprogramm.
- Es kann eine EDS-Datei erzeugt werden, die die konfigurierten Eigenschaften des CANopen-Slaves so beschreibt, dass das Device als Slave unter einem CANopen-Master eingebunden und konfiguriert werden kann.

Der CANopen-Slave stellt ausdrücklich folgende, in CANopen beschriebene, Funktionalitäten nicht zur Verfügung (alle hier und im obigen Abschnitt nicht genannten Möglichkeiten des CANopen-Protokolls sind ebenfalls nicht implementiert):

- Dynamische SDO- und PDO-Identifier
- SDO Block-Transfer
- Automatische Erzeugung von Emergency-Nachrichten. Emergency-Nachrichten müssen immer vom Anwendungsprogramm erzeugt werden. Die CANopen-Slave-Bibliothek stellt Ihnen dazu diese FBs zur Verfügung:
  - **Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:**

CANx_SLAVE_EMCY_HANDLER	verwaltet den geräteeigenen Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x: • Error Register (Index 0x1001) und • Error Field (Index 0x1003) des CANopen Objektverzeichnis x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SLAVE_SEND_EMERGENCY	versendet anwendungsspezifische Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

- **Für alle CR04nn, CR1nnn, CR253n gilt:**

CANOPEN_GETERRORREGISTER	= Get CANopen Error-Register liest die Fehler-Register 0x1001 und 0x1003 der Steuerung aus Die Register können durch Setzen der entsprechenden Eingänge zurückgesetzt werden.
CANOPEN_GETEMCYMESSAGES	= Get CANopen Emergency Messages listet alle Emergency-Nachrichten auf, die die Steuerung seit dem letzten Löschen der Nachrichten von anderen Knoten am Netz empfangen hat Die Liste kann durch Setzen des entsprechenden Eingangs zurückgesetzt werden.
CANOPEN_SENDEMCYMESSAGE	= CANopen Send Emergency-Message versendet eine EMCY-Nachricht. Die Nachricht wird aus den entsprechenden Parametern zusammengebaut und ins Register 0x1003 eingetragen

- Dynamische Änderungen der PDO-Eigenschaften werden z.Z. immer nur beim Eintreffen einer StartNode NMT-Nachricht übernommen, nicht mit den in CANopen definierten Mechanismen.

CANopen-Slave konfigurieren

Inhalt

19163
1980

Wunsch: Das Gerät als CANopen-Slave nutzen:

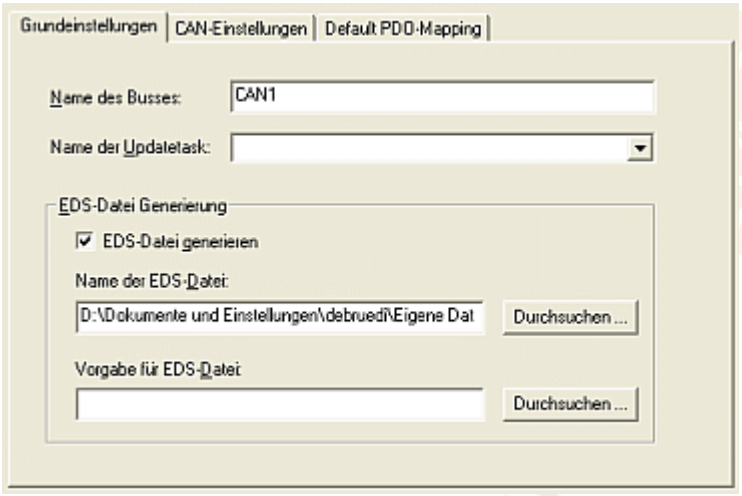
► In der Steuerungskonfiguration den CANopen-Slave einfügen:
Rechtsklick auf die erste Zeile ("CRnnnn Configuration Vnn")
[Unterelement anhängen] > [CANopen Slave]

> Wenn für ein Gerät mehrere CANopen-fähige Schnittstellen verfügbar sind, dann gilt (abhängig vom Gerät) für die Zuordnung des CANopen-Protokolls zur CAN-Schnittstelle:
→ Gerätehandbuch Kapitel CAN-Schnittstellen und CAN-Protokolle

> Alle erforderlichen Bibliotheken werden automatisch in den Bibliotheksverwalter eingefügt.

Register [Grundeinstellungen]

1981



Grundeinstellungen: Name des Busses

10049

Parameter wird im Moment nicht benutzt.

Grundeinstellungen: Name der Updatetask

10050

Name der Task, in der der Aufruf des CANopen-Slave erfolgt.

Grundeinstellungen: EDS-Datei generieren

10051

Wunsch: aus den Einstellungen eine EDS-Datei erzeugen, um den CANopen-Slave in eine beliebige Masterkonfiguration einfügen zu können. Dazu:

► Option [EDS-Datei generieren] aktivieren UND
[Name der EDS-Datei] mit Pfad angeben

► Optional: eine Vorlagendatei angeben, deren Einträge zur EDS-Datei des CANopen-Slave hinzugefügt werden. Bei Überschneidungen werden Vorgaben der Vorlage nicht überschrieben.

Beispiel für ein Objektverzeichnis

1991

Folgende Einträge könnten zum Beispiel im Objektverzeichnis stehen:

```
[FileInfo]
FileName=D:\CoDeSys\lib2\plcconf\MyTest.eds
FileVersion=1
FileRevision=1
Description=EDS for CoDeSys-Project: D:\CoDeSys\CANopenTestprojekte\TestHeartbeatODsettings_Device.pro
CreationTime=13:59
CreationDate=09-07-2005
CreatedBy=CoDeSys
ModificationTime=13:59
ModificationDate=09-07-2005
ModifiedBy=CoDeSys

[DeviceInfo]
VendorName=3S Smart Software Solutions GmbH
ProductName=TestHeartbeatODsettings_Device
ProductNumber=0x33535F44
ProductVersion=1
ProductRevision=1
OrderCode=xxxx.yyyy.zzzz
LMT_ManufacturerName=3S GmbH
LMT_ProductName=3S_Dev
BaudRate_10=1
BaudRate_20=1
BaudRate_50=1
BaudRate_100=1
BaudRate_125=1
BaudRate_250=1
BaudRate_500=1
BaudRate_800=1
BaudRate_1000=1
SimpleBootUpMaster=1
SimpleBootUpSlave=0
ExtendedBootUpMaster=1
ExtendedBootUpSlave=0

...

[1018sub0]
ParameterName=Number of entries
ObjectType=0x7
DataType=0x5
AccessType=ro
DefaultValue=2
PDOMapping=0

[1018sub1]
ParameterName=VendorID
ObjectType=0x7
DataType=0x7
AccessType=ro
DefaultValue=0x0
PDOMapping=0

[1018sub2]
ParameterName=Product Code
ObjectType=0x7
DataType=0x7
AccessType=ro
DefaultValue=0x0
PDOMapping=0
```

Bedeutung der einzelnen Objekte entnehmen Sie bitte der CANopen-Spezifikation DS301.

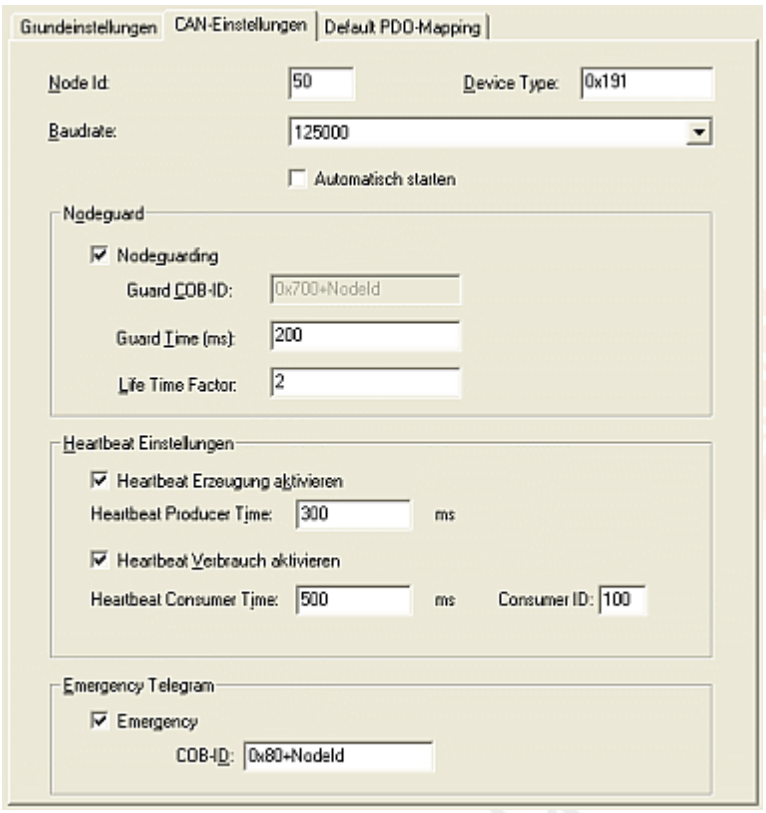
Die EDS-Datei enthält, neben den vorgeschriebenen Einträgen, die Definitionen für SYNC, Guarding, Emergency und Heartbeat. Wenn diese Objekte nicht benutzt werden, sind die Werte auf 0 gesetzt (voreingestellt). Da die Objekte aber im Objektverzeichnis des Slaves zur Laufzeit vorhanden sind, werden sie in der EDS-Datei auch beschrieben.

Das Gleiche gilt für die Einträge für die Kommunikations- und Mapping-Parameter. Es sind immer alle 8 möglichen Subindizes der Mapping-Objekte 0x16nn oder 0x1Ann vorhanden, aber u.U. im Subindex 0 nicht berücksichtigt.

Bit-Mapping wird von der Bibliothek nicht unterstützt!

Register [CAN-Einstellungen]

1982



Hier können Sie den **Node-ID** und die **Baudrate** einstellen.

Device Type
(das ist der Default-Wert des Objekts 0x1000, der im EDS eingetragen wird) wird mit 0x191 (Standard-IO-Device) vorbelegt und kann von Ihnen beliebig geändert werden.
Der Index des CAN-Controllers ergibt sich aus der Position des CANopen-Slave in der Steuerungskonfiguration.

Die **Nodeguarding**-Parameter, die **Heartbeat**-Parameter und den Emergency-COB-ID können Sie ebenfalls auf diesem Register festlegen. Der CANopen-Slave kann nur für die Überwachung eines Heartbeats konfiguriert werden.
Wir empfehlen: Für aktuelle Geräte besser mit Heartbeat arbeiten, weil dann die Buslast niedriger ist.

HINWEIS

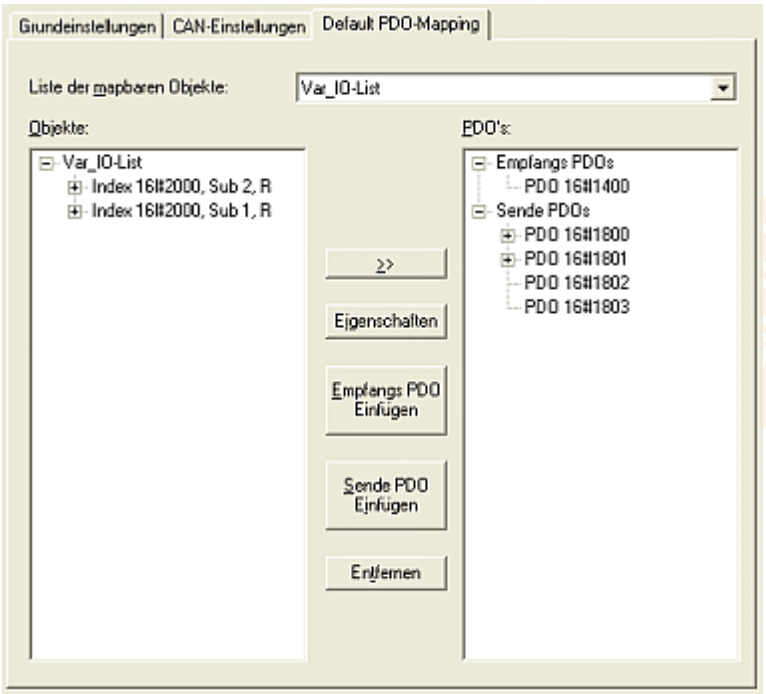
Beim Verwenden von Guarding oder Heartbeat UND
wenn Sie ein EDS-File erzeugen, das bei einem CANopen-Master eingebunden werden soll:

► Guard Time = 0 eintragen
Life Time Factor = 0 eintragen
Heartbeat Time = 0 eintragen

> Die beim CANopen-Master eingestellten Werte werden während der Konfiguration zum CANopen-Slave gesendet. Dadurch hat der CANopen-Master das Guarding oder den Heartbeat für diesen Knoten sicher aktiviert.

Register [Default PDO-Mapping]

1983



In diesem Register können Sie die Zuordnung zwischen lokalem Objektverzeichnis (OD-Editor) und den PDOs festlegen, die vom CANopen-Slave gesendet/empfangen werden. Eine solche Zuordnung wird als "Mapping" bezeichnet.

In den verwendeten Objektverzeichniseinträgen (Variablen OD) wird zwischen Objektindex/Subindex die Verbindung zu Variablen des Anwendungsprogramms hergestellt. Dabei müssen Sie nur darauf achten, dass der Subindex 0 eines Indexes, der mehr als einen Subindex enthält, die Information über die Anzahl der Subindizes enthält.

Beispiel: Variablenliste

10052

Auf dem ersten Empfangs-PDO (COB-ID = 512 + Node-ID) des CANopen-Slaves sollen die Daten für die Variable PLC_PRG.a empfangen werden.

Info

Als Listentyp kann [Variablen] oder [Parameter] gewählt werden.

Zum Datenaustausch (z.B. über PDOs oder sonstige Einträge im Objektverzeichnis) wird eine Variablenliste angelegt.

Die Parameterliste sollten Sie einsetzen, wenn Sie Objektverzeichniseinträge nicht mit Anwendungs-Variablen verknüpfen wollen. Für die Parameterliste ist zurzeit nur der Index 0x1006 / SubIdx 0 vordefiniert. In diesen Eintrag kann vom Master der Wert für die [Com. Cycle Period] eingetragen werden. Damit wird das Ausbleiben der SYNC-Nachricht gemeldet.

Also müssen Sie im Objektverzeichnis (Parametermanager) eine Variablenliste anlegen und einen Index/SubIndex mit der Variablen PLC_PRG.a verknüpfen:

- ► Dazu fügen Sie in der Variablenliste eine Zeile hinzu (rechte Maustaste öffnet das Kontextmenü) und tragen einen Variablen-Namen (beliebig) sowie den Index und den Subindex ein.
- ► Als Zugriffsrichtung ist für ein Empfangs-PDO nur [write only] (schreiben) zugelassen.
- ► In die Spalte [Variable] tragen Sie dann "`PLC_PRG.a`" ein, oder drücken [F2] und wählen die Variable aus.

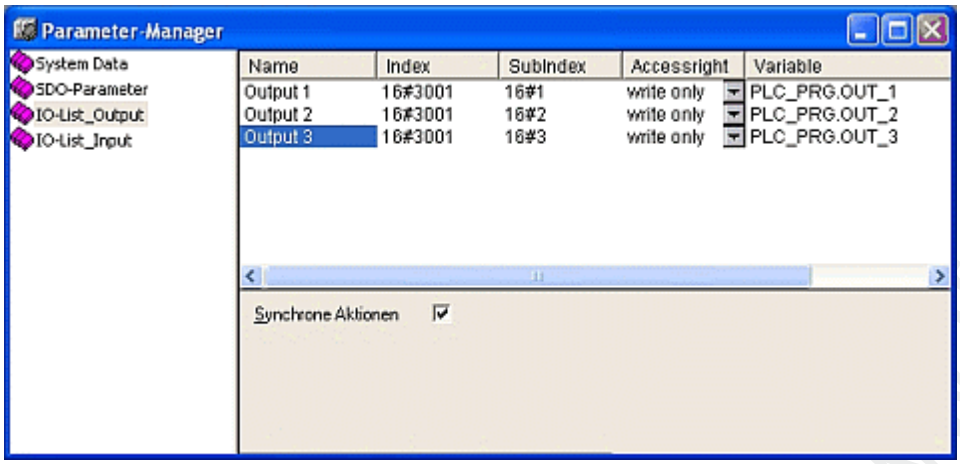
HINWEIS

Daten, die vom CANopen-Master gelesen werden sollen (z.B. Eingänge, Systemvariablem), müssen die Zugriffsrichtung (Accessright) [read only] (lesen) haben.

Daten, die vom CANopen-Master geschrieben werden sollen (z.B. Ausgänge im Slave), müssen die Zugriffsrichtung (Accessright) [write only] (schreiben) haben.

SDO-Parameter, die vom CANopen-Master geschrieben und gleichzeitig aus der Slave-Anwendung gelesen und geschrieben werden sollen, müssen die Zugriffsrichtung (Accessright) [read-write] (lesen+schreiben) haben.

Damit Sie den Parametermanager öffnen können, muss in den Zielsystemeinstellungen unter [Netzfunktionen] der Parametermanager aktiviert sein. Die Bereiche für Index/Subindex sind bereits mit sinnvollen Werten vorbelegt und sollten nicht geändert werden.



Im Default PDO-Mapping des CANopen-Slaves wird anschließend der Index-/Subindex-Eintrag als Mapping-Eintrag einem Empfangs-PDO zugewiesen. Die Eigenschaften des PDOs lassen sich über den Dialog festlegen, der aus Kapitel **CANopen-Slaves einfügen und konfigurieren** (→ S. 42) bekannt ist.

Nur Objekte aus dem Parametermanager, die mit dem Attribut [read only] (lesen) oder [write only] (schreiben) versehen sind, werden in der evtl. erzeugten EDS-Datei als mapbar (= zuordnungsfähig) markiert und tauchen in der Liste der mapbaren Objekte auf. Alle anderen Objekte werden in der EDS-Datei als nicht mapbar markiert.

Verändern des Standard-Mappings durch Master-Konfiguration

1984

Sie können das vorgegebene PDO-Mapping (in der CANopen-Slave-Konfiguration) in bestimmten Grenzen durch den Master verändern.

Dabei gilt:
Der CANopen-Slave kann nur Objektverzeichniseinträge neu anlegen, die bereits im Standard-Mapping (Default PDO-Mapping in der CANopen-Slave-Konfiguration) vorhanden sind.
Also kann z.B. für ein PDO, das im Default PDO-Mapping ein gemapptes Objekt enthält, in der Masterkonfiguration kein zweites Objekt gemappt werden.

Das durch die Masterkonfiguration veränderte Mapping kann also höchstens die im Standard-Mapping vorhandenen PDOs enthalten. Innerhalb dieser PDOs sind 8 Mapping-Einträge (Subindizes) vorhanden.

Eventuelle Fehler, die hierbei auftreten können, werden Ihnen nicht angezeigt, d.h. die überzähligen PDO-Definitionen / die überzähligen Mapping-Einträge werden so behandelt, als seien sie nicht vorhanden.

Die PDOs müssen im Master immer wie folgt angelegt sein:
- von 0x1400 beginnend (Empfangs-PDO-Kommunikationsparameter) oder
- von 0x1800 beginnend (Sende-PDO-Kommunikationsparameter)
- und lückenlos aufeinander folgend.

Zugriff auf den CANopen-Slave zur Laufzeit

1985

Einstellen der Knotennummer eines CANopen-Slaves

19165

Beim CANopen-Slave kann zur Laufzeit des Anwendungsprogramms die Knotennummer eingestellt werden:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
den FB CANx_SLAVE_NODEID aus der CANopen-Slave-Bibliothek nutzen	den FB CANOPEN_SETSTATE aus der Bibliothek `ifm_CANopen_NT_Vxxyyzz.LIB` nutzen

Einstellen der Baudrate eines CANopen-Slaves

19166

Beim CANopen-Slave kann zur Laufzeit des Anwendungsprogramms die Baudrate eingestellt werden:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
einen der folgenden FBs aus der Gerätebibliothek `ifm_CRnnnn_Vxxyyzz.LIB` nutzen: • CAN1_BAUDRATE oder • CAN1_EXT oder • CANx.	den FB CANOPEN_ENABLE aus der Bibliothek `ifm_CANopen_NT_Vxxyyzz.LIB` nutzen

Zugriff auf die OD-Einträge vom Anwendungsprogramm

19167

Standardmäßig gibt es Objektverzeichniseinträge, die auf Variablen gemappt sind (Parametermanager).

Es gibt jedoch auch die automatisch erzeugten Einträge des CANopen-Slave, auf die Sie nicht über den Parametermanager in einen Variableninhalt mappen können. Diese Einträge stehen hier zur Verfügung:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
den FB CANx_SLAVE_STATUS aus der CANopen-Slave-Bibliothek nutzen	• CANOPEN_READOBJECTDICT • CANOPEN_WRITEOBJECTDICT aus der Bibliothek `ifm_CANopen_NT_Vxxyyzz.LIB`

Ändern der PDO-Eigenschaften zur Laufzeit

1988

Sollen die Eigenschaften eines PDOs zur Laufzeit verändert werden, so funktioniert das durch einen anderen Knoten über SDO-Schreibzugriffe, wie dies von CANopen beschrieben wird.

Alternativ kann man auch direkt eine neue Eigenschaft, wie z.B. die "Event time" eines Sende-PDOs schreiben und anschließend einen Befehl "StartNode-NMT" an den Knoten schicken, obwohl er bereits gestartet ist. Das führt dazu, dass das Device die Werte im Objektverzeichnis neu interpretiert.

Emergency-Messages durch das Anwendungsprogramm senden

19168

Eine Emergency-Message vom Anwendungsprogramm versenden:

- **Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:**

CANx_SLAVE_EMCY_HANDLER	verwaltet den geräteeigenen Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x: • Error Register (Index 0x1001) und • Error Field (Index 0x1003) des CANopen Objektverzeichnis x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SLAVE_SEND_EMERGENCY	versendet anwendungsspezifische Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

- **Für alle CR04nn, CR1nnn, CR253n gilt:**

CANOPEN_GETERRORREGISTER	= Get CANopen Error-Register liest die Fehler-Register 0x1001 und 0x1003 der Steuerung aus Die Register können durch Setzen der entsprechenden Eingänge zurückgesetzt werden.
CANOPEN_GETEMCYMESSAGES	= Get CANopen Emergency Messages listet alle Emergency-Nachrichten auf, die die Steuerung seit dem letzten Löschen der Nachrichten von anderen Knoten am Netz empfangen hat Die Liste kann durch Setzen des entsprechenden Eingangs zurückgesetzt werden.
CANOPEN_SENDEMCYMESSAGE	= CANopen Send Emergency-Message versendet eine EMCY-Nachricht. Die Nachricht wird aus den entsprechenden Parametern zusammengebaut und ins Register 0x1003 eingetragen

3.4.5 CANopen-Tabellen

Inhalt

9941

Die folgenden Tabellen informieren Sie über wichtige Werte und Einstellungen der CANopen-Schnittstellen.

Aufbau von CANopen-Meldungen

Inhalt

9971

Eine CANopen-Meldung besteht aus der COB-ID und bis zu 8 Bytes Daten:

COB-ID			DLC	Byte 1		Byte 2		Byte 3		Byte 4		Byte 5		Byte 6		Byte 7		Byte 8	
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Details erfahren Sie in den folgenden Kapiteln.

Beachten Sie die umgekehrte Byte-Reihenfolge! (⇨ Little Endian oder Intel-Format)

Beispiele:

Wert [hex]	Datentyp	Byte 1		Byte 2		Byte 3		Byte 4		Byte 5		Byte 6		Byte 7		Byte 8	
12	BYTE	1	2	–	–	–	–	–	–	–	–	–	–	–	–	–	–
1234	WORD	3	4	1	2	–	–	–	–	–	–	–	–	–	–	–	–
12345678	DWORD	7	8	5	6	3	4	1	2	–	–	–	–	–	–	–	–

Aufbau der COB-ID

9972

Der erste Teil einer Meldung ist die COB-ID. Aufbau der 11-Bit COB-ID:

Nibble 0				Nibble 1				Nibble 2			
11	10	9	8	7	6	5	4	3	2	1	0
--	**3**	**2**	**1**	**0**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
--	**Funktions-Code**				**Node-ID**						

Die COB-ID besteht aus **Funktions-Code / Predefined Connectionset** (→ S. 72) und Node-ID.

Beispiel:
Das Kommunikations-Objekt = TPDO1 (TX)
Die Knoten-Nummer des Geräts = 0x020 = 32

Berechnung:
Der Funktions-Code für das Kommunikations-Objekt TPDO1 = 0x03
Die Wertigkeit des Funktions-Code in der 11-Bit-COB-ID = 0x03 • 0x80 = 0x180
Dazu die Knoten-Nummer (0x020) addieren ⇨ die COB-ID lautet: 0x1A0

1				A				0			
3	2	1	0	3	2	1	0	3	2	1	0
0	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
--	0x03 = 3				0x020 = 32						

Funktions-Code / Predefined Connectionset

9966

Im "CANopen Predefined Connectionset" sind einige Funktions-Codes vorbelegt.

Wenn Sie das Predefined Connectionset verwenden, können Sie ein CANopen-Netzwerk von bis zu 127 Teilnehmern in Betrieb nehmen, ohne dass es zu einer doppelten Vergabe von COB-IDs kommt.

Broadcast- oder Multicast-Nachrichten:

Kommunikations-Objekt	Funktions-Code [hex]	COB-ID [hex]	zugehörige Parameter-Objekte [hex]
NMT	0	000	
SYNC	1	080	1005, 1006, 1007, 1028
TIME	2	100	1012, 1013

Punkt-zu-Punkt-Nachrichten:

Kommunikations-Objekt	Funktions-Code [hex]	COB-ID [hex]	zugehörige Parameter-Objekte [hex]
EMERGENCY	1	080 + Node-ID	1014, 1015
TPDO1 (TX)	3	180 + Node-ID	1800
RPDO1 (RX)	4	200 + Node-ID	1400
TPDO2 (TX)	5	280 + Node-ID	1801
RPDO2 (RX)	6	300 + Node-ID	1401
TPDO3 (TX)	7	380 + Node-ID	1802
RPDO3 (RX)	8	400 + Node-ID	1402
TPDO4 (TX)	9	480 + Node-ID	1803
RPDO4 (RX)	A	500 + Node-ID	1403
Default SSDO (TX)	B	580 + Node-ID	1200
Default CSDO (RX)	C	600 + Node-ID	1280
NMT Error Control	E	700 + Node-ID	1016, 1017

TX = Slave sendet an Master
RX = Slave empfängt von Master

SSDO = Server-SDO
CSDO = Client-SDO

SDO-Kommando-Bytes

9968

Aufbau einer SDO-Nachricht:

COB-ID	DLC	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
XXX	8	Kommando	Index		Sub-Index	Daten abhängig von den zu transportierenden Daten			

Beachten Sie die umgekehrte Byte-Reihenfolge! (⇨ Little Endian oder Intel-Format)

Eine SDO-COB-ID setzt sich wie folgt zusammen:

CANopen	
Node-ID	**COB-ID SDO**
1…127	TX: 0x580 + Node-ID
	RX: 0x600 + Node-ID

TX = Slave sendet an Master
RX = Slave empfängt von Master

DLC = **D**ata **L**ength **C**ode = bei CANopen die Anzahl der Daten-Bytes in einer Nachricht.
Für →SDO: DLC = 8

SDO-Kommando-Bytes:

Kommando hex	dez	Nachricht	Datenlänge	Beschreibung
21	33	Anforderung	mehr als 4 Bytes	Daten an Slave senden
22	34	Anforderung	1…4 Bytes	Daten an Slave senden
23	35	Anforderung	4 Bytes	Daten an Slave senden
27	39	Anforderung	3 Bytes	Daten an Slave senden
2B	43	Anforderung	2 Bytes	Daten an Slave senden
2F	47	Anforderung	1 Byte	Daten an Slave senden
40	64	Anforderung	---	Daten von Slave anfordern
42	66	Antwort	1…4 Bytes	Daten von Slave an Master senden
43	67	Antwort	4 Bytes	Daten von Slave an Master senden
47	71	Antwort	3 Bytes	Daten von Slave an Master senden
4B	75	Antwort	2 Bytes	Daten von Slave an Master senden
4F	79	Antwort	1 Byte	Daten von Slave an Master senden
60	96	Antwort	---	Datentransfer in Ordnung: Empfangsbestätigung von Slave an Master senden
80	128	Antwort	4 Bytes	Datentransfer fehlgeschlagen: Abbruch-Nachricht von Slave an Master senden → Kapitel **SDO-Abbruch-Code** (→ S. 74)

SDO-Abbruch-Code

9970

Der SDO-Abbruch-Code gehört NICHT zum Emergency-Telegramm!

