ifm electronic

CE

ecomat mobile

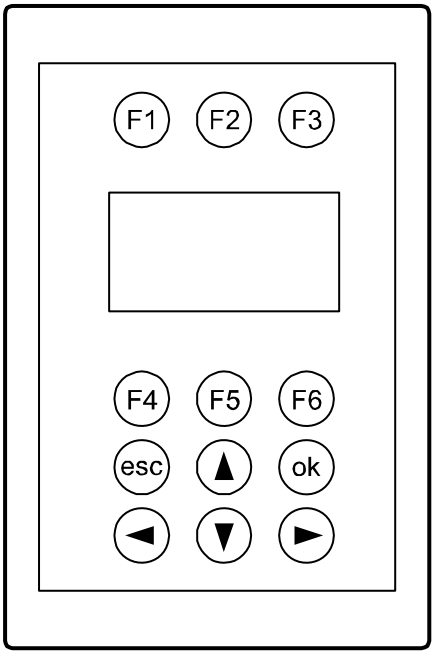
Systemhandbuch
PDM360smart Monitor

ecomat100

CR1070
CR1071

CoDeSys® V2.3
Target V05

Deutsch



7390674_01_DE 2012-03-20

Inhaltsverzeichnis

1 Über diese Anleitung

Inhalt

202

Im ergänzenden "Programmierhandbuch CoDeSys V2.3" erhalten Sie weitergehende Informationen über die Nutzung des Programmiersystems "CoDeSys for Automation Alliance". Dieses Handbuch steht auf der **ifm**-Homepage als kostenloser Download zur Verfügung:
a) → www.ifm.com > Land wählen > [Service] > [Download] > [Steuerungssysteme]
b) → **ecomat*mobile***-DVD "Software, tools and documentation"

Niemand ist vollkommen. Wenn Sie uns Verbesserungsvorschläge zu dieser Anleitung melden, erhalten Sie von uns ein kleines Geschenk als Dankeschön.

1.1 Was bedeuten die Symbole und Formatierungen?

2979

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.

HINWEIS

Wichtige Hinweise auf Fehlfunktionen oder Störungen.

Info

Weitere Hinweise.

► ...	Handlungsaufforderung
> ...	Reaktion, Ergebnis
→ ...	"siehe"
abc	Querverweis
[...]	Bezeichnung von Tasten, Schaltflächen oder Anzeigen

1.2 Wie ist diese Anleitung aufgebaut?

204

Diese Dokumentation ist eine Kombination aus verschiedenen Anleitungstypen. Sie ist eine Lernanleitung für den Einsteiger, aber gleichzeitig auch eine Nachschlageanleitung für den versierten Anwender.

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:
→ www.ifm.com > Land wählen > [Kontakt].

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 > Land wählen > [Service] > [Download] > [Steuerungssysteme]
⇨ Unsere Online-Hilfen sind meist "tagesaktuell".
⇨ Die PDF-Handbücher aktualisieren wir nur in großen zeitlichen Abständen.

2 Sicherheitshinweise

Inhalt

213

2.1 Wichtig!

9884

Mit den in dieser Anleitung gegebenen Informationen, Hinweisen und Beispielen werden keine Eigenschaften zugesichert. Die abgebildeten Zeichnungen, Darstellungen und Beispiele enthalten weder Systemverantwortung noch applikationsspezifische Besonderheiten.

Die Sicherheit der Maschine/Anlage muss auf jeden Fall eigenverantwortlich durch den Hersteller der Maschine/Anlage gewährleistet werden.

⚠ WARNUNG

Sach- oder Körperschäden sind möglich bei Nichtbeachten der Hinweise in dieser Anleitung!
Die **ifm electronic gmbh** übernimmt hierfür keine Haftung.

- ► Die handelnde Person muss vor allen Arbeiten an und mit diesem Gerät die Sicherheitshinweise und die betreffenden Kapitel dieser Anleitung gelesen und verstanden haben.
- ► Die handelnde Person muss zu Arbeiten an der Maschine/Anlage autorisiert sein.
- ► Beachten Sie die Technischen Daten der betroffenen Geräte!
  Das aktuelle Datenblatt finden Sie auf der **ifm**-Homepage:
  → www.ifm.com > Land wählen > [Datenblattsuche] > (Artikel-Nr.) > [Technische Daten im PDF-Format]
- ► Beachten Sie die Montage- und Anschlussbedingungen sowie die bestimmungsgemäße Verwendung der betroffenen Geräte!
  → mitgelieferte Montageanleitung oder auf der **ifm**-Homepage:
  → www.ifm.com > Land wählen > [Datenblattsuche] > (Artikel-Nr.) > [Betriebsanleitungen]

ACHTUNG

Der Treiberbaustein der seriellen Schnittstelle kann beschädigt werden!

Beim Trennen der seriellen Schnittstelle unter Spannung kann es zu undefinierten Zuständen kommen, die zu einer Schädigung des Treiberbausteins führen.

- ► Die serielle Schnittstelle nur im spannungslosen Zustand trennen!

ACHTUNG

Bei zu intensiver Beleuchtung kann das Display temporär "erblinden"!

- ► Das Display aus kurzer Entfernung nicht mit Blitzlicht fotografieren!

Anlaufverhalten der Steuerung

Der Hersteller der Maschine/Anlage muss mit seinem Applikations-Programm gewährleisten, dass beim Anlauf oder Wiederanlauf der Steuerung keine gefahrbringenden Bewegungen gestartet werden können.

Ein Wiederanlauf kann z.B. verursacht werden durch:

- Spannungswiederkehr nach Spannungsausfall
- Reset nach Watchdog-Ansprechen wegen zu langer Zykluszeit

2.2 Welche Vorkenntnisse sind notwendig?

215

Das Dokument richtet sich an Personen, die über Kenntnisse der Steuerungstechnik und SPS-Programmierkenntnisse mit IEC 61131-3 verfügen.

Wenn dieses Gerät über eine SPS verfügt, sollten die Personen zusätzlich mit der Software CoDeSys vertraut sein.

Das Dokument richtet sich an Fachkräfte. Dabei handelt es sich um Personen, die aufgrund ihrer einschlägigen Ausbildung und ihrer Erfahrung befähigt sind, Risiken zu erkennen und mögliche Gefährdungen zu vermeiden, die der Betrieb oder die Instandhaltung eines Produkts verursachen kann. Das Dokument enthält Angaben zum korrekten Umgang mit dem Produkt.

Lesen Sie dieses Dokument vor dem Einsatz, damit Sie mit Einsatzbedingungen, Installation und Betrieb vertraut werden. Bewahren Sie das Dokument während der gesamten Einsatzdauer des Gerätes auf.

Befolgen Sie die Sicherheitshinweise.

3 Systembeschreibung

Inhalt

975

3.1 Angaben zum Gerät

1320

Diese Anleitung beschreibt die PDM360-Monitor-Gerätefamilie der **ifm electronic gmbh** mit 16 Bit Mikrocontroller für den mobilen Einsatz:

- PDM360smart: CR1070, CR1071

3.2 Angaben zur Software

1796

Wir beziehen uns in dieser Anleitung auf CoDeSys Version 2.3.

Im "Programmierhandbuch CoDeSys 2.3" erhalten Sie weitergehende Informationen über die Nutzung des Programmiersystems "CoDeSys for Automation Alliance". Dieses Handbuch steht auf der **ifm**-Internetseite als kostenloser Download zur Verfügung:
→ www.ifm.com > Land wählen > [Service] > [Download] > [Steuerungssysteme]
→ **ecomat*mobile***-DVD "Software, tools and documentation"

Die Applikations-Software nach IEC 61131-3 kann vom Anwender komfortabel mit dem Programmiersystem CoDeSys selbst erstellt werden. Für den Einsatz dieser Software auf dem PC gelten folgende Mindest-Systemvoraussetzungen:

- CPU Pentium II, 500 MHz
- Arbeitsspeicher (RAM) 128 MB, empfohlen: 256 MB
- Freier Festplattenspeicher (HD) 100 MB
- Betriebssystem Windows 2000 oder höher
- CD-ROM-Laufwerk

Weitere Details zur aktuellen CoDeSys-Software:
DE: → http://www.3s-software.com/index.shtml?de_oem1
UK: → http://www.3s-software.com/index.shtml?en_oem1
FR: → http://www.3s-software.com/index.shtml?fr_oem1

Der Anwender muss außerdem beachten, welcher Softwarestand (speziell beim R360-Betriebssystem und den Funktionsbibliotheken) zum Einsatz kommt.

HINWEIS

Es müssen immer die zum gewählten Target passenden Software-Stände zum Einsatz kommen:

- des Betriebssystems (`CRnnnn_Vxxyyzz.H86`),
- der Steuerungskonfiguration (`CRnnnn_Vxxyyyzz.CFG`),
- der Gerätebibliothek (`ifm_CRnnnn_Vxxyyzz.LIB`)
- und der weiteren Dateien
  (→ Kapitel *Übersicht der verwendeten Dateien und Bibliotheken* (→ Seite 344)).

CRnnnn	Geräte-Artikelnummer
Vxx: 00...99	Versionsnummer
yy: 00...99	Release-Nummer
zz: 00...99	Patch-Nummer

Dabei müssen der Basisdateiname (z.B. "`CR0020`") und die Software-Versionsnummer "xx" (z.B. "`04`") überall den gleichen Wert haben! Andernfalls geht das Gerät in den STOP-Zustand.

Die Werte für "yy" (Release-Nummer) und "zz" (Patch-Nummer) müssen **nicht** übereinstimmen.

WICHTIG: Folgende Dateien müssen ebenfalls geladen sein:

- die zum Projekt erforderlichen internen Bibliotheken (in IEC 1131 erstellt),
- die Konfigurationsdateien (`*.CFG`)
- und die Target-Dateien (`*.TRG`).

⚠ WARNUNG

Für die sichere Funktion der Applikations-Programme, die vom Anwender erstellt werden, ist dieser selbst verantwortlich. Bei Bedarf muss er zusätzlich entsprechend der nationalen Vorschriften eine Abnahme durch entsprechende Prüf- und Überwachungsorganisationen durchführen lassen.

3.3 Steuerungskonfiguration

1797

Bei dem Steuerungssystem **ecomat*mobile*** handelt es sich um ein Gerätekonzept für den Serieneinsatz. Das bedeutet, dass die Geräte optimal auf den jeweiligen Einsatzfall konfiguriert werden können. Wenn notwendig, können auch Sonderfunktionen und spezielle Hardwarelösungen realisiert werden. Zusätzlich kann auch die aktuelle Version der **ecomat*mobile***-Software über www.ifm.com aus dem Internet geladen werden. → *Target einrichten* (→ Seite 28)

Ob bestimmte in der Dokumentation beschriebene Funktionen, Hardwareoptionen, Ein- und Ausgänge in der betreffenden Hardware verfügbar sind, muss in jedem Fall vor Einsatz der Geräte überprüft werden.

4 Konfigurationen

Inhalt

3615

4.1 Geräteparameter einstellen (Setup)

Inhalt

7306

In diesem Abschnitt erfahren Sie, wie Sie das Gerät mit dem internen Geräte-Setup einstellen können.

Info

Darstellung und Funktionsmöglichkeiten des Setups sind vom Gerät abhängig und können bei kundenspezifschen Geräten von der in dieser Anleitung gezeigten Version abweichen.

4.1.1 Setup starten

9863

HINWEIS

Während sich das Gerät im Setup-Menü befindet, ist keine Kommunikation über die Schnittstellen (CAN und RS232) möglich.

So erreichen Sie das Setup-Menü:

► Beim Einschalten der Versorgungsspannung ca. 1 Sekunde lang die Tastenkombination [F1]+[F5] betätigen.

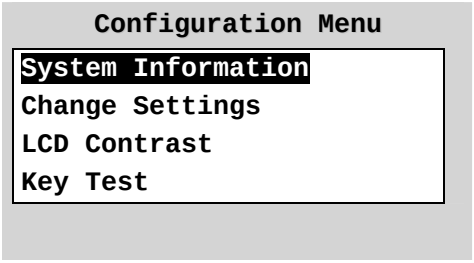


Bild: Setup-Startseite

> Ein dunkler Balken mit invertierter Schrift markiert den gewählten Menüpunkt.

► Mit [▼]/[▲] gewünschten Setup-Menüpunkt wählen und mit [OK] aktivieren.

> Das PDM wechselt auf das gewählte Setup-Menü.

► Wenn Setup-Menübild aktuell:
Mit [esc] auf die Menü-Ebene darüber wechseln.

► Wenn Setup-Startseite aktuell:
Mit [esc] das Setup-Menü verlassen.
→ *PDM-Setup verlassen, Gerät neu starten* (→ Seite 23)

Bedeutung der Setup-Menüpunkte:

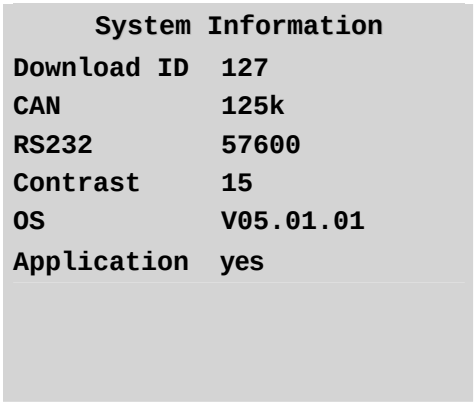
Setup-Feld	Bedeutung
System Information	Aktuelle Einstellungen des Geräts anzeigen → *Aktuelle Geräte-Einstellungen anzeigen* (→ Seite 16)
Change Settings	Einstellungen des Geräts ändern → *Geräte-Einstellungen ändern* (→ Seite 17) • Node-ID der CAN-Schnittstelle anzeigen oder verändern → *CAN Download-ID einstellen* (→ Seite 18) • Übertragungsrate der CAN-Schnittstelle anzeigen oder verändern → *CAN-Baudrate einstellen* (→ Seite 19) • Übertragungsrate der seriellen Schnittstelle anzeigen oder verändern → *Serielle Schnittstelle einstellen* (→ Seite 19) • Passwort ändern → *Passwort ändern* (→ Seite 20) • Gerät auf Werkseinstellungen zurücksetzen → *Gerät auf Werkseinstellungen zurücksetzen* (→ Seite 21)
LCD Contrast	Helligkeit / Kontrast des Displays einstellen → *Helligkeit / Kontrast des Displays einstellen* (→ Seite 22)
Key Test	Tasten und LEDs testen → *Funktion von Tasten und LEDs prüfen* (→ Seite 22)

4.1.2 Aktuelle Geräte-Einstellungen anzeigen

9864

► Im Setup-Startbild mit [OK] umschalten auf das Menübild [System Information].

HINWEIS: In diesem Menübild kann nichts eingestellt werden.



Menübild [System Information] (Beispiel)

> Download ID = Download-Identifier für CoDeSys

> CAN = Übertragungsrate der CAN-Schnittstelle

> RS232 = Übertragungsrate der seriellen Schnittstelle

> Contrast = Kontrasteinstellung für das Display

> OS = Version des geladenen Laufzeitsystems
Wenn kein Laufzeitsystem geladen: OS = no

> Application = yes, wenn Applikation geladen ist,
sonst = no

► Mit [esc] zurück zum Setup-Startbild.

4.1.3 Geräte-Einstellungen ändern

Inhalt

9866

HINWEIS

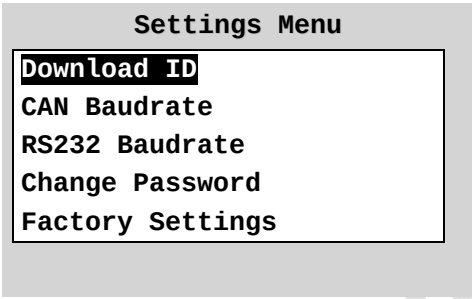
In diesem Menü kann das Gerät zurückgesetzt und das Passwort geändert werden.
Aus Sicherheitsgründen empfehlen wir daher:

► In der IEC-Applikation mit dem FB *SET_PASSWORD* (→ Seite 278) ein Passwort vergeben.

> Dieses Passwort wird beim ersten Start der Applikation aktiviert.

> Mit dem Passwort sind folgende Zugänge zum Gerät geschützt:
- Zugriff auf das Menu [Change Settings],
- Zugriff über den **ifm**-Downloader

► Im Setup-Startbild mit [OK] umschalten auf das Menübild [Change Settings].



Menübild [Change Settings]

- Download ID:
  Download-Identifier für CoDeSys einstellen
- CAN Baudrate:
  Übertragungsrate der CAN-Schnittstelle einstellen
- RS232 Baudrate:
  Übertragungsrate der seriellen Schnittstelle einstellen
- Change Password:
  Passwort ändern
- Factory Settings:
  Gerät auf Werkseinstellungen zurücksetzen

► Mit [esc] zurück zum Setup-Startbild.

CAN Download-ID einstellen

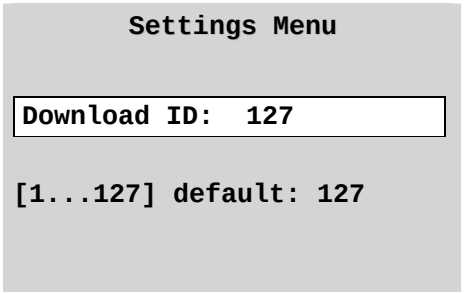
9868

HINWEIS

Der CAN-Download-ID des Geräts muss mit dem in CoDeSys eingestellten CAN-Download-ID übereinstimmen!

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

► Im Menübild [Change Settings] mit [OK] umschalten auf das Menübild [Download ID].



Menübild [Download ID]

Der Identifier dient zur Kommunikation mit dem Programmiersystem und dem **ifm**-Downloader. Der Identifier wird unabhängig vom CAN-Node-ID eingestellt.
Voreingestellt = 127

> Die editierbare Ziffer erscheint invertiert.

► Mit [◀]/[▶] die zu ändernde Ziffer wählen.

► Mit [▼]/[▲] die Ziffer ändern (1...127).

► Mit [OK] den geänderten Wert speichern.

ODER:

► Mit [esc] das Menübild ohne Änderung verlassen.

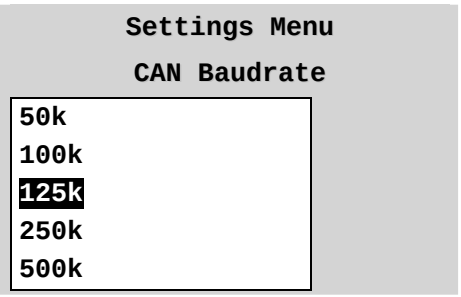
► Mit [esc] zurück zum Menübild [Change Settings].

> Nach einem Neustart (Spannungsversorgung Aus/Ein) arbeitet das Gerät mit den neuen Einstellungen.

CAN-Baudrate einstellen

9869

► Im Menübild [Change Settings] mit [OK] umschalten auf das Menübild [CAN Baudrate].



Menübild [CAN Baudrate]

> Der voreingestellte Wert erscheint invertiert.

► Mit [▼]/[▲] den gewünschten Wert wählen.

► Mit [OK] den geänderten Wert speichern.

ODER:

► Mit [esc] das Menübild ohne Änderung verlassen.

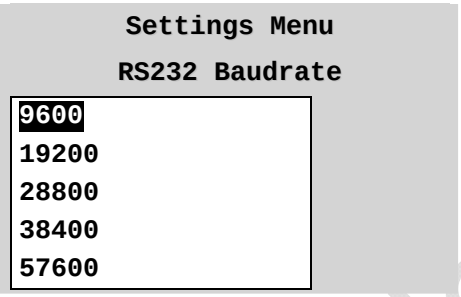
► Mit [esc] zurück zum Menübild [Change Settings].

> Nach einem Neustart (Spannungsversorgung Aus/Ein) arbeitet das Gerät mit den neuen Einstellungen.

Serielle Schnittstelle einstellen

9871

► Im Menübild [Change Settings] mit [OK] umschalten auf das Menübild [RS232 Baudrate].



Menübild [RS232 Baudrate]

> Der voreingestellte Wert erscheint invertiert.

► Mit [▼]/[▲] den gewünschten Wert wählen.

► Mit [OK] den geänderten Wert speichern.

ODER:

► Mit [esc] das Menübild ohne Änderung verlassen.

► Mit [esc] zurück zum Menübild [Change Settings].

> Nach einem Neustart (Spannungsversorgung Aus/Ein) arbeitet das Gerät mit den neuen Einstellungen.

Passwort ändern

9872

In diesem Menübild kann das Passwort geändert werden.

> Mit dem Passwort sind folgende Zugänge zum Gerät geschützt:
- Zugriff auf das Menu [Change Settings],
- Zugriff über den **ifm**-Downloader.

Das Passwort darf bis zu 16 Zeichen haben, jedoch kein Leerzeichen.
Es werden Groß- und Kleinbuchstaben unterschieden.

ACHTUNG

Das Passwort erscheint im Menübild im Klartext lesbar!

► Sorgen Sie dafür, dass kein Unbefugter die Passworteingabe mitlesen kann!

► Im Menübild [Change Settings] mit [OK] umschalten auf das Menübild [Change Password].

Menübild [Change Password]

> Die editierbare Schreibstelle erscheint invertiert.

► Mit [▼]/[▲] das gewünschte Zeichen aus der internen Liste wählen.

► Mit [◀]/[▶] die nächste zu ändernde Schreibstelle wählen.

► usw.

► Mit [OK] das Passwort speichern.

> Die Änderung ist sofort wirksam.

ODER:

► Mit [esc] das Menübild ohne Änderung verlassen.

► Mit [esc] zurück zum Menübild [Change Settings].

HINWEIS

Soll das Passwort (zusätzlich) in der IEC-Applikation gesetzt werden (mittels *SET_PASSWORD* (→ Seite 278))?

► Der FB darf nur beim allerersten Start der Applikation aufgerufen werden.