Abbruch-Code [hex]	Beschreibung
0503 0000	toggle bit not alternated
0504 0000	SDO protocol timed out
0504 0001	client/server command specifier not valid or unknown
0504 0002	invalid block size (block mode only)
0504 0003	invalid sequence number (block mode only)
0504 0004	CRC error (block mode only)
0504 0005	out of memory
0601 0000	unsupported access to an object
0601 0001	attempt to read a write only object
0601 0002	attempt to write a read only object
0602 0000	object does not exist in the object dictionary
0604 0041	object cannot be mapped to the PDO
0604 0042	the number and length of the objects to be mapped would exceed PDO length
0604 0043	general parameter incompatibility reason
0604 0047	general internal incompatibility in the device
0606 0000	access failed due to an hardware error
0607 0010	data type does not match, length of service parameter does not match
0607 0012	data type does not match, length of service parameter too high
0607 0013	data type does not match, length of service parameter too low
0609 0011	sub-index does not exist
0609 0030	value range of parameter exceeded (only for write access)
0609 0031	value of parameter written too high
0609 0032	value of parameter written too low
0609 0036	maximum value is less than minimum value
0800 0000	general error
0800 0020	data cannot be transferred or stored to the application
0800 0021	data cannot be transferred or stored to the application because of local control
0800 0022	data cannot be transferred or stored to the application because of the present device state
0800 0023	object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error)

Bootup-Nachricht

9961

Der CAN-Teilnehmer sendet nach dem Booten einmalig die Bootup-Nachricht:

	COB-ID	DLC	Byte 1
hex	0x700 + Node-ID	0x1	0x00
dez	1 792 + Node-ID	1	0

Somit ist der Teilnehmer im CAN-Netzwerk lauffähig.

DLC = **D**ata **L**ength **C**ode = bei CANopen die Anzahl der Daten-Bytes in einer Nachricht.
Für →SDO: DLC = 8

Beispiel:
Die Node-ID des Teilnehmers ist 0x7D = 125.
Dann lautet die COB-ID der Bootup-Nachricht: 0x77D = 1 917

Abweichung:
Es gibt Geräte, die kein [0x700 + Node-ID] senden können (das sind Geräte, die vor der Version 4 der CANopen-Spezifikation entstanden sind).
Diese Geräte senden stattdessen folgende Bootup-Nachricht und ohne Status:

	COB-ID	DLC
hex	0x080 + Node-ID	0x0
dez	128 + Node-ID	0

Netzwerk-Management (NMT)

Inhalt

9974

Netzwerk-Management-Kommandos

9962

Mit folgenden Netzwerk-Management-Kommandos kann der Anwender den Betriebsmodus von einzelnen oder allen CAN-Teilnehmern beeinflussen. Muster:

COB-ID	DLC	Byte 1	Byte 2
0x000	X	Kommando	Node-ID

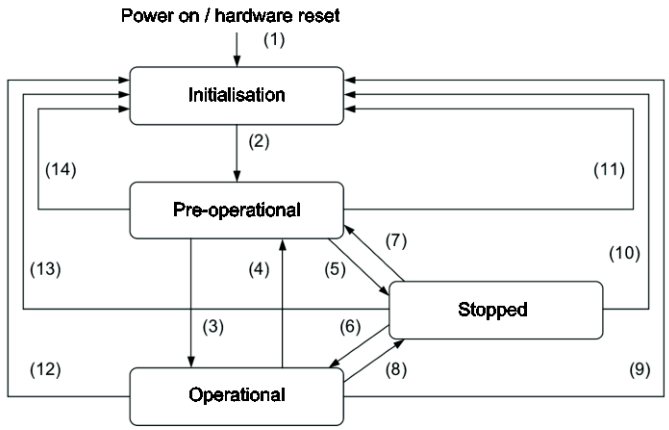
Node-ID = 00 ⇨ Kommando gilt zeitgleich für alle Knoten im Netz

COB-ID	NMT-Kommando		Beschreibung	
0x000	0x01 = 01	Node-ID	start_remode_node	Knoten in den Zustand Operational versetzen
0x000	0x02 = 02	Node-ID	stop_remode_node	Knoten in den Zustand STOPPED versetzen
0x000	0x80 = 128	Node-ID	enter_pre-operational	Knoten in den Zustand PRE-OPERATIONAL versetzen
0x000	0x81 = 129	Node-ID	reset_node	Knoten zurücksetzen
0x000	0x82 = 130	Node-ID	reset_communication	CAN-Kommunikation des Knotens zurücksetzen

NMT-Status

9963

Das Status-Byte gibt Auskunft über den Zustand des CAN-Teilnehmers.



Grafik: Zustandsübergänge unter CANopen

Erlaubte Übergänge:

(1) Zustand wird bei Power On automatisch erreicht.

(2) interne Initialisierung ist beendet – Knoten geht automatisch nach PRE-OPERATIONAL

(3) NMT Service "Start Remote Node"

(4) + (7) NMT Service "Enter PRE-OPERATIONAL"

(5) + (8) NMT Service "Stop Remote Node"

(6) NMT Service "Start Remote Node"

(9)...(11) NMT Service "Reset Node"

(12)...(14) NMT Service "Reset Communication"

NMT-Status für CANopen-Master

9964

Diese Status zeigen den internen Zustand des CANopen-Master-Stack.
Sie sind nicht durch die CANopen-Spezifikation vorgegeben.

Status hex	dez	Beschreibung
00	0	nicht definiert
01	1	Master wartet auf die Bootup-Nachricht des Slaves. ODER: Master wartet auf Ablauf der GuardTime.
02	2	• Master wartet 300 ms. • Master fordert das Objekt 0x1000 an. • Danach wechselt der Master auf Status 3.
03	3	Der Master konfiguriert seine Slaves. Dazu sendet der Master an die Slaves der Reihe nach alle vom Konfigurator erzeugten SDOs. Danach wechselt der Master auf Status 5.
05	5	Nachdem an alle Slaves die SDOs übertragen wurden, geht der Master in den Status 5 und bleibt in diesem Status. Status 5 ist für den Master der normale Betriebszustand.

Knoten-Status aus FB lesen:

verwendeter Funktionsbaustein	hier steht dieser Knoten-Status
CANx_MASTER_STATUS	Ausgang NODE_STATE
CANOPEN_GETSTATE	Ausgang MASTERSTATE

NMT-Status für CANopen-Slave

9965

Diese Status zeigen den internen Zustand des CANopen-Master-Stack im Bezug auf die Initialisierung eines CANopen-Slave.
Sie sind nicht durch die CANopen-Spezifikation vorgegeben.

Die Struktur CANx_NODE_STATE liegt in einem Array, dessen Adresse dem FB CANx_MASTER_STATUS über den Eingang NOTE_STATE_SLAVES übergeben werden muss. Die folgenden Werte kann in der Struktur CANx_NODE_STATE der Ausgang NODE_STATE annehmen:

Status hex	dez	Beschreibung
FF	-1	Initialer Status Der CANopen-Slave wird durch das NMT-Kommando [Reset_Node] zurückgesetzt. Anschließend Wechsel in den Status 1. Ist in der CODESYS-Steuerungskonfiguration beim CANopen-Slave die Option [nicht initialisieren] aktiviert, wird der Status -1 übersprungen und der Status 1 ist der initiale Status.
00	0	nicht definiert
01	1	Warten auf die Bootup-Nachricht vom Slave. Nach dem Empfang der Bootup-Nachricht ODER spätestens nach 2 s Wartezeit Wechsel in den Status 2.
02	2	Auslesen des Objekts 0x1000 aus dem Objektverzeichnis des CANopen-Slaves per SDO-Zugriff. Nach einer Antwort vom CANopen-Slave ODER einer Wartezeit von 500 ms erfolgt Wechsel in den Status 3. Ist der CANopen-Slave in der CODESYS-Steuerungskonfiguration als "optional" konfiguriert, erfolgt nach Ablauf der Wartezeit ein Wechsel in den Status 97. Entspricht der aus dem Objekt 0x1000 ausgelesene Gerätetyp nicht der Angabe der in der CODESYS-Steuerungskonfiguration eingebundenen EDS-Datei, erfolgt zwar ein Wechsel in den Status 3, aber am Ende von Status 3 ein Wechsel in den Status 98.
03	3	Der CANopen-Slave wird vom Master per SDO-Zugriff konfiguriert. Ist in der CODESYS-Steuerungskonfiguration beim CANopen-Slave die Option [Knoten zurücksetzen] aktiviert, wird während der ersten Konfiguration die Zeichenkette "load" an das Objekt 0x1011/01 gesendet und anschließend der CANopen-Slave mit dem NMT-Kommando [Reset_Node] neu gestartet. Anschließend Wechsel in den Status 1, der [load]-Befehl mit anschließendem Reset wird im weiteren Verlauf im Status 3 nicht mehr ausgeführt. CANopen-Slaves, bei denen während der Konfigurationsphase ein Problem auftritt, bleiben entweder im Status 3 oder wechseln in einen Fehlerstatus (Status > 5). Über das Strukturelement SET_TIMEOUT_STATE der Struktur CANx_NODE_STATE ist es möglich, einen nicht vorhandenen CANopen-Slave, der in der CODESYS-Steuerungskonfiguration nicht als "optional" konfiguriert wurde, in den Status 4 wechseln zu lassen. Ansonsten würde der fehlende CANopen-Slave die Initialisierung des CANopen-Netzwerks blockieren.
04	4	CANopen-Slave ist konfiguriert und im CANopen-Status PRE-OPERATIONAL. Befinden sich alle CANopen-Slaves im Zustand 4 ODER 97 und ist in der CODESYS-Steuerungskonfiguration beim CANopen-Master die Option [Automatisch starten] aktiviert, wird das NMT-Kommando [start] versendet. Ist in der CODESYS-Steuerungskonfiguration beim CANopen-Master die Option [Automatisch starten] nicht aktiviert, müssen die CANopen-Slaves manuell über das ihnen zugeordnete Strukturelement START_NODE der Struktur CANx_NODE_STATE oder alle zusammen über den Eingang START_ALL_NODES des FB CANx_MASTER_STATUS gestartet werden. Anschließend Wechsel in den Status 5.
05	5	[Normal Operation], der CANopen-Slave ist im CANopen-Status OPERATIONAL. PDOs werden übertragen.
61	97	CANopen-Slave ist als [optional] konfiguriert und ein Zugriff auf das Objekt 0x1000 blieb ohne Antwort. Wird im späteren Verlauf eine Bootup-Nachricht vom CANopen-Slave empfangen und ist in der CODESYS-Steuerungskonfiguration beim CANopen-Master die Option [Automatisch starten] aktiviert, erfolgt ein Wechsel in den Status 2.
62	98	Gerätetyp im Objekt 0x1000 entspricht nicht dem Wert in der EDS-Datei, die in der CODESYS-Steuerungskonfiguration für den CANopen-Slave eingebunden wurde. Wechsel in den Zustand 4 über das Strukturelement SET_NODE_STATE der Struktur CANx_NODE_STATE möglich. Sollte der CANopen-Slave über das globale NMT-Kommando [start] (Node-ID = 0) in den CANopen-Zustand OPERATIONAL versetzt worden sein, werden keine PDOs vom CANopen-Master an den CANopen-Slave versendet und empfangene PDOs werden ignoriert.
63	99	Es ist ein Node-Guarding oder Heartbeat Timeout aufgetreten. Sobald der CANopen-Slave wieder auf Node-Guarding reagiert bzw. Heartbeat-Nachrichten versendet und in der Steuerungskonfiguration beim CANopen-Master die Option [Automatisch starten] aktiviert ist, wird der CANopen-Slave abhängig vom in der Node-Guarding oder Heartbeat-Nachricht empfangenen Status neu konfiguriert oder sofort wieder gestartet.

Der Master sendet Nodeguard-Nachrichten an den Slave, ...
- wenn sich der Slave im Status 4 oder höher befindet UND
- wenn Nodeguarding konfiguriert wurde.

Knoten-Status aus FB lesen:

verwendeter Funktionsbaustein	hier steht dieser Knoten-Status
CANx_MASTER_STATUS CANx_SLAVE_STATUS	Ausgang NODE_STATE_SLAVE
CANOPEN_GETSTATE	Ausgang NODESTATE

CANopen-Status des Knotens

1973

Knotenstatus nach CANopen (mit diesen Werten wird der Status auch in den entsprechenden Nachrichten vom Knoten her codiert).

Status hex	dez	CANopen-Status	Beschreibung
00	0	BOOTUP	BOOTUP-Nachricht des Knotens
04	4	STOPPED	Knoten befindet sich im Zustand STOPPED. Es findet kein Datenaustausch statt und der Knoten kann auch nicht konfiguriert werden.
05	5	OPERATIONAL	Knoten befindet sich im Zustand OPERATIONAL und nimmt am normalen Datenaustausch teil.
7F	127	PRE-OPERATIONAL	Knoten befindet sich im Zustand PRE-OPERATIONAL und kann vom Master konfiguriert werden.

Wenn Nodeguarding aktiv: das höchstwertige Status-Bit wechselt (toggelt) von Nachricht zu Nachricht.

Knoten-Status aus FB lesen:

verwendeter Funktionsbaustein	hier steht dieser Knoten-Status
CANx_MASTER_STATUS	Strukturelement LAST_STATE aus dem Array NODE_STATE_SLAVE
CANx_SLAVE_STATUS	Ausgang NODE_STATE
CANOPEN_GETSTATE	Ausgang LASTNODESTATE

3.5 CANopen-Netzwerkvariablen

Inhalt

1868

3.5.1 Allgemeine Informationen

2076

CAN-Netzwerkvariablen sind eine Möglichkeit, Daten zwischen zwei oder mehreren Steuerungen auszutauschen. Der Mechanismus sollte dabei für den Anwender möglichst einfach zu handhaben sein. Derzeit sind Netzwerkvariablen auf Basis von CAN und Ethernet (UDP/IP) implementiert.

Die Variablenwerte werden dabei auf der Basis von Broadcast-Nachrichten automatisch ausgetauscht:
- in UDP/IP als Broadcast-Telegramme,
- in CAN als mit den PDOs von CANopen vergleichbare Telegramme.

Dem Protokoll entsprechend, sind diese Dienste nicht bestätigte Dienste:
es gibt keine Kontrolle, ob die Nachricht auch beim Empfänger ankommt.

Netzwerkvariablen-Austausch entspricht einer "1-zu-n-Verbindung" (1 Sender zu n Empfängern).

Das CANopen-Objektverzeichnis ist eine weitere Möglichkeit, Variablen auszutauschen. Dabei handelt es sich um eine 1-zu-1-Verbindung, die ein bestätigtes Protokoll verwendet. Hier kann der Anwender also kontrollieren, ob die Nachricht den Empfänger erreichte. Der Austausch erfolgt nicht automatisch, sondern über den Aufruf von Funktionsbausteinen aus dem Anwendungsprogramm.
→ Kapitel **Das Objektverzeichnis des CANopen-Masters** (→ S. 57)

3.5.2 CAN-Netzwerkvariablen konfigurieren

Inhalt

1869

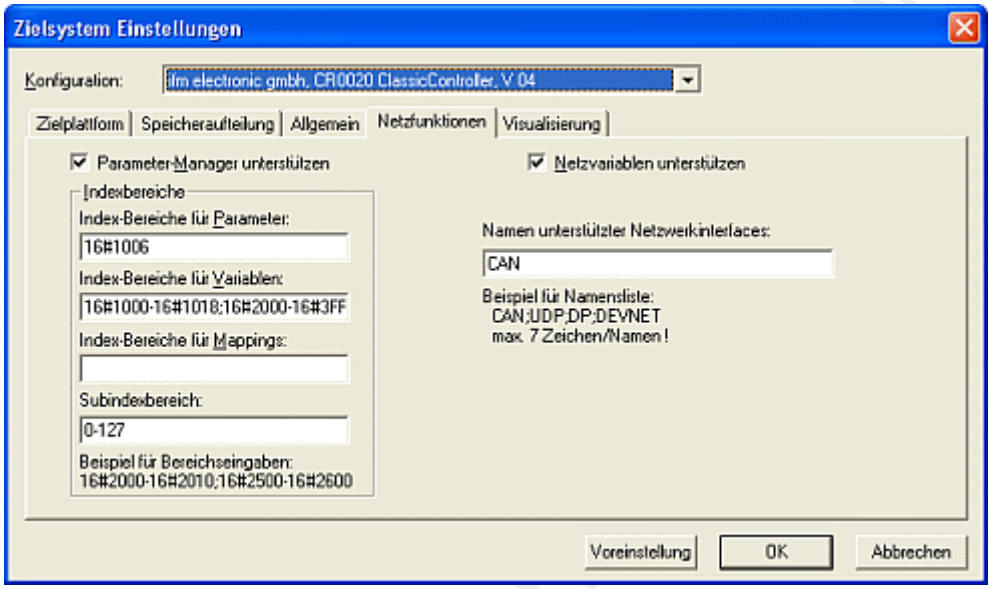
Um die Netzwerkvariablen mit CODESYS zu nutzen, benötigen Sie die folgenden Bibliotheken:

- `3s_CanDrv.lib`
- `3S_CANopenManager.lib`
- `3S_CANopenNetVar.lib`
- `SysLibCallback.lib`.

CODESYS erzeugt automatisch den nötigen Initialisierungscode sowie den Aufruf der Netzwerk-Bausteine am Zyklusanfang und Zyklusende.

Einstellungen in den Zielsystemeinstellungen

1994



Beispiel: Zielsystemeinstellungen für ClassicController CR0020

- ► Dialogbox [Zielsystemeinstellungen] wählen
- ► Register [Netzfunktionen] wählen
- ► Aktivieren Sie das Kontrollkästchen [Netzvariablen unterstützen].
- ► Bei [Namen unterstützter Netzwerkinterfaces] geben Sie den Namen des gewünschten Netzwerks an, hier CAN.
- ► Um Netzwerkvariablen zu nutzen, müssen Sie außerdem einen CANopen-Master oder CANopen-Slave in der Steuerungskonfiguration einfügen.
- ► Bitte beachten Sie die Besonderheiten bei der Anwendung von Netzwerkvariablen für die jeweiligen Gerätetypen
  → Kapitel **Besonderheiten bei Netzwerkvariablen** (→ S. 85)

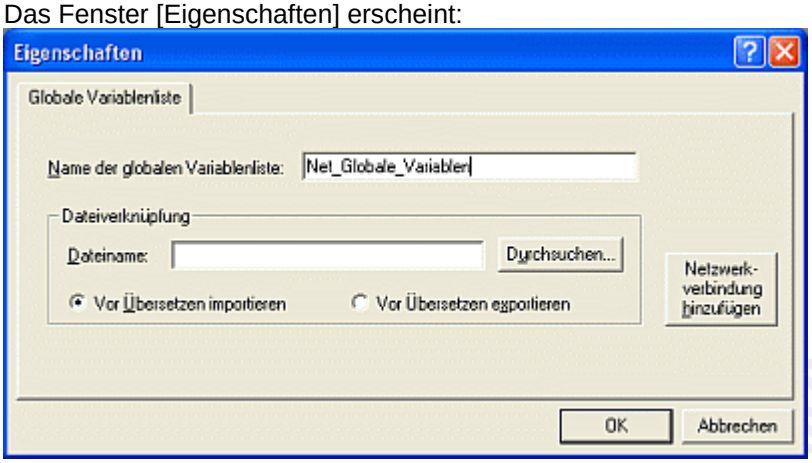
Einstellungen in den globalen Variablenlisten

1995

► Legen Sie eine neue globale Variablenliste an. Hier definieren Sie die Variablen, die sie mit anderen Steuerungen austauschen wollen.

► Öffnen Sie den Dialog mit dem Kontextmenü [Objekt Eigenschaften...].

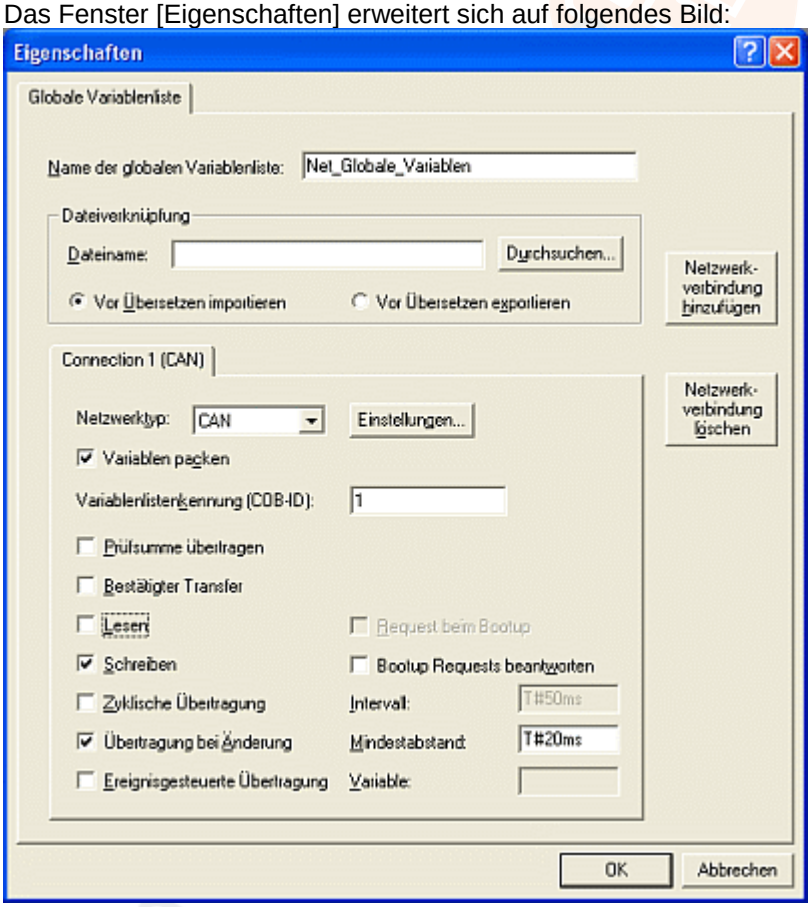
> Das Fenster [Eigenschaften] erscheint:



Wenn Sie die Netzwerkeigenschaften dieser Variablenliste definieren wollen:

► Schaltfläche [Netzwerkverbindung hinzufügen] klicken.
Wenn Sie mehrere Netzwerkverbindungen konfiguriert haben, können Sie hier auch pro Variablenliste mehrere Verbindungen konfigurieren.

> Das Fenster [Eigenschaften] erweitert sich auf folgendes Bild:



Die Optionen haben dabei folgende Bedeutungen:

Globale Variablenliste: Netzwerktyp

10055

Als Netzwerktyp können Sie einen der bei den Zielsystemeinstellungen angegebenen Netzwerknamen angeben.
Wenn Sie daneben auf die Schaltfläche [Einstellungen] klicken, können Sie die CAN-Schnittstelle wählen:

1. CAN-Schnittstelle: Wert = 0
2. CAN-Schnittstelle: Wert = 1

usw.

Globale Variablenliste: Variablen packen

10056

Option [Variablen packen] = aktiviert:
Die Variablen werden nach Möglichkeit in einer Übertragungseinheit zusammengefasst.
Bei CAN ist eine Übertragungseinheit 8 Bytes groß.

Passen nicht alle Variablen der Liste in eine Übertragungseinheit, dann werden für diese Liste automatisch mehrere Übertragungseinheiten gebildet.

Ist die Option nicht aktiviert, kommt jede Variable in eine eigene Übertragungseinheit.

Globale Variablenliste: Variablenlisten-Kennung (COB-ID)

10057

Der Basis-Identifier wird als eindeutige Kennung benutzt, um Variablenlisten verschiedener Projekte auszutauschen. Variablenlisten mit gleichem Basis-Identifier werden ausgetauscht.

► Darauf achten, dass die Definitionen der Variablenlisten mit gleichem Basis-Identifier in den verschiedenen Projekten übereinstimmen!

HINWEIS

Die COB-ID wird in CAN-Netzwerken direkt als Identifier der CAN-Nachrichten benutzt. Es gibt keine Überprüfung, ob der Identifier auch in der übrigen CAN-Konfiguration benutzt wird.

Damit die Daten korrekt zwischen zwei Steuerungen ausgetauscht werden, müssen die globalen Variablenlisten in den beiden Projekten übereinstimmen. Sie können das Feature [Dateiverknüpfung] benutzen, um dies sicherzustellen. Ein Projekt kann die Variablenlisten-Datei vor dem Übersetzen exportieren. Die anderen Projekte sollten diese Datei vor dem Übersetzen importieren.

Neben einfachen Datentypen kann eine Variablenliste auch Strukturen und Arrays enthalten. Die Elemente dieser zusammengesetzten Datentypen werden einzeln versendet.

► Keine Strings über Netzwerkvariablen versenden!
Ansonsten tritt ein Laufzeitfehler auf und der Watchdog wird aktiviert.

Wenn eine Variablenliste größer ist als ein PDO des entsprechenden Netzwerks, dann werden die Daten auf mehrere PDOs aufgeteilt. Es kann darum nicht zugesichert werden, dass alle Daten der Variablenliste in einem Zyklus empfangen werden. Teile der Variablenliste können in verschiedenen Zyklen empfangen werden. Dies ist auch für Variablen mit Struktur- und Array-Typen möglich.

Globale Variablenliste: Prüfsumme übertragen

10058

Diese Option wird nicht unterstützt.

Globale Variablenliste: Bestätigter Transfer

10059

Diese Option wird nicht unterstützt.

Globale Variablenliste: Lesen

10060

Es werden die Variablenwerte von einer (oder mehreren) Steuerungen gelesen.

Globale Variablenliste: Schreiben

10061

Die Variablen dieser Liste werden zu anderen Steuerungen gesendet.

> **HINWEIS**
>
> Sie sollten für jede Variablenliste nur eine dieser Möglichkeiten auswählen, also entweder nur lesen oder nur schreiben.
>
> Wollen Sie verschiedene Variablen eines Projekts lesen und schreiben, so verwenden Sie bitte mehrere Variablenlisten (eine zum Lesen, eine zum Schreiben).
>
> Für die Kommunikation zwischen 2 Teilnehmern sollten Sie die Variablenliste von einer Steuerung auf die andere kopieren, um die gleiche Datenstruktur zu erhalten.
>
> Zwecks besserer Übersichtlichkeit sollten Ihre Variablenlisten jeweils nur für ein Teilnehmerpaar gelten. Es ist nicht sinnvoll, dieselbe Liste für alle Teilnehmer zu verwenden.

Globale Variablenliste: Zyklische Übertragung

10062

Option [Zyklische Übertragung] = aktiviert
UND Option [Schreiben] = aktiviert:
Die Werte werden im angegebenen [Intervall] gesendet, unabhängig davon, ob sie sich geändert haben.

Globale Variablenliste: Übertragung bei Änderung

10063

Option [Übertragung bei Änderung] = aktiviert:
Für jede Übertragungseinheit wird getrennt geprüft, ob sie geändert ist und gesendet werden muss.

Mit [Mindestabstand] (Wert > 0) kann eine Mindestzeit zwischen den Nachrichtenpaketen festgelegt werden.

Globale Variablenliste: Ereignisgesteuerte Übertragung

13327

Option = [Ereignisgesteuerte Übertragung] = aktiviert:
Die CAN-Nachricht wird nur dann übertragen, wenn die angegebene binäre [Variable] auf TRUE gesetzt wird. Diese Variable kann nicht über die Eingabehilfe aus der Liste der definierten Variablen gewählt werden.

3.5.3 Besonderheiten bei Netzwerkvariablen

11811

Netzwerkvariablen werden auf folgenden CAN-Schnittstellen unterstützt:
- CAN 1 (Wert = 0)
- CAN 2 (Wert = 1)
- CAN 3 (Wert = 2)
- CAN 4 (Wert = 3)

11818

HINWEIS

► Die Identifier der Netzwerkvariablen und der Empfangs-PDOs als **dezimale** Werte eingeben!

3.6 Zusammenfassung CAN / CANopen / Netzwerkvariablen

7946

- Die COB-ID der Netzwerkvariablen muss sich unterscheiden von der CANopen-Slave-ID in der Steuerungskonfiguration und von den IDs der Sende- und Empfangs-Funktionsbausteine!
- Wenn mehr als 8 Bytes von Netzwerkvariablen in eine COB-ID gepackt werden, erweitert CODESYS das Datenpaket automatisch auf mehrere aufeinander folgende COB-IDs. Dies kann zu Konflikten mit manuell definierten COB-IDs führen!
- Netzwerkvariablen können keine String-Variablen transportieren.
- Netzwerkvariablen können transportiert werden...
  - wenn eine Variable TRUE wird (Event),
  - bei Datenänderung in der Netzwerkvariablen oder
  - zyklisch nach Zeitablauf.
- Die Intervall-Zeit beschreibt die Periode zwischen Übertragungen bei zyklischer Übertragung. Der Mindestabstand beschreibt die Wartezeit zwischen zwei Übertragungen, wenn die Variable sich zu oft ändert.
- Um die Buslast zu mindern, die Nachrichten via Netzwerkvariablen oder Sende-FBs mit Hilfe von verschiedenen Events auf mehrere SPS-Zyklen verteilen.
- In der Steuerungskonfiguration sollten die Werte für [Com Cycle Period] und [Sync. Window Length] gleich groß sein.
- Wenn die [Com Cycle Period] für einen Slave eingestellt ist, sucht der Slave in genau dieser Zeit nach einem Sync-Objekt des Masters. Deshalb muss der Wert für [Com Cycle Period] größer sein als die [Master Synch Time].
- Wir empfehlen, Slaves als "optional startup" und das Netzwerk als "automatic startup" zu setzen. Dies reduziert unnötige Buslast und ermöglicht einem kurzzeitig verlorenen Slave, sich wieder in das Netzwerk zu integrieren.
- Wir empfehlen, Analog-Eingänge auf "synchrone Übertragung" zu setzen, um Busüberlastung zu vermeiden.
- Binäre Eingänge, insbesondere die unregelmäßig schaltenden, sollten am besten auf "asynchrone Übertragung" gesetzt werden. Um bei selten auftretenden Änderungen an den binären Eingängen trotzdem eine regelmäßige Übertragung des Status zu erreichen, sollte der Event-Timer aktiviert werden.
- Beim Überwachen des Slave-Status beachten:
  - Nach dem Starten von Slaves dauert es etwas, bis die Slaves "operational" sind.
  - Beim Abschalten des Systems können Slaves wegen vorzeitigem Spannungsverlust eine scheinbare Status-Änderung anzeigen.

3.7 CAN für die Antriebstechnik

Inhalt

7678

Unter der Norm J1939 bietet die SAE dem Anwender ein CAN-Busprotokoll für die Antriebstechnik an. Hierbei werden die CAN-Nachrichten mit einem 29 Bit-Identifier übertragen. Durch den längeren Identifier kann eine große Anzahl von Nachrichten direkt dem Identifier zugeordnet werden.

Bei der Protokollerstellung hat man sich diesen Vorteil zu Nutze gemacht und gruppiert bestimmte Nachrichten in ID-Gruppen. Die Zuordnung der IDs ist in den Normen SAE J1939 und ISO 11992 festgeschrieben.

Norm	Einsatzbereich
SAE J1939	Antriebsmanagement
ISO 11992	Truck & Trailer Interface

Vom Software-Protokoll unterscheiden sich die beiden Normen nicht, da die ISO 11992 auf der SAE J1939 aufbaut. Bezüglich der Hardwareschnittstelle besteht aber ein Unterschied: höhere Spannungspegel bei der ISO 11992.

Zur Nutzung der Funktionen nach SAE J1939 / ISO 11992 benötigt man auf jeden Fall die Protokollbeschreibung des Aggregat-Herstellers (z.B. für Motor, Getriebe). Aus dieser müssen die in das Aggregat-Steuergerät implementierten Nachrichten entnommen werden, da nicht jeder Hersteller alle Nachrichten implementiert oder die Implementierung nicht für alle Aggregate sinnvoll ist.

Folgende Informationen und Hilfsmittel sollten zur Entwicklung von Programmen für Funktionen nach SAE J1939 vorhanden sein:
- Aufstellung, welche Daten von den Aggregaten genutzt werden sollen
- Übersichtsliste des Aggregatherstellers mit allen relevanten Daten
- CAN-Monitor mit 29 Bit-Unterstützung
- Wenn benötigt, die Norm SAE J1939

3.7.1 Identifier nach SAE J1939

7675

Für den Datenaustausch unter SAE J1939 ist die Bildung des 29-Bit-Identifiers entscheidend. Dieser ist schematisch nachfolgend dargestellt:

A	SOF	Identifier 11 Bits											SRR	IDE	Identifier 18 Bits																	RTR
B	SOF	Priorität			R	DP	PDU Format (PF) 6+2 Bits						SRR	IDE	noch PF		PDU specific (PS) Ziel-Adresse Gruppe extern oder proprietär								Quell-Adresse								RTR
	1	3	2	1	1	1	8	7	6	5	4	3	1	1	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	1
C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
D	-	28	27	26	25	24	23	22	21	20	19	18	-	-	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-

Legende:

A = CAN erweitertes Nachrichten-Format
B = J1939-Nachrichten-Format
C = J1939-Nachricht Bit-Position
D = CAN 29 Bit ID-Position

SOF = **S**tart **o**f **f**rame
SRR = **S**ubstitute **r**emote **r**equest
IDE = **Id**entifier **e**xtension flag
RTR = **R**emote **t**ransmission **r**equest
PDU = **P**rotocol **D**ata **U**nit
PGN = **P**arameter **G**roup **N**umber = PDU Format (PF) + PDU Source (PS)

(→ **COB-ID** (→ S. 30))

Dabei sind die 3 wesentlichen Kommunikationsmethoden unter SAE J1939 zu berücksichtigen:
- zielspezifische Kommunikation mit PDU1 (PDU-Format 0...239)
- Rundruf-Kommunikation mit PDU2 (PDU-Format 240...255)
- proprietäre Kommunikation mit PDU1 oder PDU2

3.7.2 Beispiel: ausführliche Nachrichten-Dokumentation

7679

ETC1: Electronic Transmission Controller #1 (3.3.5)
Wert = 0x0CF00203
Daraus ergeben sich folgende Details:

Bezeichnung	Parameter	Wert im Beispiel oben
transmission repetition rate	RPT	10 ms
data length	LEN	8 Bytes
PDU format	PF	240
PDU specific	PS	2
default priority	PRIO	3
data page	PG	0
source address destination address	SA DA	3
parameter group number	PGN	0x00F002
identifier	ID	0x0CF00203
data field	SRC DST	Array-Adresse (Bedeutung der Datenbytes 1...8 à Herstellerdokumentation)

Da im Beispiel vom Hersteller alle relevanten Daten bereits aufbereitet wurden, können diese direkt an die Funktionsbausteine übertragen werden.

Je nach benötigter Funktion werden die entsprechenden Werte eingesetzt. Bei den Feldern SA / DA oder SRC / DST ändert sich die Bedeutung (aber nicht der Wert), entsprechend der Empfangs- oder der Sendefunktion.

Die einzelnen Datenbytes müssen aus dem Array ausgelesen und entsprechend ihrer Bedeutung weiterverarbeitet werden.

3.7.3 Beispiel: kurze Nachrichten-Dokumentation

7680

Aber auch wenn vom Aggregathersteller nur eine Kurzdokumentation zur Verfügung steht, kann man sich die FB-Parameter aus dem Identifier herleiten. Neben dem ID werden zusätzlich in jedem Fall die "Transmission repetition rate" und die Bedeutung der Datenfelder benötigt.

Wenn es sich nicht um herstellerspezifische Protokollnachrichten handelt, kann auch die Norm SAE J1939 oder ISO 11992 als Informationsquelle dienen.

Der Identifier 0x0CF00203 setzt sich wie folgt zusammen:

DATA	PRIO, reserv., PG	PF	PS	SA / DA
hex	0C	F0	02	03
dez	→ folgende Tabelle	240	2	3

Da es sich bei diesen Werten um hexadezimale Zahlen handelt, von denen man teilweise einzelne Bits benötigt, müssen die Zahlen weiter zerlegt werden:

DATA	PRIO, reserv., PG
hex	0C
bin	0000 1100

das wird zerlegt zu:

DATA	nicht relevant	PRIO	reserviert	PG
bin	000	011	0	0
dez	---	3	0	0

Weitere typische Kombinationen für "PRIO, reserv., PG "

0x18:

DATA	PRIO, reserv., PG
hex	18
bin	0001 1000

das wird zerlegt zu:

DATA	nicht relevant	PRIO	reserviert	PG
bin	000	110	0	0
dez	---	6	0	0

0x1C:

DATA	PRIO, reserv., PG
hex	1C
bin	0001 1100

das wird zerlegt zu:

DATA	nicht relevant	PRIO	reserviert	PG
bin	000	111	0	0
dez	---	7	0	0

3.8 CAN / CANopen: Fehler und Fehlerbehandlung

Inhalt

1171

Die hier beschriebenen Fehlermechanismen werden von dem im Controller integrierten CAN-Controller automatisch abgearbeitet. Der Anwender hat darauf keinen Einfluss. Der Programmierer sollte (je nach Anwendung) auf gemeldete Fehler im Anwendungsprogramm reagieren.

Ziel der CAN-Fehler-Mechanismen ist es:

- Sicherstellung einheitlicher Datenobjekte im gesamten CAN-Netz
- Dauerhafte Funktionsfähigkeit des Netzes auch im Falle eines defekten CAN-Teilnehmers
- Unterscheidung zwischen zeitweiliger und dauerhafter Störung eines CAN-Teilnehmers
- Lokalisierung und Selbstabschaltung eines defekten Teilnehmers in 2 Stufen:
  - fehlerpassiv (error-passiv)
  - trennen vom Bus (bus-off)

  Dies ermöglicht einem zeitweilig gestörten Teilnehmer eine "Erholungspause".

Um dem interessierten Anwender einen Überblick über das Verhalten des CAN-Controllers im Fehlerfall zu geben, soll an dieser Stelle vereinfacht die Fehlerbehandlung beschrieben werden. Nach der Fehlererkennung werden die Informationen automatisch aufbereitet und stehen im Anwendungsprogramm dem Programmierer als CAN-Fehler-Bits zur Verfügung.

3.8.1 CAN-Fehler

8589

Fehlertelegramm

1172

Erkennt ein Busteilnehmer eine Fehlerbedingung, so sendet er sofort ein Fehlertelegramm und veranlasst damit den Abbruch der Übertragung bzw. das Verwerfen der von anderen Teilnehmern schon empfangenen fehlerfreien Nachrichten. Dadurch wird sichergestellt, dass allen Teilnehmern fehlerfreie und einheitliche Daten zur Verfügung stehen. Da das Fehlertelegramm unmittelbar übertragen wird, kann im Gegensatz zu anderen Feldbussystemen (diese warten eine festgelegte Quittierungszeit ab) sofort mit der Wiederholung der gestörten Nachricht durch den Absender begonnen werden. Dies ist eines der wichtigsten Merkmale von CAN.

Eine der grundsätzlichen Problematiken der seriellen Datenübertragung ist, dass ein dauerhaft gestörter oder defekter Busteilnehmer das gesamte System blockieren kann. Gerade die Fehlerbehandlung bei CAN würde solche Gefahr fördern. Um diesen Fall auszuschließen, ist ein Mechanismus erforderlich, welcher den Defekt eines Teilnehmers erkennt und diesen Teilnehmer gegebenenfalls vom Bus abschaltet.

Fehlerzähler

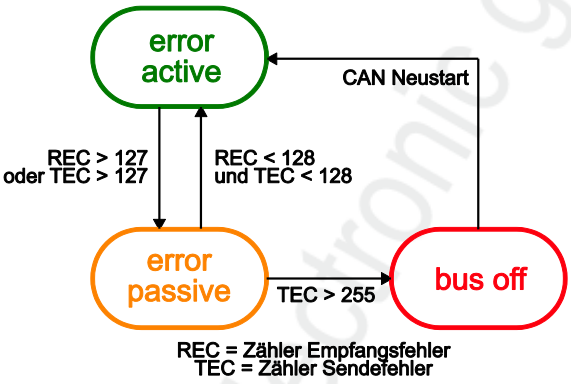
1173

Dazu sind im CAN-Controller ein Sende- und ein Empfangsfehlerzähler enthalten. Diese werden bei jedem fehlerhaften Sende- oder Empfangsvorgang heraufgezählt (inkrementiert). War eine Übertragung fehlerfrei, werden diese Zähler wieder heruntergezählt (dekrementiert).

Die Fehlerzähler werden jedoch im Fehlerfall stärker inkrementiert, als sie im Erfolgsfalle dekrementiert werden. Über eine bestimmte Zeitspanne kann dies zu einem merklichen Anstieg der Zählerstände führen, selbst wenn die Anzahl der ungestörten Nachrichten größer ist, als die Anzahl der gestörten Nachrichten. Längere fehlerfreie Zeitspannen bauen die Zählerstände langsam wieder ab. Die Zählerstände sind somit ein Maß für die relative Häufigkeit von gestörten Nachrichten.

Werden Fehler von einem Teilnehmer selbst als erster erkannt (= selbstverschuldete Fehler), wird bei diesem Teilnehmer der Fehler stärker "bestraft" als bei den anderen Busteilnehmern. Dazu wird der Zähler um einen höheren Betrag inkrementiert.

Übersteigt nun der Zählerstand eines Teilnehmers einen bestimmten Wert, kann davon ausgegangen werden, dass dieser Teilnehmer defekt ist. Damit dieser Teilnehmer den folgenden Busverkehr nicht weiter durch aktive Fehlermeldungen (error active) stört, wird er "fehlerpassiv" geschaltet (error passiv).



error active
→ **Teilnehmer fehleraktiv** (→ S. 93)
error passive
→ **Teilnehmer fehlerpassiv** (→ S. 93)
bus off
→ **Teilnehmer bus-off** (→ S. 93)
CAN Restart
→ Teilnehmer bus-off

Grafik: Mechanismus des Fehlerzählers

Teilnehmer fehleraktiv

1174

Ein fehleraktiver Teilnehmer nimmt voll am Busverkehr teil und darf erkannte Fehler durch Senden des aktiven Fehlertelegramms signalisieren. Wie bereits beschrieben, wird dadurch die übertragene Nachricht zerstört.

Teilnehmer fehlerpassiv

19173

Ein fehlerpassiver Teilnehmer ist ebenfalls noch voll kommunikationsfähig. Er darf allerdings einen von ihm erkannten Fehler nur durch ein – den Busverkehr nicht störendes – passives Fehlerflag kenntlich machen. Ein fehlerpassiver Teilnehmer wird beim Unterschreiten eines festgelegten Zählerwertes wieder fehleraktiv.

Zur Reaktion im Anwendungsprogramm:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
Bei einem Wert des Fehlerzählers ≥ 96: Systemvariable CANx_WARNING = TRUE.	Bei Überschreiten der Warnschwelle für den TX-Fehlerzähler: im FB CAN_STATUS wird Ausgang WARNING_TX = TRUE Bei Überschreiten der Warnschwelle für den RX-Fehlerzähler: im FB CAN_STATUS wird Ausgang WARNING_RX = TRUE

Der Teilnehmer ist in diesem Zustand fehlerpassiv.

Teilnehmer bus-off

19174

Wird der Fehlerzählerwert weiter inkrementiert, wird nach Überschreiten eines Maximalzählerwertes der Teilnehmer vom Bus abgeschaltet (bus-off).

Zur Reaktion im Anwendungsprogramm:

Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:	Für alle CR04nn, CR1nnn, CR253n gilt:
Die Systemvariable CANx_BUSOFF = TRUE	Bei Überschreiten der oberen Schwelle für den TX-Fehlerzähler: im FB CAN_STATUS wird der Ausgang BUSOFF = TRUE Bei Überschreiten der oberen Schwelle für den RX-Fehlerzähler: im FB CAN_STATUS wird der Ausgang BUSOFF = TRUE
> Der Fehler CANx_BUSOFF wird vom Laufzeitsystem automatisch behandelt und zurückgesetzt. Soll eine genauere Fehlerbehandlung und Auswertung über das Anwendungsprogramm erfolgen: ► Den FB CANx_ERRORHANDLER einsetzen!	> Der Fehler BUSOFF wird vom Laufzeitsystem automatisch behandelt (Recovery). Neustart-Versuch der betroffenen CAN-Schnittstelle bis zu 4-mal im 1s-Abstand. > Wenn der Bus-Off nach dem 4. Versuch noch nicht behoben ist, schaltet sich das Gerät von der Schnittstelle weg und nimmt nicht mehr am Bus-Verkehr teil.

► Die Anzeige der Busfehler im Anwendungsprogramm immer zurücksetzen:

► Den Fehler CANx_BUSOFF explizit im Anwendungsprogramm zurücksetzen!	► Im FB CAN_STATUS den Eingang CLEAR = TRUE setzen. > Die Anzeige der Fehler wird zurückgesetzt. Wenn diese Fehler im nächsten Zyklus nicht mehr neu gesetzt werden: > der Bus-Off ist behoben > das Gerät ist wieder ERROR ACTIVE, nimmt also wieder ganz normal an der Bus-Kommunikation teil.

3.8.2 CANopen-Fehler

Inhalt

11670

Aufbau einer EMCY-Nachricht

Inhalt

19177

Die Signalisierung von Fehlerzuständen erfolgt unter CANopen über einen sehr einfachen, standardisierten Mechanismus. Jedes Auftreten eines Fehlers bei einem CANopen-Gerät wird über eine spezielle Nachricht signalisiert, die den Fehler genauer beschreibt.

Verschwindet ein Fehler oder seine Ursache nach einer bestimmten Zeit wieder, wird dieses Ereignis ebenfalls über die EMCY-Nachricht signalisiert. Die zuletzt aufgetretenen Fehler werden im Objektverzeichnis (Objekt 0x1003) abgelegt und können über einen SDO-Zugriff ausgelesen werden. Zusätzlich spiegelt sich die aktuelle Fehlersituation im Error-Register (Objekt 0x1001) wider.

Fehler via SDO-Zugriff auslesen:

- **Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:**

CANx_SDO_READ	CAN-Schnittstelle x: liest das SDO mit den angegebenen Indizes aus dem Knoten aus x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

- **Für alle CR04nn, CR1nnn, CR253n gilt:**

CANOPEN_SDOREAD	= CANopen Read SDO liest ein "expedited SDO" = beschleunigtes Nachrichten-Objekt mit Servicedaten
CANOPEN_SDOREADBLOCK	= CANopen Read SDO Block liest den angegeben Eintrag im Objektverzeichnis eines Knotens im Netz per SDO-Blocktransfer
CANOPEN_SDOREADMULTI	= CANopen Read SDO Multi liest den angegeben Eintrag im Objektverzeichnis eines Knotens im Netz

Man unterscheidet folgende Fehler:

8046

Kommunikationsfehler (Beispiele:)

- Der CAN-Controller signalisiert CAN-Fehler.
  (Das gehäufte Auftreten ist ein Indiz für physikalische Probleme. Diese Fehler können einen erheblichen Einfluss auf das Übertragungsverhalten und damit auf den Datendurchsatz eines Netzwerks haben.)
- Life-Guarding- oder Heartbeat-Fehler
- SYNC-Fehler (nur Slave)

Anwendungsfehler (Beispiele:)

- Kurzschluss oder Leiterbruch
- Temperatur zu hoch

Emergency-Nachrichten

9973

Gerätefehler im Slave oder Probleme im CAN-Bus lösen Emergency-Nachrichten aus:

COB-ID	DLC	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x080 + Node-ID	X	Error-Code		Objekt 0x1001	gerätespezifisch				

Beachten Sie die umgekehrte Byte-Reihenfolge! (⇨ Little Endian oder Intel-Format)

Identifier

8048

Der Identifier für die Fehlernachricht besteht aus der Summe folgender Elemente:
EMCY-Default-Identifier 128 (0x80)
+
Node-ID

EMCY-Fehler-Code

8049

Er gibt detailliert Auskunft darüber, welcher Fehler aufgetreten ist. Eine Liste möglicher Fehler-Codes ist bereits im Kommunikationsprofil definiert. Fehler-Codes, die nur für eine bestimmte Geräteklasse gültig sind, werden im jeweiligen Geräteprofil dieser Geräteklasse festgelegt.

Übersicht CANopen-Error-Codes

8545

Error Code (hex)	Meaning / Bedeutung
00xx	Reset or no Error (Fehler rücksetzen / kein Fehler)
10xx	Generic Error (allgemeiner Fehler)
20xx	Current (Stromfehler)
21xx	Current, device input side (Stromfehler, eingangsseitig)
22xx	Current inside the device (Stromfehler im Geräteinnern)
23xx	Current, device output side (Stromfehler, ausgangsseitig)
30xx	Voltage (Spannungsfehler)
31xx	Mains Voltage
32xx	Voltage inside the device (Spannungsfehler im Geräteinnern)
33xx	Output Voltage (Spannungsfehler, ausgangsseitig)
40xx	Temperature (Temperaturfehler)
41xx	Ambient Temperature (Umgebungstemperaturfehler)
42xx	Device Temperature (Gerätetemperaturfehler)
50xx	Device Hardware (Geräte-Hardware-Fehler)
60xx	Device Software (Geräte-Software-Fehler)
61xx	Internal Software (Firmware-Fehler)
62xx	User Software (Applications-Software)
63xx	Data Set (Daten-/Parameterfehler)
70xx	Additional Modules (zusätzliche Module)
80xx	Monitoring (Überwachung)
81xx	Communication (Kommunikation)
8110	CAN Overrun-objects lost (CAN Überlauf-Datenverlust)
8120	CAN in Error Passiv Mode (CAN im Modus "fehlerpassiv")
8130	Life Guard Error or Heartbeat Error (Guarding-Fehler oder Heartbeat-Fehler)
8140	Recovered from Bus off (Bus-Off zurückgesetzt)
8150	Transmit COB-ID collision (Senden "Kollision des COB-ID")
82xx	Protocol Error (Protokollfehler)
8210	PDO not processed due to length error (PDO nicht verarbeitet, fehlerhafte Längenangabe)
8220	PDO length exceeded (PDO Längenfehler, ausgangsseitig)
90xx	External Error (Externer Fehler)
F0xx	Additional Functions (zusätzliche Funktionen)
FFxx	Device specific (gerätespezifisch)

Objekt 0x1001 (Error-Register)

8547

Dieses Objekt spiegelt den allgemeinen Fehlerzustand eines CANopen-Gerätes wider. Das Gerät ist dann als fehlerfrei anzusehen, wenn das Objekt 0x1001 keinen Fehler mehr signalisiert.

Bit	Meaning (Bedeutung)
0	Generic Error (allgemeiner Fehler)
1	Current (Stromfehler)
2	Voltage (Spannungsfehler)
3	Temperature (Temperaturfehler)
4	Communication Error (Kommunikationsfehler)
5	Device Profile specific (Geräteprofil spezifisch)
6	Reserved – always 0 (reserviert – immer 0)
7	manufacturer specific (herstellerspezifisch)

Für eine Fehlermeldung können mehrere Bits im Error-Register gleichzeitig gesetzt sein.

Beispiel: CR2033, Meldung "Leiterbruch" an Kanal 2 (→ Installationsanleitung des Geräts):

COB-ID	DLC	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x80 + Node-ID		**00**	**FF**	**81**	**10**	**00**	00	00	00

Error-Code = 0xFF00

Error-Register = 0x81 = 0b1000 0001, besteht also aus folgenden Fehlern:
- generic error (allgemeiner Fehler)
- manufacturer specific (herstellerspezifisch)

Betroffener Kanal = 0x0010 = 0b0000 0000 0001 0000 = Kanal 2

Objekt 0x1003 (Error Field)

8050

Das Objekt 0x1003 stellt den Fehlerspeicher eines Gerätes dar. Die Subindizes enthalten die zuletzt aufgetretenen Fehler, die ein Fehler-Telegramm ausgelöst haben.

Tritt ein neuer Fehler auf, dann wird sein EMCY-Fehler-Code immer im Subindex 0x1 gespeichert. Alle anderen, älteren Fehler werden im Fehlerspeicher um einen Platz nach hinten geschoben, also der Subindex um 1 erhöht. Falls alle unterstützten Subindizes belegt sind, wird der älteste Fehler gelöscht. Der Subindex 0x0 wird auf die Anzahl der gespeicherten Fehler erhöht. Nachdem alle Fehler behoben sind, wird in das Fehlerfeld des Subindex 0x1 der Wert "0" geschrieben.

Um den Fehlerspeicher zu löschen, kann der Subindex 0x0 mit dem Wert "0" beschrieben werden. Andere Werte dürfen nicht eingetragen werden.

Gerätefehler signalisieren

19178

Wie beschrieben, werden EMCY-Nachrichten versendet, wenn Fehler in einem Gerät auftreten. Im Unterschied zu frei programmierbaren Geräten, werden beispielsweise von dezentralen Ein-/Ausgangsmodulen (z.B. CompactModule CR2033) Fehlermeldungen automatisch verschickt. Entsprechende Fehler-Codes → jeweiliges Gerätehandbuch.

Die programmierbaren Geräte erzeugen nur dann automatisch eine EMCY-Nachricht (z.B. für "Kurzschluss am Ausgang Q07"), wenn einer der folgenden FBs in das Anwendungsprogramm eingebunden wird:

- **Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:**

CANx_MASTER_EMCY_HANDLER	verwaltet den geräteeigenen Fehlerstatus des CANopen-Masters an der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SLAVE_EMCY_HANDLER	verwaltet den geräteeigenen Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x: • Error Register (Index 0x1001) und • Error Field (Index 0x1003) des CANopen Objektverzeichnis x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

- **Für alle CR04nn, CR1nnn, CR253n gilt:**

CANOPEN_GETERRORREGISTER	= Get CANopen Error-Register liest die Fehler-Register 0x1001 und 0x1003 der Steuerung aus Die Register können durch Setzen der entsprechenden Eingänge zurückgesetzt werden.
CANOPEN_GETEMCYMESSAGES	= Get CANopen Emergency Messages listet alle Emergency-Nachrichten auf, die die Steuerung seit dem letzten Löschen der Nachrichten von anderen Knoten am Netz empfangen hat Die Liste kann durch Setzen des entsprechenden Eingangs zurückgesetzt werden.

Übersicht der automatisch verschickten EMCY-Fehler-Codes für alle mit CODESYS programmierbaren **ecomat*mobile***-Geräte → Kapitel **Übersicht CANopen-Error-Codes** (→ S. 96).

Sollen zusätzlich noch anwendungsspezifische Fehler durch das Anwendungsprogramm verschickt werden, dann einen der folgenden FBs einsetzen:

- **Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:**

CANx_MASTER_SEND_EMERGENCY	versendet anwendungsspezifische Fehlerstatus des CANopen-Masters an der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SLAVE_SEND_EMERGENCY	versendet anwendungsspezifische Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

- **Für alle CR04nn, CR1nnn, CR253n gilt:**

CANOPEN_SENDEMCYMESSAGE	= CANopen Send Emergency-Message versendet eine EMCY-Nachricht. Die Nachricht wird aus den entsprechenden Parametern zusammengebaut und ins Register 0x1003 eingetragen

Herstellerspezifische Informationen

8548

Hier kann ein Gerätehersteller zusätzliche Fehlerinformationen mitteilen. Das Format ist dabei frei wählbar.