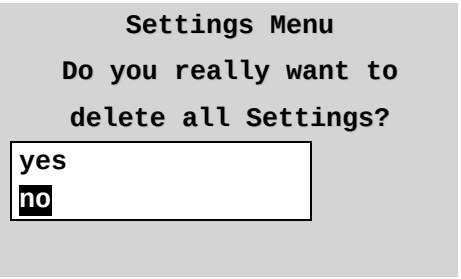
> Andernfalls wird beim nächsten Start der Applikation die im Setup-Menü vorgenommene Änderung des Passworts mit dem Passwort aus dem FB überschrieben.

Gerät auf Werkseinstellungen zurücksetzen

9874

Beim Rücksetzen des Geräts auf die Werkseinstellungen werden folgende Werte gesetzt:
- Download-ID = 127
- CAN-Baudrate = 125 kBaud
- RS232-Baudrate = 9600 Baud
- Kontrast = 10
- Laufzeitsystem wird gelöscht
- Applikation wird gelöscht
- Passwort wird gelöscht

► Im Menübild [Change Settings] mit [OK] umschalten auf das Menübild [Factory Settings].



Menübild [Factory Settings]

► Mit [▲] den Eintrag [yes] wählen.

► Mit [OK] das Gerät auf die Werkseinstellungen zurücksetzen.

ODER:

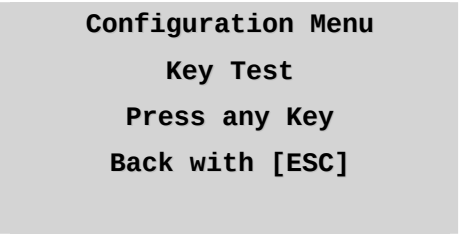
► Mit [esc] das Menübild ohne Änderung verlassen.

► Mit [esc] zurück zum Menübild [Change Settings].

4.1.4 Helligkeit / Kontrast des Displays einstellen

9876

► Im Setup-Startbild mit [OK] umschalten auf das Menübild [LCD Contrast].



Menübild [LCD Contrast]

In diesem Menübild können die Tasten und LEDs auf ihre Funktionsfähigkeit überprüft werden.

Das Drücken einer Taste aktiviert die entsprechende Tastenbeleuchtung.

► Eine Taste betätigen.

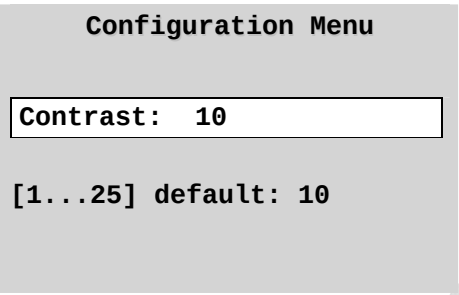
► Die zugehörige LED leuchtet.

► Mit [esc] zurück zum Setup-Startbild.

4.1.5 Funktion von Tasten und LEDs prüfen

9877

► Im Setup-Startbild mit [OK] umschalten auf das Menübild [Key Test].



Menübild [Key Test]

Der Kontrast des Displays kann nur im Geräte Setup geändert werden. Die hier getätigte Einstellung wird spannungsausfallsicher gespeichert.
Voreingestellt = 10

> Die editierbare Ziffer erscheint invertiert.

► Mit [◀]/[▶] die zu ändernde Ziffer wählen.

► Mit [▼]/[▲] die Ziffer ändern (1...25).

► Mit [OK] den geänderten Wert speichern.

> Die Änderung wird sofort wirksam.

ODER:

► Mit [esc] das Menübild ohne Änderung verlassen.

► Mit [esc] zurück zum Setup-Startbild.

4.1.6 PDM-Setup verlassen, Gerät neu starten

9878

In diesem Menü können Sie wählen, ob und wie Sie das PDM-Setup verlassen wollen.

► Im Setup-Startbild die Taste [esc] betätigen.

Wenn eine gültige Applikation gespeichert ist:

> Das PDM startet neu und startet anschließend die Applikation.

Wenn keine gültige Applikation gespeichert ist:

> Das PDM startet neu und zeigt anschließend die Meldung
- "Bootloader..." oder
- "No Application..."

Info

Grundsätzlich können Sie bei jedem Geräte-Neustart mit der Tastenkombination [F1]+[F5] (ca. 1 Sekunde lang betätigen) in das Setup-Menü gelangen.

4.2 Programmierschnittstellen

Inhalt

9880

Zur Programmierung stehen im PDM derzeit folgende Schnittstellen zur Verfügung:

- Programmierung über die serielle Schnittstelle RS232,
- Programmierung über die CAN-Schnittstelle.

4.2.1 Programmierung über die serielle Schnittstelle RS232

9827

Auf der Geräte-Rückseite auf Steckanschluss 2 gibt es 1 serielle Schnittstelle (technische Details → Datenblatt).

Über ein Nullmodemkabel (gekreuzte Datenleitungen) kann die Verbindung zwischen PDM und der seriellen Schnittstelle am Computer hergestellt werden.

CoDeSys-Kommunikationsparameter für die serielle Schnittstelle einstellen

3074

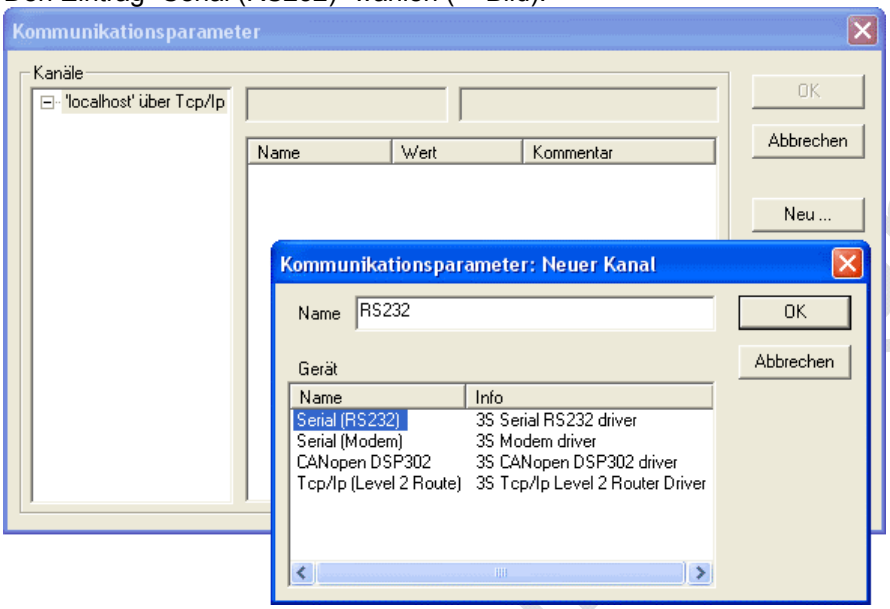
► In CoDeSys [Online] > [Kommunikationsparameter...] klicken.

► Klicken auf [Neu...]

► Fenster "Kommunikationsparameter: Neuer Kanal" erscheint.

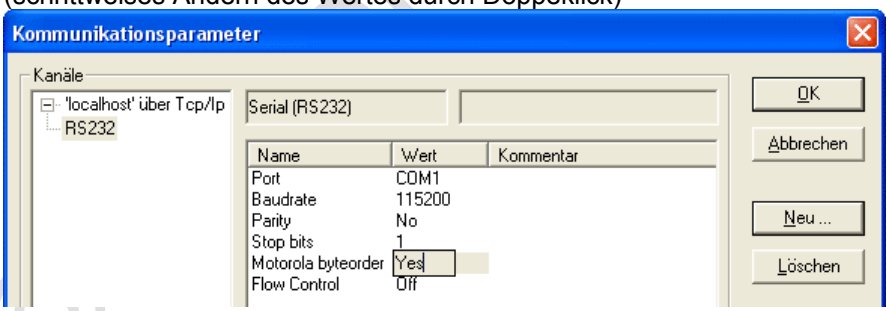
► Einen selbsterklärenden Namen vergeben, z.B. "ifm_RS232".

► Den Eintrag "Serial (RS232)" wählen (→ Bild):



► Für den neuen Kanal die folgenden Kommunikationsparameter eintragen (→ Bild):
- [Baudrate] = 115200
- [Motorola byteorder] = Yes (für alle PDM, außer CR107n)
- [Motorola byteorder] = No (für alle Controller und CR107n)

(schrittweises Ändern des Wertes durch Doppeklick)



► Kommunikationsparameter mit [OK] übernehmen.

> Nun sollten CoDeSys und das Gerät über die serielle Schnittstelle kommunizieren können.

4.2.2 Programmierung über die CAN-Schnittstelle

3028

Info

Wegen der geringen Übertragungsgeschwindigkeit und der großen Datenmengen ist die Programmierung von PDMs über die CAN-Schnittstelle weniger zu empfehlen.

Voraussetzungen:

- ► Den CAN-Adapter (optional, z.B. Artikel Nr. EC2112) mit dem PC verbinden.
- ► Kabelverbindung zwischen CAN-Adapter und PDM herstellen. Dazu muss zwischen CAN-H und CAN-L auf beiden Seiten der Kabelverbindung je ein Abschlusswiderstand (120 Ohm) vorhanden sein.

CAN-Interface konfigurieren:

- ► Die PC-seitige Konfiguration des CAN-Adapters entnehmen Sie bitte der dazu gehörenden Dokumentation.

HINWEIS

Der CAN-Download-ID des Geräts muss mit dem in CoDeSys eingestellten CAN-Download-ID übereinstimmen!

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

CoDeSys-Kommunikationsparameter für die CAN-Schnittstelle einstellen

3073

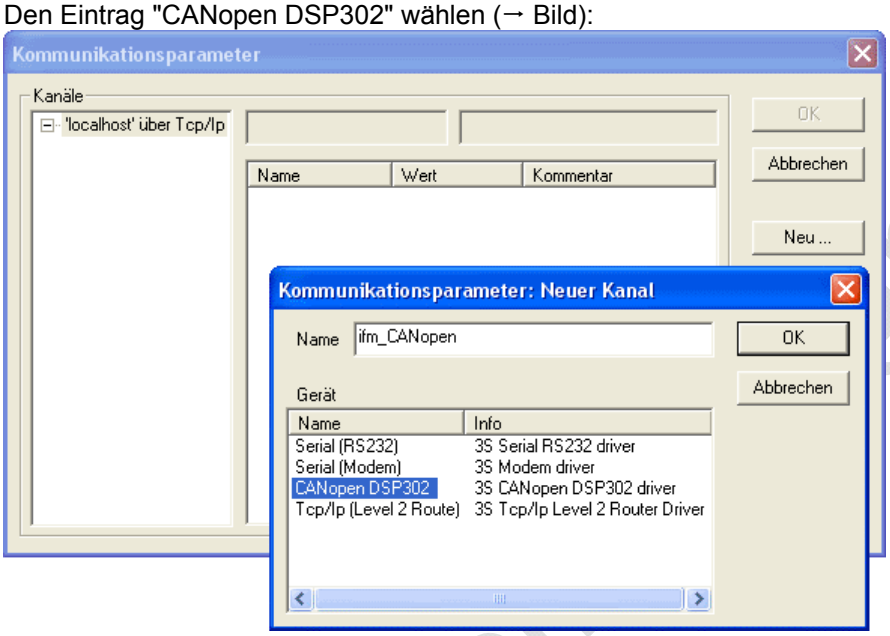
► In CoDeSys [Online] > [Kommunikationsparameter...] klicken.

► Klicken auf [Neu...]

► Fenster "Kommunikationsparameter: Neuer Kanal" erscheint.

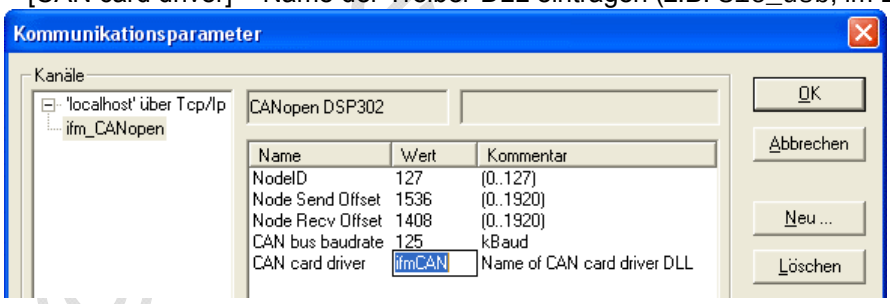
► Einen selbsterklärenden Namen vergeben, z.B. "ifm_CANopen".

► Den Eintrag "CANopen DSP302" wählen (→ Bild):



► Neue Parameter mit [OK] übernehmen.

Für den neuen Kanal z.B. die folgenden Kommunikationsparameter eintragen (→ Bild):
- [NodeID] = `127` eintragen (Default-Einstellung für alle **ecomat*mobile***-Controller und PDM360-Geräte)
- [CAN card driver] = Name der Treiber-DLL eintragen (z.B. `Sie_usb`; im Bild: `ifmCAN`)



► Kommunikationsparameter mit [OK] übernehmen.

> Nun sollten CoDeSys und das Gerät über die CAN-Schnittstelle kommunizieren können.

4.3 Programmiersystem einrichten

Inhalt

3968

4.3.1 Programmiersystem manuell einrichten

Inhalt

3963

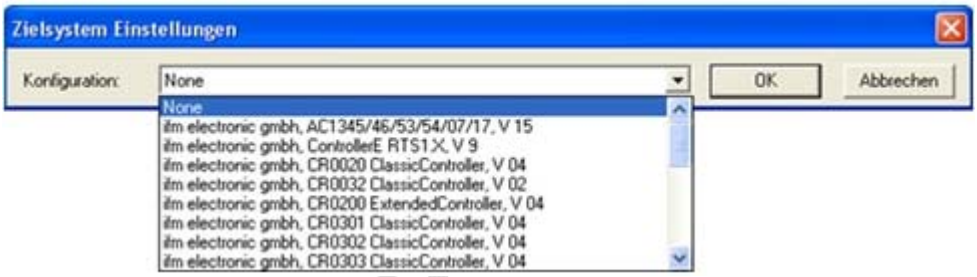
Target einrichten

2687

11379

Beim Erstellen eines neuen Projektes in CoDeSys muss die dem Gerät entsprechende Target-Datei geladen werden.

► Die gewünschte Target-Datei im Dialogfenster wählen (→ Screenshot).



Grafik: Zielsystem Einstellungen (Beispiel)

> Die Target-Datei stellt für das Programmiersystem die Schnittstelle zur Hardware her.

> Gleichzeitig mit Wahl des Targets werden automatisch einige wichtige Bibliotheken und die Steuerungskonfiguration geladen.

► Bei Bedarf geladene Bibliotheken wieder entfernen oder durch weitere Bibliotheken ergänzen.

► Immer die passende Geräte-Bibliothek `ifm_CRnnnn_Vxxyyzz.LIB` manuell ergänzen!

HINWEIS

Es müssen immer die zum gewählten Target passenden Software-Stände zum Einsatz kommen:

- des Betriebssystems (`CRnnnn_Vxxyyzz.H86` / `CRnnnn_Vxxyyzz.RESX`),
- der Steuerungskonfiguration (`CRnnnn_Vxx.CFG`),
- der Gerätebibliothek (`ifm_CRnnnn_Vxxyyzz.LIB`)
- und der weiteren Dateien
  (→ Kapitel *Übersicht der verwendeten Dateien und Bibliotheken* (→ Seite 344)).

CRnnnn	Geräte-Artikelnummer
Vxx: 00...99	Versionsnummer
yy: 00...99	Release-Nummer
zz: 00...99	Patch-Nummer

Dabei müssen der Basisdateiname (z.B. "`CR0032`") und die Software-Versionsnummer "xx" (z.B. "`02`") überall den gleichen Wert haben! Andernfalls geht das Gerät in den STOP-Zustand

Die Werte für "yy" (Release-Nummer) und "zz" (Patch-Nummer) müssen **nicht** übereinstimmen.

WICHTIG: Folgende Dateien müssen ebenfalls geladen sein:

- die zum Projekt erforderlichen internen Bibliotheken (in IEC 1131 erstellt),
- die Konfigurationsdateien (`*.CFG`)
- und die Target-Dateien (`*.TRG`).

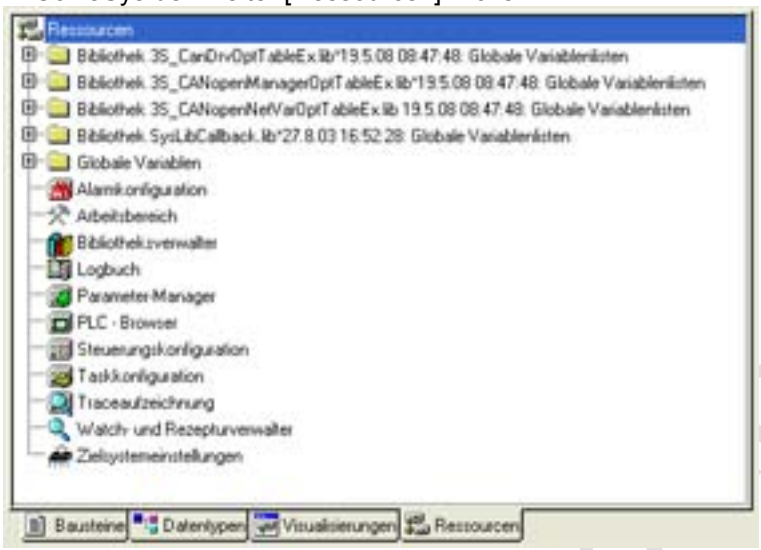
Steuerungskonfiguration aktivieren (z.B. CR0020)

2688

Bei der Konfiguration des Programmiersystems (→ vorheriger Abschnitt) erfolgte automatisch auch die Steuerungskonfiguration.

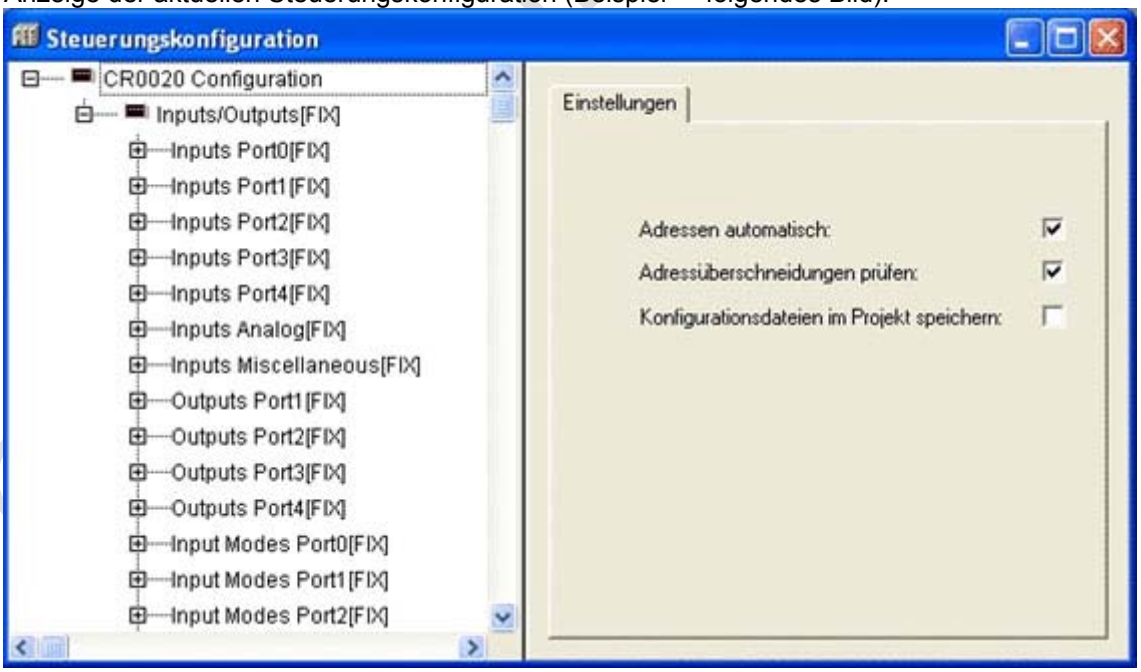
Den Punkt [Steuerungskonfiguration] erreicht man über den Reiter [Ressourcen]. Über einen Doppelklick auf den Punkt [Steuerungskonfiguration] öffnet sich das entsprechende Fenster.

► In CoDeSys den Reiter [Ressourcen] klicken:



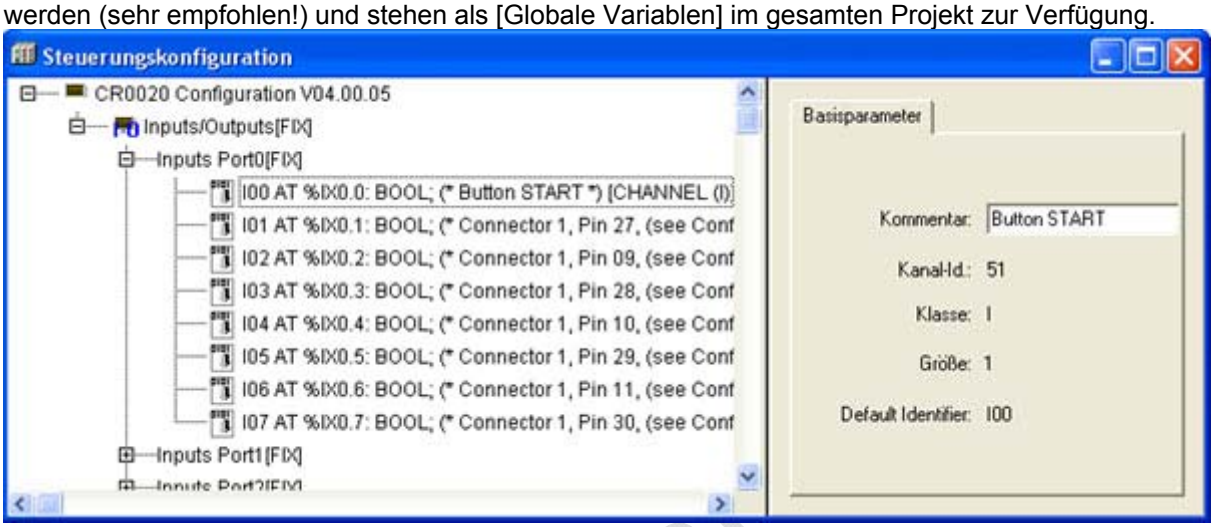
► In der linken Spalte Doppelklick auf [Steuerungskonfiguration]

> Anzeige der aktuellen Steuerungskonfiguration (Beispiel → folgendes Bild):



Durch die Konfiguration erhält der Anwender in der Programmumgebung Folgendes verfügbar:

- alle wichtigen System- und Fehlermerker
  Je nach Anwendung und Applikations-Programm müssen diese Merker bearbeitet und ausgewertet werden. Der Zugriff erfolgt über deren symbolischen Namen.
- die Struktur der Ein- und Ausgänge
  Diese können im Fenster [Steuerungskonfiguration] (→ Bild unten) direkt symbolisch bezeichnet werden (sehr empfohlen!) und stehen als [Globale Variablen] im gesamten Projekt zur Verfügung.