Beispiel:

In einem Gerät treten zwei Fehler auf und werden über den Bus gemeldet:

- Kurzschluss der Ausgänge:
  Fehler-Code 0x2308,
  im Objekt 0x1001 wird der Wert 0x03 (0b0000 0011) eingetragen
  (allg. Fehler und Stromfehler)
- CAN-Überlauf:
  Fehler-Code 0x8110,
  im Objekt 0x1001 wird der Wert 0x13 (0b0001 0011) eingetragen
  (allg. Fehler, Stromfehler und Kommunikationsfehler)

>> CAN-Überlauf bearbeitet:
Fehler-Code 0x0000,
im Objekt 0x1001 wird der Wert 0x03 (0b0000 0011) eingetragen
(allg. Fehler, Stromfehler, Kommunikationsfehler zurückgesetzt.)

Nur aus dieser Information kann man entnehmen, dass der Kommunikationsfehler nicht mehr anliegt.

Übersicht CANopen-EMCY-Codes (Standard-Seite)

18071

- **Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:**
  Die folgenden EMCY-Meldungen werden automatisch versendet, wenn der FB CANx_MASTER_EMCY_HANDLER zyklisch aufgerufen wird.
- **Für alle CR04nn, CR1nnn, CR253n gilt:**
  Im CANopen-Stack ist noch keiner dieser EMCY-Codes fix implementiert. Vorschlag:
  ▶ Diese EMCY-Codes mit dem FB CANOPEN_SENDEMCYMESSAGE erzeugen.

EMCY-Code Objekt 0x1003		Objekt 0x1001	herstellerspezifische Informationen					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Beschreibung
00	21	03						Leiterbruch Eingänge
08	21	03						Kurzschluss Eingänge
10	21	03						Überstrom 0…20 mA
00	23	03						Leiterbruch Ausgänge
08	23	03						Kurzschluss Ausgänge
10	23	03						Überlast Ausgänge
00	31	05						Versorgungsspannung VBBS
00	33	05						Ausgangsspannung VBBO
08	33	05						Ausgangsspannung VBBR
00	42	09						Übertemperatur

Die Einträge für die Bytes 3...7 richten sich nach der konkreten Verteilung der Ein- und Ausgänge des Geräts (→ Programmierhandbuch).

Übersicht CANopen-EMCY-Codes (Extended-Seite)

18072

- **Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:**
  Die folgenden EMCY-Meldungen werden automatisch versendet, wenn der FB CANx_MASTER_EMCY_HANDLER zyklisch aufgerufen wird.
- **Für alle CR04nn, CR1nnn, CR253n gilt:**
  Im CANopen-Stack ist noch keiner dieser EMCY-Codes fix implementiert. Vorschlag:
  ▶ Diese EMCY-Codes mit dem FB CANOPEN_SENDEMCYMESSAGE erzeugen.

EMCY-Code Objekt 0x1003		Objekt 0x1001	herstellerspezifische Informationen					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Beschreibung
01	21	03						Leiterbruch Eingänge
09	21	03						Kurzschluss Eingänge
11	21	03						Überstrom 0…20 mA
01	23	03						Leiterbruch Ausgänge
09	23	03						Kurzschluss Ausgänge
10	23	03						Überlast Ausgänge
10	33	05						Ausgangsspannung VBB1
11	33	05						Ausgangsspannung VBB2
12	33	05						Ausgangsspannung VBB3
13	33	05						Ausgangsspannung VBB4
18	33	05						Versorgung Relais

Die Einträge für die Bytes 3...7 richten sich nach der konkreten Verteilung der Ein- und Ausgänge des Geräts (→ Programmierhandbuch).

Übersicht CANopen-EMCY-Codes (CANx)

18073

- **Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:**
  Die folgenden EMCY-Meldungen werden automatisch versendet, wenn der FB CANx_MASTER_EMCY_HANDLER zyklisch aufgerufen wird.
- **Für alle CR04nn, CR1nnn, CR253n gilt:**
  Im CANopen-Stack ist noch keiner dieser EMCY-Codes fix implementiert. Vorschlag:
  ▶ Diese EMCY-Codes mit dem FB CANOPEN_SENDEMCYMESSAGE erzeugen.

13094

Die Angaben für CANx gelten für jede der CAN-Schnittstellen.

EMCY-Code Objekt 0x1003		Objekt 0x1001	herstellerspezifische Informationen					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Beschreibung
00	80	11	---	---	---	---	---	CANx Monitoring SYNC-Error (nur Slave)
00	81	11	---	---	---	---	---	CANx Warngrenze (> 96)
10	81	11	---	---	---	---	---	CANx Empfangspuffer Überlauf
11	81	11	---	---	---	---	---	CANx Sendepuffer Überlauf
30	81	11	---	---	---	---	---	CANx Guard-/Heartbeat-Error (nur Slave)

4 Ausgänge steuern – Beschreibung

Inhalt

13741

4.1 PWM-Funktionen – Beschreibung

Inhalt

13831

In diesem Kapitel erfahren Sie mehr über die Pulsweitenmodulation im **ecomat*mobile***-Gerät.

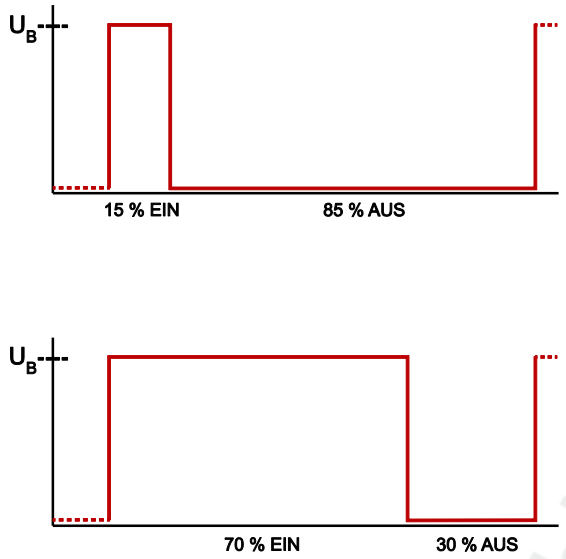
Verfügbarkeit von PWM oder PWMi:
→ Datenblatt des Geräts
→ Gerätehandbuch des Geräts

4.1.1 PWM-Signalverarbeitung – Beschreibung

Inhalt

13832
6889

PWM steht als Abkürzung für die **P**uls-**W**eiten-**M**odulation, zuweilen auch "Puls-Pausen-Modulation" (PPM) genannt. Sie wird im Bereich der Steuerungen für den mobilen und robusten Einsatz hauptsächlich zur Ansteuerung von proportionalen Ventilen (PWM-Ventilen) genutzt. Ferner kann durch eine entsprechende Zusatzbeschaltung eines PWM-Ausganges (Zubehör) aus dem pulsweitenmodulierten Ausgangssignal eine analoge Ausgangsspannung erzeugt werden.



Grafik: Prinzip PWM

Bei dem PWM-Ausgangssignal handelt es sich um ein getaktetes Signal zwischen GND und Versorgungsspannung. Innerhalb einer festen Periode (PWM-Frequenz) wird das Puls-/Pausenverhältnis variiert. Durch die angeschlossene Last stellt sich je nach Puls-/Pausenverhältnis der entsprechende Effektivstrom ein.

Die PWM-Funktion der Controller ist eine Hardware-Funktion, die vom Prozessor zur Verfügung gestellt wird. Um die integrierten PWM-Ausgänge des Controllers zu nutzen, müssen diese im Anwendungsprogramm initialisiert und entsprechend dem gewünschten Ausgangssignal parametriert werden.

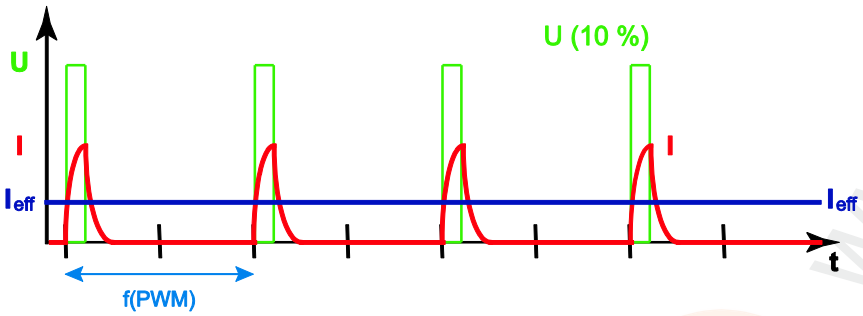
PWM: Was macht ein PWM-Ausgang?

1563

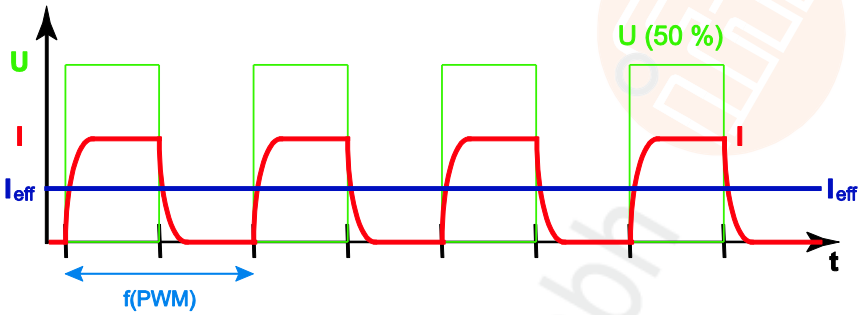
PWM steht für "**P**uls-**W**eiten-**M**odulation" und meint folgendes Prinzip:

Digitale Ausgänge liefern in der Regel eine feste Ausgangsspannung, sobald sie eingeschaltet sind. Der Wert der Ausgangsspannung lässt sich hier **nicht** verändern. PWM-Ausgänge dagegen zerlegen die Spannung in eine schnelle Folge von vielen Rechteck-Impulsen. Das Verhältnis der Impulsdauer "eingeschaltet" zur Impulsdauer "ausgeschaltet" bestimmt den Effektivwert der gewünschten Ausgangsspannung. Man spricht dann von der Einschaltdauer (in [%]).

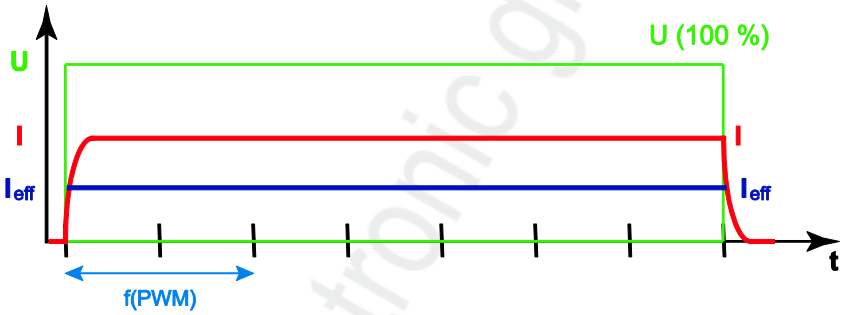
In den folgenden Skizzen sind die Strom-Verläufe nur stilisiert als Gerade dargestellt. Tatsächlich verläuft der Strom nach einer e-Funktion.



Grafik: Verlauf von PWM-Spannung U und Spulenstrom I bei 10 % Einschaltdauer:
Der effektive Spulenstrom I_{eff} beträgt ebenfalls 10 %



Grafik: Verlauf von PWM-Spannung U und Spulenstrom I bei 50 % Einschaltdauer:
Der effektive Spulenstrom I_{eff} beträgt ebenfalls 50 %



Grafik: Verlauf von PWM-Spannung U und Spulenstrom I bei 100 % Einschaltdauer:
Der effektive Spulenstrom I_{eff} beträgt ebenfalls 100 %

PWM: Was ist der Dither?

1564

Wenn ein Proportional-Hydraulikventil angesteuert wird, bewegt sich sein Kolben nicht sofort los und anfangs auch nicht proportional zum Spulenstrom. Durch diesen "Slip-Stick-Effekt" – eine Art "Losbrechmoment" – benötigt das Ventil zu Anfang einen etwas höheren Strom, um die Kraft aufzubringen, den Kolben aus der Ruhelage zu bewegen. Das gleiche geschieht auch bei jeder anderen Positionsänderung des Ventilkolbens. Gerade bei sehr geringen Stellgeschwindigkeiten zeigt sich dieser Effekt in Form einer ruckenden Bewegung des Ventilkolbens.

Diesem Problem begegnet die Technik, indem der Ventilkolben ständig einer kleinen Hin- und Herbewegung (dem Dither) unterworfen wird. Dabei vibriert der Kolben ständig hin und her und kann nicht "festkleben". Eine auch kleine Positionsänderung erfolgt nun ohne Verzögerung quasi als "fliegender Wechsel".

Vorteil: Der so angesteuerte Hydraulikzylinder kann wesentlich feinfühliger bewegt werden.

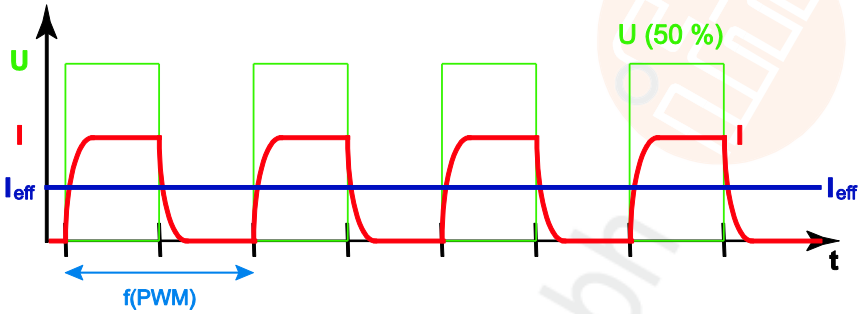
Nachteil: Das Ventil wird mit Dither messbar heißer als ohne, weil die Ventilspule nun dauernd arbeitet.

Es gilt also, eine "goldene Mitte" zu finden.

Wann ist ein Dither sinnvoll?

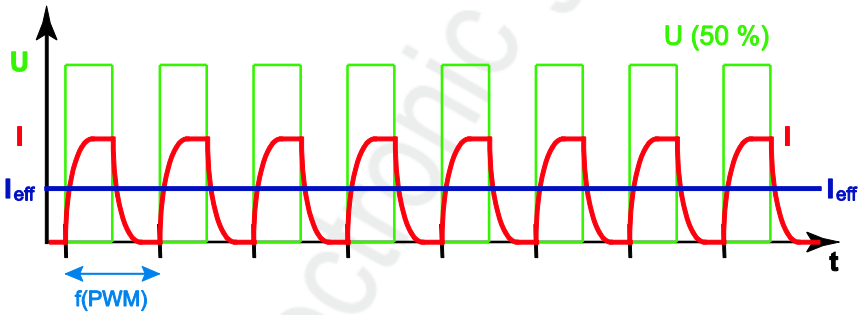
1565

Wenn der PWM-Ausgang eine Puls-Frequenz ausgibt, die klein genug ist (Richtwert: bis 250 Hz), dass sich der Ventilkolben ständig mit einem Mindesthub bewegt, dann ist kein zusätzlicher Dither erforderlich (→ nächstes Bild):

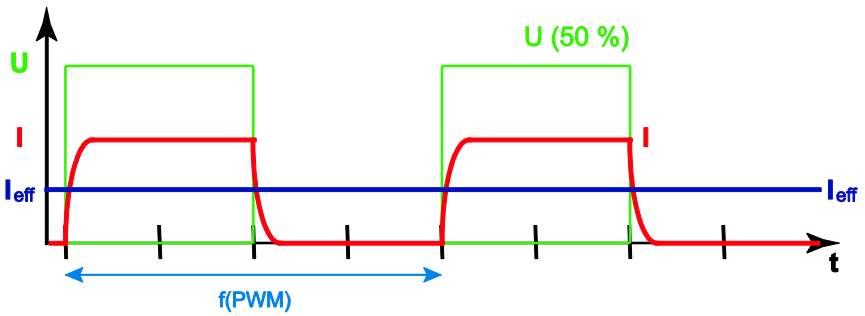


Grafik: Ausgewogenes PWM-Signal; kein Dither erforderlich.

Bei einer höheren PWM-Frequenz (Richtwert: 250 Hz bis 1 kHz) ist die Restbewegung des Ventilkolbens so kurz oder so langsam, dass dies effektiv als Stillstand resultiert, der Ventilkolben also wieder in der momentanen Position festkleben kann (und auch wird!, → nächste Grafiken):



Grafik: Hohe Frequenz des PWM-Signals führt annähernd zu einem resultierenden Gleichstrom in der Spule. Der Ventilkolben bewegt sich nicht mehr genug. Bei jeder Signaländerung muss der Ventilkolben erneut das Losbrechmoment überwinden.



Grafik: Zu niedrige Frequenz des PWM-Signals lässt nur seltene, ruckende Bewegungen des Ventilkolbens entstehen. Jeder Impuls bewegt den Ventilkolben erneut aus seiner Ruhelage; jedes Mal muss der Ventilkolben erneut das Losbrechmoment überwinden.

> **HINWEIS**
>
> Bei einer PWM-Einschaltdauer unter 10 % und größer 90 % ist es sinnvoll und notwendig, dem PWM-Signal ein Dither-Signal zu überlagern.

Dither-Frequenz und -Amplitude

1566

Das Puls-/Pausenverhältnis (die Einschaltdauer) des PWM-Ausgangssignals wird mit der Dither-Frequenz umgeschaltet. Die Dither-Amplitude bestimmt, wie groß der Unterschied der Einschaltdauer in den beiden Dither-Halbwellen ist.

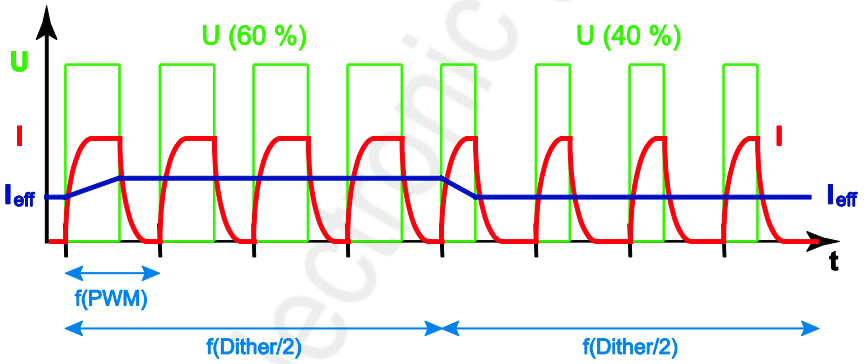
> Die Dither-Frequenz muss ein ganzzahliger Teil der PWM-Frequenz sein. Andernfalls wird das hydraulische System nicht gleichförmig arbeiten, sondern schwingen.

Beispiel Dither

1567

Die Dither-Frequenz beträgt den 8-ten Teil der PWM-Frequenz.
Die Dither-Amplitude beträgt 10 %.

Bei der im Bild anstehenden Einschaltdauer von 50 % wird die tatsächliche Einschaltdauer für 4 Impulse 60 % betragen und für die nächsten 4 Impulse 40 %, was im Mittel wieder 50 % Einschaltdauer ausmacht. Der resultierende effektive Spulenstrom wird 50 % des maximalen Spulenstroms betragen.



Im Ergebnis wird der Ventilkolben wieder um seine jeweilige Ruhelage schwingen, um bei der nächsten Signaländerung sofort seine neue Position annehmen zu können, ohne zuvor das Losbrechmoment überwinden zu müssen.

PWM: Bausteine

19183

PWM-Funktionen für die PWM-fähigen Ausgänge erreichen Sie mit den folgenden Bausteinen:

Einige hier aufgeführten Bausteine sind nur für einzelne Geräte verfügbar.

- **Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:**

OUTPUT_BRIDGE	H-Brücke an einem PWM-Kanalpaar
OUTPUT_CURRENT	misst den Strom (Mittelung über Dither-Periode) an einem Ausgangskanal
OUTPUT_CURRENT_CONTROL	Stromregler für einen PWMi-Ausgangskanal
PWM1000	initialisiert und parametriert einen PWM-fähigen Ausgangskanal das Puls-Pausen-Verhältnis kann in 1 ‰-Schritten angegeben werden

- **Für alle CR04nn, CR253n gilt:**

CURRENT_CONTROL	Stromregler für einen PWMi-Ausgangskanal
H_BRIDGE	H-Brücke an einem PWM-Kanalpaar
PWM1000	initialisiert und parametriert einen PWM-fähigen Ausgangskanal das Puls-Pausen-Verhältnis kann in 1 ‰-Schritten angegeben werden
PWM1000_LOW	initialisiert und parametriert einen PWM-fähigen Ausgangskanal minus-schaltend das Puls-Pausen-Verhältnis kann in 1 ‰-Schritten angegeben werden

PWM: Beschreibung der Parameter

13833

Der FB PWM... enthält eine Reihe von Parametern. Hier erklären wir einige davon ausführlich.

PWM-Frequenz

2304

Abhängig vom Ventiltyp wird eine entsprechende PWM-Frequenz benötigt. Die PWM-Frequenz wird bei PWM1000 direkt als Zahlenwert in [Hz] übergeben.

Alle PWM-Kanäle verhalten sich gleich. Jeder PWM-Kanal kann unabhängig auf eine eigene Frequenz eingestellt werden. Die PWM-Frequenz liegt im Bereich 20...250 Hz.

PWM-Dither

2306

Bei bestimmten Hydraulikventiltypen muss die PWM-Frequenz zusätzlich von einer sogenannten Dither-Frequenz (Zitter-Frequenz) überlagert werden. Würden diese Ventile über einen längeren Zeitraum mit einem konstanten PWM-Wert angesteuert, so könnten sie sich durch die hohen Systemtemperaturen festsetzen.

Um dieses Blockieren zu verhindern, wird der PWM-Wert in Abhängigkeit von der Dither-Frequenz um einen festgelegten Wert (DITHER_VALUE) vergrößert oder verkleinert.

Die Folge ist: der konstante PWM-Wert wird überlagert...
- von einer Schwebung mit der Dither-Frequenz
- und der Amplitude DITHER_VALUE.

► Bei der Definition des Parameters DITHER_VALUE darauf achten, dass das resultierende PWM-Ratio im Arbeitsbereich der Regelung zwischen 0...1000 ‰ bleibt:
  - PWM-Ratio + DITHER_VALUE < 1000 ‰ und
  - PWM-Ratio - DITHER_VALUE > 0 ‰.

► Die Dither-Frequenz muss ein ganzzahliger Teil der PWM-Frequenz sein.
Wertebereich für Dither-Frequenz = 0...(PWM-Frequenz) / 2
Das Ergebnis von (PWM-Frequenz) / (Dither-Frequenz) muss geradzahlig sein!

4.1.2 Hydraulikregelung mit PWMi

Inhalt

19185
1559

Als Spezialgebiet der Stromregelung mit PWM (= PWMi) bietet **ifm electronic** dem Anwender spezielle Funktionen zur Regelung von Hydrauliksystemen.

Eine Hydraulik-Bibliothek ist derzeit nur für Controller verfügbar, jedoch nicht für CR04nn, CR253n.

Wozu diese Bibliothek? – Eine Einführung

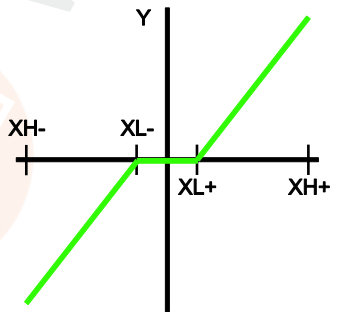
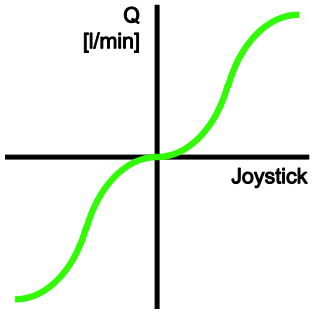
1560

Mit den Funktionen dieser Bibliothek können Sie folgende Aufgaben erfüllen:

Ausgangssignale von Joysticks normieren

1561

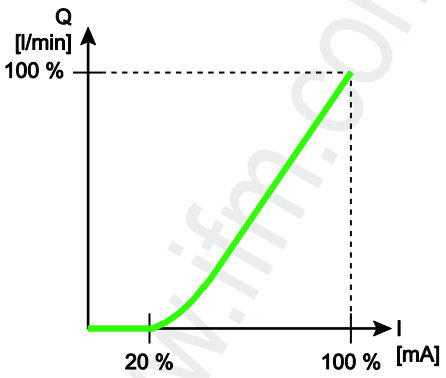
Nicht immer will man, dass sich der volle Bewegungsbereich des Joysticks auf die Maschinenbewegung auswirkt.

Oft soll der Bereich um die Neutralstellung des Joysticks herum ausgespart werden, weil der Joystick in der Neutralstellung nicht sicher 0 V liefert. Hier im Bild soll der Bereich zwischen XL- und XL+ ausgespart bleiben.	
Die FBs dieser Bibliothek ermöglichen Ihnen, die Kennlinie Ihres Joysticks nach Ihrem Bedarf anzupassen – auf Wunsch sogar frei parametrierbar:	

Hydraulikventile mit stromgeregelten Ausgängen ansteuern

1562

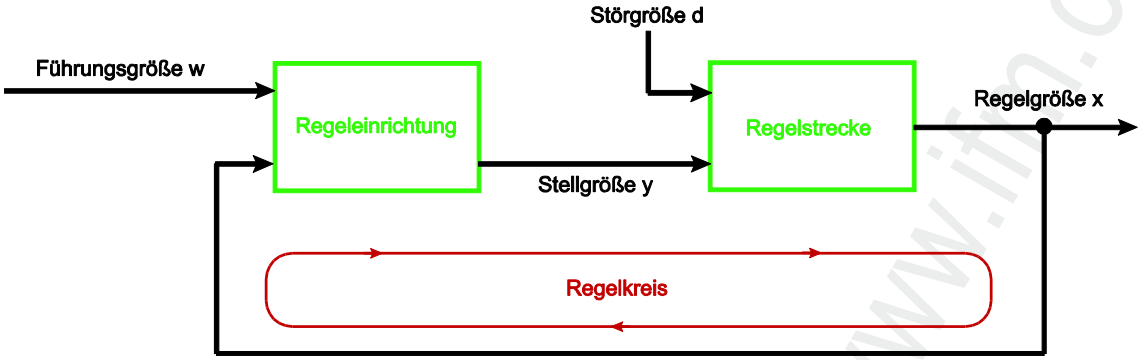
Hydraulikventile haben in der Regel keine völlig lineare Kennlinie:

Typischer Kennlinienverlauf eines Hydraulikventils: Erst bei ca. 20 % des Spulenstroms beginnt der Ölfluss. Der Ölfluss ist anfänglich nicht linear. Dies muss bei der Berechnung der Sollwerte für den Spulenstrom berücksichtigt werden. Die FBs aus dieser Bibliothek unterstützen Sie hierbei.	

4.2 Regler – Beschreibung

13830
1623

Die Regelung ist ein Vorgang, bei dem die zu regelnde Größe (Regelgröße x) fortlaufend erfasst und mit der Führungsgröße w verglichen wird. In Abhängigkeit vom Ergebnis dieses Vergleiches wird zur Angleichung an die Führungsgröße die Regelgröße beeinflusst.



Grafik: Prinzip einer Regelung

Die Auswahl einer geeigneten Regeleinrichtung und deren optimale Einstellung setzt genaue Angaben über das Beharrungsverhalten und das dynamische Verhalten der Regelstrecke voraus. In den meisten Fällen können diese Kennwerte aber nur experimentell ermittelt werden und sind kaum beeinflussbar.

Man kann drei Typen von Regelstrecken unterscheiden:

4.2.1 Regelstrecke mit Ausgleich

1624

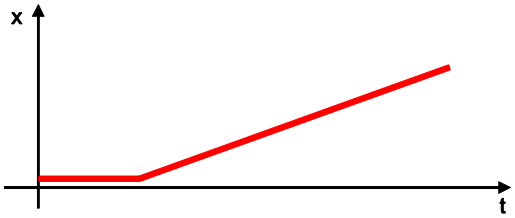
Bei einer Regelstrecke mit Ausgleich strebt die Regelgröße x nach einer bestimmten Stellgrößenänderung einem neuen Endwert (Beharrungszustand) zu. Entscheidend ist bei diesen Regelstrecken die Verstärkung (Übertragungsbeiwert KS). Je kleiner die Verstärkung ist, umso besser lässt sich die Strecke regeln. Man bezeichnet diese Regelstrecken als P-Systeme (P = proportional).

Grafik: P-Regler = Regelstrecke mit Ausgleich

4.2.2 Regelstrecke ohne Ausgleich

1625

Regelstrecken mit einem Verstärkungsfaktor gegen unendlich werden als Regelstrecken ohne Ausgleich bezeichnet. Dieses ist meistens auf ein integrierendes Verhalten zurückzuführen. Diese hat zur Folge, dass nach der Änderung der Stellgröße oder durch Einfluss einer Störgröße die Regelgröße stetig wächst. Durch dieses Verhalten erreicht sie nie einen Endwert. Man bezeichnet diese Regelstrecken als I-Systeme (I = integral).

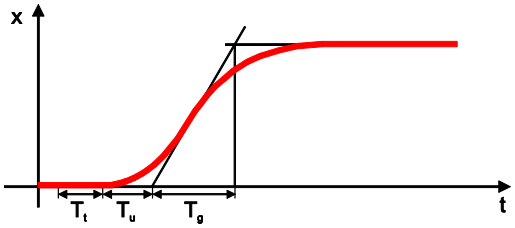


Grafik: I-Regler = Regelstrecke ohne Ausgleich

4.2.3 Regelstrecke mit Verzögerung

1626

Die meisten Regelstrecken entsprechen der Reihenschaltung von P-Systemen (Strecken mit Ausgleich) und einem oder mehreren T1-Systemen (Strecken mit Trägheit). Eine Regelstrecke 1. Ordnung entsteht z.B. durch die Reihenschaltung einer Drosselstelle und einem dahinter liegenden Speicher.



Grafik: PT-System = Regelstrecke mit Verzögerung

Bei Regelstrecken mit Totzeit reagiert die Regelgröße erst nach Ablauf der Totzeit T_t auf eine Veränderung der Stellgröße. Die Totzeit T_t bzw. die Summe aus $T_t + T_u$ ist das Maß für die Regelbarkeit der Strecke. Die Regelbarkeit einer Strecke ist umso besser, je größer das Verhältnis T_g/T_u ist.

Die Regler, die in die Bibliothek integriert sind, stellen eine Zusammenfassung der vorgestellten Grundfunktionen dar. Welche Funktionen zum Einsatz kommen und wie sie kombiniert werden, hängt von der jeweiligen Regelstrecke ab.

5 Arbeiten mit dem User-Flash-Speicher

Inhalt

11607

Einige **ifm**-Geräte bieten einen User-Flash-Speicher an. Dies ist ein Flash-Speicher-Bereich, der dem Kunden für seine Anwendungsdaten zur Verfügung steht.

Anwendungsbeispiele:
- Meldetexte (mehrsprachig umschaltbar) zur Anzeige in PDM und Display
- Belastungsgrenzwerte-Tabellen für z.B. Aufzüge, Krane und Drehleitern

Der Programmierer erstellt dazu Listen oder Tabellen.
Das hierfür verwendete Programm muss die Quelldatei in eine CSV-Datei wandeln können. Geeignet sind z.B. Tabellenkalkulationsprogramme wie Microsoft Excel oder OpenOffice Calc.

7317

HINWEIS

Eine CSV-Datei darf keine sicherheitsrelevanten Daten enthalten.
Hierfür sind keine geeigneten Sicherungsmaßnahmen vorhanden.

5.1 Flash-Speicher – was ist das?

11608

Flash-ROM (oder Flash-EPROM oder Flash-Memory) kombiniert die Vorteile von Halbleiterspeicher und Festplatten. Die Daten werden allerdings wie bei einer Festplatte blockweise in Datenblöcken zu 64, 128, 256, 1024, ... Byte zugleich geschrieben und gelöscht.

Vorteile von Flash-Speicher

- Die gespeicherten Daten bleiben auch bei fehlender Versorgungsspannung erhalten.
- Wegen fehlender beweglicher Teile ist Flash geräuschlos, unempfindlich gegen Erschütterungen und magnetische Felder.

Nachteile von Flash-Speicher

- Begrenzte Zahl von Schreib- bzw. Löschvorgängen, die eine Speicherzelle vertragen kann:
  - Multi-Level-Cells: typ. 10 000 Zyklen
  - Single-Level-Cells: typ. 100 000 Zyklen
- Da ein Schreibvorgang Speicherblöcke zwischen 16 und 128 kByte gleichzeitig beschreibt, werden auch Speicherzellen beansprucht, die gar keiner Veränderung bedürfen.

5.2 CSV-Datei – was ist das?

11627

CSV = **C**omma **S**eparated **V**alues (auch: **C**haracter **S**eparated **V**alues)
Eine CSV-Datei ist eine Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten.
Die Dateinamen-Erweiterung lautet `.csv`.

Beispiel: Quell-Tabelle mit Zahlenwerten:

Wert 1.0	Wert 1.1	Wert 1.2	Wert 1.3
Wert 2.0	Wert 2.1	Wert 2.2	Wert 2.3
Wert 3.0	Wert 3.1	Wert 3.2	Wert 3.3

Daraus entsteht folgende CSV-Datei:

```
Wert 1.0;Wert 1.1;Wert 1.2;Wert 1.3
Wert 2.0;Wert 2.1;Wert 2.2;Wert 2.3
Wert 3.0;Wert 3.1;Wert 3.2;Wert 3.3
```

Bitte beachten:

- Downloader und Maintenance-Tool erwarten zwischen den Spalten der Quell-Tabelle ein Trennzeichen, z.B. ein Semikolon (;).
- CODESYS erwartet als Endezeichen einer Zeichenkette (String) ein Null-Byte (NUL).
- Jeder Datensatz (jede zu übertragene Tabellenzeile) sollte die gleiche Anzahl von Tabellenspalten haben.

5.3 CSV-Datei und das ifm-Maintenance-Tool

Inhalt

11619

Folgende Geräte können mit dem ifm-Maintenance-Tool kommunizieren:
via AddIn BasicSystem
- BasicController: CR040n, CR041n, CR043n
- BasicDisplay: CR045n
- SmartController: CR253n

via Addin R360System
- Controller: CR0n3n, CR7n3n
- Controller: CR0020, CR0200, CR0505
- CabinetController: CR0303
- SmartController: CR2500

5.3.1 Voraussetzungen für die CSV-Datei

11630

► Die CSV-Datei muss eine bestimmte Kopfdaten-Struktur haben.
Alle Kopfdaten beginnen mit '#'.
  - 1. Zeile: CSV-Datei-Typ
    z.B.: `#File Type=0`
    zulässig: 0/1
  - 2. Zeile (optional): Projektname der CSV-Datei
    z.B.: `#Name=Demo Textmessages`
    zulässig: 0...20 Zeichen
  - 3. Zeile (optional): Version der CSV-Datei
    z.B.: `#Version=V01.00.00`
    zulässig: 0...12 Zeichen

- Das ifm-Maintenance-Tool kennt selbst die Startadresse des User-Flash-Speichers.
Die Adresse muss daher nicht in der CSV-Datei stehen.

► Direkt nach den Kopfdaten-Zeilen müssen lückenlos die Daten folgen!
Struktur:
relative Adresse;Datum oder Text;Datentyp;{Kommentar}
Beispiel:
31;Übertemperatur;string(20);Text 02
Nach dem Datentyp MUSS ein Semikolon (;) folgen!

- Das ifm-Maintenance-Tool erzeugt selbst aus dem Datentyp die richtige Datenlänge.
Die String-Daten müssen daher nicht mit der vollen Länge in der CSV-Datei stehen.

► Als Feld-Trenner gilt das Semikolon (;).

5.3.2 CSV-Datei erstellen mittels Tabellenkalkulationsprogramm

11639

Beispiel:

- Zweisprachige Meldetexte für BasicDisplay CR0451
- 20 Texte je bis zu 30 Zeichen in Deutsch
- 20 Texte je bis zu 30 Zeichen in Englisch

DemoFlashTextMessageD_E.xls

	A	B	C	D
1	#FileType=0			
2	#Name=Messages D/E CR0451			
3	#Version=V01.00.00			
4	0	Dies ist Text 1	String(30)	Text 01
5	31	Dies ist Text 2	String(30)	Text 02
6	62	Dies ist Text 3	String(30)	Text 03
7	93	Dies ist Text 4	String(30)	Text 04
8	124	Dies ist Text 5	String(30)	Text 05
9	155	Dies ist Text 6	String(30)	Text 06
10	186	Dies ist Text 7	String(30)	Text 07
11	217	Dies ist Text 8	String(30)	Text 08
12	248	Dies ist Text 9	String(30)	Text 09
13	279	Dies ist Text 10	String(30)	Text 10
14	310	Dies ist Text 11	String(30)	Text 11
15	341	Dies ist Text 12	String(30)	Text 12
16	372	Dies ist Text 13	String(30)	Text 13
17	403	Dies ist Text 14	String(30)	Text 14
18	434	Dies ist Text 15	String(30)	Text 15
19	465	Reserve	String(30)	Text 16
20	496	Reserve	String(30)	Text 17
21	527	Reserve	String(30)	Text 18
22	558	Reserve	String(30)	Text 19
23	589	Reserve	String(30)	Text 20
24	620	This is text 1	String(30)	Text 21
25	651	This is text 2	String(30)	Text 22
26	682	This is text 3	String(30)	Text 23
27	713	This is text 4	String(30)	Text 24
28	744	This is text 5	String(30)	Text 25
29	775	This is text 6	String(30)	Text 26
30	806	This is text 7	String(30)	Text 27
31	837	This is text 8	String(30)	Text 28
32	868	This is text 9	String(30)	Text 29
33	899	This is text 10	String(30)	Text 30
34	930	This is text 11	String(30)	Text 31
35	961	This is text 12	String(30)	Text 32
36	992	This is text 13	String(30)	Text 33
37	1023	This is text 14	String(30)	Text 34
38	1054	This is text 15	String(30)	Text 35
39	1085	reserve	String(30)	Text 36
40	1116	reserve	String(30)	Text 37
41	1147	reserve	String(30)	Text 38
42	1178	reserve	String(30)	Text 39
43	1209	reserve	String(30)	Text 40

Legende:

Feldnr.	Beschreibung
A1...A3	Tabellenkopf (Header); Einträge beginnen mit '#'
A1	#FileType= 0: Beim Übersetzen einer CSV-Datei werden die erfassten Parameter direkt in der Reihenfolge im User-Flash abgelegt 1: Beim Übersetzen einer CSV-Datei werden die erfassten Parameter so im User-Flash abgelegt, dass die Daten mittels CODESYS-Struktur direkt ausgelesen werden können.
A2	#Name= Name zum Definieren der Tabelle und zum Finden der Tabelle im Anwendungsprogramm Länge = 0...20 Zeichen
A3	#Version= Version der Tabelle (z.B. für verschiedene Fahrzeuge) Länge = 0...12 Zeichen
A4...A43	Byte-Nummer für Beginn eines Meldetextes
A4	erster Text der ersten Sprache (Byte-Nummer = 0)
A24	hier: erster Text der zweiten Sprache (Byte-Nummer = 620 = Offset)

Feldnr.	Beschreibung
B4...B23 B24...B43	Meldetexte (erste Sprache) dieselben Meldetexte (zweite Sprache) Nur diese Texte überträgt das ifm-Maintenance-Tool in das Gerät
C4...C43	Datentyp, hier: String(30)
D4...D43	Kommentare (optional) • nur zur Information beim Erstellen der Tabelle • Kommentare werden nicht in das Gerät übertragen

► Diese Struktur ist erforderlich, damit die daraus erzeugte CSV-Datei von CODESYS verstanden wird.

► Speichern Sie die Tabelle:
Speicherort wählen und Dateiname eintragen.
Wählen Sie dabei einen sinnvollen Dateinamen, um später die richtige Datei identifizieren zu können.

► Wandel Sie die Tabellen-Datei in eine CSV-Datei.
Wählen Sie das Semikolon ';' als Spaltentrenner.

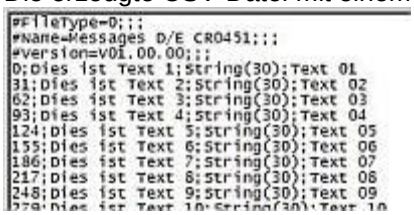
Bei **Excel**: [Speichern unter...] > [Dateityp] = CSV-Datei

Bei **OpenOffice**: [Speichern unter...] > [Dateityp] = Text CSV > [Format beibehalten]
im Fenster [Textexport] einstellen:
• Semikolon als Feldtrenner
• Texttrenner = (leer)

► Warnmeldungen (wegen Formatierungsverlust) quittieren.

► Schließen Sie das Tabellenkalkulations-Programm.

► Die erzeugte CSV-Datei mit einem Editor öffnen:



```
#FileType=0;;;
#Name=Messages D/E CR0451;;;
#version=V01.00.00;;;
0;Dies ist Text 1;String(30);Text 01
31;Dies ist Text 2;String(30);Text 02
62;Dies ist Text 3;String(30);Text 03
93;Dies ist Text 4;String(30);Text 04
124;Dies ist Text 5;String(30);Text 05
155;Dies ist Text 6;String(30);Text 06
186;Dies ist Text 7;String(30);Text 07
217;Dies ist Text 8;String(30);Text 08
248;Dies ist Text 9;String(30);Text 09
```

► Hinter den Header-Zeilen (beginnen mit '#') alle Semikola entfernen.

► Schließen Sie den Editor mit Speichern der Datei.

5.3.3 CSV-Datei erstellen mittels Editor

11640

► Gewünschten Inhalt der CSV-Datei manuell eintragen.

► Datei als CSV-Datei speichern.
Dateityp = ANSI

Beispiel:

- Zweisprachige Meldetexte für BasicDisplay CR0451
- 20 Texte je bis zu 30 Zeichen in Deutsch
- 20 Texte je bis zu 30 Zeichen in Englisch

```
#FileType=0
#Name=Messages D/E CR0451
#version=V01.00.00
0;Dies ist Text 1;String(30);
31;Dies ist Text 2;String(30);
62;Dies ist Text 3;String(30);
93;Dies ist Text 4;String(30);
124;Dies ist Text 5;String(30);
155;Dies ist Text 6;String(30);
186;Dies ist Text 7;String(30);
217;Dies ist Text 8;String(30);
248;Dies ist Text 9;String(30);
279;Dies ist Text 10;String(30);
310;Dies ist Text 11;String(30);
341;Dies ist Text 12;String(30);
372;Dies ist Text 13;String(30);
403;Dies ist Text 14;String(30);
434;Dies ist Text 15;String(30);
465;Reserve;String(30);
496;Reserve;String(30);
527;Reserve;String(30);
558;Reserve;String(30);
589;Reserve;String(30);
620;This is text 1;String(30);
651;This is text 2;String(30);
682;This is text 3;String(30);
713;This is text 4;String(30);
744;This is text 5;String(30);
775;This is text 6;String(30);
806;This is text 7;String(30);
837;This is text 8;String(30);
868;This is text 9;String(30);
899;This is text 10;String(30);
930;This is text 11;String(30);
961;This is text 12;String(30);
992;This is text 13;String(30);
1023;This is text 14;String(30);
1054;This is text 15;String(30);
1085;reserve;String(30);
1116;reserve;String(30);
1147;reserve;String(30);
1178;reserve;String(30);
1209;reserve;String(30);
```

Legende:

Feldnr.	Beschreibung
Zeilen 1...3	Tabellenkopf (Header); Einträge beginnen mit '#'
Zeile 1	#FileType= 0: Beim Übersetzen einer CSV-Datei werden die erfassten Parameter direkt in der Reihenfolge im User-Flash abgelegt 1: Beim Übersetzen einer CSV-Datei werden die erfassten Parameter so im User-Flash abgelegt, dass die Daten mittels CODESYS-Struktur direkt ausgelesen werden können.
Zeile 2 (optional)	#Name= Name zum Definieren der Tabelle und zum Finden der Tabelle im Anwendungsprogramm Länge = 0...20 Zeichen
Zeile 3 (optional)	#Version= Version der Tabelle (z.B. für verschiedene Fahrzeuge) Länge = 0...12 Zeichen
0; 31; 62; ...	Byte-Nummer für Beginn eines Meldetextes • Beginn mit der relativen Adresse 0 • folgende Adressen in Schritten von (Zeilenlänge plus 1) Bytes
;Text;	Meldetexte (oder Reserve) • jede Zeile ist exakt 20 Zeichen lang (mit beliebigen Zeichen auffüllen, hier: Punkte) • der Datentyp ergibt sich automatisch aus der Zeilenlänge • nur diese Daten werden nachher ins Gerät übertragen

Feldnr.	Beschreibung
String(30)	Datentyp Nach dem Datentyp MUSS ein Semikolon (;) folgen! Das ifm-Maintenance-Tool generiert aus der Angabe automatisch die richtige Länge der Meldetexte

5.3.4 CSV-Datei mit Maintenance-Tool übertragen

19189

► Verbinden Sie die Programmierschnittstelle des **ifm**-Geräts mit dem PC.
► Wenn noch nicht erfolgt, dann mit CODESYS das Anwendungsprogramm (als Bootprojekt) in das **ifm**-Gerät übertragen.
► Das ifm-Maintenance-Tool starten.

Z.B. für das BasicSystem:

► Wenn noch nicht geschehen, folgendes Menü herstellen:
[ecomat mobile] > [CAN] > [Basic System]
► Im Maintenance-Tool in der linken Spalte der Programmoberfläche folgendes Menü wählen:
[ecomat mobile] > [CAN] > [Basic System]
► In der mittleren Spalte der Programmoberfläche folgendes Menü wählen:
[Basic System] > [System-Information] > [Identität]
> Nach Klick auf [Lesen] erscheint im rechten Teil der Programmoberfläche die Geräteinformation.

Wenn die Daten des richtigen Geräts erscheinen:

► In der mittleren Spalte der Programmoberfläche folgendes Menü wählen:
[Basic System] > [Software] > [Laden]
► Im rechten Teil der Programmoberfläche klicken auf [nach Basic System]
► Im Feld [Software laden] klicken auf [*.csv-Datei importieren...].
> Das Fenster [Software laden] erscheint.
► Speicherort und Datei wählen und mit [Öffnen] bestätigen.
> Ein Informationsfenster erscheint mit folgenden Angaben:
- Speicherort, -Pfad
- Speicher: n Bytes von m Bytes verwendet
- Datei-Typ
- Name (aus den CSV-Kopfdaten, editierbar)
- Version (aus den CSV-Kopfdaten, editierbar)

► Mit [Importieren] die Datei in die Liste der zu übertragenden Dateien aufnehmen.
► Die zu übertragenden Dateien (oder: alle) markieren.
► Mit [Laden] die CSV-Datei in das **ifm**-Gerät übertragen.
> Ein Fortschrittsbalken zeigt an, wie weit der Vorgang bereits abgeschlossen ist.
> Abschließend erscheint eine Fertigmeldung.
► Das Gerät neu starten.

5.3.5 Zugriff auf die Flash-Daten: Bausteine

19190

Das Anwendungsprogramm kann nun mit folgenden Bausteinen auf die Daten zugreifen:

- **Für alle Controller (jedoch nicht für CR04nn, CR253n) gilt:**

FLASHREAD	liest unterschiedliche Datentypen direkt aus dem Flash-Speicher in den RAM
FLASHWRITE	schreibt unterschiedliche Datentypen direkt in den Flash-Speicher

- **Für alle CR04nn, CR1nnn, CR253n gilt:**

FLASH_INFO	liest die Informationen aus dem User-Flash-Speicher: • Name des Speicherbereichs (vom User vorgegeben), • Software-Version, • Startadresse (für einfaches Lesen mit IEC-Struktur)
FLASH_READ	liest unterschiedliche Datentypen direkt aus dem Flash-Speicher in den RAM

6 Visualisierungen im Gerät

Inhalt

3111

In diesem Kapitel finden Sie wichtige Informationen über Bitmap-Grafiken in CODESYS-Visualisierungen.

6.1 Grundsätzliches

10465
10464

Grundsätzlich können Sie neben den grafischen Elementen, die Sie mit dem CODESYS-Visualisierungs-Editor erstellen, auch Grafiken einbinden, die Sie mit anderen Programmen erstellt haben. Solche Grafikdateien können zum Beispiel Piktogramme, Logos oder auch kleine Bilder sein. Bevor Sie aber so eine "externe Grafik" einbinden, sind einige grundlegende Dinge zu beachten, die in den folgenden Kapiteln erläutert werden.

Weitere Hinweise finden Sie z.B. hier:

- Visualisierungen erstellen und parametrieren:
  → CODESYS-Programmierhandbuch (→ **ecomat*mobile***-DVD "Software, tools and documentation")
  → **ifm**-Lehrbuch "PDM – Handbuch zur Einführung"
- Beachten Sie die **Begrenzungen und Programmierhinweise**!

HINWEIS

► Das neue oder aus einem Template erzeugte Projekt sofort unter einen Projektnamen auf dem PC unter CODESYS speichern!

Wird ein Projekt ohne Dateiname auf das Gerät geladen, wird keine Visualisierung angezeigt. Dem Gerät fehlt ein Dateiname und deshalb kann die Visualisierung nicht starten.

6.2 Empfehlungen für Bedienoberflächen

Inhalt

7435

Entscheidend für die Akzeptanz und den Gebrauch von technischen Produkten ist in hohem Maß ihre Benutzerfreundlichkeit!

In diesem Kapitel geben wir einige Empfehlungen, wie die Benutzeroberfläche (auch **H**uman-**M**achine-**I**nterface HMI genannt) einer Maschine möglichst nutzerfreundlich zu gestalten ist.

6.2.1 Empfehlungen zur nutzerfreundlichen Produktgestaltung

7436

Alle wichtigen Schnittstellen zwischen Mensch und Maschine werden durch Oberfläche und Gestaltung bestimmt. Wichtigen Kriterien für Gestaltung von Schnittstellen zwischen Mensch und Maschine sind…

- Eindeutigkeit:
  - Für jede Funktion eine eindeutige Funktionsbeschreibung.
  - Erwartungskonforme Gestaltung, Erlerntes bleibt gleich
- Ablesbarkeit:
  - Umgebung (Beleuchtung, Lese-Abstand) berücksichtigen.
- Intuitive Bedienbarkeit:
  - Stellteil / Funktion muss erkennbar sein.
  - Bedienoberfläche muss sich selbst erklären.
- Sinnlichkeit
  - Bedienelemente müssen nutzerfreundlich sein.
  - Gute Unterscheidbarkeit von anderen Anzeigen und Bedienelementen.
- Feedback
  - Zeitnahe Reaktion auf Nutzer-Aktivitäten.
  - Ursache für eine Meldung muss eindeutig erkennbar sein.
- Umgebung des Produkts wegen Ablenkung oder Irritation durch...
  - Lärm
  - Dunkelheit
  - Lichtreflexe
  - Vibrationen
  - extreme Temperaturen

Aus Sicht des Herstellers ist zusätzlich wichtig:

- Anzeige als markenspezifisches Merkmal.
- Anzeige muss Standards und Normen erfüllen.

6.2.2 Kennen Sie die künftigen Nutzer?

7444

Die künftigen Nutzer des Produkts sollten bekannt sein:

- Alter
- Geschlecht
- Sinne:
  - Sehfähigkeit
  - Höhrfähigkeit
  - bevorzugte Hand (Rechts- oder Linkshänder)
  - Tastfähigkeit
- Ausbildung:
  - allgemeines Ausbildungsniveau
  - spezifische Schulungen und Erfahrungen
- Motivation und kognitive Fähigkeiten:
  - Wahrnehmen (Sinnesorgane): Nicht alle zur Verfügung stehenden Informationen werden genutzt, sondern massiv gefiltert, integriert und auf viele andere Weisen verändert, bevor sie ins Bewusstsein gelangen.
  - Denken: Das Arbeitsgedächtnis, in dem die geistige Manipulation von Informationen stattfindet, hat eine sehr kleine Kapazität.
  - Lernen: Die im Langzeitgedächtnis gespeicherten Informationen werden häufig sowohl im Voraus (z.B. durch Erwartungen), als auch im Nachhinein (z.B. durch nachfolgende Informationen) verändert.
  - Erinnern: Die im Langzeitgedächtnis "eigentlich" vorhandenen Informationen sind häufig nicht abrufbar.
  - Motivation und Konzentration: Müdigkeit, Lustlosigkeit, Ablenkbarkeit usw. können die kognitive Leistungsfähigkeit beeinträchtigen.
- Vertrautheit mit dem Problem oder Anwendungsgebiet:
  - Gefahren erkennen können
  - Wissen, was nach einer Bedienung geschehen soll
- Intensität der Anwendung (wie oft und wie intensiv wird das Produkt benutzt)
- Kulturkreis, z.B.:
  - Sprache
  - Bedeutung von Farben und Symbolen
  - Leserichtung

6.2.3 Gebrauchstauglichkeit prüfen

7422

In vielen Fällen kann eine Versuchsanordnung mit potentiellen Nutzern wichtige Ergebnisse liefern, wo und wie das Produkt verbessert werden soll/muss, um am Markt erfolgreich zu sein.

Für desen sogenannten "Usability-Test" müssen nacheinander folgende Schritte durchlaufen werden:

- Benutzergruppe (Zielgruppe) feststellen:
  - Wer soll mit dem Produkt umgehen können?
- Interview-Leitfaden erstellen:
  - Mit welcher Methode befrage ich welche Nutzer (Bediener, Einrichter, Wartungspersonal)?
  - Was will ich mit den Interviews erreichen? (Verbesserungspotentiale)
- Interviews durchführen und auswerten.
- Kontext-Szenarien verfassen:
  - Auswertbare Prüfumgebung erstellen.
  - Kritische Nutzungs-Szenarien identifizieren.
- Nutzungstest durchführen:
  - Wie kommen die Prüfpersonen mit dem Produkt in der Versuchsanordnung zurecht?
  - Wo ergibt sich welcher Korrekturbedarf am Produkt?
- Nach erfolgter Optimierung des Produkts bei Bedarf die Tests wiederholen.

6.2.4 Sprache als Hindernis

7454

Um Geräte zu produzieren, die weltweit die Endkunden zufrieden stellen, muss die Sprache berücksichtigt werden. Der Bediener kann seine Aufgaben nicht effektiv erledigen, wenn er die Anweisungen auf dem Bildschirm nicht versteht. Hersteller versuchen immer noch, dieses Problem angesichts der vielen verschiedenen Sprachen weltweit zu lösen. Einige Sprachen sind nachstehend aufgeführt:

Chinesische Zeichen

Das chinesische Schriftzeichen, auch bekannt als Han-Chinesisch, ist ein Wortzeichen, d.h. es kann als Wort dargestellt werden. Die Anzahl der Zeichen in dem Kangxi-Wörterbuch liegt über 47 000, doch in China reicht es aus, wenn drei- bis viertausend Zeichen bekannt sind. In der Neuzeit sind die chinesischen Schriftzeichen sehr vereinfacht worden und werden in Festlandchina verwendet, während die traditionellen chinesischen Schriftzeichen noch in Hongkong und Taiwan verwendet werden. Die Chinesischen Zeichen sind romanisiert worden. Diese werden Pinyin genannt und sind in China auch weit verbreitet.

Japanische Schriftzeichen

Das moderne japanische Schriftsystem verwendet drei Hauptschriften:

- Kanji sind Ideogramme aus chinesischen Schriftzeichen
- Hiragana wird verwendet für muttersprachliche japanische Wörter und
- Katakana wird verwendet für Lehnwörter
- Romanisierte japanische Zeichen, Romanji genannt, werden ebenfalls in japanischen Texten verwendet.

Koreanische Schriftzeichen

Das moderne koreanische Schriftsystem wird Hangul genannt und offiziell in Nord- und Südkorea verwendet. Daneben wird Hanja verwendet, das sich auf die dem Chinesischen entlehnten Zeichen bezieht.

Arabisches Alphabet

Diese Schrift wird verwendet, um mehrere Sprachen in Asien (z.B. Mittlerer und Naher Osten, Pakistan) und Afrika (z.B. Arabisch und Urdu) zu schreiben. Sie ist eine Schreibschrift von rechts nach links und umfasst 28 Buchstaben.