4.3.2 Programmiersystem über Templates einrichten

Inhalt

3977

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

HINWEIS

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

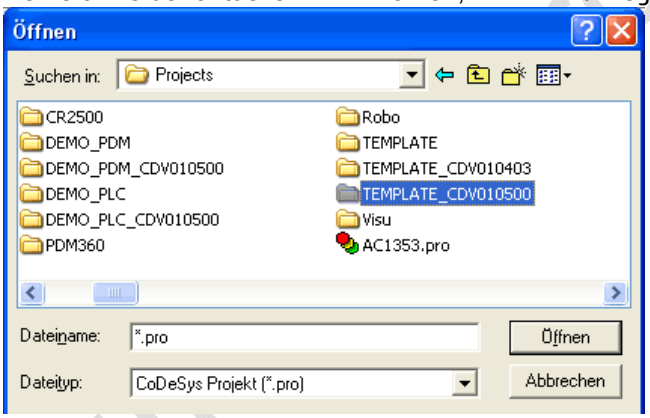
► Die gewünschte dort gespeicherte Vorlage in CoDeSys öffnen mit:
[Datei] > [Neu aus Vorlage…]

> CoDeSys legt ein neues Projekt an, dem der prinzipielle Programmaufbau entnommen werden kann. Es wird dringend empfohlen, dem gezeigten Schema zu folgen.
→ Kapitel *Programmiersystem über Templates einrichten* (→ Seite 32)

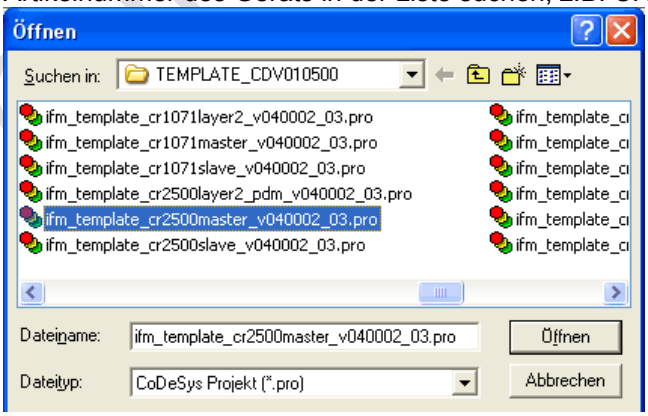
Wie richten Sie das Programmiersystem schnell und einfach ein? (z.B. CR2500)

► Im CoDeSys-Menü wählen: [Datei] > [Neu aus Vorlage...].

► Verzeichnis der aktuellen DVD wählen, z.B. `...\Projects\TEMPLATE_CDV010500`:



► Artikelnummer des Geräts in der Liste suchen, z.B. CR2500 als CANopen-Master:

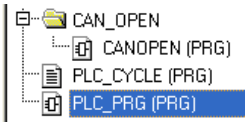
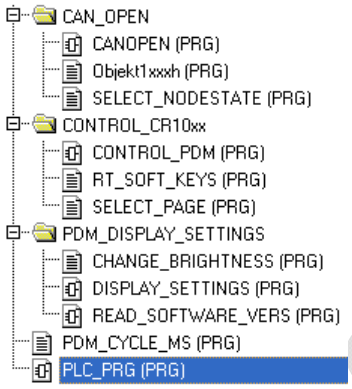


► Achten Sie auch auf die richtige Programm-Version!

► 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:
	

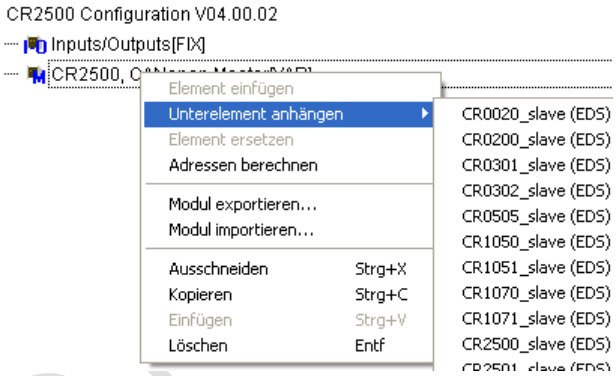
(Über die Ordnerstrukturen in Templates → Kapitel *Über die ifm-Templates* (→ Seite 35)).

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

► 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.
Hinweis: [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* (→ Seite 39)

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 010500

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

3980

Die vorgenannten Ordner enthalten die folgenden Programme und Bausteine:

Bausteine im Ordner CAN_OPEN	Beschreibung
CANOPEN	für Controller und PDM, CAN-Betrieb als Master: Enthält folgende parametrierte Bausteine: - CAN1_MASTER_EMCY_HANDLER (→ *CANx_MASTER_EMCY_HANDLER* (→ Seite 164)), - CAN1_MASTER_STATUS (→ *CANx_MASTER_STATUS* (→ Seite 169)), - SELECT_NODESTATE (→ unten).
CANOPEN	für Controller und PDM, CAN-Betrieb als Slave: Enthält folgende parametrierte Bausteine: - CAN1_SLAVE_EMCY_HANDLER (→ *CANx_SLAVE_EMCY_HANDLER* (→ Seite 176)), - CAN1_SLAVE_STATUS (→ *CANx_SLAVE_STATUS* (→ Seite 181)), - SELECT_NODESTATE (→ unten).
Objekt1xxxh	für Controller und PDM, 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	für PDM, 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 I_O_CONFIGURATION	Beschreibung
CONF_IO_CRnnnn	für Controller, 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	für Controller, 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	für Controller, 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 Ordner CONTROL_CR10nn	Beschreibung
CONTROL_PDM	für 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_MAIN_MAPPER*), - PDM_PAGECONTROL (→ *PDM_PAGECONTROL* (→ Seite 296)), - PDM_TO_PLC (→ unten), - SELECT_PAGE (→ unten).
PDM_TO_PLC	für PDM, 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_x_TRANSMIT*), - CAN_1_RECEIVE (→ *CAN_x_RECEIVE*).
RT_SOFT_KEYS	für PDM, 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	für PDM, 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	für PDM, CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Organisiert Helligkeit / Kontrast des Monitors.
DISPLAY_SETTINGS	für PDM, 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 (→ *PDM_SET_RTC*), - READ_SOFTWARE_VERS (→ unten), - TP (→ 3S).
READ_SOFTWARE_VERS	für PDM, CAN-Betrieb mit Layer 2 oder als Master oder als Slave: Zeigt die Software-Version. Enthält folgende parametrierte Bausteine: - DEVICE_KERNEL_VERSION1 (→ *DEVICE_KERNEL_VERSION1*), - DEVICE_RUNTIME_VERSION (→ *DEVICE_RUNTIME_VERSION*), - LEFT (→ 3S).

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

Struktur der Visualisierungen in den Templates

3979

Für folgende Geräte verfügbar:
- BasicDisplay: CR0451
- PDM: CR10nn

Die Visualisierungen sind wie folgt in Ordnern strukturiert:

Ordner	Bild-Nr.	Beschreibung Inhalt
START_PAGE	P00001	Einstellung / Anzeige von... - Node-ID - CAN-Baudrate - Status - GuardErrorNode - SPS-Zykluszeit
__MAIN_MENUES	P00010	Menübild: - Display-Setup
____MAIN_MENUE_1		
______DISPLAY_SETUP		
________1_DISPLAY_SETUP1	P65000	Menübild: - Software-Version - Helligkeit / Kontrast - Echtzeituhr anzeigen / setzen
__________1_SOFTWARE_VERSION	P65010	Anzeige der Software-Version
__________2_BRIGHTNESS	P65020	Einstellen von Helligkeit / Kontrast
__________3_SET_RTC	P65030	Echtzeituhr anzeigen / setzen

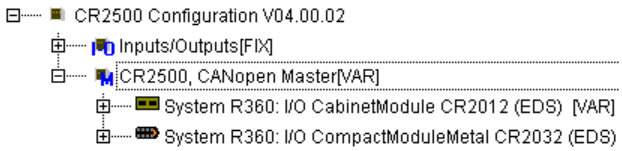
In den Templates haben wir die Bildnummern in 10er-Schritten organisiert. So können Sie mit Hilfe eines Bildnummer-Offsets in verschiedene Sprachversionen der Visualisierungen schalten.

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:

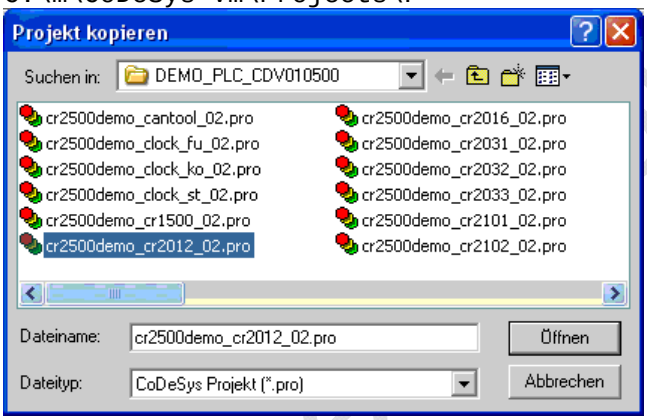


Beispiel: Steuerungskonfiguration

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

► CoDeSys-Projekt speichern!

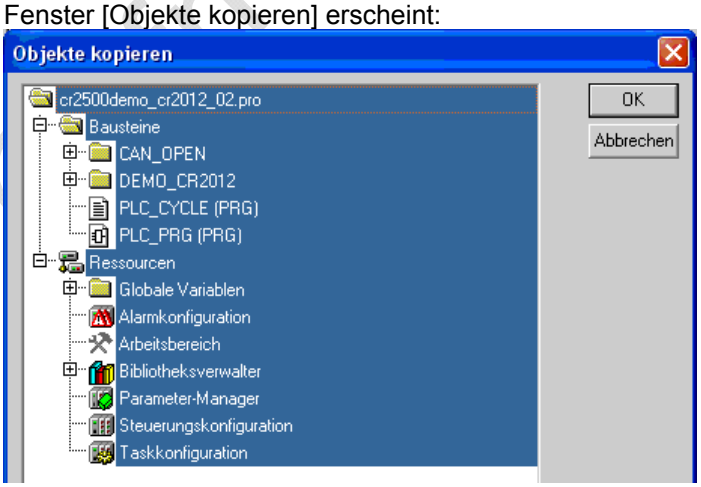
► In CoDeSys mit [Projekt] > [kopieren...] das Projekt öffnen, das die gewünschte Funktion enthält: z.B. `CR2500Demo_CR2012_02.pro` aus dem Verzeichnis `DEMO_PLC_CDV…` unter `C:\…\CoDeSys V…\Projects\`:



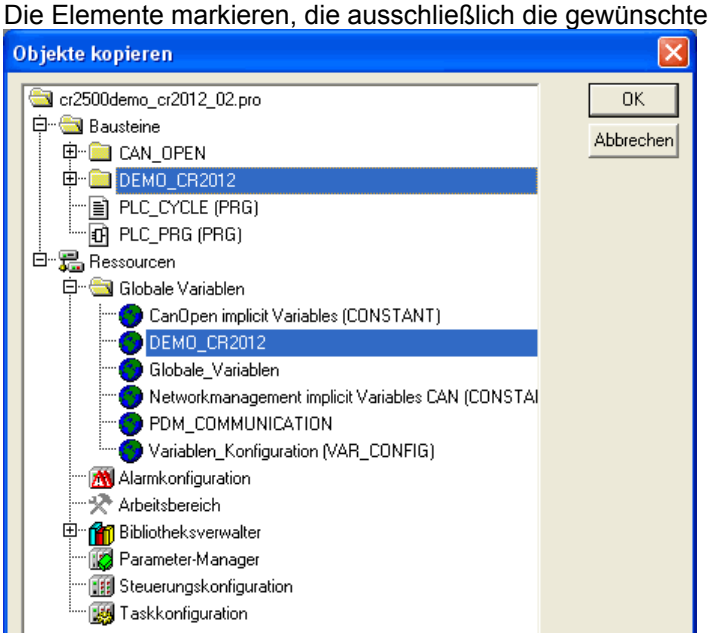
► Wahl mit [Öffnen] bestätigen.

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

> Fenster [Objekte kopieren] erscheint:



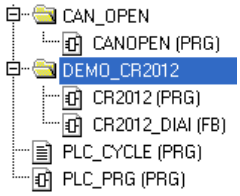
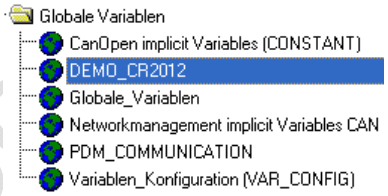
► Die Elemente markieren, die ausschließlich die gewünschte Funktion enthalten, hier z.B.:



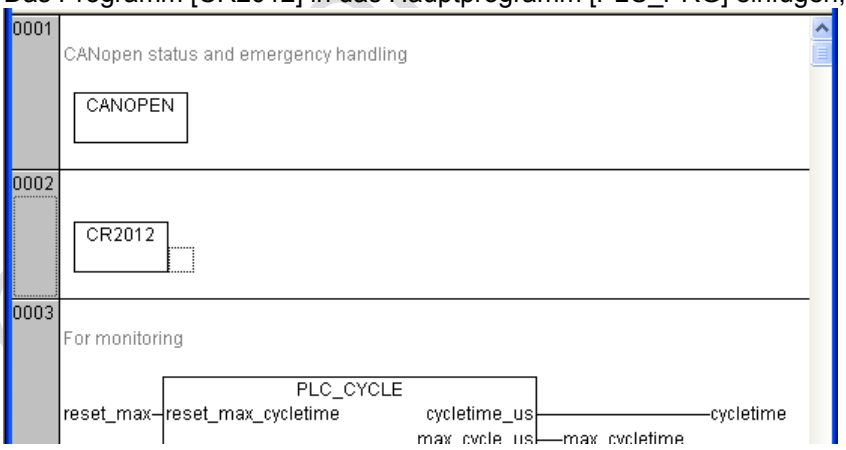
HINWEIS: In anderen Fällen können auch Bibliotheken und / oder Visualisierungen erforderlich sein.

► 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].

4.3.3 ifm-Demo-Programme

Inhalt

3982

Im Verzeichnis `DEMO_PLC_CDV…` (für Controller) oder `DEMO_PDM_CDV…` (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**-Demos 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 Demos 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 010500

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 Highside, 4x Digital-Eingang Lowside, 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_Interrupt_xx.pro`	Beispiel mit *SET_INTERRUPT_XMS* (→ Seite 283).
`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* (→ Seite 201), - *PWM100* (→ Seite 231).
`CR2500Demo_RS232_xx.pro`	Beispiel für den Empfang von Daten auf der seriellen Schnittstelle mit Hilfe des Windows-Hyperterminal.

`StartersetDemo.pro`
`StartersetDemo2.pro`
`StartersetDemo2_fertig.pro`
	Verschiedene Übungen zum E-Learning mit dem Starterset EC2074.

_xx = Angabe der Demo-Version

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.

`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].
`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].

Demo-Programm	Funktion
`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

_xx = Angabe der Demo-Version

4.4 Hinweise zur Anschlussbelegung

1426

Die Anschlussbelegungen (→ Montageanleitungen der Geräte, Kapitel "Anschlussbelegung") beschreiben die Standard-Gerätekonfigurationen. Die Anschlussbelegung dient der Zuordnung der Ein- und Ausgangskanäle zu den IEC-Adressen und den Geräteanschlussklemmen.

Beispiele:

12 GND_A

12	Klemmennummer
GND_A	Klemmenbezeichnung

30 %IX0.7 BL

30	Klemmennummer
%IX0.7	IEC-Adresse für einen binären Eingang
BL	hardwaremäßige Ausführung des Eingangs, hier: **B**inär **L**ow-Side

47 %QX0.3 BH/PH

47	Klemmennummer
%QX0.3	IEC-Adresse für einen binären Ausgang
BH/PH	Hardwaremäßige Ausführung des Ausgangs, hier: **B**inär-**H**igh-Side oder **P**WM-**H**igh-Side

Die einzelnen Kürzel haben folgende Bedeutung:

A	Analog-Eingang
BH	Binärer Eingang/Ausgang, High-Side
BL	Binärer Eingang/Ausgang, Low-Side
CYL	Eingang Periodendauermessung
ENC	Eingang Drehgebersignale
FRQ	Frequenzeingang
H-Bridge	Ausgang mit H-Brücken-Funktion
PWM	**P**uls**w**eiten-**m**oduliertes Signal
PWM_I	PWM-Ausgang mit Strommessung
IH	Impuls-/Zählereingang, High-Side
IL	Impuls-/Zählereingang, Low-Side
R	Rücklesekanal für einen Ausgang

Zuordnung der Ein-/Ausgangskanäle:

Je nach Gerätekonfiguration steht an einer Geräteklemme ein Eingang und/oder ein Ausgang zur Verfügung (→ Katalog, Montageanleitung oder Datenblatt des jeweiligen Gerätes).

HINWEIS

Kontakte von Reed-Relais können (reversibel) verkleben, wenn sie ohne Vorwiderstand an den Geräte-Eingängen angeschlossen werden.

- ► **Abhilfe:** Vorwiderstand zum Reed-Relais installieren:
  Vorwiderstand = max. Eingangsspannung / zulässiger Strom im Reed-Relais
  Beispiel: 32 V / 500 mA = 64 Ohm
- ► Der Vorwiderstand darf 5 % des Eingangswiderstands RE des Geräte-Eingangs (→ Datenblatt) nicht überschreiten. Sonst wird das Signal nicht als TRUE erkannt.
  Beispiel:
  RE = 3 000 Ohm
  ⇨ max. Vorwiderstand = 150 Ohm

4.5 Erste Schritte

Inhalt

3044

► Geräteparameter einstellen (→ *Geräteparameter einstellen (Setup)* (→ Seite 14)).

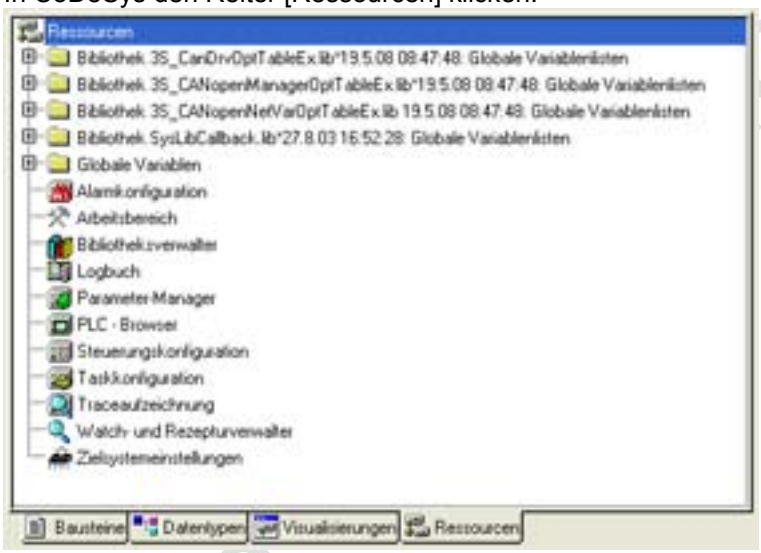
► Programmiersystem einrichten (→ *Programmiersystem einrichten* (→ Seite 28)).

4.5.1 Fehlende Bibliotheken einfügen

9882

Die Gerätedaten sind dem CoDeSys-Projekt bekannt, die Steuerungskonfiguration ist aktiviert. Einige Bibliotheken werden bereits automatisch geladen. Abhängig von der Applikation müssen Sie dem Projekt noch einige Bibliotheken hinzufügen. Die Beschreibung dazu folgt hier.

► In CoDeSys den Reiter [Ressourcen] klicken:



► In der linken Spalte Doppelklick auf [Bibliotheksverwalter]

► Mit Taste [Einfg] oder Menü [Einfügen] > [weitere Bibliothek ...] die Bibliotheks-Übersicht dieses Geräts anfordern.

> Das Fenster [Öffnen] erscheint mit der Bibliotheks-Übersicht.

Die hier gezeigten Bibliotheken haben folgende Funktionen:

Bibliothek	Bedeutung
`ifm_CRnnnn_CAN1openMaster_Vxxyyzz`	CANopen-Master für Schnittstelle CAN1
`ifm_CRnnnn_CAN1openSlave_Vxxyyzz`	CANopen-Slave für Schnittstelle CAN1
`ifm_CRnnnn_CAN2openMaster_Vxxyyzz`	CANopen-Master für Schnittstelle CAN2
`ifm_CRnnnn_CAN2openSlave_Vxxyyzz`	CANopen-Slave für Schnittstelle CAN2
`ifm_CRnnnn_Vxxyyzz`	Geräte-Bibliothek

HINWEIS

Es müssen immer die zum gewählten Target passenden Software-Stände zum Einsatz kommen:

- des Betriebssystems (`CRnnnn_Vxxyyzz.H86` / `CRnnnn_Vxxyyzz.RESX`),
- der Steuerungskonfiguration (`CRnnnn_Vxx.CFG`),
- der Gerätebibliothek (`ifm_CRnnnn_Vxxyyzz.LIB`)
- und der weiteren Dateien
  (→ Kapitel *Übersicht der verwendeten Dateien und Bibliotheken* (→ Seite 344)).