Unicode

Unicode ist ein Standard für die konsequente Darstellung und Verwendung von Zeichen, die in den weltweiten Schriftsystemen vorkommen. Es war nicht leicht, Sprachen an Computer anzupassen, teilweise wegen der großen Anzahl von Zeichen in einigen Sprachen. Es ist möglich, ein englisches Zeichen mit nur einem Byte zu kodieren, weil Schriftenglisch nur wenige Zeichen benötigt. Das gilt nicht für Sprachen wie Japanisch, Chinesisch oder Koreanisch, die über 256 Zeichen haben und somit eine Doppel- oder Multibyte-Kodierung erfordern. Mehrere Kodierverfahren werden verwendet und Unicode scheint das universellste Verfahren zu sein. Es kodiert offensichtlich in alle Sprachen der Welt.

Zum Beispiel handelt es sich bei der Han-Vereinheitlichung, die zu Unihan zusammengezogen wird, um ein Unterfangen von Unicode und Universal Character Set (nach ISO 10646), mehrere Zeichensätze des Chinesischen, Japanischen und Koreanischen in einen einzigen Satz vereinheitlichter Zeichen abzubilden.

Arabische Schriftzeichen können kodiert werden durch Unicode ab Version 5.0 (mehrere Zeichensätze nach ISO 8859-6).

ISO 10646 spezifiziert den Universal Multiple Octet Coded Character Set. Er wird angewendet für Darstellung, Austausch, Verarbeitung, Speicherung und Eingabe der schriftlichen Form der weltweiten Sprachen sowie für zusätzliche Symbole.

Die Unicode-Standard-Versionen 4...6 entsprechen alle ISO 10646.

Piktogramm

Dies ist ein grafisches Symbol, auch Bildzeichen genannt, das ein Konzept, Objekt, Ereignis oder eine Aktivität durch Abbildung darstellt. Piktogramme gibt es seit vielen tausend Jahren. Sie spielen immer noch eine wichtige Rolle bei Sprachbarrieren und Analphabetismus in der modernen Welt und werden als Bildzeichen, Repräsentationszeichen, Anweisungen oder statistische Diagramme verwendet. Aufgrund ihrer grafischen Darstellung werden sie in unterschiedlichen Lebensbereichen eingesetzt. Um zum Beispiel auf Toiletten und Flughäfen hinzuweisen, wird ein Standardsatz von Piktogrammen definiert in der ISO 7001 "Graphische Symbole zur Information der Öffentlichkeit".

Ein Piktogramm ist zu einer funktionellen visuellen Sprache für Leute mit kognitiven Schwierigkeiten entwickelt worden. Jedes Bild steht für ein Wort oder ein Konzept. Es enthält zwei Elemente, gezeichnete Bilder und Text. Die Symbole sind meistens weiß auf einem schwarzen Quadrat.

6.2.5 Kulturelle Details sind oft nicht übertragbar

7461

Länder-, kultur- oder sprachspezifische Details sollten im Ausgangstext vermieden werden, da ihre Verwendung oft unnötig und ihre Anpassung an die Zielkultur sehr zeitraubend ist. Meistens weiß der Autor nicht, dass seine Texte oder Grafiken kulturell oder sprachlich geprägt sind oder dass sie durch andere gestalterische Entscheidungen Lokalisierungsprobleme erzeugen. Probleme können z.B. in folgenden Bereichen entstehen:

- Farben
- Symbole
- Abbildungen
- Leserichtung

Farben

7464

Die Wahl der "richtigen" Farbe ist ein wichtiges Element bei der Gestaltung von Text und Produkt. Viele Farben sind kulturspezifisch belegt und können bei falscher Verwendung zu Missverständnissen und über Fehlbedienungen sogar zum Imageverlust des Produkts führen.

Beispiele:

	Farbe	Bedeutung in Europa + USA	Bedeutung in anderen Kulturen
	rot	Dramatik, Umbruch, Blut (Kampf, Rache und Tod), Liebe, Gefahr, Adel	China: Glück, fröhlich Russland: schön Ägypten: Tod Indien: Leben, kreativ Japan: Ärger, Gefahr
	gelb	Vorsicht, Warnung, Sonnenlicht, Ewigkeit, Neid, Hass	China: Geburt, Gesundheit, Kraft Ägypten: fröhlich, Besitz Indien: Erfolg Japan: Adel
	grün	Natur, Ökologie, Hoffnung, unsterblich, Glück	China: Ewigkeit, Familie, Harmonie, Gesundheit, Frieden, die Nachwelt Agypten: fruchtbar, Stärke Indien: Besitz, fruchtbar Japan: Zukunft, Jugend, Energie
	blau	Wasser, Himmel, Treue, Freiheit, beständig, Freude, Freundschaft, männlich	Asien: Reichtum, Stärke Ägypten: Tugend, Glaube, Wahrheit
	weiß	Licht, rein, weise, Leben, vollkommen, ideal, gut, sachlich, klar, unschuldig, ehrlich	Asien: Tod, Trauer, Reinheit Ägypten: Freude
	schwarz	Tod, Trauer, Finsternis, das Böse. Auch: Brüderlichkeit, Macht und Einigkeit	(Trauer nicht im Buddhismus) Ägypten: Auferstehung
	grau	Weisheit und Alter	Asien: hilfreich

Symbole

7465

Da Symbole oft in Analogie zu kulturspezifischen Konzepten entstehen oder Anspielungen auf vertraute Bereiche der Ausgangskultur nutzen, stellen sie ein Problem für die Lokalisierung dar.

Beispiel:

Das Symbol für ein Haus, das für Start oder Anfang stehen soll, ist nicht eindeutig verständlich, da sich die englische Benennung "home" nicht problemlos übertragen lässt.

Abbildungen

7466

Nicht immer kann ein Bild einen Text sinnvoll ersetzen.

Die Darstellung komplexerer Prozesse kann unmöglich werden. Denn wie soll z.B. die Abbildung für die Aufforderung aussehen "Drücken Sie die Taste, bis Sie einen leichten Widerstand spüren"?

Und selbst wenn eine Abbildung einen Sachverhalt gut darstellen kann, muss ihr Einsatz auf internationaler Ebene gut durchdacht werden. Das Ersetzen von Text durch Bilder ist nämlich nur dann sinnvoll und kostensenkend, wenn die Abbildungen kulturneutral, also in ALLEN angestrebten Zielländern ohne Anpassungen einsetzbar sind. Viele Dinge, die uns hier völlig selbstverständlich erscheinen, sind es in anderen Kulturen nicht.

Die Abbildung von Menschen kann zu Problemen führen: Welches Geschlecht soll oder darf die Person haben? Welche Hautfarbe? Welches Alter? Schließlich sollen sich die Adressaten in allen Zielländern gleichermaßen angesprochen fühlen. Kleidung, die in Westeuropa unauffällig ist, kann in arabischen oder afrikanischen Ländern zu Irritationen führen. Auch die Darstellung von Gesten und einzelnen Körperteilen, speziell von Händen und Augen, sollte unterbleiben, da diese oft eine anstößige oder beleidigende Assoziation auslösen.

Leserichtung

7468

In den meisten Kulturen wird von links nach rechts und von oben nach unten gelesen.

Einige asiatische Kulturen lesen jedoch von unten nach oben und von hinten nach vorn.

Viele arabische Kulturen lesen von rechts nach links.

Diese Besonderheiten sind auch bei rein grafischen Anleitungen zu beachten!

6.2.6 Richtlinien und Normen

Inhalt

7445

Die folgende Aufstellung ist nur eine Auswahl und erhebt keinen Anspruch auf Vollständigkeit.

ISO 7001 _ Graphische Symbole zur Information der Öffentlichkeit

7456

Ein grafisches Symbol, auch Bildzeichen genannt, stellt ein Konzept, Objekt, Ereignis oder eine Aktivität durch Abbildung dar. Piktogramme gibt es seit vielen tausend Jahren. Sie spielen immer noch eine wichtige Rolle bei Sprachbarrieren und Analphabetismus in der modernen Welt und werden als Bildzeichen, Repräsentationszeichen, Anweisungen oder statistische Diagramme verwendet. Aufgrund ihrer grafischen Darstellung werden sie in unterschiedlichen Lebensbereichen eingesetzt.

Beispiele:

ISO 9126 _ Qualitätsmerkmale für Software-Produkte

7446

Die Norm beschreibt folgende Kriterien:

Funktionalität: Inwieweit besitzt die Software die geforderten Funktionen?

- Angemessenheit: Eignung von Funktionen für spezifizierte Aufgaben, z. B. aufgabenorientierte Zusammensetzung von Funktionen aus Teilfunktionen.
- Richtigkeit: Liefern der richtigen oder vereinbarten Ergebnisse oder Wirkungen, z. B. die benötigte Genauigkeit von berechneten Werten.
- Interoperabilität: Fähigkeit, mit vorgegebenen Systemen zusammenzuwirken.
- Sicherheit: Fähigkeit, unberechtigten Zugriff (versehentlich oder vorsätzlich) auf Programme und Daten zu verhindern.
- Ordnungsmäßigkeit: Merkmale von Software, die bewirken, dass die Software anwendungsspezifische Normen oder Vereinbarungen oder gesetzliche Bestimmungen und ähnliche Vorschriften erfüllt.

Zuverlässigkeit: Kann die Software ein bestimmtes Leistungsniveau unter bestimmten Bedingungen über einen bestimmten Zeitraum aufrechterhalten?

- Reife: Geringe Versagenshäufigkeit durch Fehlerzustände.
- Fehlertoleranz: Fähigkeit, ein spezifiziertes Leistungsniveau bei Software-Fehlern oder Nicht-Einhaltung ihrer spezifizierten Schnittstelle zu bewahren.
- Robustheit: Fähigkeit, ein stabiles System bei Eingaben zu gewährleisten, die nicht vorgesehen sind. Die Software hält "DAUs" stand.
- Wiederherstellbarkeit: Fähigkeit, bei einem Versagen das Leistungsniveau wiederherzustellen und die direkt betroffenen Daten wiederzugewinnen. Zu berücksichtigen sind die dafür benötigte Zeit und der benötigte Aufwand.
- Konformität: Grad, in dem die Software Normen oder Vereinbarungen zur Zuverlässigkeit erfüllt.

Benutzbarkeit: Welchen Aufwand fordert der Einsatz der Software von den Benutzern und wie wird er von diesen beurteilt?

- Verständlichkeit: Aufwand für den Benutzer, das Konzept und die Anwendung zu verstehen.
- Erlernbarkeit: Aufwand für den Benutzer, die Anwendung zu erlernen (z. B. Bedienung, Ein-, Ausgabe).
- Bedienbarkeit: Aufwand für den Benutzer, die Anwendung zu bedienen.
- Attraktivität: Anziehungskraft der Anwendung gegenüber dem Benutzer.
- Konformität: Grad, in dem die Software Normen oder Vereinbarungen zur Benutzbarkeit erfüllt.

Effizienz: Wie liegt das Verhältnis zwischen Leistungsniveau der Software und eingesetzten Betriebsmitteln?

- Zeitverhalten: Antwort- und Verarbeitungszeiten sowie Durchsatz bei der Funktionsausführung.
- Verbrauchsverhalten: Anzahl und Dauer der benötigten Betriebsmittel bei der Erfüllung der Funktionen. Ressourcenverbrauch, wie CPU-Zeit, Festplattenzugriffe usw.
- Konformität: Grad, in dem die Software Normen oder Vereinbarungen zur Effizienz erfüllt.

Änderbarkeit: Welchen Aufwand erfordert die Durchführung vorgegebener Änderungen an der Software? Änderungen können Korrekturen, Verbesserungen oder Anpassungen an Änderungen der Umgebung, der Anforderungen oder der funktionalen Spezifikationen einschließen.

- Analysierbarkeit: Aufwand, um Mängel oder Ursachen von Versagen zu diagnostizieren oder um änderungsbedürftige Teile zu bestimmen.
- Modifizierbarkeit: Aufwand zur Ausführung von Verbesserungen, zur Fehlerbeseitigung oder Anpassung an Umgebungsänderungen.
- Stabilität: Wahrscheinlichkeit des Auftretens unerwarteter Wirkungen von Änderungen.
- Testbarkeit: Aufwand, der zur Prüfung der geänderten Software notwendig ist.

Übertragbarkeit: Wie leicht lässt sich die Software in eine andere Umgebung übertragen? Umgebung kann organisatorische Umgebung, Hardware- oder Software-Umgebung sein.

- Anpassbarkeit: Fähigkeit der Software, diese an verschiedene Umgebungen anzupassen.
- Installierbarkeit: Aufwand, der zum Installieren der Software in einer festgelegten Umgebung notwendig ist.
- Koexistenz: Fähigkeit der Software neben einer anderen mit ähnlichen oder gleichen Funktionen zu arbeiten.
- Austauschbarkeit: Möglichkeit, diese Software anstelle einer spezifizierten anderen in der Umgebung jener Software zu verwenden, sowie der dafür notwendige Aufwand.
- Konformität: Grad, in dem die Software Normen oder Vereinbarungen zur Übertragbarkeit erfüllt.

ISO 9241 _ Ergonomie der Mensch-System-Interaktion

7447

Die Norm ISO 9241 ist ein internationaler Standard, der Richtlinien der Interaktion zwischen Mensch und Computer beschreibt. Die Normenreihe beschreibt Anforderungen an die Arbeitsumgebung, Hardware und Software. Ziel der Richtlinie ist es, gesundheitliche Schäden beim Arbeiten am Bildschirm zu vermeiden und dem Benutzer die Ausführung seiner Aufgaben zu erleichtern.

Die folgenden Teile (jedoch nicht ausschließlich) sind Bestandteile der Norm:

Teil 1: Allgemeine Einführung
Teil 2: Anforderungen an die Arbeitsaufgaben – Leitsätze
Teil 3: Anforderungen an visuelle Anzeigen
Teil 4: Anforderungen an Tastaturen
Teil 5: Anforderungen an die Arbeitsplatzgestaltung und Körperhaltung
Teil 6: Anforderungen an die Arbeitsumgebung
Teil 7: Anforderungen an visuelle Anzeigen bezüglich Reflexionen
Teil 8: Anforderungen an Farbdarstellungen
Teil 9: Anforderungen an Eingabegeräte - außer Tastaturen
(Teil 10: Grundsätze der Dialoggestaltung (veraltet, da seit 2006 ersetzt durch Teil 110))
Teil 11: Anforderungen an die Gebrauchstauglichkeit – Leitsätze
Teil 12: Informationsdarstellung
Teil 13: Benutzerführung
Teil 14: Dialogführung mittels Menüs
Teil 15: Dialogführung mittels Kommandosprachen
Teil 16: Dialogführung mittels direkter Manipulation
Teil 17: Dialogführung mittels Bildschirmformularen
Teil 110: Grundsätze der Dialoggestaltung (ersetzt den bisherigen Teil 10)
Teil 151: Leitlinien zur Gestaltung von Benutzungsschnittstellen für das World Wide Web
Teil 171: Leitlinien für die Zugänglichkeit von Software (im Oktober 2008 veröffentlicht)
Teil 300: Einführung in Anforderungen und Messtechniken für elektronische optische Anzeigen
Teil 302: Terminologie für elektronische optische Anzeigen (zurzeit im Entwurfsstadium)
Teil 303: Anforderungen an elektronische optische Anzeigen (zurzeit im Entwurfsstadium)
Teil 304: Prüfverfahren zur Benutzerleistung
Teil 305: Optische Laborprüfverfahren für elektronische optische Anzeigen (zurzeit im Entwurfsstadium)
Teil 306: Vor-Ort-Bewertungsverfahren für elektronische optische Anzeigen (zurzeit im Entwurfsstadium)
Teil 307: Analyse und Konformitätsverfahren für elektronische optische Anzeigen (zurzeit im Entwurfsstadium)
Teil 400: Grundsätze und Anforderungen für physikalische Eingabegeräte
Teil 410: Gestaltungskriterien für physikalische Eingabegeräte (zurzeit im Entwurfsstadium)

Die Teile 5 und 6 umfassen den Themenbereich Arbeitsumgebung. Die Teile 3, 4, 7, 8 und 9 beschäftigen sich mit Anforderungen an Hardware, während die Teile 11...17 und 110 Aspekte der Software-Ergonomie behandeln. Vor allem in den Teilen **ISO 9241-110 _ Grundsätze der Dialoggestaltung** (→ S. 134) und **ISO 9241-11 _ Anforderungen an die Gebrauchstauglichkeit** (→ S. 134) finden sich einige Kriterien für die ergonomische Gestaltung interaktiver Systeme.

ISO 9241-11 _ Anforderungen an die Gebrauchstauglichkeit

7448

Die Gebrauchstauglichkeit einer Software ist von ihrem Nutzungskontext abhängig. Im Teil 11 der ISO 9241 werden drei Leitkriterien für die Gebrauchstauglichkeit einer Software bestimmt:

- Effektivität zur Lösung einer Aufgabe,
- Effizienz der Handhabung des Systems,
- Zufriedenheit der Nutzer einer Software.

ISO 9241-110 _ Grundsätze der Dialoggestaltung

7450

Benutzungsschnittstellen von interaktiven Systemen, wie Webseiten oder Software, sollten vom Benutzer leicht zu bedienen sein. Der Teil 110 der ISO 9241 beschreibt folgende Grundsätze für die Gestaltung und Bewertung einer Schnittstelle zwischen Benutzer und System (Dialoggestaltung):

- Aufgabenangemessenheit
  geeignete Funktionalität, Minimierung unnötiger Interaktionen
- Selbstbeschreibungsfähigkeit
  Verständlichkeit durch Hilfen / Rückmeldungen
- Lernförderlichkeit
  Anleitung des Benutzers, Verwendung geeigneter Metaphern, Ziel: minimale Erlernzeit
- Steuerbarkeit
  Steuerung des Dialogs durch den Benutzer
- Erwartungskonformität
  Konsistenz, Anpassung an das Benutzermodell
- Individualisierbarkeit
  Anpassbarkeit an Benutzer und an seinen Arbeitskontext
- Fehlertoleranz
  Intelligente Dialoggestaltung zur Fehlervermeidung seitens der Benutzer steht an erster Stelle.
  Ansonsten: erkannte Fehler des Benutzers verhindern nicht das Benutzerziel.
  Unerkannte Fehler: leichte Korrektur durch den Benutzer.

ISO 10646 _ Informationstechnik – Universeller Mehrfach-8-bit-codierter Zeichensatz (UCS)

7455

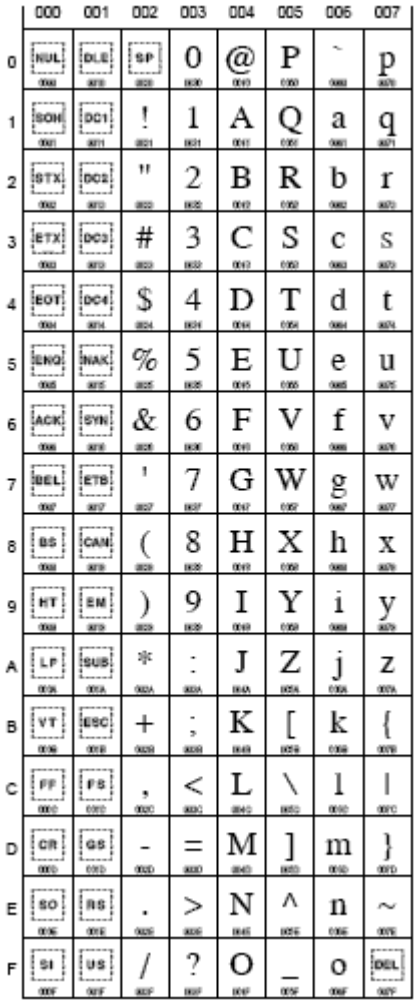
Der Universal Character Set (UCS) ist eine Zeichenkodierung, die im internationalen Standard ISO 10646 definiert ist. Für alle praktischen Belange ist dies dasselbe wie Unicode.

Pro Zeichen werden 2 Byte Speicherplatz verwendet. Entsprechend ist Unicode ein 16-Bit-Code, mit dem man 2^{16} = 65 536 Zeichen repräsentieren kann. Erstes Ziel ist, die Schriftzeichen aller Staatssprachen eindeutig und einheitlich zu kodieren.

Nicht alle dieser 65 536 Zeichenadressen werden dabei standardisiert belegt: Ein nutzerdefinierter Bereich erlaubt es, ca. 2 000 Adressen mit nutzerspezifischen Zeichen zu belegen.

Über die Kombination von zwei 16-Bit-Codes kann man weitere 1 408 576 Zeichen ansprechen. Damit hofft man, alle Schriftzeichen erfassen zu können, die es gibt und jemals gegeben hat. Darüberhinaus werden auch technische Symbole, musikalische Zeichen, Lautschrift etc. abgebildet. Noch sind aber bei weitem nicht alle Zeichenadressen belegt.

Beispiele:



Unicode: Steuerzeichen und Basiszeichen

Unicode: Pfeile

ISO 13406 _ Ergonomische Anforderungen für Tätigkeiten an optischen Anzeigeeinheiten in Flachbauweise

7453

Teil 2: Ergonomische Anforderungen an Flachbildschirme

Gemäß der internationalen Norm ISO 13406-2 werden LCD-Bildschirme nach folgenden Kriterien klassifiziert:

- Leuchtdichte, Kontrast und Farbe gemessen an der Blickrichtung des Betrachters
- Reflexionen und Kontrast bei einfallender Beleuchtung
- Bildaufbauzeit
- Defekte (Pixelfehler)

ISO 13407 _ Benutzer-orientierte Gestaltung interaktiver Systeme

7452

Die ISO 13407 ist eine Norm, die einen prototypischen benutzerorientierten Softwareentwicklungsprozess beschreibt. Ein spezieller Entwicklungsprozess kann als zu ihr konform betrachtet werden, wenn ihre Empfehlungen erfüllt werden.

Die Norm stellt nutzerorientierte Gestaltung als eine fachübergreifende Aktivität dar, die Wissen über menschliche Faktoren und ergonomische Kenntnisse und Techniken umfasst. Der ISO-Prozess besteht aus vier wesentlichen Teilaktivitäten:

- Nutzungskontext verstehen:
  Das Ergebnis dieser Aktivität ist eine dokumentierte Beschreibung der relevanten Benutzer, ihrer Arbeitsaufgaben und ihrer Umgebung.
- Anforderungen spezifizieren:
  Während dieser Phase werden die Zielgrößen aus der bereits vorhandenen Dokumentation auf einer Kompromissebene abgeleitet. Dabei wird die Teilung der Systemaufgaben bestimmt in...
  - solche, die von Menschen durchgeführt werden sollen
  - solche, die von der Technik durchgeführt werden sollen.
- Lösungen produzieren:
  Dies kann im Sinne einer Prototyp-Entwicklung oder eines anderen iterativen Prozesses erfolgen. Diese Prototypen können noch reine Papierentwürfe (Attrappen) oder aber schon lauffähige Programmversionen sein. Falls es unternehmensinterne Gestaltungsregeln für Benutzerschnittstellen gibt, sollten diese genutzt werden.
- Lösungen bewerten:
  Die Lösungen werden auf die Erfüllung der festgelegten Anforderungen geprüft. Dazu können Experten-Bewertungen, Gebrauchstauglichkeitstests (Usability-Tests), Befragungen oder auch eine Mischung daraus dienen. Die dabei entdeckten Abweichungen werden dann auf ihre Relevanz hin bewertet und sind Ausgangspunkt der nächsten Iteration des Entwicklungsprozesses.

Dieses Verfahren ist komplementär zu bestehenden Prozessmodellen der Software-Entwicklung und ergänzt diese. Der benutzerorientierte Gestaltungsprozess sollte der Norm zufolge bereits im frühesten Stadium des Projekts beginnen und sollte dann wiederholt durchlaufen werden, bis das System die Anforderungen erfüllt. Die Bedeutung und der Aufwand für die benutzerorientierte Gestaltung misst sich an der Größe und Art des zu entwickelnden Produkts und wird für kleinere Projekte durch Einzelpersonen gesteuert.

ISO 20282 _ Bedienungsfreundlichkeit von Produkten des täglichen Gebrauchs

7443

Dieser Normenentwurf besteht aus:

- Teil 1: Gebrauchsumfeld und Benutzerkriterien
  Beschreibt folgende Kriterien:
  - den Anwendungsbereich
  - die Benutzerschnittstelle
  - den Nutzer
  - seine psychischen und sozialen Charakteristika
  - die physische und soziale Umgebung
  - die physische und sensorische Kategorie.
- Teil 2: Prüfverfahren für öffentlich zugängliche Produkte
  Definiert als Technische Spezifikation die Prüfverfahren.

6.3 Grundlegende Informationen zu Farben und Bitmap-Grafiken

Inhalt

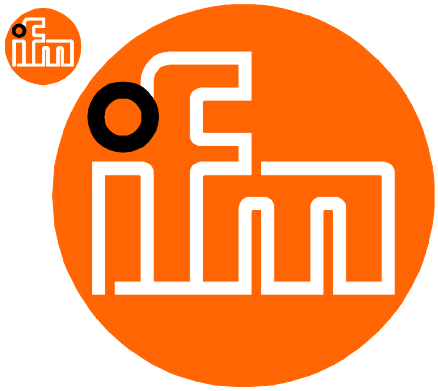
3112

Bei Grafiken und Bilddateien unterscheidet man vereinfacht zwei grundsätzliche Typen:

	Vektorgrafiken	**Pixelgrafiken**
Beispiele:	Zeichnungen von CAD-Programmen. Zeichensätze vom Typ TrueType, PostScript oder OpenType.	Digitalfotos. Dateien aus dem Scanner oder aus Capture-Programmen.
Prinzip:	Vektorgrafiken basieren auf einer Bildbeschreibung, die die Objekte, aus denen das Bild aufgebaut ist, exakt definiert. Z.B. ein Kreis wird definiert über Lage (Koordinaten) des Mittelpunktes, Radius, Linienstärke und Farbe.	Eine Rastergrafik, auch Pixelgrafik oder Bitmap, ist eine Form der Beschreibung eines Bildes, bestehend aus einer rasterförmigen Anordnung von so genannten Pixeln (Bildpunkten), denen jeweils eine Farbe zugeordnet ist. Die Hauptmerkmale einer Rastergrafik sind daher die Breite und die Höhe in Pixeln (Bildauflösung) sowie die Farbtiefe.
Speicherbedarf:	Speicherbedarf relativ gering.	Je nach Auflösung ist Speicherbedarf hoch bis sehr hoch: Die Dateien werden mit jedem zusätzlich zu speichernden Bildpunkt immer größer.
Verluste beim Skalieren:	Verlustfreie Umrechnung (skalieren) in beliebige Bildgrößen möglich.	Umrechnung (skalieren) in andere Bildgrößen meist nur mit Qualitätsverlusten möglich.
Leistungsfähigkeit der Hardware:	Da Monitore grundsätzlich auf einer Raster-Matrix basieren, müssen alle Grafiken in einzelne Bildpunkte umgerechnet (= gerastert) werden, um sie auf dem Monitor anzeigen zu können. Je nach Komplexität der Grafik sehr leistungsfähige Rechner erforderlich, um eine schnelle Bearbeitung und Anzeige zu ermöglichen.	Anforderung relativ gering.
Typische Datei-Endungen:	*.cdr (Corel Draw) *.dwg (AutoCAD) *.ai (Adobe Illustrator) *.svg (Scalable Vector Graphics)	*.bmp (Bitmap) *.gif (Compuserv GIF) *.jpg (Joint Photographic Experts Group) *.png (Portable Network Graphics)

6.3.1 Bildgröße Vektorgrafik / Pixelgrafik

7380

Vektorgrafiken	Pixelgrafiken
Grafische Elemente werden als Vektoren beschrieben: Informationen über Start- und Endpunkt, Dicke und Farbe einer Linie, ggf. Füllmuster und Farbverlauf.	Pixelgrafiken aus modernen Digitalkameras haben 5 Millionen und mehr Bildpunkte (Auflösung = 5 Megapixel). Spezielle Datenkompression versucht, den hohen Speicherbedarf zu mindern. Leider arbeitet die Kompression nur mit Qualitätsverlust.
Vergrößern oder Verkleinern erfolgt einfach und ohne Qualitätsverluste (→ Beispiel unten).	Beim Vergrößern entstehen entweder Klötzchen-Grafiken oder verschwommene Bilder (→ Beispiel unten). Einen hohen Verlust an Bildinformationen hat man beim Verkleinern eines solchen Megapixel-Bildes.
Beispiel:  Original Ø 10 mm / Vergrößerung 5-fach EPS-Datei jeweils 35 kB	Beispiel:  Original 30x30 px / Vergrößerung 5-fach BMP-Datei 3 kB / 62 kB

Beispiel: Verkleinern eines Pixelbildes für CR108n

19193
7402

Aufgabe: Ein vorhandenes Digitalfoto mit einer Auflösung von 5 Megapixeln hat z.B. eine Bildgröße von 2 560 x 1 920 Bildpunkten (= 4 915 200 Pixel). Dieses Foto soll nun in einer Bildgröße von nur 800 x 480 Bildpunkten (= Monitorgröße beim PDM360NG) dargestellt werden.