CRnnnn	Geräte-Artikelnummer
Vxx: 00...99	Versionsnummer
yy: 00...99	Release-Nummer
zz: 00...99	Patch-Nummer

Dabei müssen der Basisdateiname (z.B. "`CR0032`") und die Software-Versionsnummer "xx" (z.B. "`02`") überall den gleichen Wert haben! Andernfalls geht das Gerät in den STOP-Zustand

Die Werte für "yy" (Release-Nummer) und "zz" (Patch-Nummer) müssen **nicht** übereinstimmen.

WICHTIG: Folgende Dateien müssen ebenfalls geladen sein:

- die zum Projekt erforderlichen internen Bibliotheken (in IEC 1131 erstellt),
- die Konfigurationsdateien (`*.CFG`)
- und die Target-Dateien (`*.TRG`).

► Wenn noch nicht im Projekt integriert, nacheinander die folgenden Bibliotheken einfügen:
 - Standard-Bibliothek `Standard.Lib` aus `C:\...\CoDeSys\Library\`
 - Geräte-Bibliothek `CRnnnn_Vxxyyzz.Lib` aus `C:\...\CoDeSys\Targets\ifm\Library\ifm_CRnnnn\`

► Das Projekt mit [Strg]+[s] sichern.

> Das Projekt ist nun vorbereitet für das PLC-Programm der Applikation.

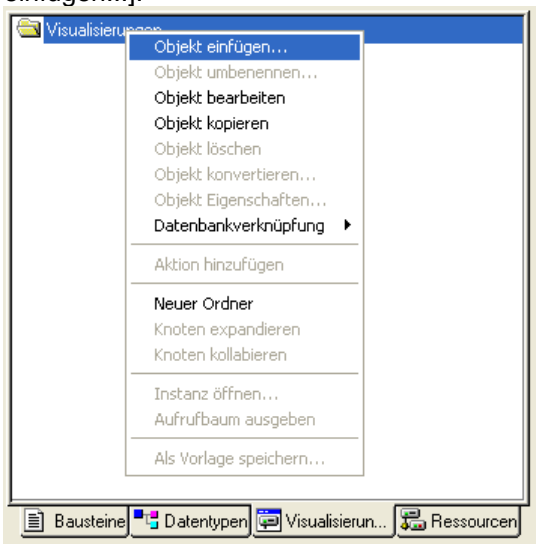
4.5.2 Visualisierung erstellen

3100

Für dieses Beispiel erstellen wir zuerst die Visualisierung, erst anschließend das PLC-Programm dazu.

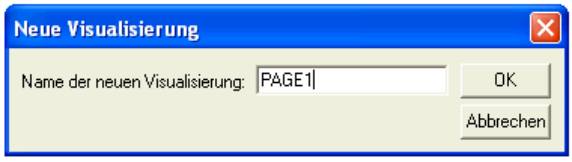
► In CoDeSys den Reiter [Visualisierungen] klicken.

► Neben dem Ordner-Symbol Rechtsklick auf [Visualisierungen], gefolgt von Klick auf [Objekt einfügen...]:



> Das Fenster [Neue Visualisierung] erscheint.

► Hinter [Name der neuen Visualisierung] in Großbuchstaben (!) den Namen des ersten Bildes eintragen (max. 8 Zeichen, keine Leerzeichen!):



► Mit [OK] übernehmen.

> CoDeSys öffnet das Zeichenfeld für diese Visualisierung:

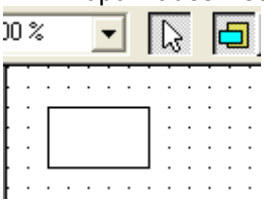
WICHTIG: Das Zeichenfeld entspricht der Größe des Displays.

Zum Umgang mit dem Visualisierungs-Editor:
→ CoDeSys-Onlinehilfe oder
→ CoDeSys-Programmierhandbuch → **ecomatmobile**-DVD "Software, tools and documentation".

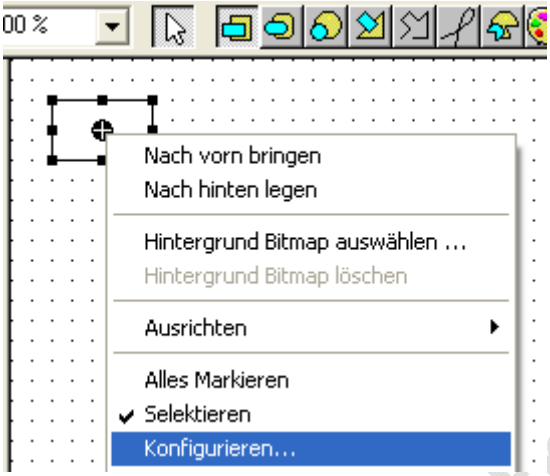
Zur Komplettierung unseres Test-Programms erstellen wir nun eine einfache Darstellung.

► Das Symbol "Rechteck" markieren.

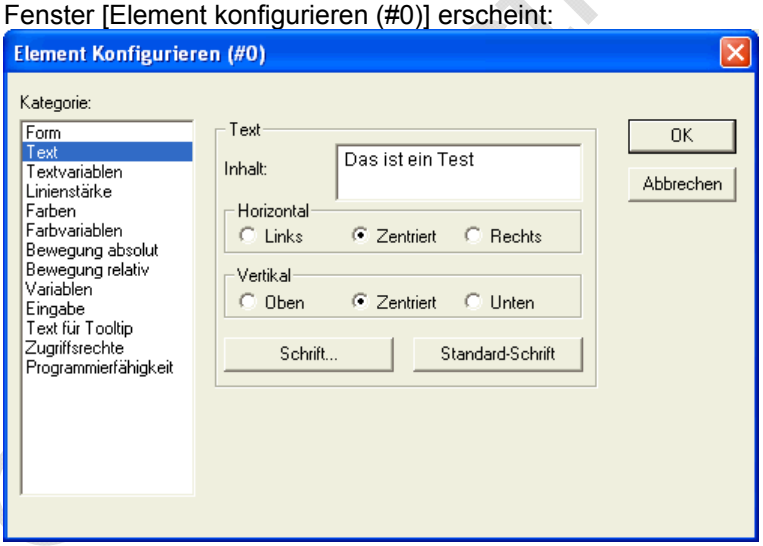
► Auf der Zeichenfläche auf einen Punkt zeigen als Beginn eines Rechtecks.
Linke Maustaste drücken und festhalten, dabei ein Rechteck in beliebiger Richtung aufziehen.
Am Endpunkt des Rechtecks die Maustaste wieder loslassen:



► Mit Rechtsklick auf das Rechteck Kontextmenü öffnen und [Konfigurieren...] wählen:



> Fenster [Element konfigurieren (#0)] erscheint:



► Im Feld [Kategorie] den Eintrag [Text] markieren.

► Im Feld [Text] > [Inhalt] einen Anzeigetext eintragen (→ Bild oben).

► Eintrag mit [OK] übernehmen.

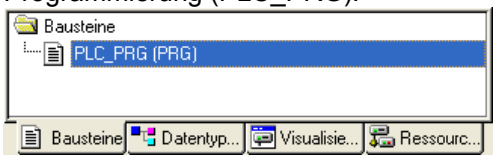
► Projekt zwischendurch mit [Strg]+[s] sichern!

4.5.3 PLC-Programm erstellen

9885

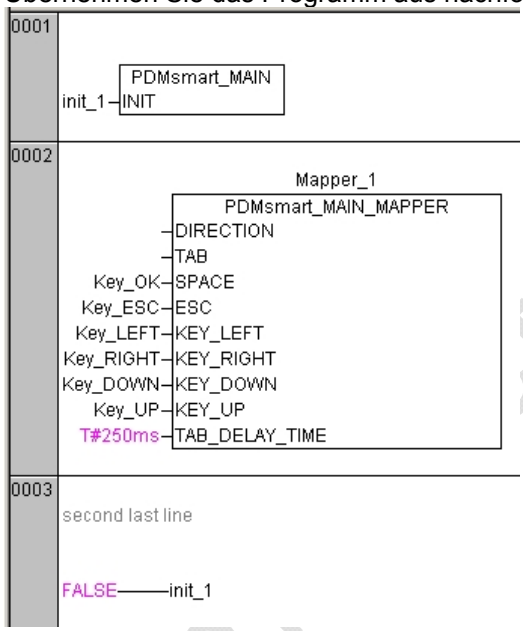
Für dieses Beispiel erstellen wir zuerst die Visualisierung, erst anschließend das PLC-Programm dazu.

► Für die eigentliche Programmierung wechseln Sie nun über den Reiter [Bausteine] wieder in die Programmierung (PLC_PRG):



Für ein lauffähiges Programm sind nur wenige Netzwerke erforderlich. Um wesentliche Funktionalitäten des Gerätes nutzen zu können, benötigen Sie lediglich folgende Bausteine:
- PDMsmart_MAIN aus der Bibliothek `ifm_CR1071_init_Vxxyyzz.LIB` und
- PDMsmart_MAIN_MAPPER aus der Bibliothek `ifm_CRnnnn_Vxxyyzz.LIB`.

► Übernehmen Sie das Programm aus nachfolgendem Beispiel:



> Sie können jetzt bereits Folgendes nutzen:
- den Tastenstatus abfragen oder
- die LEDs setzen.

Die Variable `init_1` wird bei der Definition bereits auf TRUE gesetzt:
`init_1: BOOL := TRUE`

► Am Ende des ersten Zyklus müssen Sie die Variable `init_1` wieder zurücksetzen:
→ Netzwerk 3 im obigen Beispiel.

Info

Alle wichtigen Systemvariablen für das PDM360smart, wie z.B. Taste `F1`, finden Sie hier:
→ unter dem Reiter [Ressourcen] oben in der Liste:
—→ Bibliothek `ifm_CRnnnn_Vxxyyzz.LIB`
———→ Globale Variablen <R> und
———→ PDMsmart_MAIN <R>

4.6 Geräte-Update auf neue Software-Version

Inhalt

3084

Immer, wenn es zu wesentlichen Verbesserungen in der Betriebsystem-Software oder des CoDeSys-Laufzeitsystems kommt, gibt **ifm** davon eine neue Version heraus. Die Versionen werden fortlaufend durchnummeriert (V01, V02, V03, ...).

Welche neuen Zusatzfunktionen die neue Softwareversion enthält, entnehmen Sie bitte der jeweiligen Dokumentation. Beachten Sie, ob in der Dokumentation auf besondere Anforderungen an die Hardware-Version hingewiesen wird.

Wenn Sie im Besitz eines Gerätes mit einer älteren Version sind und wenn die Bedingungen für die Hardware und Ihr Projekt stimmen, können Sie Ihr Gerät durch Aktualisieren der Software auf den neuen Software-Stand bringen.

4.6.1 Was wird benötigt?

9888

Was wird benötigt?	Woher?
aktuelle CoDeSys-Version	z.B. **ecomat*mobile***-DVD "Software, tools and documentation"
Programm **ifm**-Downloader	z.B. **ecomat*mobile***-DVD "Software, tools and documentation"
aktuelle Dateien des Software-Updates	• **ecomat*mobile***-DVD "Software, tools and documentation" • ifm-Downloadbereich → www.ifm.com > Land wählen > [Service] > [Download] > [Steuerungssysteme]

4.6.2 Applikations-Programm übernehmen?

9891

Soll das im Gerät gespeicherte Applikations-Programm nach dem Geräte-Update wieder zur Verfügung stehen? Dann müssen vor dem Geräte-Update folgende Punkte abgearbeitet werden:

► In CoDeSys mit [Projekt] > [Exportieren...] das Applikations-Programm exportieren.

► Mit [Start Programme] > [ifm electronic] > [CoDeSys V2.3] > [InstallTarget] das dem Geräte-Update entsprechende Zielsystem installieren.

► In CoDeSys ein neues Projekt anlegen mit der aktuellen Version des Zielsystems.

► In CoDeSys mit [Projekt] > [Importieren...] das exportierte Applikations-Programm importieren.

► Falls erforderlich, die Bibliotheken im Projekt aktualisieren.

► In CoDeSys mit [Projekt] > [Alles bereinigen] das Projekt zum Übersetzen vorbereiten.

► Das Projekt speichern.

► In CoDeSys mit [Projekt] > [Alles übersetzen] das Projekt zur Übertragung auf das Gerät vorbereiten.

► Das Geräte-Update durchführen (→ folgendes Kapitel).

4.6.3 Geräte-Update mit dem Downloader

9889

Das Betriebssystem wird mit dem eigenständigen Programm **ifm**-Downloader in die Steuerung übertragen.

► Im Menü mit [Interface] die Schnittstelle wählen (RS232 oder CAN).

► Im Menü mit [Download] die Betriebssystemdatei wählen (z.B. `ifm_CR1071_V030002.H86`).

> Der Download startet automatisch nach dem Wählen der Betriebssystemdatei.

> Das Applikations-Programm wird dabei gelöscht.

> Das Geräte-Update des Betriebssystems ist erfolgreich abgeschlossen.

4.6.4 Applikations-Programm in die Steuerung laden

9892

Wenn das Geräte-Update erfolgreich abgeschlossen ist, dann kann das Applikations-Programm in das Gerät geladen werden.

► In CoDeSys das (entsprechend dem Update) aktualisierte Projekt öffnen.

► In CoDeSys mit [Online] > [Einloggen] das Programmiersystem mit dem Gerät verbinden.

► Mit [Online] > [Daten in Steuerung schreiben] das aktualisierte Projekt in die Steuerung laden.

> FERTIG!

5 Begrenzungen und Programmierhinweise

Inhalt

3055

Hier zeigen wir Ihnen die Grenzen des Geräts und helfen Ihnen mit Programmierhinweisen.

5.1 Leistungsgrenzen des Geräts

7358

HINWEIS

Beachten Sie die Grenzen des Geräts! → Datenblatt

5.1.1 CPU-Frequenzen

8005

► Beachten Sie, welche CPU in dem eingesetzten Gerät verwendet wird:

Controller-Familie / Artikel-Nr.	CPU-Frequenz [MHz]
BasicController: CR040n	50
CabinetController: CR0301, CR0302	20
CabinetController: CR0303	40
ClassicController: CR0020, CR0505	40
ClassicController: CR0032, CR0033	150
ExtendedController: CR0200	40
ExtendedController: CR0232, CR0233	150
SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506	40
SmartController: CR25nn	20

Monitor-Familie / Artikel-Nr.	CPU-Frequenz [MHz]
BasicDisplay: CR0451	50
PDM360: CR1050, CR1051	50
PDM360compact: CR1052, CR1053, CR1055, CR1056	50
PDM360NG: CR108n	400
PDM360smart: CR1070, CR1071	20

Je höher die CPU-Frequenz, desto größer ist die Leistungsfähigkeit für den gleichzeitigen Einsatz von komplexen Bausteinen.

5.1.2 Verhalten des Watchdog

1490

Bei (fast) allen programmierbaren **ecomat*mobile***-Geräten wird CoDeSys-intern die Programmlaufzeit über einen Watchdog überwacht.

Wird die maximale Watchdog-Zeit überschritten:
- das Gerät führt einen Reset durch und startet neu

JEDOCH:

BasicController: CR040n:
- alle Prozesse werden angehalten (Reset)
- alle Ausgänge werden ausgeschaltet
- die Status-LED leuchtet rot
- Neustart über Spannung Aus/Ein erforderlich

BasicDisplay: CR0451:
- alle Prozesse werden angehalten (Reset)
- alle Ausgänge werden ausgeschaltet
- der Bildschirm wird dunkel
- die Status-LED leuchtet rot
- Neustart über Spannung Aus/Ein erforderlich

SafetyController: CR7nnn:
- alle Prozesse werden angehalten (Reset)
- alle Ausgänge werden ausgeschaltet
- die Status-LED erlischt
- Neustart über Spannung Aus/Ein erforderlich

PDM360NG: CR108n:
- alle Prozesse werden angehalten (Reset)
- alle Ausgänge werden ausgeschaltet
- der Bildschirm wird dunkel
- die Status-LED blinkt rot mit 5 Hz
- Neustart über Spannung Aus/Ein erforderlich

Je nach Hardware haben die einzelnen Geräte ein unterschiedliches Zeitverhalten:

Controller	Watchdog [ms]
BasicController: CR040n	100
BasicDisplay: CR0451 (Applikations-Programm) BasicDisplay: CR0451 (Visualisierung)	100 1.200
CabinetController: CR030n	100…200
ClassicController: CR0020, CR0032, CR0033, CR0505	100
ExtendedController: CR0200, CR0232, CR0233	100
Platinensteuerung: CS0015	100…200
SafetyController: CR7nnn	100
SmartController: CR25nn	100…200
PDM360: CR1050, CR1051	kein Watchdog
PDM360compact: CR1052, CR1053, CR1055, CR1056	kein Watchdog
PDM360NG: CR108n	Linux-Überwachung *)
PDM360smart: CR1070, CR1071	100…200

*) Der Linux-Kernel und kritische Prozesse werden einzeln überwacht (unterschiedliche Zeiten).

5.1.3 Begrenzungen beim PDM360smart

9895

HINWEIS

Beachten Sie die Grenzen des Geräts! → Datenblatt

Beachten Sie insbesondere folgende Begrenzungen:

Bezeichnung	PDM360smart CR1070, CR1071
Länge Strings	≤ 80 Zeichen
Länge Pfadnamen	≤ 80 Zeichen
Anzahl grafische Objekte pro Visualisierungsseite	50...100
Anzahl Bitmaps ¹) pro Projekt	≤ 100
Anzahl Zeichensätze pro Projekt	≤ 5
Anzahl POUs ²) pro Projekt	≤ 24 576

¹) Vorgaben für das Startbild → Kapitel *Visualisierungsgrenzen* (→ Seite 58).

²) POU (Program Organization Unit) = Funktion, Funktionsblock oder Programmblock

5.1.4 Verfügbarer Speicher

9896

Gilt nur für folgende Geräte:
- PDM360smart: CR1070, CR1071

Physikalischer Speicher	Physikalisch vorhandener FLASH-Speicher (nichtflüchtiger, langsamer Speicher)	1 MByte
	Physikalisch vorhandener SRAM ¹) (flüchtiger, schneller Speicher)	256 kByte
	Physikalisch vorhandener EEPROM (nichtflüchtiger, langsamer Speicher)	---
	Physikalisch vorhandener FRAM ²) (nichtflüchtiger, schneller Speicher)	2 kByte
Nutzung des FLASH-Speichers	Speicher reserviert für den Code der IEC-Applikation	448 kByte
	Speicher für Daten außerhalb der IEC-Applikation, die vom Anwender beschrieben werden können, wie z.B. Files, Bitmaps, Fonts	176 kByte
	Speicher für Daten außerhalb der IEC-Applikation, die vom Anwender mit FBs wie FLASHREAD, FLASHWRITE bearbeitet werden	16 kByte
RAM	Speicher für die von der IEC-Applikation reservierten Daten im RAM	48 kByte
Remanenter Speicher	Speicher für in der IEC-Applikation als VAR_RETAIN deklarierten Daten	128 Byte
	Speicher für in der IEC-Applikation als RETAIN vereinbarten Merker	---
	Vom Anwender frei verfügbarer remanenter Speicher Der Zugriff erfolgt über die FBs FRAMREAD, FRAMWRITE	1536 Byte
	Vom Anwender frei verfügbarer FRAM ²) Der Zugriff erfolgt über Adressoperator	---

¹) SRAM steht hier allgemein für alle Arten von flüchtigen, schnellen Speichern.
²) FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.

5.1.5 Visualisierungsgrenzen

9908

Embeded-Displays, wie sie z.B. in diesem Gerät verbaut sind, können den vollen Farbumfang von Bitmap-Grafiken nicht zur Verfügung stellen, weil nur eingeschränkte Leistungsreserven verfügbar sind. Folgende Vorbereitungen ermöglichen trotzdem Bitmap-Bilder im PDM:

- richtige Auswahl der Motive,
- das richtige Skalieren der Bitmaps vor dem Einsatz auf dem PDM.

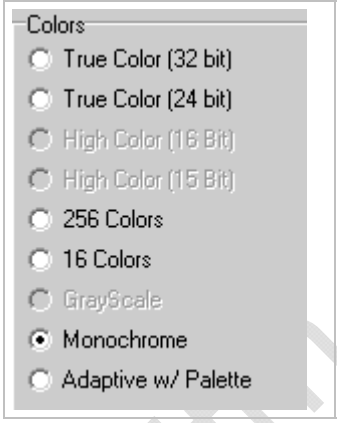
Leistungsreserven des Geräts → Kapitel *Leistungsgrenzen des Geräts* (→ Seite 55)

Bild-Vorgaben für das Startbild:

Parameter	Begrenzung
Datei-Typ	Bitmap (`*.bmp`) RLE-komprimiert
Dateiname	nur Kleinbuchstaben, Namenskonvention = 8.3
Bildgröße	128 x 64 Pixel
Farben	1 Bit = nur Schwarz und Weiß, keine Graustufen
Speicherbedarf	ca. 1 kByte, abhängig vom Bildinhalt bei RLE-Komprimierung

Die im Projekt eingesetzten Grafiken dürfen durchaus größer sein als die vorgegebene Bildgröße. Dann wird von dem Bild jedoch nur ein (wählbarer) Ausschnitt sichtbar sein.

Farben:



- unterstützt nur die 2 Farben Schwarz und Weiß
- Monochrom-Bitmap Bilevel
- Beim Monochrom-Bitmap sollten nur die Farben Weiß (R=0, G=0, B=0) und Schwarz (R=224, G=224, B=224) oder das Monochrom-Farbformat Bilevel verwendet werden.

Bild umrechnen / skalieren

9910

Wird im Gerät ein Bild geladen, welches nicht den Größen- oder den Farbanforderungen entspricht, wird es nicht dargestellt.

► Alle Umformungen des Bitmaps oder des Bildes zuvor auf dem Computer in einer Bildverarbeitung durchführen. Auf dem Gerät selbst werden keine Anpassungen vorgenommen (Größe, Skalierung, Farbe).

► Nur die passend gewandelten Bilder in der Visualisierung des Geräts speichern.

► Nur RLE-codierte Bitmaps in das Gerät laden.

→ Kapitel *Bildgröße Vektorgrafik / Pixelgrafik* (→ Seite 342)

CoDeSys-Visualisierungs-Elemente

9902

Gilt nur für folgende Geräte:
- PDM360smart: CR1070, CR1071

HINWEIS: Nicht alle CoDeSys-Funktionen können auf dem PDM erfolgreich arbeiten:

Visualisierungselement	Funktions-Sicherheit beim PDM	
Linie	o	Linienstärke ≤ 1 mm
Linientyp für Rahmen	—	wird nicht unterstützt
Rechteck	+	keine Probleme bekannt
abgerundetes Rechteck	—	wird nicht unterstützt
Kreis, Ellipse	+	keine Probleme bekannt
Polygone	o	möglich, jedoch zu viele Elemente davon auf einer Seite bremsen das System
Tortengrafik	—	wird nicht unterstützt
BMP-Grafikdateien	+	≤ 100 pro Projekt Dateiname: ≤ 27 Zeichen Bei Größe von 128 x 64 Pixel: ≤ 60 Bitmaps pro Projekt
Visualisierung	o	möglich, jedoch zu viele Elemente davon auf einer Seite bremsen das System
Schaltflächen	+	keine Probleme bekannt
WMF-Grafikdateien	—	wird nicht unterstützt
Tabellen	—	nicht sinnvoll nutzbar
Trendkurven	—	wird nicht unterstützt
Alarmtabelle	—	nicht sinnvoll nutzbar
Skalen	—	wird nicht unterstützt → nachfolgenden Hinweis
Balkendiagramm	+	keine Probleme bekannt
Histogramm	+	keine Probleme bekannt
Dynamischer Text (XML)	—	wird nicht unterstützt
Platzhalter %t (Systemzeit)	—	wird nicht unterstützt
Online Change	—	wird nicht unterstützt

Zum Vermeiden zu langer Bild-Ladezeiten beachten Sie bitte:

- In der Grafik grafische Elemente nicht gruppieren!
- Grafiken möglichst nicht überlagern.
- Manche Visualisierungen mit den CoDeSys-Möglichkeiten sind nicht sehr befriedigend, z.B. runde Skalen. Abhilfe:
  Integrieren Sie die gewünschten Elemente als (extern erzeugte) BMP-Grafik. In der Visualisierung muss dann nur ein Pfeil werteabhängig gedreht werden, der bei Bedarf bei Überschreiten von Grenzwerten seine Farbe wechseln könnte.

Texte

9903

Gilt nur für folgende Geräte:
- PDM360smart: CR1070, CR1071

► Zum Vermeiden zu langer Bild-Ladezeiten:
Reduzieren Sie die Anzahl verschiedener Zeichensätze (Fonts) je Projekt.

Unterstützte Zeichensätze / Schriftgrößen:

Zeichensatz (Font)	Schriftgröße [Punkt]	Hinweis
Arial	6, 8, 10, 13, 20, 26	normal
Arial	32	nur Zahlen
Arial Black	8, 10, 13, 20, 26	fett

- NICHT unterstützte Attribute:
  - unterstrichen
  - kursiv
  - durchgestrichen.
- Wird der eingestellte Zeichensatz nicht unterstützt, werden die Zeichen in "Arial" mit Schriftgröße 6 Punkt dargestellt.
- Wird die eingestellte Schriftgröße nicht unterstützt, werden die Zeichen in der nächst kleineren Schriftgröße dargestellt.
- Die kleinste auf dem PDM gut lesbare Schriftgröße ist 8 Punkt.

5.2 Programmierhinweise für CoDeSys-Projekte

Inhalt

7426

Hier erhalten Sie Tipps zum Programmieren des Geräts.

► Beachten Sie die Hinweise im CoDeSys-Programmierhandbuch
→ **ecomat*mobile***-DVD "Software, tools and documentation".

5.2.1 FB, FUN, PRG in CoDeSys

8473

In CoDeSys unterscheiden wir folgende Typen von Bausteinen (POUs):

FB = function block = Funktionsblock

- Ein FB kann mehrere Eingänge und mehrere Ausgänge haben.
- Ein FB darf in einem Projekt mehrmals aufgerufen werden.
- Für jeden Aufruf muss eine Instanz deklariert werden.
- Erlaubt: Im FB aufrufen von FB und FUN.

FUN = function = Funktion

- Eine Funktion kann mehrere Eingänge, aber nur einen Ausgang haben.
- Der Ausgang ist vom gleichen Datentyp wie die Funktion selbst.

PRG = program = Programm

- Ein PRG kann mehrere Eingänge und mehrere Ausgänge haben.
- Ein PRG darf in einem Projekt nur einmal aufgerufen werden.
- Erlaubt: im PRG aufrufen von PRG, FB und FUN.

HINWEIS

Funktionsblöcke dürfen NICHT in Funktionen aufgerufen werden.
Sonst: Bei der Ausführung stürzt das Applikations-Programm ab.

Alle Bausteine (POUs) dürfen NICHT rekursiv aufgerufen werden, auch nicht indirekt.

Hintergrund:

Alle Variablen von Funktionen...
- werden beim Aufruf initialisiert und
- werden nach der Rückkehr zum Aufrufer ungültig.

Funktionsbausteine haben 2 Aufrufe:
- einen Initialisierungsaufruf und
- den eigentlichen Aufruf, um irgend etwas zu tun.

Folglich heißt das für den FB-Aufruf in einer Funktion, dass jedesmal ein zusätzlicher Initialisierungsaufruf über die Schnittstelle ginge.

5.2.2 Zykluszeit beachten!

8006

Bei den frei programmierbaren Geräten aus der Controller-Familie **ecomat*mobile*** stehen in einem großen Umfang Bausteine zur Verfügung, die den Einsatz der Geräte in den unterschiedlichsten Applikationen ermöglichen.

Da diese Bausteine je nach Komplexität mehr oder weniger Systemressourcen belegen, können nicht immer alle Bausteine gleichzeitig und mehrfach eingesetzt werden.

ACHTUNG

Gefahr von zu trägem Verhalten des Controllers! Zykluszeit darf nicht zu lang werden!

► Beim Erstellen des Applikations-Programms müssen die oben aufgeführten Empfehlungen beachtet und durch Austesten überprüft werden. Bei Bedarf muss durch Neustrukturieren der Software und des Systemaufbaus die Zykluszeit vermindert werden.

5.2.3 Bibliotheken

9938

Folgende Bibliotheken sollten die CoDeSys-Projekte mindestens enthalten:

- Standard-Bibliothek `Standard.Lib` in `C:\...\CoDeSys\Library\`
- Geräte-Bibliothek `ifm_CRnnnn_Vxxyyzz.LIB` in `C:\...\CoDeSys\Targets\ifm\Library\ifm_CRnnnn`

Bei Einsatz von PDM als CANopen-Master sind folgende Bibliotheken mindestens erforderlich:

- `3S_CanDrvOptTableEx.lib` in `C:\...\CoDeSys\Library\`
- `3S_CanOpenNetVarOptTableEx.lib` in `C:\...\CoDeSys\Library\`
- `3S_CanOpenManagerOptTableEx.lib` in `C:\...\CoDeSys\Library\`
- `3S_CanOpenMasterOptTableEx.lib` in `C:\...\CoDeSys\Library\`

Bei Einsatz von PDM als CANopen-Slave sind folgende Bibliotheken mindestens erforderlich:

- `3S_CanDrvOptTableEx.lib` in `C:\...\CoDeSys\Library\`
- `3S_CanOpenNetVarOptTableEx.lib` in `C:\...\CoDeSys\Library\`
- `3S_CanOpenManagerOptTableEx.lib` in `C:\...\CoDeSys\Library\`
- `3S_CanOpenDeviceOptTableEx.lib` in `C:\...\CoDeSys\Library\`

Zur Behandlung von Dateien und zum Mitschreiben von Daten:
HINWEIS: Gefahr für das System bei falscher Handhabung! Erfahrung erforderlich!

- Bibliothek `SysLibFile.Lib` in `C:\...\CoDeSys\Library\` **ODER:**
- Bibliothek `ifm_CRnnnn_Vxxyyzz.LIB` in `C:\...\CoDeSys\Targets\ifm\Library\ifm_CRnnnn`

5.2.4 Arbeitsreihenfolge

7427

Es gibt grundsätzlich zwei Reihenfolgen, ein Projekt für ein PDM oder Display zu erstellen:

A) Zuerst die Visualisierung, anschließend das PLC-Programm.	
Vorteile: • Im Programm kann auf die Parameter in den fertigen Bildern querverwiesen werden. • Beim Testen des PLC-Programms existieren die Bilder bereits.	**Nachteil:** • Die in den Bildern benötigten PLC-Parameter und Variablen sind noch nicht definiert.
B) Zuerst das PLC-Programm, anschließend die Visualisierung.	
Vorteil: • Alle Parameter und Variablen sind im PLC-Programm definiert, bevor in den Visualisierungen auf sie verwiesen wird.	**Nachteile:** • Die Parameter aus den Bildern (Bildnummer, Taste, LED usw.) müssen anderweitig ermittelt werden. • Das PLC-Programm kann erst nach dem Erstellen der Visualisierung getestet werden.

In beiden Fällen empfehlen wir dringend, **vor** Beginn eine möglichst genaue Struktur der Visualisierung und ihrer Inhalte zu entwerfen.

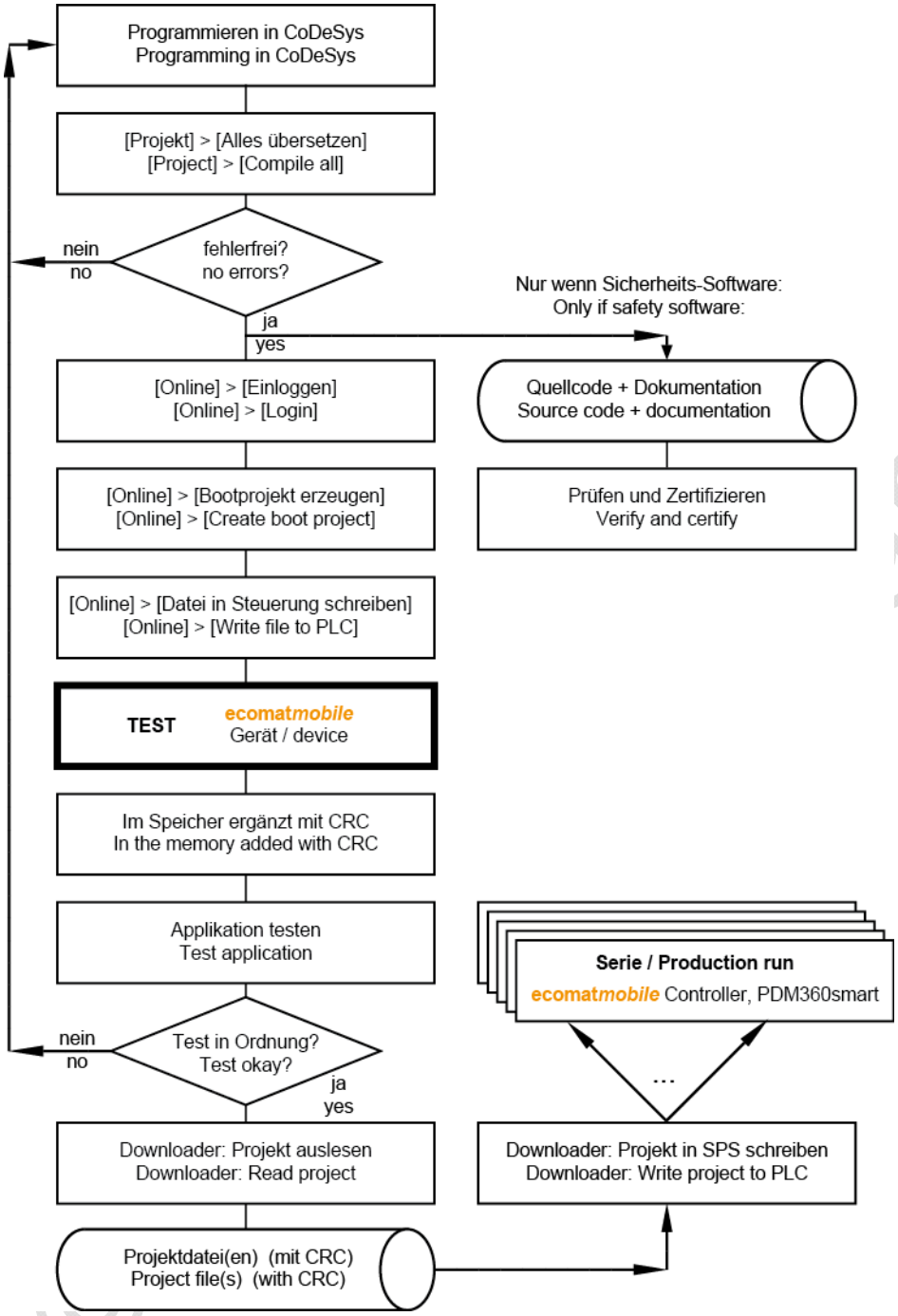
5.2.5 Applikations-Programm erstellen

8007

Das Applikations-Programm wird mit dem Programmiersystem CoDeSys erstellt und während der Programmentwicklung mehrfach zum Testen in die Steuerung geladen:
In CoDeSys: [Online] > [Datei in Steuerung schreiben].

Für jeden derartigen Download via CoDeSys wird dazu der Quellcode neu übersetzt. Daraus resultiert, dass auch jedes Mal im Speicher der Steuerung eine neue Prüfsumme gebildet wird. Auch für Sicherheitssteuerungen ist dieses Verfahren bis zur Freigabe der Software zulässig.

Zumindest für sicherheitsrelevante Applikationen muss aber für die Serienproduktion der Maschine eine Einheitlichkeit der Software und ihrer Prüfsumme gewährleistet sein.



Grafik: Erstellen und Verteilen der (zertifizierten) Software

5.2.6 ifm-Downloader nutzen

8008

Der **ifm**-Downloader dient dem einfachen Übertragen des Programmcodes vom Programmierplatz in die Steuerung. Grundsätzlich kann jede Applikations-Software mit dem **ifm**-Downloader auf die Steuerungen kopiert werden. Vorteil: Dazu ist kein Programmiersystem mit einer CoDeSys-Lizenz erforderlich.

Sicherheitsrelevante Applikations-Software MUSS mit dem **ifm**-Downloader auf die Steuerungen kopiert werden, um die Prüfsumme CRC, mit der die Software zertifiziert wurde, nicht zu verfälschen.

HINWEIS

Der **ifm**-Downloader kann nicht eingesetzt werden für folgende Geräte:
- BasicController: CR040n
- BasicDisplay: CR0451
- PDM360: CR1050, CR1051,
- PDM360compact: CR1052, CR1053, CR1055, CR1056,
- PDM360NG: CR108n

6 CAN einsetzen

Inhalt

1163

6.1 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.

6.1.1 Topologie

1244

Das CAN-Netzwerk wird als Linienstruktur aufgebaut. Stichleitungen sind in eingeschränktem Umfang zulässig. Des Weiteren sind auch ein ringförmiger Bus (Infotainment Bereich) sowie ein sternförmiger Bus (Zentralverrieglung) möglich. Beide Varianten haben im Vergleich zum linienförmigen Bus jeweils einen Nachteil:

- Im ringförmigen Bus sind alle Steuergeräte in Reihe geschaltet, so dass bei einem Ausfall eines Steuergeräts der gesamte Bus ausfällt.
- Der sternförmige Bus wird meist von einem Zentralrechner gesteuert, da diesen alle Informationen passieren müssen, mit der Folge, dass bei einem Ausfall des Zentralrechners keine Informationen weitergeleitet werden können. Bei einem Ausfall eines einzelnen Steuergeräts funktioniert der Bus weiter.

Der lineare Bus hat den Vorteil, dass alle Steuergeräte parallel zu einer zentralen Leitung gehen. Nur wenn diese ausfällt, funktioniert der Bus nicht mehr.

HINWEIS

Die Linie muss an ihren beiden Enden jeweils mit einem Abschlusswiderstand von der Größe 120 Ohm abgeschlossen werden, um ein Verfälschen der Signalqualität zu verhindern.

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

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

6.1.2 CAN-Schnittstellen

2269

Die Controller 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 (→ Kapitel *ifm-CANopen-Bibliotheken* (→ Seite 121)), ein Protokoll nach CiA 301/401 für Master/Slave-Betrieb (via CoDeSys)
- *CANopen-Netzwerkvariablen* (→ Seite 156) (via CoDeSys)
- Protokoll SAE J1939 (für Antriebs-Management, → Kapitel *CAN-Bausteine nach SAE J1939* (→ Seite 104))
- Buslast-Erkennung
- Errorframe-Zähler
- Download-Schnittstelle (nicht alle Geräte)
- 100 % Buslast ohne Paketverlust

Welche CAN-Schnittstelle des Geräts welche konkreten Möglichkeiten bietet, → Datenblatt des Geräts.
Das aktuelle Datenblatt finden Sie auf der **ifm**-Homepage:
→ www.ifm.com > Land wählen > [Datenblattsuche] > Artikel-Nr.

Informativ: Weitere interessante CAN-Protokolle sind:

- "Truck & Trailer Interface" nach ISO 11992
  Für folgende Geräte verfügbar: SmartController: CR2501
- ISOBUS nach ISO 11783 für Landmaschinen
- NMEA 2000 für den maritimen Einsatz
- CANopen Truck Gateway nach CiA 413 (Umsetzung zwischen ISO 11992 und SAE J1939)

6.1.3 Verfügbare CAN-Schnittstellen und CAN-Protokolle

6467

In den **ifm**-Geräten sind folgende CAN-Schnittstellen und CAN-Protokolle verfügbar:

Schnittstelle voreingestellter Download-ID **Gerät**	**CAN 1** ID 127	**CAN 2** ID 126	**CAN 3** ID 125	**CAN 4** ID 124	**Standard Baudrate** [kBit/s]
BasicController: CR040n	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 CANopen SAE J1939	---	---	250
BasicDisplay: CR0451	CAN Layer 2 CANopen SAE J1939	---	---	---	250
CabinetController: CR0301, CR0302	CAN Layer 2 CANopen SAE J1939	---	---	---	125
CabinetController: CR0303	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 SAE J1939	---	---	125

Schnittstelle voreingestellter Download-ID **Gerät**	**CAN 1** ID 127	**CAN 2** ID 126	**CAN 3** ID 125	**CAN 4** ID 124	**Standard Baudrate** [kBit/s]
ClassicController: CR0020, CR0505	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 SAE J1939	---	---	125
ClassicController: CR0032, CR0033	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 CANopen SAE J1939	125
ExtendedController: CR0200	**CPU 1 CAN 1** ID 127 CAN Layer 2 CANopen SAE J1939	**CPU 1 CAN 2** ID 126 CAN Layer 2 SAE J1939	**CPU 2 CAN 1** ID 127 CAN Layer 2 CANopen SAE J1939	**CPU 2 CAN 2** ID 126 CAN Layer 2 SAE J1939	125
ExtendedController: CR0232, CR0233	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 CANopen SAE J1939	125
Platinensteuerung: CS0015	CAN Layer 2 CANopen SAE J1939	---	---	---	Drehschalter
SafetyController: CR7021, CR7506	CAN Layer 2 CANopen CANopen Safety SAE J1939	CAN Layer 2 CANopen Safety SAE J1939	---	---	125
ExtendedSafetyController: CR7201	**CPU 1 CAN 1** ID 127 CAN Layer 2 CANopen CANopen Safety SAE J1939	**CPU 1 CAN 2** ID 126 CAN Layer 2 CANopen Safety SAE J1939	**CPU 2 CAN 1** ID 127 CAN Layer 2 CANopen SAE J1939	**CPU 2 CAN 2** ID 126 CAN Layer 2 SAE J1939	125
SmartController: CR2500	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 SAE J1939	---	---	125
PDM360: CR1050, CR1051	CAN Layer 2 CANopen	CAN Layer 2 CANopen SAE J1939	---	---	125
PDM360compact: CR1052, CR1053, CR1055, CR1056	CAN Layer 2 CANopen	---	---	---	125
PDM360smart: CR1070, CR1071	CAN Layer 2 CANopen SAE J1939	---	---	---	125
PDM360NG: CR108n	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 CANopen SAE J1939	CAN Layer 2 CANopen SAE J1939	125

6.1.4 System-Konfiguration

2270

Die Controller werden mit folgenden Download-Identifier (= ID) ausgeliefert:
- ID 127 für CAN-Schnittstelle 1
- ID 126 für CAN-Schnittstelle 2 (wenn vorhanden)
- ID 125 für CAN-Schnittstelle 3 (wenn vorhanden)
- ID 124 für CAN-Schnittstelle 4 (wenn vorhanden)

Das Download-System benutzt diesen Identifier für die erste Kommunikation mit einem nicht konfigurierten Modul über CAN.
Die Download-IDs können auf folgenden Wegen eingestellt werden:
- über den PLC-Browser des Programmiersystems,
- über den Downloader oder das MaintenanceTool oder
- über das Applikations-Programm.

Über den Modus "Autoconfig" des Bootloaders kann nur die CAN-Schnittstelle 1 eingestellt werden.