Problem 1: Das Verhältnis Seite zu Höhe beträgt für die Quelle 4:3 (1,33:1) aber für das Ziel 15:9 (1,66:1).

Lösung (anisotropisch): Höhe und Seite des Bildes in verschiedenen Maßstäben skalieren, um das Bild auf dem Display unverzerrt darzustellen.
Bei gleichmäßiger (isotropischer) Skalierung wird das Bild gegenüber dem Original verzerrt.

Problem 2: Nach dem Skalieren sind nur noch 384 000 Bildpunkte (= 7,8 % des ursprünglichen Bildes) verblieben, die anderen 4 531 200 Pixel entfallen ersatzlos. Oder anders ausgedrückt: Waagerecht wird etwa nur jedes 3. Pixel verwendet, senkrecht sogar nur jedes 4. Pixel.

Daher kann ein so gewandeltes Foto nicht mehr die Qualität des Originals besitzen. Wichtige Informationen gehen verloren und das Bild wird verzerrt dargestellt.

► **Abhilfe:** Bilder von Anfang an in der benötigten Größe und Auflösung anfertigen.

Für andere Geräte mit anderen Monitorgrößen gilt das Problem entsprechend.

Bitmap-Grafiken anpassen

9996

Vorhandene Bitmap-Grafiken können Sie mit gängigen Grafikprogrammen anpassen.
Fragen Sie Ihren **ecomat*mobile***-Fachberater!

6.3.2 Farbe bei Bitmap-Grafiken

3121

Ein zweiter wichtiger Faktor ist die Farbinformation (der RGB-Wert), der für jeden Bildpunkt gespeichert wird.

RGB steht für Rot, Grün und Blau. Für jede dieser drei Grundfarben stehen 255 Intensitätsstufen zur Verfügung. Durch Mischen dieser drei Grundfarben in unterschiedlichen Intensitäten entstehen über die so genannte **Additive Farbmischung** (→ S. 141) ca. 16,6 Millionen Farben. Um diese Menge darzustellen, benötigt man geeignete Monitore und leistungsfähige Rechner.

6.3.3 Welche Farben werden dargestellt?

19194

Farben im CR108n

19195
7381

Das Display kann eine Farbtiefe von 6 Bit je Grundfarbe darstellen, also jeweils 64 Farbstufen. Aus dem Gesamtspektrum von 256 adressierbaren Farbstufen kann folglich nur jede vierte verwendet werden:

Farbe		zulässige Farbwerte
Rot		R = 0, 4, 8, 12, 16, ..., 236, 240, 244, 248, 252
Grün		G = 0, 4, 8, 12, 16, ..., 236, 240, 244, 248, 252
Blau		B = 0, 4, 8, 12, 16, ..., 236, 240, 244, 248, 252

Werte, die nicht in das Schema passen, werden nicht dargestellt.

Farben im CR045n

19196
8367

Das Gerät kann eine Farbtiefe von 8 Bit darstellen, also insgesamt 256 Farbstufen. Aus dem Gesamtspektrum von 256 adressierbaren Farbstufen je Farbkanal (= 16 777 216 Farben) kann folglich nur jede 65 536. verwendet werden:

Farbe		zulässige Farbwerte
Rot		R (3 Bit = 8 Abstufungen) = 0, 32, 64, 96, 128, 160, 192, 224
Grün		G (3 Bit = 8 Abstufungen) = 0, 32, 64, 96, 128, 160, 192, 224
Blau		B (2 Bit = 4 Abstufungen) = 0, 64, 128, 192

Die Farbpalette wurde werksseitig festgelegt und ist fest im Gerät hinterlegt.

Werte, die nicht in das Schema passen, werden nicht dargestellt.

6.4 Spezielle Informationen zu Bitmap-Grafiken

Inhalt

3113

Für den interessierten Leser gibt es hier tiefergehende Informationen über Bitmap-Grafiken.

6.4.1 Additive Farbmischung

3123

RGB	Mischfarben
Monitore und viele Drucker stellen Mischfarben aus den 3 Grundfarben Rot, Grün und Blau zusammen.	Mischfarben entstehen durch Addieren der Farben im benötigten Mischungsverhältnis. Diese Methode heißt deshalb additive Farbmischung.

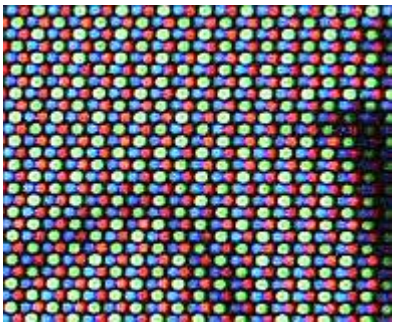
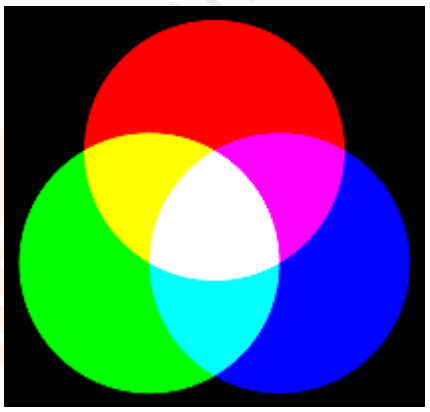

	

Bild: RGB-Raster eines Monitors, stark vergrößert dargestellt	Bild: Additive Farbmischung

Tabelle: Beispiele von Farbmischungen

100 % Rot		+ 100 % Grün		= 100 % Gelb	
100 % Grün		+ 100 % Blau		= 100 % Cyan	
100 % Blau		+ 100 % Rot		= 100 % Magenta	

Abstufungen in der Farbsättigung ergeben sich durch geringere Anteile der jeweiligen Grundfarbe:

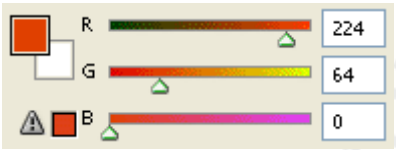


Bild: RGB-Farbmischung bei Photoshop; 100 % ⇨ 255_{dez} = FF_{hex}

6.4.2 Welche Grafiken sind für das Gerät geeignet und welche Schritte muss man durchführen?

7387

Nicht alle Bitmaps sind für die Anzeige auf dem Gerät geeignet.

- Grundsätzlich sollten Fotos so gewandelt werden, dass sie bei der gegebenen Auflösung und Farbtiefe optimal dargestellt werden.
- Bilder mit einem geringen Kontrastumfang sind nicht geeignet, da sich die Farbunterschiede auf dem Gerät nicht darstellen lassen.
- Logos und Symbole sollten bei Bedarf für die Darstellung auf dem Gerät optimiert oder neu gezeichnet werden.

7 Übersicht der verwendeten Dateien und Bibliotheken

Inhalt

2711

(Stand: 25.06.2014)

Je nach Gerät und gewünschter Funktion kommen verschiedene Bibliotheken und Dateien zum Einsatz. Teilweise werden sie automatisch geladen oder müssen vom Programmierer eingefügt oder geladen werden.

7.1 Allgemeine Übersicht

2712

Dateiname	Beschreibung und Speicherort [1])
ifm_CRnnnn_Vxx.CFG	Steuerungskonfiguration je Gerät nur 1 gerätespezifische Datei enthält: IEC- und symbolische Adressen der Ein- und Ausgänge, der Systemmerker sowie die Speicherverteilung …\CoDeSys V*\Targets\ifm\ifm_CRnnnncfg\Vxxyyzz
CAA-*.CHM	Online-Hilfe je Gerät nur 1 gerätespezifische Datei enthält: Online-Hilfe zu diesem Gerät …\CoDeSys V*\Targets\ifm\Help\… (Sprache)
ifm_CRnnnn_Vxxyyzz.H86 ifm_CRnnnn_Vxxyyzz.RESX	Laufzeitsystem (muss bei Erstbenutzung in den Controller / Monitor geladen werden) je Gerät nur 1 gerätespezifische Datei …\CoDeSys V*\Targets\ifm\Library\ifm_CRnnnn
ifm_Browser_CRnnnn.INI	CODESYS-Bowser-Kommandos (CODESYS benötigt die Datei zum Projektstart) je Gerät nur 1 gerätespezifische Datei enthält: Kommandos für Browser in CODESYS …\CoDeSys V*\Targets\ifm
ifm_Errors_CRnnnn.INI	CODESYS-Fehler-Datei (CODESYS benötigt die Datei zum Projektstart) je Gerät nur 1 gerätespezifische Datei enthält: gerätespezifische Fehlermeldungen aus CoDeSys …\CoDeSys V*\Targets\ifm
ifm_CRnnnn_Vxx.TRG	Target-Datei je Gerät nur 1 gerätespezifische Datei enthält: Hardware-Beschreibung für CODESYS, z.B.: Speicher, Dateiablageorte …\CoDeSys V*\Targets\ifm
ifm_*_Vxxyyzz.LIB	allgemeine Bibliotheken je Gerät mehrere Dateien möglich …\CoDeSys V*\Targets\ifm\Library
ifm_CRnnnn_Vxxyyzz.LIB	gerätespezifische Bibliothek je Gerät nur 1 gerätespezifische Datei enthält: Programmbausteine dieses Geräts …\CoDeSys V*\Targets\ifm\Library\ifm_CRnnnn
ifm_CRnnnn_*_Vxxyyzz.LIB	gerätespezifische Bibliotheken je Gerät mehrere Dateien möglich → folgende Tabellen …\CoDeSys V*\Targets\ifm\Library\ifm_CRnnnn

Legende:

*	beliebige Zeichen
CRnnnn	Artikelnummer des Geräts
V*	CODESYS-Version
Vxx	Versionsnummer der ifm-Software
yy	Release-Nummer der ifm-Software
zz	Patch-Nummer der ifm-Software

[1]) Speicherort der Dateien:
System-Laufwerk (C: / D:) \ Programme-Ordner \ ifm electronic

7.2 Wozu dienen die einzelnen Dateien und Bibliotheken?

Inhalt

2713

Die nachfolgende Übersicht zeigt, welche Dateien/Bibliotheken mit welchem Gerät eingesetzt werden können und dürfen. Dateien/Bibliotheken, die in dieser Liste nicht aufgeführt werden, können nur unter bestimmten Bedingungen eingesetzt werden oder die Funktionalität wurde noch nicht getestet.

7.2.1 Dateien für Laufzeitsystem

2714

Dateiname	Funktion	verfügbar für:
`ifm_CRnnnn_Vxxyyzz.H86` `ifm_CRnnnn_Vxxyyzz.RESX`	Laufzeitsystem	• alle ecomat*mobile*-Controller • BasicDisplay: CR045n • PDM: CR1nnn
`ifm_Browser_CRnnnn.INI`	CODESYS-Browser-Kommandos	• alle ecomat*mobile*-Controller • PDM: CR1nnn
`ifm_Errors_CRnnnn.INI`	CODESYS-Fehler-Datei	• alle ecomat*mobile*-Controller • PDM: CR1nnn

7.2.2 Target-Datei

2715

Dateiname	Funktion	verfügbar für:
`ifm_CRnnnn_Vxx.TRG`	Target-Datei in CODESYS	• alle ecomat*mobile*-Controller • BasicDisplay: CR045n • PDM: CR1nnn

7.2.3 Steuerungskonfigurations-Datei

2716

Dateiname	Funktion	verfügbar für:
`ifm_CRnnnn_Vxxyyzz.CFG`	Steuerungskonfiguration in CODESYS	• alle ecomat*mobile*-Controller • BasicDisplay: CR045n • PDM: CR1nnn

7.2.4 ifm-Gerätebibliotheken

2717

Dateiname	Funktion	verfügbar für:
`ifm_CRnnnn_Vxxyyzz.LIB`	gerätespezifische Bibliothek	• alle ecomat*mobile*-Controller • BasicDisplay: CR045n • PDM: CR1nnn
`ifm_CR0200_MSTR_Vxxyyzz.LIB`	Bibliothek ohne Extended-Funktionen	• ExtendedController: CR0200
`ifm_CR0200_SMALL_Vxxyyzz.LIB`	Bibliothek ohne Extended-Funktionen, reduzierter Funktionsumfang	• ExtendedController: CR0200

7.2.5 ifm-CANopen-Hilfsbibliotheken Master/Slave

2718

Diese Bibliotheken setzen auf CODESYS-Bibliotheken (3S-CANopen-Funktionen) auf und stellen die Funktionen dem Anwender übersichtlich zur Verfügung.

Dateiname	Funktion	verfügbar für:

`ifm_CRnnnn_CANopenMaster_Vxxyyzz.LIB`
`ifm_CRnnnn_CANopen`**`x`**`Master_Vxxyyzz.LIB`
`ifm_CRnnnn_CAN`**`x`**`openMaster_Vxxyyzz.LIB`
	CANopen Master Emergency- und Status-Handler	• alle ecomat*mobile*-Controller *)
		• PDM: CR1nnn *)

`ifm_CRnnnn_CANopenSlave_Vxxyyzz.LIB`
`ifm_CRnnnn_CANopen`**`x`**`Slave_Vxxyyzz.LIB`
`ifm_CRnnnn_CAN`**`x`**`openSlave_Vxxyyzz.LIB`
	CANopen Slave Emergency- und Status-Handler	• alle ecomat*mobile*-Controller *)
		• PDM: CR1nnn *)

`ifm_CANx_SDO_Vxxyyzz.LIB`	CANopen SDO Read und SDO Write	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
`ifm_CANopen_NT_Vxxyyzz.LIB`	CANopen-Bausteine im CAN-Stack	• BasicController: CR040n, CR041n, CR043n • BasicDisplay: CR045n • PDM360 NG: CR108n, CR120n • SmartController: CR253n

*) jedoch NICHT für...
- BasicController: CR040n, CR041n, CR043n
- BasicDisplay: CR045n
- PDM360 NG: CR108n, CR120n
- SmartController: CR253n

7.2.6 CODESYS-CANopen-Bibliotheken

2719

Diese Bibliotheken sind für folgende Geräte NICHT verwendbar:
- BasicController: CR040n, CR041n, CR043n
- BasicDisplay: CR045n
- PDM360 NG: CR108n, CR120n
- SmartController: CR253n

Dateiname	Funktion	verfügbar für:
3S_CanDrvOptTableEx.LIB	CANopen Treiber	• alle ecomat*mobile*-Controller • PDM360smart: CR1070, CR1071
3S_CanDrv.LIB [1])		• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
3S_CANopenDeviceOptTableEx.LIB	CANopen Slave-Treiber	• alle ecomat*mobile*-Controller • PDM360smart: CR1070, CR1071
3S_CANopenDevice.LIB [1])		• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
3S_CANopenManagerOptTableEx.LIB	CANopen Netzwerkmanager	• alle ecomat*mobile*-Controller • PDM360smart: CR1070, CR1071
3S_CANopenManager.LIB [1])		• PDM360: CR1050, CR1051 • PDM360compact: CR1053, CR1056
3S_CANopenMasterOptTableEx.LIB	CANopen Master	• alle ecomat*mobile*-Controller • PDM360smart: CR1070, CR1071
3S_CANopenMaster.LIB [1])		• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
3S_CANopenNetVarOptTableEx.LIB	Treiber für Netzwerkvariablen	• alle ecomat*mobile*-Controller • PDM360smart: CR1070, CR1071
3S_CANopenNetVar.LIB [1])		• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056

[1]) Für folgende Geräte gilt: diese Bibliothek ist funktionslos als Platzhalter enthalten:
- BasicController: CR040n, CR041n, CR043n
- BasicDisplay: CR045n
- PDM360 NG: CR108n, CR120n
- SmartController: CR253n

7.2.7 spezielle ifm-Bibliotheken

2720

Dateiname	Funktion	verfügbar für:
ifm_RawCAN_NT_Vxxyyzz.LIB	CAN-Bausteine im CAN-Stack auf Basis Layer 2	• BasicController: CR040n, CR041n, CR043n • BasicDisplay: CR045n • PDM360 NG: CR108n, CR120n • SmartController: CR253n
ifm_J1939_NT_Vxxyyzz.LIB	J1939 Kommunikationsfunktionen im CAN-Stack	• BasicController: CR040n, CR041n, CR043n • BasicDisplay: CR045n • PDM360 NG: CR108n, CR120n • SmartController: CR253n
ifm_NetVarLib_NT_Vxxyyzz.LIB	zusätzlicher Treiber für Netzwerkvariablen	• BasicController: CR040n, CR041n, CR043n • BasicDisplay: CR045n • PDM360 NG: CR108n, CR120n • SmartController: CR253n
ifm_J1939_Vxxyyzz.LIB	J1939 Kommunikationsfunktionen	bis Laufzeitsystem V04 für: • CabinetController: CR0303 • ClassicController: CR0020, CR0505 • ExtendedController: CR0200 • SafetyController: CR7020, CR7200, CR7505 • SmartController: CR2500 • PDM360smart: CR1070, CR1071
ifm_J1939_x_Vxxyyzz.LIB	J1939 Kommunikationsfunktionen	ab Laufzeitsystem V05 für: • CabinetController: CR0303 • ClassicController: CR0020, CR0505 • ExtendedController: CR0200 • SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506 • SmartController: CR2500 • PDM360smart: CR1070, CR1071
ifm_CRnnnn_J1939_Vxxyyzz.LIB	J1939 Kommunikationsfunktionen	• Controller: CR0n3n, CR7n3n
ifm_PDM_J1939_Vxxyyzz.LIB	J1939 Kommunikationsfunktionen	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
ifm_CANx_LAYER2_Vxxyyzz.LIB	CAN-Bausteine auf Basis Layer 2: CAN Transmit, CAN Receive	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
ifm_CAN1E_Vxxyyzz.LIB	Stellt den CAN-Bus von 11 Bit auf 29 Bit um	bis Laufzeitsystem V04 für: • PDM360smart: CR1070, CR1071
ifm_CAN1_EXT_Vxxyyzz.LIB	Stellt den CAN-Bus von 11 Bit auf 29 Bit um	ab Laufzeitsystem V05 für: • CabinetController: CR030n • ClassicController: CR0020, CR0505 • ExtendedController: CR0200 • Platinensteuerung: CS0015 • SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506 • SmartController: CR250n • PDM360smart: CR1070, CR1071
ifm_CAMERA_O2M_Vxxyyzz.LIB	Kamera-Funktionen	• PDM360: CR1051
CR2013AnalogConverter.LIB	Analogwertkonvertierung für E/A-Modul CR2013	• alle ecomat*mobile*-Controller • PDM: CR1nnn
ifm_Hydraulic_16bitOS04_Vxxyyzz.LIB	Hydraulikfunktionen für Controller	bis Laufzeitsystem V04 für: • ClassicController: CR0020, CR0505 • ExtendedController: CR0200 • SafetyController: CR7020, CR7200, CR7505 • SmartController: CR250n

Dateiname	Funktion	verfügbar für:
ifm_Hydraulic_16bitOS05_Vxxyyzz.LIB	Hydraulikfunktionen für Controller	ab Laufzeitsystem V05 für: • ClassicController: CR0020, CR0505 • ExtendedController: CR0200 • SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506 • SmartController: CR250n
ifm_Hydraulic_32bit_Vxxyyzz.LIB	Hydraulikfunktionen für Controller	• Controller: CR0n3n, CR7n3n
ifm_Hydraulic_CR0303_Vxxyyzz.LIB	Hydraulikfunktionen für Controller	ab Laufzeitsystem V05 für: • CabinetController: CR0303
ifm_SafetyIO_Vxxyyzz.LIB	Sicherheitsfunktionen	• SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
ifm_SafetyPLCopen_Vxxyyzz.LIB	Sicherheitsfunktionen	• SafetyController: CR7032, CR7132
ifm_PDM_UTIL_Vxxyyzz.LIB	Hilfsfunktionen PDM	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
ifm_PDMng_UTIL_Vxxyyzz.LIB	Hilfsfunktionen PDM	• PDM360 NG: CR108n, CR120n
ifm_PDMsmart_UTIL_Vxxyyzz.LIB	Hilfsfunktionen PDM	• PDM360smart: CR1070, CR1071
ifm_PDM_Input_Vxxyyzz.LIB	alternative Eingabefunktionen PDM	• PDM: CR1nnn
ifm_CR107n_Init_Vxxyyzz.LIB	Initialisierungsfunktion PDM360smart	• PDM360smart: CR1070, CR1071
ifm_PDM_File_Vxxyyzz.LIB	Dateifunktionen PDM360	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056 • PDM360 NG: CR108n, CR120n
ifm_PDM360NG_linux_syscall_asynch.LIB	Linux-Kommandos an das System senden	• PDM360 NG: CR108n, CR120n
ifm_PDM360NG_USB_Vxxyyzz.LIB	Geräte an der USB-Schnittstelle verwalten	• PDM360 NG: CR108n, CR120n
ifm_PDM360NG_USB_LL_Vxxyyzz.LIB	Hilfsbibliothek für ifm_PDM360NG_USB_Vxxyyzz.LIB	• PDM360 NG: CR108n, CR120n
Instrumente_x.LIB	vordefinierte Anzeige-Instrumente	• PDM: CR1nnn
Symbols_x.LIB	vordefinierte Symbole	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
Segment_x.LIB	vordefinierte 7-Segment-Anzeigen	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056

Weitere Bibliotheken auf Anfrage.

8 Diagnose und Fehlerbehandlung

Inhalt

19598

Das Laufzeitsystem (LZS) überprüft das Gerät durch interne Fehler-Checks:
• in der Startphase (Reset-Phase)
• während der Ausführung des Anwendungsprogramms
→ Kapitel **Betriebszustände**

So wird eine möglichst hohe Betriebssicherheit gewährleistet.

8.1 Übersicht

12217

Bei erkannten Fehlern kann im Anwendungsprogramm zusätzlich der Systemmerker ERROR gesetzt werden. Im Fehlerfall reagiert die Steuerung dann wie folgt:

> die Betriebs-LED leuchtet rot,
> die Ausgangsrelais schalten ab,
> die darüber gesicherten Ausgänge sind spannungsfrei,
> die logischen Signalzustände der Ausgänge ändern sich dadurch NICHT.

HINWEIS

Bei Abschalten der Ausgänge durch die Relais bleiben die logischen Signalzustände unverändert.

► Der Programmierer muss das ERROR-Bit auswerten und so im Fehlerfall die Ausgänge auch logisch zurücksetzen.

Vollständige Aufstellung der gerätespezifischen Fehler-Codes und Diagnosemeldungen
→ Kapitel **Systemmerker**

9 Begriffe und Abkürzungen

A

Adresse

Das ist der „Name“ des Teilnehmers im Bus. Alle Teilnehmer benötigen eine unverwechselbare, eindeutige Adresse, damit der Austausch der Signale fehlerfrei funktioniert.

Anleitung

Übergeordnetes Wort für einen der folgenden Begriffe:
Montageanleitung, Datenblatt, Benutzerinformation, Bedienungsanleitung, Gerätehandbuch, Installationsanleitung, Onlinehilfe, Systemhandbuch, Programmierhandbuch, usw.

Anwendungsprogramm

Software, die speziell für die Anwendung vom Hersteller in die Maschine programmiert wird. Die Software enthält üblicherweise logische Sequenzen, Grenzwerte und Ausdrücke zum Steuern der entsprechenden Ein- und Ausgänge, Berechnungen und Entscheidungen.

Architektur

Spezifische Konfiguration von Hardware- und/oder Software-Elementen in einem System.

B

Baud

Baud, Abk.: Bd = Maßeinheit für die Geschwindigkeit bei der Datenübertragung. Baud ist nicht zu verwechseln mit "bits per second" (bps, Bit/s). Baud gibt zwar die Anzahl von Zustandsänderungen (Schritte, Takte) pro Sekunde auf einer Übertragungsstrecke an. Aber es ist nicht festgelegt, wie viele Bits pro Schritt übertragen werden. Der Name Baud geht auf den französischen Erfinder J. M. Baudot zurück, dessen Code für Telexgeräte verwendet wurde.
1 MBd = 1024 x 1024 Bd = 1 048 576 Bd

Bestimmungsgemäße Verwendung

Das ist die Verwendung eines Produkts in Übereinstimmung mit den in der Anleitung bereitgestellten Informationen.

Bootloader

Im Auslieferungszustand enthalten **ecomat*mobile***-Controller nur den Bootloader.
Der Bootloader ist ein Startprogramm, mit dem das Laufzeitsystem und das Anwendungsprogramm auf dem Gerät nachgeladen werden können.
Der Bootloader enthält Grundroutinen...
- zur Kommunikation der Hardware-Module untereinander,
- zum Nachladen des Laufzeitsystems.

Der Bootloader ist das erste Software-Modul, das im Gerät gespeichert sein muss.

Bus

Serielle Datenübertragung mehrerer Teilnehmer an derselben Leitung.

C

CAN

CAN = **C**ontroller **A**rea **N**etwork
CAN gilt als Feldbussystem für größere Datenmengen, das prioritätengesteuert arbeitet. Es gibt mehrere höhere Protokolle, die auf CAN aufsetzen, z. B. 'CANopen' oder 'J1939'.

CAN-Stack

CAN-Stack = Software-Komponente, die sich um die Verarbeitung von CAN-Telegramme kümmert.

CiA

CiA = CAN in Automation e.V.
Anwender- und Herstellerorganisation in Erlangen, Deutschland. Definitions- und Kontrollorgan für das CANopen-Protokoll.
Homepage → www.can-cia.org

CiA DS 304

DS = **D**raft **S**tandard
CANopen-Geräteprofil für sichere Kommunikation

CiA DS 401

DS = **D**raft **S**tandard
CANopen-Geräteprofil für digitale und analoge E/A-Baugruppen

CiA DS 402

DS = **D**raft **S**tandard
CANopen-Geräteprofil für Antriebe

CiA DS 403

DS = **D**raft **S**tandard
CANopen-Geräteprofil für Bediengeräte

CiA DS 404

DS = **D**raft **S**tandard
CANopen-Geräteprofil für Messtechnik und Regler

CiA DS 405

DS = **D**raft **S**tandard
CANopen-Spezifikation der Schnittstelle zu programmierbaren Steuerungen (IEC 61131-3)

CiA DS 406

DS = **D**raft **S**tandard
CANopen-Geräteprofil für Drehgeber / Encoder

CiA DS 407

DS = **D**raft **S**tandard
CANopen-Anwendungsprofil für den öffentlichen Nahverkehr

COB-ID

COB = **C**ommunication **Ob**ject = Kommunikationsobjekt
ID = **Id**entifier = Kennung
ID eines CANopen-Kommunikationsobjekts
Entspricht dem Identifier der CAN-Nachricht, mit der das Kommunikationsobjekt über den CAN-Bus gesendet wird.

CODESYS

CODESYS® ist eingetragene Marke der 3S – Smart Software Solutions GmbH, Deutschland.
'CODESYS for Automation Alliance℠' vereinigt Firmen der Automatisierungsindustrie, deren Hardware-Geräte alle mit dem weit verbreiteten IEC 61131-3 Entwicklungswerkzeug CODESYS® programmiert werden.
Homepage → www.codesys.com

CSV-Datei

CSV = **C**omma **S**eparated **V**alues (auch: **C**haracter **S**eparated **V**alues)
Eine CSV-Datei ist eine Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten.
Die Dateinamen-Erweiterung lautet `.csv`.