Da der Download-Mechanismus auf Basis des CANopen-SDO-Dienstes arbeitet (auch wenn der Controller nicht im CANopen-Modus betrieben wird), müssen alle Steuerungen im Netzwerk einen eindeutigen Identifier besitzen. Die eigentlichen COB-IDs werden nach dem "predefined connection set" aus den Modulnummern abgeleitet. Es darf jeweils nur ein nicht konfiguriertes Modul mit dem Netz verbunden werden. Nachdem die neue Teilnehmernummer 1...126 zugewiesen wurde, kann ein Download oder ein Debugging stattfinden und danach ein weiteres Gerät ins System eingebunden werden.

HINWEIS

► Der Download-ID wird unabhängig von dem CANopen-Identifier eingestellt. Es muss beachtet werden, dass sich diese IDs nicht mit den Download-IDs und den CANopen-Knotennummern der anderen Controller oder Netzwerkteilnehmer überschneiden.

Vergleich Download-ID vs. COB-ID:

Controller Programm-Download		CANopen	
Download-ID	**COB-ID SDO**	**Node-ID**	**COB-ID SDO**
1…127	TX: 580_{16} + Download-ID	1…127	TX: 580_{16} + Node-ID
	RX: 600_{16} + Download-ID		RX: 600_{16} + Node-ID

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

HINWEIS

Der CAN-Download-ID des Geräts muss mit dem in CoDeSys eingestellten CAN-Download-ID übereinstimmen!

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

6.2 Physikalische Anbindung des CAN

Inhalt

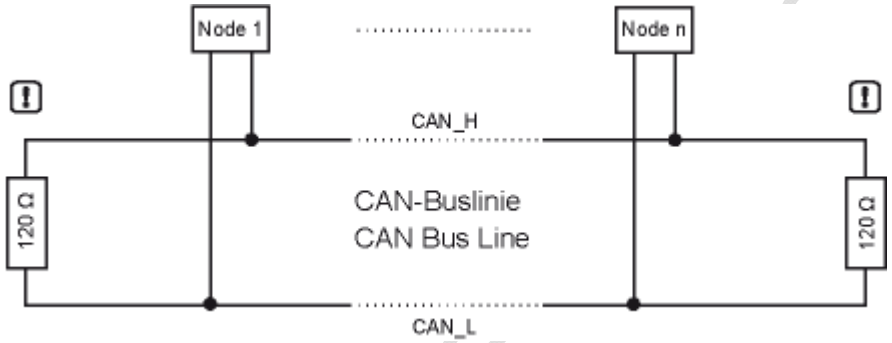
1177

Die in den Kapiteln *CAN-Datenaustausch* (→ Seite 76) und *CAN-Fehler und Fehlerbehandlung* (→ Seite 189) beschriebenen Mechanismen der Datenübertragung und der Fehlerbehandlung sind direkt im CAN-Controller implementiert. Die physikalische Verbindung der einzelnen CAN-Teilnehmer wird von der ISO 11898 in der Schicht 1 beschrieben.

6.2.1 Netzaufbau

1178

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



Grafik: CAN-Netzaufbau Linienstruktur

HINWEIS

Die Linie muss an ihren beiden Enden jeweils mit einem Abschlusswiderstand von der Größe 120 Ohm abgeschlossen werden, um ein Verfälschen der Signalqualität zu verhindern.

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

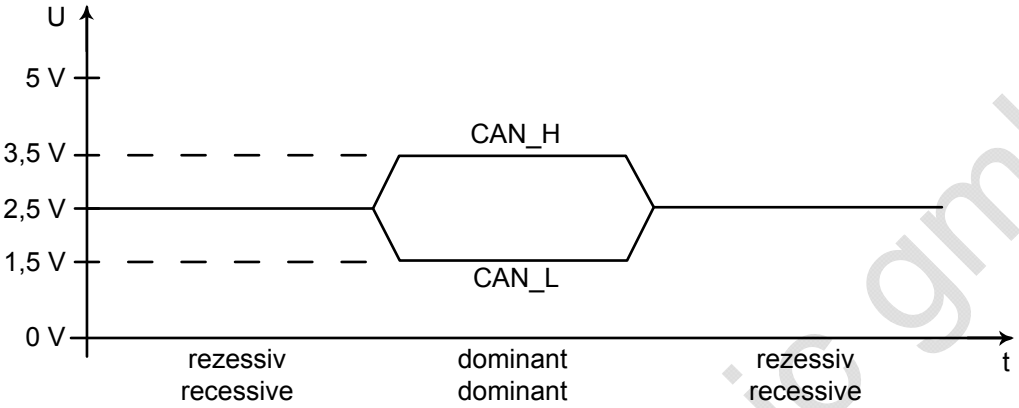
Stichleitungen

Idealerweise sollte zu den Busteilnehmern (Node 1 ... Node n) keine Stichleitung führen, da in Abhängigkeit von der Gesamtleitungslänge und den zeitlichen Abläufen auf dem Bus Reflektionen auftreten. Damit diese nicht zu Systemfehlern führen, sollten die Stichleitungen zu einem Busteilnehmer (z.B. einem E/A-Modul) eine gewisse Länge nicht überschreiten. Stichleitungen mit einer Länge von 2 m (bezogen auf 125 kBit/s) werden als unkritisch angesehen. Die Summe aller Stichleitungen im Gesamtsystem sollte 30 m nicht übersteigen. In besonderen Fällen müssen die Leitungslängen der Linie und der Stiche genau berechnet werden.

6.2.2 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

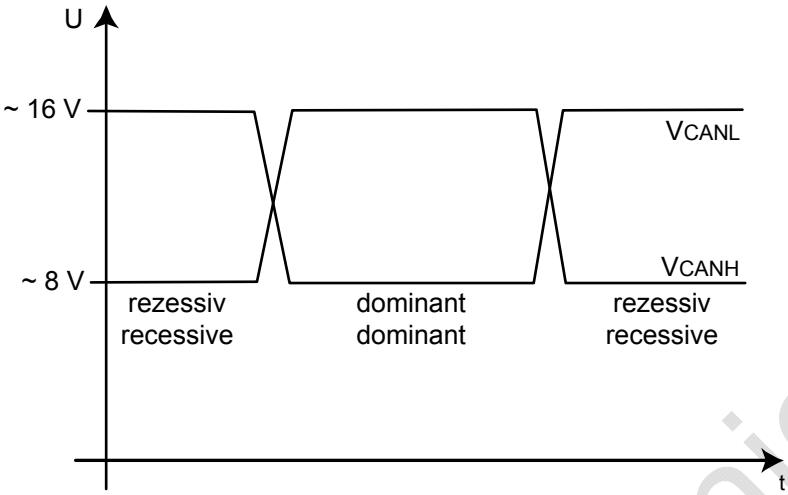
Durch diese differentielle Übertragung mit gemeinsamem Rückleiter wird die Übertragungssicherheit entscheidend verbessert. Störspannungen, die von außen auf das System einwirken, oder Massepotential-Verschiebungen beeinflussen beide Signalleitungen mit gleichen Störgrößen. Dadurch fallen die Störungen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

6.2.3 CAN-Buspegel nach ISO 11992-1

1182

Für folgende Geräte verfügbar: nur SmartController: CR2501 auf 2. CAN-Schnittstelle.

Die physikalische Schicht der ISO 11992-1 unterscheidet sich von der ISO 11898 durch höhere Spannungspegel. Die Netzwerke werden hier als Point-to-Point-Verbindung ausgeführt. Die Abschlussnetzwerke sind bereits integriert.



Grafik: Spannungspegel nach ISO 11992-1 (hier: 12-V-System)

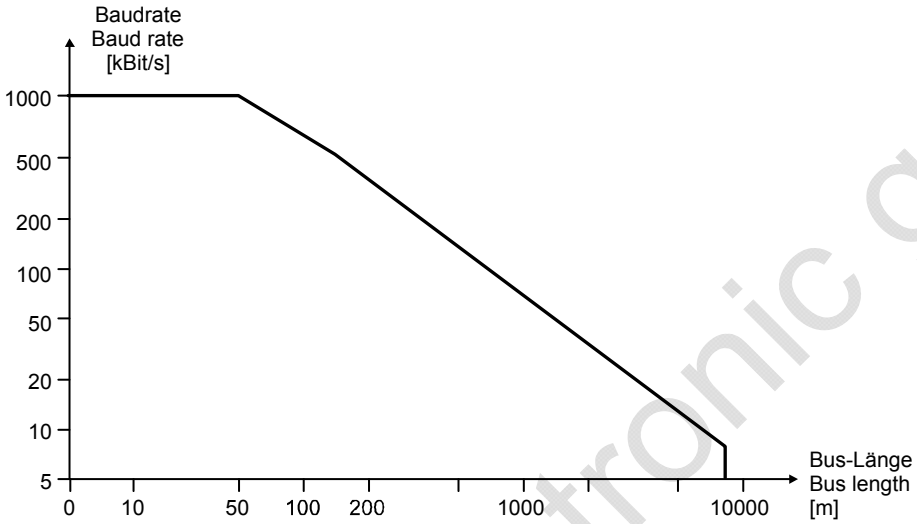
6.2.4 Busleitungslänge

1180

Die Länge der Busleitung ist abhängig von:

- Beschaffenheit der Busverbindung (Kabel, Steckverbinder),
- Leitungswiderstand,
- benötigte Übertragungsrate (Baud-Rate),
- Länge der Stichleitungen.

Vereinfachend kann man von folgender Abhängigkeit zwischen Buslänge und Baud-Rate ausgehen:



Grafik: Busleitungslänge

Baud-Rate [kBit/s]	Buslänge [m]	nominelle Bit-Länge [µs]
1 000	30	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 / Bitzeit

6.2.5 Leitungsquerschnitte

1181

Für die Auslegung des CAN-Netzes ist auch der Leitungsquerschnitt der eingesetzten Busleitung zu beachten. Die folgende Tabelle beschreibt die Abhängigkeit des Leiterquerschnitts bezogen auf die Leitungslänge und der Anzahl der daran angeschlossenen Teilnehmer (Knoten).

Leitungslänge [m]	**Leiterquerschnitt** [mm^2] **bei 32 Knoten**	**Leiterquerschnitt** [mm^2] **bei 64 Knoten**	**Leiterquerschnitt** [mm^2] **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,
- als Twisted-Pair
- und/oder abgeschirmt.

6.3 CAN-Datenaustausch

Inhalt

1168

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 Betriebssystem automatisch durchgeführt.

Für den normalen CAN-Datenaustausch muss der Programmierer lediglich bei der Softwareerstellung die Datenobjekte mit ihren Identifiern dem System bekannt machen. Dies erfolgt über folgende FBs:

- *CANx_RECEIVE* (→ Seite 88) (CAN-Daten empfangen) und
- *CANx_TRANSMIT* (→ Seite 93) (CAN-Daten senden).

Über diese FBs werden folgende Einheiten zu einem Datenobjekt verknüpft:

- die RAM-Adresse der Arbeitsdaten,
- der Datentyp,
- 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 CAN-Identifier (*CAN-ID* (→ Seite 77)) und maximal 8 Datenbytes. Der 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 Empfangs-Objekt deklariert ist. Beide Deklarationen müssen dem gleichen Identifier zugeordnet sein.

6.3.1 Hinweise

8394

CAN-ID

1166

Je nach CAN-ID sind folgende CAN-Identifier frei verfügbar für den Datentransfer:

CAN-ID base	CAN-ID extended
11 Bit	29 Bit
2 047 CAN-Identifier	536 870 912 CAN-Identifier
Standard-Applikationen	Motor-Management (SAE J1939), Truck & Trailer Interface (ISO 11992)

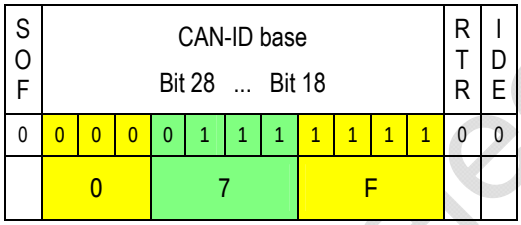
HINWEIS

Der 29-Bit-CAN-ID steht bei einigen Geräten nicht für alle CAN-Schnittstellen zur Verfügung,
→ Datenblatt.

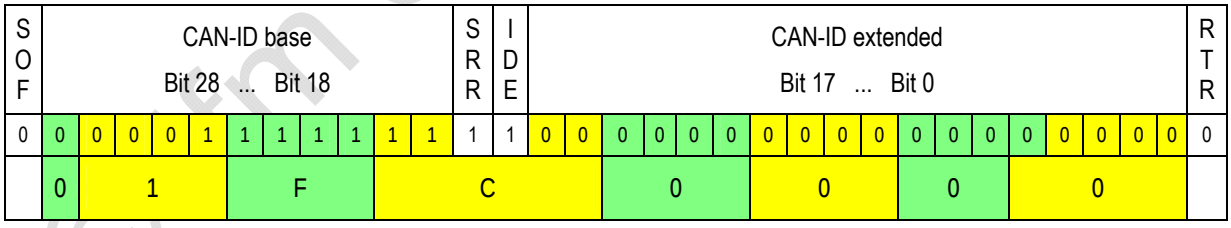
Derselbe CAN-Controller kann NICHT gleichzeitig 11 Bit und 29 Bit lange CAN-Identifier empfangen.

Wir empfehlen: In einem CAN-Netzwerk ausschließlich 11 Bit lange CAN-Identifier benutzen ODER 29 Bit lange CAN-Identifier.

Beispiel 11-Bit CAN-ID (base):



Beispiel 29-Bit CAN-ID (extended):



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

Zusammenfassung CAN / CANopen

3956

- Der COB-ID der Netzwerkvariablen muss sich unterscheiden vom CANopen-Slave-ID in der Steuerungskonfiguration und von den IDs der FBs CANx_TRANSMIT und CANx_RECEIVE!
- Wenn mehr als 8 Bytes von Netzwerkvariablen in einen COB-ID gepackt werden, erweitert CANopen das Datenpaket automatisch auf mehrere aufeinander folgende COB-IDs. Dies kann zu Konflikten mit manuell definierten COB-IDs führen!
- Netzwerkvariable können keine String-Variablen transportieren.
- Netzwerkvariable 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 CANx_TRANSMIT mit Hilfe von verschiedenen Events auf mehrere SPS-Zyklen verteilen.
- Jeder Aufruf von CANx_TRANSMIT oder CANx_RECEIVE erzeugt ein Nachrichtenpaket von 8 Bytes.
- In der Steuerungskonfiguration sollten die Werte für [Com Cycle Period] und [Sync. Window Length] gleich groß sein. Diese Werte müssen größer sein als die SPS-Zykluzeit.
- 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.
- Weil wir keinen Inhibit-Timer haben, empfehlen wir, Analog-Eingänge auf "synchrone Übertragung" zu setzen, um Busüberlastung zu vermeiden.
- Binäre Eingänge, insbesondere die unregelmäßig schaltenden, sollten am besten auf "asynchrone Übertragung" mittels Event-Timer gesetzt 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.

6.3.2 Daten empfangen

1169

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 *CANx_RECEIVE* (→ Seite 88) entleert.

6.3.3 Daten senden

1170

Durch den Aufruf von *CANx_TRANSMIT* (→ Seite 93) übergibt das Applikations-Programm genau eine CAN-Nachricht an den CAN-Controller. Als Rückgabe erhält man die Information, ob die Nachricht erfolgreich an den CAN-Controller übergeben wurde. Dieser führt dann selbständig die eigentliche Übergabe der Daten auf den CAN-Bus aus.

Der Sendeauftrag wird abgewiesen, wenn der Controller nicht bereit ist, weil er bereits ein Datenobjekt überträgt. Der Sendeauftrag muss dann durch das Applikations-Programm wiederholt werden. Der Anwender bekommt diese Information durch ein Bit angezeigt.

Bei mehreren zeitgleich zum Senden bereiten CAN-Nachrichten wird die Nachricht mit dem niedrigsten ID vorrangig gesendet. Der Programmierer muss daher den *CAN-ID* (→ Seite 77) sehr umsichtig vergeben.

6.4 Beschreibung der CAN-Standardbausteine

Inhalt

1186

Hier werden die CAN-Funktionsblöcke zur Nutzung im Applikations-Programm beschrieben.

HINWEIS

Um die volle Leistungsfähigkeit von CAN zu nutzen, ist es unbedingt erforderlich, dass sich der Programmierer vor Beginn seiner Arbeit ein genaues **Buskonzept** aufbaut:

- Wie viele Datenobjekte mit welchen Identifiern werden benötigt?
- Wie soll das Gerät auf mögliche CAN-Fehler reagieren?
- Wie oft müssen Daten übertragen werden? Dem entsprechend oft müssen *CANx_TRANSMIT* (→ Seite 93) und *CANx_RECEIVE* (→ Seite 88) aufgerufen werden.

► Dabei überwachen, ob die Sendeaufträge erfolgreich an CANx_TRANSMIT übergeben wurden (Ausgang RESULT) oder dafür sorgen, dass die empfangenen Daten mit CANx_RECEIVE aus dem Datenpuffer der Warteschlage ausgelesen und sofort im übrigen Programm entsprechend verarbeitet werden.

Damit eine Kommunikationsverbindung aufgebaut werden kann, muss zuvor bei allen Teilnehmern des CAN-Netzwerkes die gleiche Übertragungsrate (Baud-Rate) eingestellt werden. Beim Controller wird diese mit *CAN1_BAUDRATE* (→ Seite 82) (für die 1. CAN-Schnittstelle) oder über *CAN2* (für die 2. CAN-Schnittstelle) vorgenommen.

Unabhängig davon, ob die Geräte eine oder mehrere CAN-Schnittstellen unterstützen, werden die der Schnittstelle zugehörigen Funktionen durch Nummerierung im CAN-FB gekennzeichnet (z.B. CAN1_TRANSMIT oder CAN2_RECEIVE). In der Dokumentation wird aus Vereinfachungsgründen die Bezeichnung (z.B. CANx_TRANSMIT) für alle Varianten verwendet.

HINWEIS

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_CDV`…

► 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.
→ Kapitel *Programmiersystem über Templates einrichten* (→ Seite 32)

In diesem Beispiel werden über die Identifier 1 und 2 Datenobjekte mit einem weiteren CAN-Teilnehmer ausgetauscht. Dazu muss im anderen Teilnehmer zum Sende-Identifier ein Empfangs-Identifier (oder umgekehrt) existieren.

6.4.1 CAN1_BAUDRATE

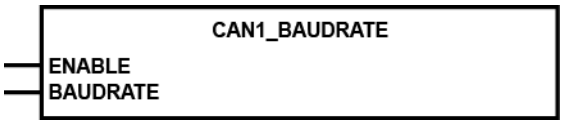
651

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

654

CAN1_BAUDRATE stellt die Übertragungsrate für den Busteilnehmer ein.

Mit dem FB wird für das Gerät die Übertragungsrate eingestellt. Dazu wird am Eingang BAUDRATE der entsprechende Wert in kBit/s angegeben. Nach Ausführen des FB wird der neue Wert im Gerät gespeichert und steht auch nach einem Spannungsausfall wieder zur Verfügung.

ACHTUNG

Für CR250n, CR0301, CR0302, CS0015 beachten:

Das EEPROM-Speichermodul kann bei Dauerbetrieb dieser Funktion zerstört werden!

► Diesen Baustein nur **einmalig** bei der Initialisierung im ersten Programmzyklus ausführen! Anschließend den Baustein wieder sperren (ENABLE = "FALSE")!

HINWEIS

Die neue Baud-Rate wird erst nach einem RESET gültig (Spannung Aus/Ein oder Soft-Reset).

ExtendedController: Im Slave-Modul wird die neue Baud-Rate erst nach Spannung Aus/Ein übernommen.

Parameter der Eingänge

655

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (steigende Flanke): Baustein wird ausgeführt (nur 1 Zyklus lang) FALSE: Baustein wird nicht ausgeführt Baustein-Ein- und Ausgänge sind nicht aktiv
BAUDRATE	WORD	Baud-Rate [kBit/s] Zulässige Werte: 50, 100, 125, 250, 500, 1000 Voreinstellung = 125 kBit/s

6.4.2 CAN1_DOWNLOADID

645

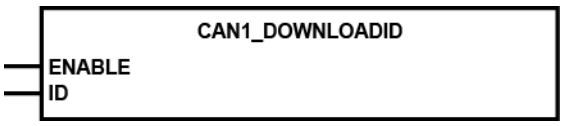
= CAN1 Download-ID

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

648

CAN1_DOWNLOADID stellt den Download-Identifier für die erste CAN-Schnittstelle ein.

Mit dem FB kann der Kommunikations-Identifier für den Programmdownload und das Debuggen eingestellt werden. Der neue Wert wird eingetragen, wenn der Eingang ENABLE auf TRUE gesetzt wird. Der neue Download-ID wird gültig nach Spannung Aus/Ein oder nach einem Softreset.

ACHTUNG

Für CR250n, CR0301, CR0302, CS0015 beachten:

Das EEPROM-Speichermodul kann bei Dauerbetrieb dieser Funktion zerstört werden!

► Diesen Baustein nur **einmalig** bei der Initialisierung im ersten Programmzyklus ausführen! Anschließend den Baustein wieder sperren (ENABLE = "FALSE")!

HINWEIS

Achten Sie darauf, dass bei jedem Gerät im selben Netzwerk ein anderer Download-ID eingestellt ist!

Wird das Gerät im CANopen-Netzwerk betrieben, darf sich der Download-ID auch mit keinem Modul-ID (Knotennummer) der anderen Teilnehmer überschneiden!

ExtendedController: Im Slave-Modul wird der Download-ID erst nach Spannung Aus/Ein gültig.

Parameter der Eingänge

649

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (nur 1 Zyklus lang): Der ID wird gesetzt FALSE: Baustein wird nicht ausgeführt
ID	BYTE	Download-Identifier Zulässige Werte: 1…127

6.4.3 CANx_ERRORHANDLER

633

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

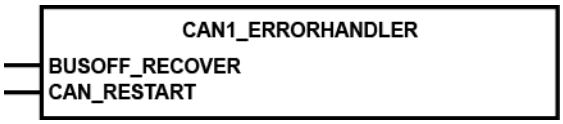
CAN1_ERRORHANDLER

9344

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

636

Fehlerroutine zur Überwachung der CAN-Schnittstellen

CANx_ERRORHANDLER überwacht die CAN-Schnittstellen und wertet die CAN-Fehler aus. Tritt eine bestimmte Anzahl von Übertragungsfehlern auf, so wird der CAN-Teilnehmer error-passiv. Verringert sich die Fehlerhäufigkeit, wird der Teilnehmer wieder error-activ (= Normalzustand).

Ist ein Teilnehmer schon error-passiv und es treten weiterhin Übertragungsfehler auf, wird er vom Bus abgeschaltet (= bus-off) und das Fehlerbit CANx_BUSOFF gesetzt. Die Rückkehr an den Bus ist nur möglich, wenn der Bus-off-Zustand behoben wird (Signal BUSOFF_RECOVER).

Der Eingang CAN_RESTART dient zur Behebung anders gearteter CAN-Fehler. Die CAN-Schnittstelle wird dadurch neu initialisiert.

Das Fehlerbit muss anschließend im Applikations-Programm zurückgesetzt werden.

Das Vorgehen für den Neustart der Schnittstellen unterscheidet sich:

- für CAN-Schnittstelle 1 oder Geräte mit nur einer CAN-Schnittstelle:
  den Eingang CAN_RESTART = TRUE (nur 1 Zyklus lang) setzen
- für CAN-Schnittstelle 2:
  in *CAN2* den Eingang START = TRUE (nur 1 Zyklus lang) setzen

HINWEIS

CAN2 muss grundsätzlich zum Initialisieren der zweiten CAN-Schnittstelle ausgeführt werden, bevor FBs für diese genutzt werden können.

Wenn die automatische Bus-Recover-Funktion genutzt werden soll (Default-Einstellung), darf CANx_ERRORHANDLER **nicht** in das Programm eingebunden und instanziert werden!

Parameter der Eingänge

637

Parameter	Datentyp	Beschreibung
BUSOFF_RECOVER	BOOL	TRUE (nur 1 Zyklus lang): Bus-off-Zustand beheben FALSE: diese Funktion wird nicht ausgeführt
CAN_RESTART	BOOL	TRUE (nur 1 Zyklus lang): CAN-Schnittstelle 1 komplett neu initialisieren FALSE: diese Funktion wird nicht ausgeführt

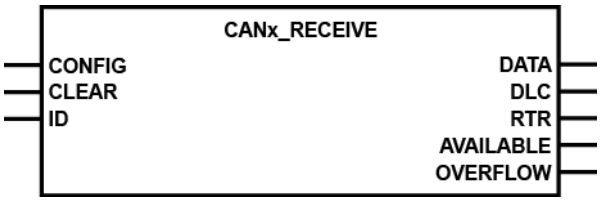
6.4.4 CANx_RECEIVE

627

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_RECEIVE

9354

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
  Baustein NICHT für Sicherheitssignale!
  (Für Sicherheitssignale → *CAN_SAFETY_RECEIVE*)
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Beschreibung

630

CANx_RECEIVE konfiguriert ein Datenempfangsobjekt und liest den Empfangspuffer des Datenobjektes aus.

Der FB muss für jedes Datenobjekt in der Initialisierungsphase einmalig aufgerufen werden, um dem CAN-Controller die Identifier der Datenobjekte bekannt zu machen.

Im weiteren Programmzyklus wird CANx_RECEIVE zum Auslesen des jeweiligen Empfangspuffers aufgerufen, bei langen Programmzyklen auch mehrfach. Der Programmierer muss durch Auswertung des Bytes AVAILABLE dafür Sorge tragen, dass neu eingegangene Datenobjekte aus dem Puffer abgerufen und weiterverarbeitet werden.

Jeder Aufruf des FB dekrementiert das Byte AVAILABLE um 1. Ist der Wert von AVAILABLE gleich 0, sind keine Daten im Puffer.

Durch Auswerten des Ausgangs OVERFLOW kann ein Überlauf des Datenpuffers erkannt werden. Wenn OVERFLOW = TRUE, dann ist mindestens 1 Datenobjekt verloren gegangen.

HINWEIS

Soll CAN2_RECEIVE verwendet werden, muss zuvor mit *CAN2* die zweite CAN-Schnittstelle initialisiert werden.

Parameter der Eingänge

631

Parameter	Datentyp	Beschreibung
CONFIG	BOOL	TRUE (nur 1 Zyklus lang): Datenobjekt konfigurieren FALSE: diese Funktion wird nicht ausgeführt
CLEAR	BOOL	TRUE: löscht den Datenpuffer (Warteschlange) FALSE: diese Funktion wird nicht ausgeführt
ID	WORD	Nummer des Datenobjekt-Identifier Zulässige Werte: 0...2 047

Parameter der Ausgänge

632

Parameter	Datentyp	Beschreibung
DATA	ARRAY[0...7] OF BYTE	Das Array enthält maximal 8 Datenbytes
DLC	BYTE	Anzahl der übertragenen Bytes im Array DATA Mögliche Werte: 0...8.
RTR	BOOL	Wird nicht unterstützt
AVAILABLE	BYTE	Anzahl der eingegangenen Meldungen
OVERFLOW	BOOL	TRUE: Überlauf des Datenpuffers ⇨ Datenverlust! FALSE: Puffer noch nicht gefüllt

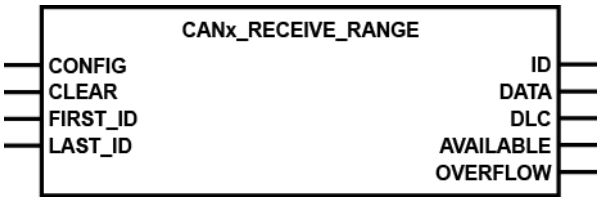
6.4.5 CANx_RECEIVE_RANGE

4179

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_RECEIVE_RANGE

9359

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB` (xx ≥ 05)

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
  Baustein NICHT für Sicherheitssignale!
  (Für Sicherheitssignale → *CAN_SAFETY_RECEIVE*)
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Beschreibung

2295

CANx_RECEIVE_RANGE konfiguriert eine Folge von Datenempfangsobjekten und liest den Empfangspuffer der Datenobjekte aus.

Für die 1. CAN-Schnittstelle sind max. 2048 IDs je 11 Bits möglich.
Für die 2. CAN-Schnittstelle sind max. 256 IDs je 11 ODER 29 Bits möglich.
Die 2. CAN-Schnittstelle benötigt eine lange Initialisierungszeit. Damit der Watchdog nicht anspricht, sollte bei größeren Bereichen der Vorgang auf mehrere Zyklen verteilt werden (→ *Beispiel: Initialisieren von CANx_RECEIVE_RANGE in 4 Zyklen* (→ Seite 92)).

Der FB muss für jede Folge von Datenobjekten in der Initialisierungsphase einmalig aufgerufen werden, um dem CAN-Controller die Identifier der Datenobjekte bekannt zu machen.

Der FB darf für die selben IDs an den selben CAN-Schnittstellen NICHT gemischt eingesetzt werden mit *CANx_RECEIVE* (→ Seite 88) oder CANx_RECEIVE_RANGE.

Im weiteren Programmzyklus wird CANx_RECEIVE_RANGE zum Auslesen des jeweiligen Empfangspuffers aufgerufen, bei langen Programmzyklen auch mehrfach. Der Programmierer muss durch Auswertung des Bytes AVAILABLE dafür Sorge tragen, dass neu eingegangene Datenobjekte aus dem Puffer SOFORT abgerufen und weiterverarbeitet werden, da die Daten nur einen Zyklus lang bereitstehen.

Jeder Aufruf des FB dekrementiert das Byte AVAILABLE um 1. Ist der Wert von AVAILABLE gleich 0, sind keine Daten im Puffer.

Durch Auswerten des Ausgangs OVERFLOW kann ein Überlauf des Datenpuffers erkannt werden. Wenn OVERFLOW = TRUE, dann ist mindestens 1 Datenobjekt verloren gegangen.

Receive-Puffer: max. 16 Software-Puffer pro Identifier.

Parameter der Eingänge

2290

Parameter	Datentyp	Beschreibung
CONFIG	BOOL	TRUE (nur 1 Zyklus lang): Datenobjekt konfigurieren FALSE: diese Funktion wird nicht ausgeführt
CLEAR	BOOL	TRUE: löscht den Datenpuffer (Warteschlange) FALSE: diese Funktion wird nicht ausgeführt
FIRST_ID	CAN1: WORD CAN2: DWORD	Nummer des ersten Datenobjekt-Identifiers der Folge. Zulässige Werte Normal Frame: 0...2 047 (2^{11}) Zulässige Werte Extended Frame: 0...536 870 912 (2^{29})
LAST_ID	CAN1: WORD CAN2: DWORD	Nummer des letzten Datenobjekt-Identifiers der Folge. Zulässige Werte Normal Frame: 0...2 047 (2^{11}) Zulässige Werte Extended Frame: 0...536 870 912 (2^{29}) LAST_ID muss größer sein als FIRST_ID.

Parameter der Ausgänge

4381

Parameter	Datentyp	Beschreibung
ID	CAN1: WORD CAN2: DWORD	ID des ausgegebenen Datenobjekts
DATA	ARRAY[0...7] OF BYTE	Das Array enthält maximal 8 Datenbytes
DLC	BYTE	Anzahl der übertragenen Bytes im Array DATA Mögliche Werte: 0...8.
AVAILABLE	BYTE	Anzahl der Meldungen im Puffer
OVERFLOW	BOOL	TRUE: Überlauf des Datenpuffers ⇨ Datenverlust! FALSE: Puffer noch nicht gefüllt

Beispiel: Initialisieren von CANx_RECEIVE_RANGE in 4 Zyklen

2294

PLC_PRG (PRG-ST) (-1/181/-1/88)

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     init : BOOL := FALSE;
0004     initstep : WORD := 1;
0005     can20 : CAN2;
0006     cr2 : CAN2_RECEIVE_RANGE;
0007     cnt : WORD;
0008 END_VAR
```