Beispiel: Quell-Tabelle mit Zahlenwerten:

Wert 1.0	Wert 1.1	Wert 1.2	Wert 1.3
Wert 2.0	Wert 2.1	Wert 2.2	Wert 2.3
Wert 3.0	Wert 3.1	Wert 3.2	Wert 3.3

Daraus entsteht folgende CSV-Datei:

```
Wert 1.0;Wert 1.1;Wert 1.2;Wert 1.3
Wert 2.0;Wert 2.1;Wert 2.2;Wert 2.3
Wert 3.0;Wert 3.1;Wert 3.2;Wert 3.3
```

D

Datentyp

Abhängig vom Datentyp können unterschiedlich große Werte gespeichert werden.

Datentyp	min. Wert	max. Wert	Größe im Speicher
BOOL	FALSE	TRUE	8 Bit = 1 Byte
BYTE	0	255	8 Bit = 1 Byte
WORD	0	65 535	16 Bit = 2 Bytes
DWORD	0	4 294 967 295	32 Bit = 4 Bytes
SINT	-128	127	8 Bit = 1 Byte
USINT	0	255	8 Bit = 1 Byte
INT	-32 768	32 767	16 Bit = 2 Bytes
UINT	0	65 535	16 Bit = 2 Bytes
DINT	-2 147 483 648	2 147 483 647	32 Bit = 4 Bytes
UDINT	0	4 294 967 295	32 Bit = 4 Bytes
REAL	$-3{,}402823466 \cdot 10^{38}$	$3{,}402823466 \cdot 10^{38}$	32 Bit = 4 Bytes
ULINT	0	18 446 744 073 709 551 615	64 Bit = 8 Bytes
STRING			number of char. + 1

DC

Direct **C**urrent = Gleichstrom

Diagnose

Bei der Diagnose wird der "Gesundheitszustand" des Gerätes geprüft. Es soll festgestellt werden, ob und gegebenenfalls welche →Fehler im Gerät vorhanden sind.

Je nach Gerät können auch die Ein- und Ausgänge auf einwandfreie Funktion überwacht werden:
- Drahtbruch,
- Kurzschluss,
- Wert außerhalb des Sollbereichs.

Zur Diagnose können Konfigurations-Dateien herangezogen werden, die während des "normalen" Betriebs des Gerätes erzeugt wurden.
Der korrekte Start der Systemkomponenten wird während der Initialisierungs- und Startphase überwacht.
Zur weiteren Diagnose können auch Selbsttests durchgeführt werden.

Dither

to dither (engl.) = schwanken / zittern.
Dither ist ein Bestandteil der →PWM-Signale zum Ansteuern von Hydraulik-Ventilen. Für die elektromagnetischen Antriebe von Hydraulik-Ventilen hat sich herausgestellt, dass sich die Ventile viel besser regeln lassen, wenn das Steuersignal (PWM-Impulse) mit einer bestimmten Frequenz der PWM-Frequenz überlagert wird. Diese Dither-Frequenz muss ein ganzzahliger Teil der PWM-Frequenz sein.

DLC

Data **L**ength **C**ode = bei CANopen die Anzahl der Daten-Bytes in einer Nachricht.
Für →SDO: DLC = 8

DRAM

DRAM = **D**ynamic **R**andom **A**ccess **M**emory.
Technologie für einen elektronischen Speicherbaustein mit wahlfreiem Zugriff (Random Access Memory, RAM). Das speichernde Element ist dabei ein Kondensator, der entweder geladen oder entladen ist. Über einen Schalttransistor wird er zugänglich und entweder ausgelesen oder mit neuem Inhalt beschrieben. Der Speicherinhalt ist flüchtig: die gespeicherte Information geht bei fehlender Betriebsspannung oder zu später Wiederauffrischung verloren.

DTC

DTC = **D**iagnostic **T**rouble **C**ode = Fehler-Code
Beim Protokoll J1939 werden Störungen und Fehler über zugeordnete Nummern – den DTCs – verwaltet und gemeldet.

E

ECU

(1) **E**lectronic **C**ontrol **U**nit = Steuergerät oder Mikrocontroller
(2) **E**ngine **C**ontrol **U**nit = Steuergerät eines Motors

EDS-Datei

EDS = **E**lectronic **D**ata **S**heet = elektronisch hinterlegtes Datenblatt, z.B. für:
- Datei für das Objektverzeichnis im CANopen-Master,
- CANopen-Gerätebeschreibungen.

Via EDS können vereinfacht Geräte und Programme ihre Spezifikationen austauschen und gegenseitig berücksichtigen.

Embedded Software

System-Software, Grundprogramm im Gerät, praktisch das →Laufzeitsystem.
Die Firmware stellt die Verbindung her zwischen der Hardware des Gerätes und dem Anwendungsprogramm. Die Firmware wird vom Hersteller der Steuerung als Teil des Systems geliefert und kann vom Anwender nicht verändert werden.

EMCY

Abkürzung für Emergency (engl.) = Notfall
Nachricht im CANopen-Protokoll, mit der Fehler gemeldet werden.

EMV

EMV = **E**lektro-**M**agnetische **V**erträglichkeit.
Gemäß der EG-Richtlinie (2004/108/EG) zur elektromagnetischen Verträglichkeit (kurz EMV-Richtlinie) werden Anforderungen an die Fähigkeit von elektrischen und elektronischen Apparaten, Anlagen, Systemen oder Bauteilen gestellt, in der vorhandenen elektromagnetischen Umwelt zufriedenstellend zu arbeiten. Die Geräte dürfen ihre Umgebung nicht stören und dürfen sich von äußerlichen elektromagnetischen Störungen nicht ungünstig beeinflussen lassen.

Ethernet

Ethernet ist eine weit verbreitete, herstellerneutrale Netzwerktechnologie, mit der Daten mit einer Geschwindigkeit von 10 bis 10 000 Millionen Bit pro Sekunde (Mbps) übertragen werden können. Ethernet gehört zu der Familie der sogenannten „bestmöglichen Datenübermittlung“ auf einem nicht exklusiven Übertragungsmedium. 1972 entwickelt, wurde das Konzept 1985 als IEEE 802.3 spezifiziert.

EUC

EUC = **E**quipment **U**nder **C**ontrol (kontrollierte Einrichtung).
EUC ist eine Einrichtung, Maschine, Gerät oder Anlage, verwendet zur Fertigung, Stoffumformung, zum Transport, zu medizinischen oder anderen Tätigkeiten (→ IEC 61508-4, Abschnitt 3.2.3). Das EUC umfasst also alle Einrichtungen, Maschinen, Geräte oder Anlagen, die →Gefährdungen verursachen können und für die sicherheitsgerichtete Systeme erforderlich sind.
Falls eine vernünftigerweise vorhersehbare Aktivität oder Inaktivität zu durch das EUC verursachten Gefährdungen mit unvertretbarem Risiko führt, sind Sicherheitsfunktionen erforderlich, um einen sicheren Zustand für das EUC zu erreichen oder aufrecht zu erhalten. Diese Sicherheitsfunktionen werden durch ein oder mehrere sicherheitsgerichtete Systeme ausgeführt.

F

Fehlanwendung

Das ist die Verwendung eines Produkts in einer Weise, die vom Konstrukteur nicht vorgesehen ist.
Eine Fehlanwendung führt meist zu einer →Gefährdung von Personen oder Sachen.
Vor vernünftigerweise, vorhersehbaren Fehlanwendungen muss der Hersteller des Produkts in seinen Benutzerinformationen warnen.

FiFo

FIFO (**F**irst **I**n, **F**irst **O**ut) = Arbeitsweise des Stapelspeichers: Das Datenpaket, das zuerst in den Stapelspeicher geschrieben wurde, wird auch als erstes gelesen. Pro Identifier steht ein solcher Zwischenspeicher (als Warteschlange) zur Verfügung.

Flash-Speicher

Flash-ROM (oder Flash-EPROM oder Flash-Memory) kombiniert die Vorteile von Halbleiterspeicher und Festplatten. Die Daten werden allerdings wie bei einer Festplatte blockweise in Datenblöcken zu 64, 128, 256, 1024, ... Byte zugleich geschrieben und gelöscht.

Vorteile von Flash-Speicher

- Die gespeicherten Daten bleiben auch bei fehlender Versorgungsspannung erhalten.
- Wegen fehlender beweglicher Teile ist Flash geräuschlos, unempfindlich gegen Erschütterungen und magnetische Felder.

Nachteile von Flash-Speicher

- Begrenzte Zahl von Schreib- bzw. Löschvorgängen, die eine Speicherzelle vertragen kann:
  - Multi-Level-Cells: typ. 10 000 Zyklen
  - Single-Level-Cells: typ. 100 000 Zyklen
- Da ein Schreibvorgang Speicherblöcke zwischen 16 und 128 kByte gleichzeitig beschreibt, werden auch Speicherzellen beansprucht, die gar keiner Veränderung bedürfen.

FRAM

FRAM, oder auch FeRAM, bedeutet **Fe**rroelectric **R**andom **A**ccess **M**emory. Der Speicher- und Löschvorgang erfolgt durch eine Polarisationsänderung in einer ferroelektrischen Schicht.
Vorteile von FRAM gegenüber herkömmlichen Festwertspeichern:

- nicht flüchtig,
- kompatibel zu gängigen EEPROMs, jedoch:
- Zugriffszeit ca. 100 ns,
- fast unbegrenzt viele Zugriffszyklen möglich.

H

Heartbeat

Heartbeat (engl.) = Herzschlag.
Die Teilnehmer senden regelmäßig kurze Signale. So können die anderen Teilnehmer prüfen, ob ein Teilnehmer ausgefallen ist.

HMI

HMI = **H**uman **M**achine **I**nterface = Mensch-Maschine-Schnittstelle

I

ID – Identifier

ID = **Id**entifier = Kennung

Name zur Unterscheidung der an einem System angeschlossenen Geräte / Teilnehmer oder der zwischen den Teilnehmern ausgetauschten Nachrichtenpakete.

IEC 61131

Norm: Grundlagen Speicherprogrammierbarer Steuerungen
- Teil 1: Allgemeine Informationen
- Teil 2: Betriebsmittelanforderungen und Prüfungen
- Teil 3: Programmiersprachen
- Teil 5: Kommunikation
- Teil 7: Fuzzy-Control-Programmierung

IEC-User-Zyklus

IEC-User-Zyklus = SPS-Zyklus im CODESYS-Anwendungsprogramm.

IP-Adresse

IP = **I**nternet **P**rotocol = Internet-Protokoll.
Die IP-Adresse ist eine Nummer, die zur eindeutigen Identifizierung eines Internet-Teilnehmers notwendig ist. Zur besseren Übersicht wird die Nummer in 4 dezimalen Werten geschrieben, z. B. 127.215.205.156.

ISO 11898

Norm: Straßenfahrzeuge – CAN-Protokoll
- Teil 1: Bit-Übertragungsschicht und physikalische Zeichenabgabe
- Teil 2: High-speed medium access unit
- Teil 3: Fehlertolerante Schnittstelle für niedrige Geschwindigkeiten
- Teil 4: Zeitgesteuerte Kommunikation
- Teil 5: High-speed medium access unit with low-power mode

ISO 11992

Norm: Straßenfahrzeuge – Austausch von digitalen Informationen über elektrische Verbindungen zwischen Zugfahrzeugen und Anhängefahrzeugen
- Teil 1: Bit-Übertragungsschicht und Sicherungsschicht
- Teil 2: Anwendungsschicht für die Bremsausrüstung
- Teil 3: Anwendungsschicht für andere als die Bremsausrüstung
- Teil 4: Diagnose

ISO 16845

Norm: Straßenfahrzeuge – Steuergerätenetz (CAN) – Prüfplan zu Konformität

J

J1939

→ SAE J1939

K

Klemme 15

Klemme 15 ist in Fahrzeugen die vom Zündschloss geschaltete Plusleitung.

L

Laufzeitsystem

Grundprogramm im Gerät, stellt die Verbindung her zwischen der Hardware des Gerätes und dem Anwendungsprogramm.
→ Kapitel **Software-Module für das Gerät**

LED

LED = **L**ight **E**mitting **D**iode = Licht aussendende Diode.
Leuchtdiode, auch Luminiszenzdiode, ein elektronisches Element mit hoher, farbiger Leuchtkraft auf kleinem Volumen bei vernachlässigbarer Verlustleistung.

Link

Ein Link ist ein Querverweis zu einer anderen Stelle im Dokument oder auf ein externes Dokument.

LSB

Least **S**ignificant **B**it/Byte = Niederwertigstes Bit/Byte in einer Reihe von Bit/Bytes.

M

MAC-ID

MAC = **M**anufacturer‘s **A**ddress **C**ode
= Hersteller-Seriennummer.
→ID = **Id**entifier = Kennung
Jede Netzwerkkarte verfügt über eine so genannte MAC-Adresse, ein unverwechselbarer, auf der ganzen Welt einzigartiger Zahlencode – quasi eine Art Seriennummer. So eine MAC-Adresse ist eine Aneinanderreihung von 6 Hexadezimalzahlen, etwa "00-0C-6E-D0-02-3F".

Master

Wickelt die komplette Organisation auf dem →Bus ab. Der Master entscheidet über den zeitlichen Buszugriff und fragt die →Slaves zyklisch ab.

MMI

MMI = **M**ensch-**M**aschine-**I**nterface
→ **HMI** (→ S. 156)

MRAM

MRAM = **M**agnetoresistive **R**andom **A**ccess **M**emory
Die Informationen werden mit magnetischen Ladungselementen gespeichert. Dabei wird die Eigenschaft bestimmter Materialien ausgenutzt, die ihren elektrischen Widerstand unter dem Einfluss magnetischer Felder ändern.
Vorteile von MRAM gegenüber herkömmlichen Festwertspeichern:
- nicht flüchtig (wie FRAM), jedoch:
- Zugriffszeit nur ca. 35 ns,
- unbegrenzt viele Zugriffszyklen möglich.

MSB

Most **S**ignificant **B**it/Byte = Höchstwertiges Bit/Byte einer Reihe von Bits/Bytes.

N

NMT

NMT = **N**etwork **M**anagemen**t** = Netzwerk-Verwaltung (hier: im CANopen-Protokoll).
Der NMT-Master steuert die Betriebszustände der NMT-Slaves.

Node

Node (engl.) = Knoten. Damit ist ein Teilnehmer im Netzwerk gemeint.

Node Guarding

Node (engl.) = Knoten, hier: Netzwerkteilnehmer
Guarding (engl.) = Schutz
Parametrierbare, zyklische Überwachung von jedem entsprechend konfigurierten →Slave. Der →Master prüft, ob die Slaves rechtzeitig antworten. Die Slaves prüfen, ob der Master regelmäßig anfragt. Somit können ausgefallene Netzwerkteilnehmer schnell erkannt und gemeldet werden.

O

Obj / Objekt

Oberbegriff für austauschbare Daten / Botschaften innerhalb des CANopen-Netzwerks.

Objektverzeichnis

Das **Ob**jekt**v**erzeichnis OBV enthält alle CANopen-Kommunikationsparameter eines Gerätes, sowie gerätespezifische Parameter und Daten.

OBV

Das **Ob**jekt**v**erzeichnis OBV enthält alle CANopen-Kommunikationsparameter eines Gerätes, sowie gerätespezifische Parameter und Daten.

OPC

OPC = **O**LE for **P**rocess **C**ontrol = Objektverknüpfung und -einbettung für Prozesssteuerung
Standardisierte Software-Schnittstelle zur herstellerunabhängigen Kommunikation in der Automatisierungstechnik
OPC-Client (z.B. Gerät zum Parametrieren oder Programmieren) meldet sich nach dem Anschließen am OPC-Server (z.B. Automatisierungsgerät) automatisch bei diesem an und kommuniziert mit ihm.

operational

Operational (engl.) = betriebsbereit
Betriebszustand eines CANopen-Teilnehmers. In diesem Modus können →SDOs, →NMT-Kommandos und →PDOs übertragen werden.

P

PC-Karte

→ PCMCIA-Karte

PCMCIA-Karte

PCMCIA = Personal Computer Memory Card International Association, ein Standard für Erweiterungskarten mobiler Computer.
Seit der Einführung des Cardbus-Standards 1995 werden PCMCIA-Karten auch als PC-Karte (engl.: PC Card) bezeichnet.

PDM

PDM = **P**rocess and **D**ialog **M**odule = **P**rozess- und **D**ialog-**M**onitor.
Gerät zur Kommunikation des Bedieners mit der Maschine / Anlage.

PDO

PDO = **P**rocess **D**ata **O**bject = Nachrichten-Objekt mit Prozessdaten.
Die zeitkritischen Prozessdaten werden mit Hilfe der "Process Data Objects" (PDOs) übertragen. Die PDOs können beliebig zwischen den einzelnen Knoten ausgetauscht werden (PDO-Linking). Zusätzlich wird festgelegt, ob der Datenaustausch ereignisgesteuert (asynchron) oder synchronisiert erfolgen soll. Je nach der Art der zu übertragenden Daten kann die richtige Wahl der Übertragungsart zu einer erheblichen Entlastung des →CAN-Bus führen.
Dem Protokoll entsprechend, sind diese Dienste nicht bestätigte Dienste: es gibt keine Kontrolle, ob die Nachricht auch beim Empfänger ankommt. Netzwerkvariablen-Austausch entspricht einer "1-zu-n-Verbindung" (1 Sender zu n Empfängern).

PDU

PDU = **P**rotocol **D**ata **U**nit = Protokoll-Daten-Einheit.
Die PDU ist ein Begriff aus dem →CAN-Protokoll →SAE J1939. Sie bezeichnet einen Bestandteil der Zieladresse (PDU Format 1, verbindungsorientiert) oder der Group Extension (PDU Format 2, nachrichtenorientiert).

PES

Programable **e**lectronic **s**ystem = Programmierbares elektronisches System ...
- zur Steuerung, zum Schutz oder zur Überwachung,
- auf der Basis einer oder mehrerer programmierbarer Geräte,
- einschließlich aller Elemente dieses Systems, wie Ein- und Ausgabegeräte.

PGN

PGN = **P**arameter **G**roup **N**umber = Parameter-Gruppennummer
PGN = 6 Null-Bits + 1 Bit reserviert + 1 Bit Data Page + 8 Bit PDU Format (PF) + 8 PDU Specific (PS)
Die Parameter-Gruppennummer ist ein Begriff aus dem →CAN-Protokoll →SAE J1939.

PID-Regler

Der PID-Regler (proportional–integral–derivative controller) besteht aus folgenden Anteilen:
- P = Proportional-Anteil
- I = Integral-Anteil
- D = Differential-Anteil (jedoch nicht beim Controller CR04nn, CR253n).

Piktogramm

Piktogramme sind bildhafte Symbole, die eine Information durch vereinfachte grafische Darstellung vermitteln (→ Kapitel **Was bedeuten die Symbole und Formatierungen?** (→ S. 6)).

Pre-Op

Pre-Op = PRE-OPERATIONAL mode (engl.) = Zustand vor 'betriebsbereit'.
Betriebszustand eines CANopen-Teilnehmers. Nach dem Einschalten der Versorgungsspannung geht jeder Teilnehmer automatisch in diesem Zustand. Im CANopen-Netz können in diesem Modus nur →SDOs und →NMT-Kommandos übertragen werden, jedoch keine Prozessdaten.

Prozessabbild

Mit Prozessabbild bezeichnet man den Zustand der Ein- und Ausgänge, mit denen die SPS innerhalb eines →Zyklusses arbeitet.

- Am Zyklus-Beginn liest die SPS die Zustände aller Eingänge in das Prozessabbild ein. Während des Zyklusses kann die SPS Änderungen an den Eingängen nicht erkennen.
- Im Laufe des Zyklusses werden die Ausgänge nur virtuell (im Prozessabbild) geändert.
- Am Zyklus-Ende schreibt die SPS die virtuellen Ausgangszustände auf die realen Ausgänge.

PWM

PWM = Puls-Weiten-Modulation
Bei dem PWM-Ausgangssignal handelt es sich um ein getaktetes Signal zwischen GND und Versorgungsspannung.
Innerhalb einer festen Periode (PWM-Frequenz) wird das Puls-/Pausenverhältnis variiert. Durch die angeschlossene Last stellt sich je nach Puls-/Pausenverhältnis der entsprechende Effektivstrom ein.

R

ratiometrisch

Ratio (lat.) = Verhältnis
Messungen können auch ratiometrisch erfolgen = Verhältnismessung. Wenn das Ausgangssinal eines Sensors proportional zu seiner Versorgungsspannung ist, kann durch ratiometrische Messung (= Messung im Verhältnis zur Versorgung) der Einfluss von Schwankungen der Versorgung reduziert, im Idealfall sogar beseitigt werden.
→ Analogeingang

RAW-CAN

RAW-CAN bezeichnet das reine →CAN-Protokoll, das ohne ein zusätzliches Kommunikationsprotokoll auf dem CAN-Bus (auf ISO/OSI-Schicht 2) arbeitet. Das CAN-Protokoll ist international nach →ISO 11898-1 definiert und garantiert zusätzlich in →ISO 16845 die Austauschbarkeit von CAN-Chips.

remanent

Remanente Daten sind gegen Datenverlust bei Spannungsausfall geschützt.
Z.B. kopiert das →Laufzeitsystem die remanenten Daten automatisch in einen →Flash-Speicher, sobald die Spannungsversorgung unter einen kritischen Wert sinkt. Bei Wiederkehr der Spannungsversorgung lädt das Laufzeitsystem die remanenten Daten zurück in den Arbeitsspeicher. Dagegen sind die Daten im Arbeitsspeicher einer Steuerung flüchtig und bei Unterbrechung der Spannungsversorgung normalerweise verloren.

ro

ro = read only (engl.) = nur lesen
Unidirektionale Datenübertragung: Daten können nur gelesen werden, jedoch nicht verändert.

RTC

RTC = **R**eal **T**ime **C**lock = Echtzeituhr
Liefert (batteriegepuffert) aktuell Datum und Uhrzeit. Häufiger Einsatz beim Speichern von Fehlermeldungsprotokollen.

rw

rw = read/write (engl.) = lesen und schreiben
Bidirektionale Datenübertragung: Daten können sowohl gelesen als auch verändert werden.

S

SAE J1939

Das Netzwerkprotokoll SAE J1939 beschreibt die Kommunikation auf einem →CAN-Bus in Nutzfahrzeugen zur Übermittlung von Diagnosedaten (z.B.Motordrehzahl, Temperatur) und Steuerungsinformationen.
Norm: Recommended Practice for a Serial Control and Communications Vehicle Network
- Teil 2: Agricultural and Forestry Off-Road Machinery Control and Communication Network
- Teil 3: On Board Diagnostics Implementation Guide
- Teil 5: Marine Stern Drive and Inboard Spark-Ignition Engine On-Board Diagnostics Implementation Guide
- Teil 11: Physical Layer – 250 kBits/s, Shielded Twisted Pair
- Teil 13: Off-Board Diagnostic Connector
- Teil 15: Reduced Physical Layer, 250 kBits/s, Un-Shielded Twisted Pair (UTP)
- Teil 21: Data Link Layer
- Teil 31: Network Layer
- Teil 71: Vehicle Application Layer
- Teil 73: Application Layer – Diagnostics
- Teil 81: Network Management Protocol

SD-Card

Eine SD Memory Card (Kurzform für **S**ecure **D**igital Memory Card; deutsch: Sichere digitale Speicherkarte) ist ein digitales Speichermedium, das nach dem Prinzip der →Flash-Speicherung arbeitet.

SDO

SDO = **S**ervice **D**ata **O**bject = Nachrichten-Objekt mit Servicedaten.
Das SDO dient dem Zugriff auf Objekte in einem CANopen-Objektverzeichnis. Dabei fordern 'Clients' die gewünschten Daten von 'Servern' an. Die SDOs bestehen immer aus 8 Bytes.
Beispiele:
- Automatische Konfiguration aller →Slaves über SDOs beim Systemstart.
- Auslesen der Fehlernachrichten aus dem →Objektverzeichnis.

Jedes SDO wird auf Antwort überwacht und wiederholt, wenn sich innerhalb der Überwachungszeit der Slave nicht meldet.

Selbsttest

Testprogramm, das aktiv Komponenten oder Geräte testet. Das Programm wird durch den Anwender gestartet und dauert eine gewisse Zeit. Das Ergebnis davon ist ein Testprotokoll (Log-Datei), aus dem entnommen werden kann, was getestet wurde und ob das Ergebnis positiv oder negativ ist.

Slave

Passiver Teilnehmer am Bus, antwortet nur auf Anfrage des →Masters. Slaves haben im Bus eine eindeutige →Adresse.

Steuerungskonfiguration

Bestandteil der CODESYS-Bedienoberfläche.

► Programmierer teilt dem Programmiersystem mit, welche Hardware programmiert werden soll.
> CODESYS lädt die zugehörigen Bibliotheken.
> Lesen und schreiben der Peripherie-Zustände (Ein-/Ausgänge) ist möglich.

stopped

stopped (engl.) = angehalten
Betriebszustand eines CANopen-Teilnehmers. In diesem Modus werden nur →NMT-Kommandos übertragen.

Symbole

Piktogramme sind bildhafte Symbole, die eine Information durch vereinfachte grafische Darstellung vermitteln (→ Kapitel **Was bedeuten die Symbole und Formatierungen?** (→ S. 6)).

Systemvariable

Variable, auf die via IEC-Adresse oder Symbolname aus der SPS zugegriffen werden kann.

T

Target

Das Target enthält für CODESYS die Hardware-Beschreibung des Zielgeräts, z.B.: Ein- und Ausgänge, Speicher, Dateiablageorte.
Entspricht einem elektronischen Datenblatt.

TCP

Das **T**ransmission **C**ontrol **P**rotocol ist Teil der Protokollfamilie TCP/IP. Jede TCP/IP-Datenverbindung hat einen Sender und einen Empfänger. Dieses Prinzip ist eine verbindungsorientierte Datenübertragung. In der TCP/IP-Protokollfamilie übernimmt TCP als verbindungsorientiertes Protokoll die Aufgabe der Datensicherheit, der Datenflusssteuerung und ergreift Maßnahmen bei einem Datenverlust. (vgl.: →UDP)

Template

Template (englisch = Schablone) ist eine Vorlage, die mit Inhalten gefüllt werden kann.
Hier: Eine Struktur von vorkonfigurierten Software-Elementen als Basis für ein Anwendungsprogramm.

U

UDP

UDP (**U**ser **D**atagram **P**rotocol) ist ein minimales, verbindungsloses Netzprotokoll, das zur Transportschicht der Internetprotokollfamilie gehört. Aufgabe von UDP ist es, Daten, die über das Internet übertragen werden, der richtigen Anwendung zukommen zu lassen.
Derzeit sind Netzwerkvariablen auf Basis von →CAN und UDP implementiert. Die Variablenwerte werden dabei auf der Basis von Broadcast-Nachrichten automatisch ausgetauscht. In UDP sind diese als Broadcast-Telegramme realisiert, in CAN als →PDOs.
Dem Protokoll entsprechend, sind diese Dienste nicht bestätigte Dienste: es gibt keine Kontrolle, ob die Nachricht auch beim Empfänger ankommt. Netzwerkvariablen-Austausch entspricht einer "1-zu-n-Verbindung" (1 Sender zu n Empfängern).

V

Verwendung, bestimmungsgemäß

Das ist die Verwendung eines Produkts in Übereinstimmung mit den in der Anleitung bereitgestellten Informationen.

W

Watchdog

Der Begriff Watchdog (englisch; Wachhund) wird verallgemeinert für eine Komponente eines Systems verwendet, die die Funktion anderer Komponenten beobachtet. Wird dabei eine mögliche Fehlfunktionen erkannt, so wird dies entweder signalisiert oder geeignete Programm-Verzweigungen eingeleitet. Das Signal oder die Verzweigungen dienen als Auslöser für andere kooperierende Systemkomponenten, die das Problem lösen sollen.

Z

Zykluszeit

Das ist die Zeit für einen Zyklus. Das SPS-Programm läuft einmal komplett durch.
Je nach ereignisgesteuerten Verzweigungen im Programm kann dies unterschiedlich lange dauern.

10 Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z

11 Notizen • Notes • Notes