```
0001 (* CAN2 init. *)
0002 can20(ENABLE:= TRUE , START:= init, EXTENDED_MODE:= FALSE, BAUDRATE:= 125);
0003
0004 (* CAN2_RECEIVE_RANGE in mehreren Steps initialisieren *)
0005 CASE initstep OF
0006     1:
0007         cr2(CONFIG:= TRUE,CLEAR:= FALSE,FIRST_ID:= 16#100,LAST_ID:= 16#10F,ID=> ,DATA=> ,DLC=> ,AVAILABLE=> ,OVERFLOW=> );
0008         initstep := initstep + 1;
0009     2:
0010         cr2(CONFIG:= TRUE,CLEAR:= FALSE,FIRST_ID:= 16#110,LAST_ID:= 16#11F,ID=> ,DATA=> ,DLC=> ,AVAILABLE=> ,OVERFLOW=> );
0011         initstep := initstep + 1;
0012     3:
0013         cr2(CONFIG:= TRUE,CLEAR:= FALSE,FIRST_ID:= 16#120,LAST_ID:= 16#12F,ID=> ,DATA=> ,DLC=> ,AVAILABLE=> ,OVERFLOW=> );
0014         initstep := initstep + 1;
0015     4:
0016         cr2(CONFIG:= TRUE,CLEAR:= FALSE,FIRST_ID:= 16#130,LAST_ID:= 16#13F,ID=> ,DATA=> ,DLC=> ,AVAILABLE=> ,OVERFLOW=> );
0017         initstep := initstep + 1;
0018     ELSE
0019         cr2(CONFIG:=FALSE,CLEAR:= FALSE,FIRST_ID:= 16#100,LAST_ID:= 16#100,ID=> ,DATA=> ,DLC=> ,AVAILABLE=> ,OVERFLOW=> );
0020 END_CASE
0021
0022 init := FALSE;
0023
0024 (* Test *)
0025 IF cr2.available > 0 THEN
0026     cnt := cnt + 1;
0027 END_IF
```

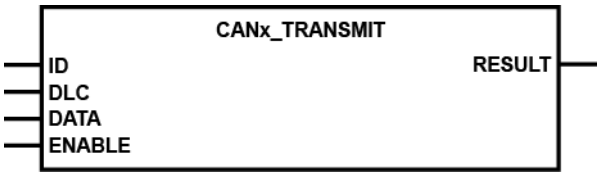
6.4.6 CANx_TRANSMIT

609

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_TRANSMIT

9362

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
  Baustein NICHT für Sicherheitssignale!
  (Für Sicherheitssignale → *CAN_SAFETY_TRANSMIT*)
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Beschreibung

612

CANx_TRANSMIT übergibt ein CAN-Datenobjekt (Message) an den CAN-Controller zur Übertragung.

Der FB wird für jedes Datenobjekt im Programmzyklus aufgerufen, bei langen Programmzyklen auch mehrfach. Der Programmierer muss durch Auswertung des Ausgangs RESULT dafür Sorge tragen, dass sein Sendeauftrag auch angenommen wurde. Vereinfacht gilt bei 125 kBit/s, dass pro 1 ms ein Sendeauftrag ausgeführt werden kann.

Über den Eingang ENABLE kann die Ausführung des FB zeitweilig gesperrt werden (ENABLE = FALSE). Damit kann z.B. eine Busüberlastung verhindert werden.

Mehrere Datenobjekte können quasi gleichzeitig verschickt werden, wenn jedem Datenobjekt ein Merkerflag zugeordnet wird und mit diesem die Ausführung des FB über den ENABLE-Eingang gesteuert wird.

HINWEIS

Soll CAN2_TRANSMIT verwendet werden, muss zuvor mit *CAN2* die zweite CAN-Schnittstelle initialisiert werden.

Parameter der Eingänge

613

Parameter	Datentyp	Beschreibung
ID	WORD	Nummer des Datenobjekt-Identifier Zulässige Werte: 0...2 047
DLC	BYTE	Anzahl der zu übertragenden Bytes aus dem Array DATA Zulässige Werte: 0...8
DATA	ARRAY[0...7] OF BYTE	Das Array enthält maximal 8 Datenbytes
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv

Parameter der Ausgänge

614

Parameter	Datentyp	Beschreibung
RESULT	BOOL	TRUE (nur 1 Zyklus lang): Der Baustein hat den Sendeauftrag angenommen

6.4.7 CAN1_EXT

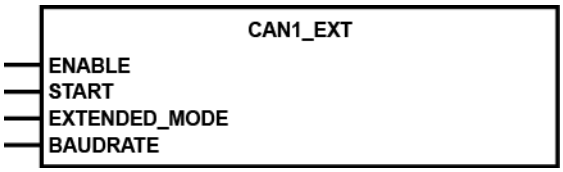
4192

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CAN1_EXT_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

4333

CAN1_EXT initialisiert die 1. CAN-Schnittstelle für den erweiterten Identifier (29 Bits).

Der FB muss aufgerufen werden, wenn die 1. CAN-Schnittstelle z.B. mit den Funktionsbibliotheken für *CAN-Bausteine nach SAE J1939* (→ Seite 104) benutzt werden soll.

Eine Änderung der Baud-Rate wird erst gültig nach Spannung Aus/Ein. Die Baud-Raten von CAN 1 und CAN 2 können unterschiedlich eingestellt werden.

Der Eingang START wird nur für einen Zyklus bei Neustart oder Restart der Schnittstelle gesetzt.

HINWEIS

Der FB muss **vor** den FBs CAN1_EXT_... ausgeführt werden.

Parameter der Eingänge

4334

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
START	BOOL	TRUE (im 1. Zyklus): Schnittstelle wird initialisiert FALSE: Initialisierungszyklus ist beendet
EXTENDED_MODE	BOOL	TRUE: Identifier der 1. CAN-Schnittstelle arbeitet mit 29 Bits FALSE: Identifier der 1. CAN-Schnittstelle arbeitet mit 11 Bits
BAUDRATE	WORD	Baud-Rate [kBit/s] Zulässige Werte: 50, 100,125, 250, 500, 1000 Voreinstellung = 125 kBit/s

6.4.8 CAN1_EXT_ERRORHANDLER

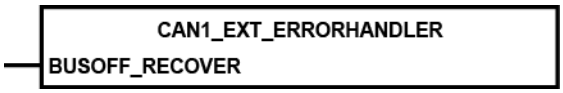
4195

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CAN1_EXT_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

4335

CAN1_EXT_ERRORHANDLER überwacht die 1. CAN-Schnittstelle und wertet die CAN-Fehler aus. Tritt eine bestimmte Anzahl von Übertragungsfehlern auf, so wird der CAN-Teilnehmer error-passiv. Verringert sich die Fehlerhäufigkeit, wird der Teilnehmer wieder error-activ (= Normalzustand).

Ist ein Teilnehmer schon error-passiv und es treten weiterhin Übertragungsfehler auf, wird er vom Bus abgeschaltet (= bus-off) und das Fehlerbit CANx_BUSOFF gesetzt. Die Rückkehr an den Bus ist nur möglich, wenn der Bus-off-Zustand behoben wird (Signal BUSOFF_RECOVER).

Das Fehlerbit CANx_BUSOFF muss anschließend im Applikations-Programm zurückgesetzt werden.

HINWEIS

Wenn die automatische Bus-Recover-Funktion genutzt werden soll (Default-Einstellung), darf CAN1_EXT_ERRORHANDLER **nicht** in das Programm eingebunden und instanziert werden!

Parameter der Eingänge

2177

Parameter	Datentyp	Beschreibung
BUSOFF_RECOVER	BOOL	TRUE (nur 1 Zyklus lang): > Neustart der CAN-Schnittstelle x > Bus-off-Zustand beheben FALSE: Baustein wird nicht ausgeführt

6.4.9 CAN1_EXT_RECEIVE

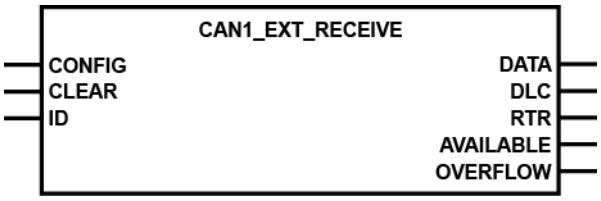
4302

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CAN1_EXT_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

4336

CAN1_EXT_RECEIVE konfiguriert ein Datenempfangsobjekt und liest den Empfangspuffer des Datenobjektes aus.

Der FB muss für jedes Datenobjekt in der Initialisierungsphase einmalig aufgerufen werden, um dem CAN-Controller die Identifier der Datenobjekte bekannt zu machen.

Im weiteren Programmzyklus wird CAN1_EXT_RECEIVE zum Auslesen des jeweiligen Empfangspuffers aufgerufen, bei langen Programmzyklen auch mehrfach. Der Programmierer muss durch Auswertung des Bytes AVAILABLE dafür Sorge tragen, dass neu eingegangene Datenobjekte aus dem Puffer abgerufen und weiterverarbeitet werden.

Jeder Aufruf des FB dekrementiert das Byte AVAILABLE um 1. Ist der Wert von AVAILABLE gleich 0, sind keine Daten im Puffer.

Durch Auswerten des Ausgangs OVERFLOW kann ein Überlauf des Datenpuffers erkannt werden. Wenn OVERFLOW = TRUE, dann ist mindestens 1 Datenobjekt verloren gegangen.

HINWEIS

Soll dieser FB verwendet werden, muss zuvor mit *CAN1_EXT* (→ Seite 95) die 1. CAN-Schnittstelle für den erweiterten ID initialisiert werden.

Parameter der Eingänge

2172

Parameter	Datentyp	Beschreibung
CONFIG	BOOL	TRUE (nur 1 Zyklus lang): Datenobjekt konfigurieren FALSE: diese Funktion wird nicht ausgeführt
CLEAR	BOOL	TRUE: löscht den Datenpuffer (Warteschlange) FALSE: diese Funktion wird nicht ausgeführt
ID	DWORD	Nummer des Datenobjekt-Identifier Zulässige Werte Normal Frame: 0...2 047 (2^{11}) Zulässige Werte Extended Frame: 0...536 870 912 (2^{29})

Parameter der Ausgänge

632

Parameter	Datentyp	Beschreibung
DATA	ARRAY[0...7] OF BYTE	Das Array enthält maximal 8 Datenbytes
DLC	BYTE	Anzahl der übertragenen Bytes im Array DATA Mögliche Werte: 0...8.
RTR	BOOL	Wird nicht unterstützt
AVAILABLE	BYTE	Anzahl der eingegangenen Meldungen
OVERFLOW	BOOL	TRUE: Überlauf des Datenpuffers ⇨ Datenverlust! FALSE: Puffer noch nicht gefüllt

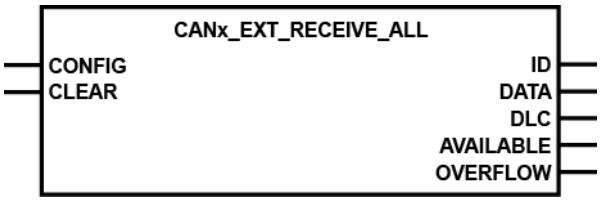
6.4.10 CANx_EXT_RECEIVE_ALL

4183

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_EXT_RECEIVE_ALL

9351

Enthalten in Bibliothek: `ifm_CAN1_EXT_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
  Baustein NICHT für Sicherheitssignale!
  (Für Sicherheitssignale → *CAN_SAFETY_RECEIVE*)
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Beschreibung

4326

CANx_EXT_RECEIVE_ALL konfiguriert alle Datenempfangsobjekte und liest den Empfangspuffer der Datenobjekte aus.

Der FB muss in der Initialisierungsphase einmalig aufgerufen werden, um dem CAN-Controller die Identifier der Datenobjekte bekannt zu machen.

Im weiteren Programmzyklus wird CANx_EXT_RECEIVE_ALL zum Auslesen des jeweiligen Empfangspuffers aufgerufen, bei langen Programmzyklen auch mehrfach. Der Programmierer muss durch Auswertung des Bytes AVAILABLE dafür Sorge tragen, dass neu eingegangene Datenobjekte aus dem Puffer abgerufen und weiterverarbeitet werden.

Jeder Aufruf des FB dekrementiert das Byte AVAILABLE um 1. Ist der Wert von AVAILABLE gleich 0, sind keine Daten im Puffer.

Durch Auswerten des Ausgangs OVERFLOW kann ein Überlauf des Datenpuffers erkannt werden. Wenn OVERFLOW = TRUE, dann ist mindestens 1 Datenobjekt verloren gegangen.

Receive-Puffer: max. 16 Software-Puffer pro Identifier.

Parameter der Eingänge

4329

Parameter	Datentyp	Beschreibung
CONFIG	BOOL	TRUE (nur 1 Zyklus lang): Datenobjekt konfigurieren FALSE: Baustein wird nicht ausgeführt
CLEAR	BOOL	TRUE: Löscht den Datenpuffer (Warteschlange) FALSE: diese Funktion wird nicht ausgeführt

Parameter der Ausgänge

2292

Parameter	Datentyp	Beschreibung
ID	DWORD	ID des ausgegebenen Datenobjekts
DATA	ARRAY[0...7] OF BYTE	Das Array enthält maximal 8 Datenbytes
DLC	BYTE	Anzahl der übertragenen Bytes im Array DATA Mögliche Werte: 0...8.
AVAILABLE	BYTE	Anzahl der Meldungen im Puffer
OVERFLOW	BOOL	TRUE: Überlauf des Datenpuffers ⇨ Datenverlust! FALSE: Puffer noch nicht gefüllt

6.4.11 CAN1_EXT_TRANSMIT

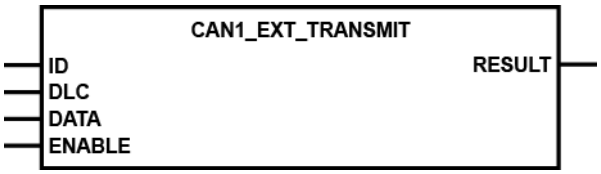
4307

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CAN1_EXT_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

4337

CAN1_EXT_TRANSMIT übergibt ein CAN-Datenobjekt (Message) an den CAN-Controller zur Übertragung.

Der FB wird für jedes Datenobjekt im Programmzyklus aufgerufen, bei langen Programmzyklen auch mehrfach. Der Programmierer muss durch Auswertung des FB-Ausgangs RESULT dafür Sorge tragen, dass sein Sendeauftrag auch angenommen wurde. Vereinfacht gilt bei 125 kBit/s, dass pro 1 ms ein Sendeauftrag ausgeführt werden kann.

Über den Eingang ENABLE kann die Ausführung der Funktion zeitweilig gesperrt werden (ENABLE = FALSE). Damit kann z.B. eine Busüberlastung verhindert werden.

Mehrere Datenobjekte können quasi gleichzeitig verschickt werden, wenn jedem Datenobjekt ein Merkerflag zugeordnet wird und mit diesem die Ausführung der Funktion über den ENABLE-Eingang gesteuert wird.

HINWEIS

Soll dieser FB verwendet werden, muss zuvor mit *CAN1_EXT* (→ Seite 95) die 1. CAN-Schnittstelle für den erweiterten ID initialisiert werden.

Parameter der Eingänge

4380

Parameter	Datentyp	Beschreibung
ID	DWORD	Nummer des Datenobjekt-Identifier Zulässige Werte: 11-Bit-ID: 0...2 047, 29-Bit-ID: 0...536 870 911
DLC	BYTE	Anzahl der zu übertragenden Bytes aus dem Array DATA Zulässige Werte: 0...8
DATA	ARRAY[0...7] OF BYTE	Das Array enthält maximal 8 Datenbytes
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv

Parameter der Ausgänge

614

Parameter	Datentyp	Beschreibung
RESULT	BOOL	TRUE (nur 1 Zyklus lang): Der Baustein hat den Sendeauftrag angenommen

6.5 CAN-Bausteine nach SAE J1939

Inhalt

7482

Das Netzwerkprotokoll SAE J1939 beschreibt die Kommunikation auf einem CAN-Bus in Nutzfahrzeugen zur Übermittlung von Diagnosedaten (z.B.Motordrehzahl, Temperatur) und Steuerungsinformationen.

6.5.1 CAN für die Antriebstechnik

Inhalt

7678

Unter der Norm SAE J1939 bietet die CiA dem Anwender ein CAN-Busprotokoll für die Antriebstechnik an. Hierbei wird der CAN-Controller der Schnittstelle in den "Extended Mode" geschaltet. Das bedeutet, dass die CAN-Nachrichten mit einem 29 Bit-Identifier übertragen werden. 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	Antriebs-Management
ISO 11992	"Truck & Trailer Interface"

Der 29 Bit-Identifier setzt sich aus zwei Teilen zusammen:
- einem 11 Bit-ID und
- einem 18 Bit-ID.

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.

HINWEIS

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

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)

(→ *CAN-ID* (→ Seite 77))

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

Beispiel: ausführliche Nachrichten-Dokumentation

7679

ETC1: Electronic Transmission Controller #1 (3.3.5) $0CF00203_{16}$

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	SA	3
Parameter group number	PGN	$00F002_{16}$
Identifier	ID	$0CF00203_{16}$
Data Field	SRC	Die Bedeutung der Datenbytes 1...8 wird an dieser Stelle nicht weiter behandelt. Sie ist der Herstellerdokumentation zu entnehmen.

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

Dabei bedeuten:

Bezeichung in der Herstellerdokumentation	Baustein-Eingang Bibliotheksfunktion	Beispielwert
Transmission repetition rate	RPT	T#10ms
Data length	LEN	8
PDU format	PF	240
PDU specific	PS	2
Default priority	PRIO	3
Data Page	PG	0
Source Address / Destination Address	SA / DA	3
Data Field	SRC / DST	Array-Adresse

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.

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 $0CF00203_{16}$ setzt sich wie folgt zusammen:

PRIO, reserv., PG		PF + PS				SA / DA	
0	C	F	0	0	2	0	3

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

SA / DA		Source / Destination Address (hexadezimal)		Source / Destination Address (dezimal)	
0	3	00	03	0	3

PF		PDU format (PF) (hexadezimal)		PDU format (PF) (dezimal)	
F	0	0F	00	16	0

PS		PDU specific (PS) (hexadezimal)		PDU specific (PS) (dezimal)	
0	2	00	02	0	2

PRIO, reserv., PG		PRIO, reserv., PG (binär)	
0	C	0000	1100

Von den 8 Bit ($0C_{16}$) werden nur die 5 niederwertigen Bits benötigt:

nicht benötigt			Priority			res.	PG
x	x	x	0_2	1_2	1_2	0_2	0_2
x			03_{10}			0_{10}	0_{10}

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

18_{16}:

nicht benötigt			Priority			res.	PG
x	x	x	1_2	1_2	0_2	0_2	0_2
x			6_{10}			0_{10}	0_{10}

$1C_{16}$:

nicht benötigt			Priority			res.	PG
x	x	x	1_2	1_2	1_2	0_2	0_2
x			7_{10}			0_{10}	0_{10}

6.5.2 Bausteine für SAE J1939

Inhalt

8566

Hier finden Sie Funktionsblöcke der CAN-Funktion für SAE J1939.

HINWEIS

Soll dieser FB verwendet werden, muss zuvor mit *CAN1_EXT* (→ Seite 95) die 1. CAN-Schnittstelle für den erweiterten ID initialisiert werden.

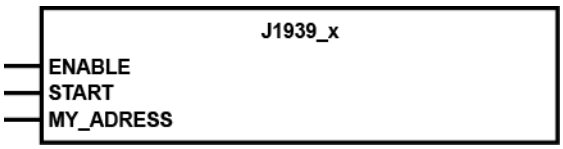
J1939_x

2274

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



J1939_1

9375

Enthalten in Bibliothek: `ifm_J1939_1_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Beschreibung

4325

J1939_x dient als Protokoll-Handler für das Kommunikationsprofil SAE J1939.

HINWEIS

J1939-Kommunikation über 1. CAN-Schnittstelle:	J1939-Kommunikation über 2. CAN-Schnittstelle:
► Schnittstelle zuvor mit *CAN1_EXT* (→ Seite 95) initialisieren!	► Schnittstelle zuvor mit *CAN2* initialisieren!

Zur Abwicklung der Kommunikation muss der Protokoll-Handler in jedem Programmzyklus aufgerufen werden. Dazu wird der Eingang ENABLE auf TRUE gesetzt.

Der Protokoll-Handler wird gestartet, wenn der Eingang START für einen Zyklus auf TRUE gesetzt wird.

Über MY_ADRESS wird dem Controller eine Geräteadresse übergeben. Sie muss sich von Adressen der anderen J1939-Busteilnehmer unterscheiden. Sie kann dann von anderen Busteilnehmern ausgelesen werden.

Parameter der Eingänge

469

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
START	BOOL	TRUE (nur 1 Zyklus): Protokoll-Handler wird gestartet FALSE: im weiteren Programmablauf
MY_ADRESS	BYTE	Node-ID des Geräts

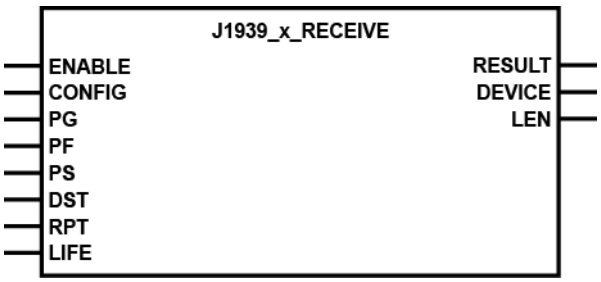
J1939_x_RECEIVE

2278

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



J1939_1_RECEIVE

9393

Enthalten in Bibliothek: `ifm_J1939_1_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Beschreibung

2288

J1939_x_RECEIVE dient dem Empfang einer einzelnen Nachricht oder eines Nachrichtenblocks.

Dazu muss der FB über den Eingang CONFIG für einen Zyklus initialisiert werden. Bei der Initialisierung werden die Parameter PG, PF, PS, RPT, LIFE und die Speicheradresse des Datenarrays DST übergeben.

► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

► Der Datenempfang muss über das RESULT-Byte ausgewertet werden. Wird RESULT = 1, können die Daten von der über DST übergebenen Speicheradresse ausgelesen und weiter verarbeitet werden.

> Der Empfang einer neuen Nachricht überschreibt die Daten auf der Speicheradresse DST.

> Die Anzahl der empfangenen Nachrichten-Bytes wird über den Ausgang LEN angezeigt.

> Wird RESULT = 3, wurden im angegebenen Zeitfenster (LIFE * RPT) keine gültigen Nachrichten empfangen.

HINWEIS

Dieser Baustein muss auch eingesetzt werden, wenn die Nachrichten mit den FBs J1939_..._REQUEST angefordert werden.

Parameter der Eingänge

457

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
CONFIG	BOOL	TRUE (nur 1 Zyklus): zur Konfiguration des Datenobjektes FALSE: im weiteren Programmablauf
PG	BYTE	Page address (normalerweise = 0)
PF	BYTE	PDU Format Byte
PS	BYTE	PDU Specific Byte
DST	DWORD	Zieladresse des Arrays, unter der die Empfangsdaten abgelegt werden ► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
RPT	TIME	Überwachungszeit Innerhalb dieses angegebenen Zeitfensters müssen die Telegramme wiederholt empfangen werden. Andernfalls erfolgt eine Fehlersignalisierung. Wird keine Überwachung gewünscht, muss RPT auf T#0s gesetzt werden.
LIFE	BYTE	Anzahl der zulässigen fehlerhaften Überwachungsaufrufe

Parameter der Ausgänge

458

Parameter	Datentyp	Beschreibung
RESULT	BYTE	0 = nicht aktiv 1 = Daten wurden empfangen 3 = Fehler-Signalisierung: Innerhalb des Zeitfensters (LIFE * RPT) wurde nichts empfangen
DEVICE	BYTE	Geräteadresse des Absenders
LEN	WORD	Anzahl der empfangenen Bytes

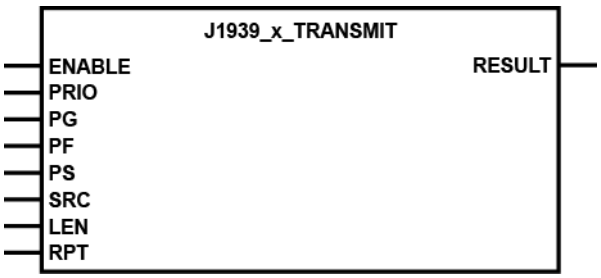
J1939_x_TRANSMIT

2279

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



J1939_1_TRANSMIT

4322

Enthalten in Bibliothek: `ifm_J1939_1_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Beschreibung

2298

J1939_x_TRANSMIT ist für das Versenden einzelner Nachrichten oder Nachrichtenblocks verantwortlich. Dazu werden dem FB die Parameter PG, PF, PS, RPT und die Adresse des Datenarrays SRC übergeben.

► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

► Zusätzlich die Anzahl der zu übertragenen Datenbytes und die Priorität (typisch 3, 6 oder 7) übergeben.

► Da das Versenden der Daten über mehrere Steuerungszyklen abgewickelt wird, muss der Vorgang über das RESULT-Byte ausgewertet werden. Wird RESULT = 1, wurden alle Daten übertragen.

Info

Wenn mehr als 8 Bytes gesendet werden sollen, wird ein "multi package transfer" durchgeführt.

Parameter der Eingänge

439

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
PRIO	BYTE	Nachrichtenpriorität (0…7)
PG	BYTE	Page address (normalerweise = 0)
PF	BYTE	PDU Format Byte
PS	BYTE	PDU Specific Byte
SRC	DWORD	Speicheradresse des Datenarrays, dessen Inhalt übertragen werden soll ► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
LEN	WORD	Anzahl der zu sendenden Bytes
RPT	TIME	Wiederholzeit, innerhalb der die Daten-Telegramme zyklisch versendet werden

Parameter der Ausgänge

440

Parameter	Datentyp	Beschreibung
RESULT	BYTE	0 = nicht aktiv 1 = Datenübertragung beendet 2 = Baustein aktiv (Datenübertragung) 3 = Fehler, Daten können nicht gesendet werden

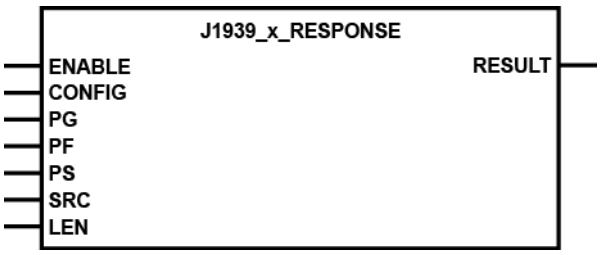
J1939_x_RESPONSE

2280

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



J1939_1_RESPONSE

9399

Enthalten in Bibliothek: `ifm_J1939_1_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Beschreibung

2299

J1939_x_RESPONSE organisiert die automatische Antwort auf ein Request-Telegramm (Anforderungstelegramm).

Der FB ist für das automatische Versenden von Nachrichten auf "Global Requests" und "Specific Requests" verantwortlich. Dazu muss der FB über den Eingang CONFIG für einen Zyklus initialisiert werden.

Dem FB werden die Parameter PG, PF, PS, RPT und die Adresse des Datenarrays SRC übergeben.

► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

► Zusätzlich die Anzahl der zu übertragenen Datenbytes übergeben.

Parameter der Eingänge

451

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
CONFIG	BOOL	TRUE (nur 1 Zyklus lang): zur Konfiguration des Datenobjektes FALSE: im weiteren Programmablauf
PG	BYTE	Page address (normalerweise = 0)
PF	BYTE	PDU Format Byte
PS	BYTE	PDU Specific Byte
SRC	DWORD	Speicheradresse des Datenarrays, dessen Inhalt übertragen werden soll ► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
LEN	WORD	Anzahl der zu sendenden Bytes

Parameter der Ausgänge

440

Parameter	Datentyp	Beschreibung
RESULT	BYTE	0 = nicht aktiv 1 = Datenübertragung beendet 2 = Baustein aktiv (Datenübertragung) 3 = Fehler, Daten können nicht gesendet werden

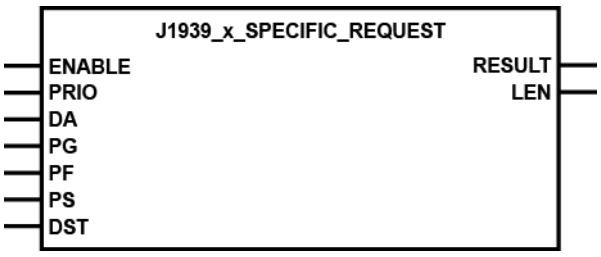
J1939_x_SPECIFIC_REQUEST

2281

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



J1939_1_SPECIFIC_REQUEST

8884

Enthalten in Bibliothek: `ifm_J1939_1_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Beschreibung

2300

J1939_x_SPECIFIC_REQUEST ist für das automatische Anfordern einzelner Nachrichten von einem bestimmten (specific) J1939-Netzwerkteilnehmer verantwortlich. Dazu werden dem FB die logische Geräteadresse DA, die Parameter PG, PF, PS und die Adresse des Arrays DST übergeben, in dem die empfangenen Daten abgelegt werden.

► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

► Zusätzlich die Priorität (typisch 3, 6 oder 7) übergeben.

► Da das Anfordern der Daten über mehrere Steuerungszyklen abgewickelt werden kann, muss dieser Vorgang über das RESULT-Byte ausgewertet werden. Wird RESULT = 1, wurden alle Daten empfangen.

> Der Ausgang LEN zeigt an, wie viele Datenbytes empfangen wurden.

Parameter der Eingänge

445

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
PRIO	BYTE	Priorität (0…7)
DA	BYTE	Logische Adresse (Zieladresse) des angeforderten Gerätes
PG	BYTE	Page address (normalerweise = 0)
PF	BYTE	PDU Format Byte
PS	BYTE	PDU Specific Byte
DST	DWORD	Zieladresse des Arrays, unter der die Empfangsdaten abgelegt werden ► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Parameter der Ausgänge

446

Parameter	Datentyp	Beschreibung
RESULT	BYTE	0 = nicht aktiv 1 = Datenübertragung beendet 2 = Baustein aktiv (Datenübertragung) 3 = Fehler, Daten können nicht gesendet werden
LEN	WORD	Anzahl der empfangenen Datenbytes

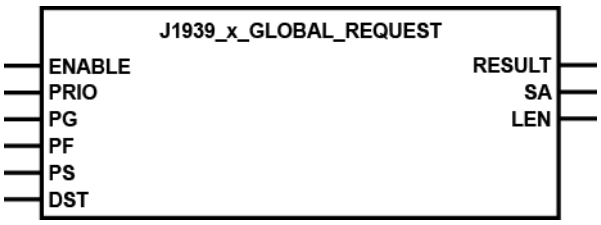
J1939_x_GLOBAL_REQUEST

2282

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



J1939_1_GLOBAL_REQUEST

4315

Enthalten in Bibliothek: `ifm_J1939_1_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Beschreibung

2301

J1939_x_GLOBAL_REQUEST organisiert globales Anfordern und Empfangen von Daten der Netzwerkteilnehmer.

Der Funktionsblock ist für das automatische Anfordern einzelner Nachrichten von allen (global) aktiven J1939-Netzwerkteilnehmern verantwortlich. Dazu werden dem FB die logische Geräteadresse DA, die Parameter PG, PF, PS und die Adresse des Arrays DST übergeben, in dem die empfangenen Daten abgelegt werden.

► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

► Zusätzlich die Priorität (typisch 3, 6 oder 7) übergeben.

► Da das Anfordern der Daten über mehrere Steuerungszyklen abgewickelt werden kann, muss dieser Vorgang über das RESULT-Byte ausgewertet werden. Wird RESULT = 1, wurden alle Daten empfangen.

> Der Ausgang LEN zeigt an, wie viele Datenbytes empfangen wurden.

Parameter der Eingänge

463

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
PRIO	BYTE	Priorität (0…7)
PG	BYTE	Page address (normalerweise = 0)
PF	BYTE	PDU Format Byte
PS	BYTE	PDU Specific Byte
DST	DWORD	Zieladresse des Arrays, unter der die Empfangsdaten abgelegt werden ► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Parameter der Ausgänge

464

Parameter	Datentyp	Beschreibung
RESULT	BYTE	0 = nicht aktiv 1 = Datenübertragung beendet 2 = Baustein aktiv (Datenübertragung) 3 = Fehler, Daten können nicht gesendet werden
SA	BYTE	Logische Geräteadresse (Sendeadresse) des angeforderten Gerätes
LEN	WORD	Anzahl der empfangenen Datenbytes

6.6 ifm-CANopen-Bibliotheken

Inhalt

1856

HINWEIS

Folgende Geräte unterstützen CANopen nur für die 1. CAN-Schnittstelle:
- Controller CR0020, CR200, CR0301, CR0302, CR0303, CR0505, CR250n, CR7021, CR7201, CR7506
- PDM360smart: CR1070, CR1071

Wurde bereits der CANopen-Master eingefügt, kann das Gerät nicht mehr als CANopen-Slave über CoDeSys genutzt werden.

Die Implementierung eines eigenen Protokolls auf Schnittstelle 2 oder Nutzung des Protokolls nach SAE J1939 oder ISO 11992 ist aber jederzeit möglich.

Folgende Geräte können auf allen CAN-Schnittstellen mit allen Protokollen genutzt werden:
- BasicController: CR040n
- BasicDisplay: CR0451
- Controller CRnn32, CRnn33
- PDM360: CR1050, CR1051
- PDM360compact: CR1052, CR1053, CR1055, CR1056
- PDM360NG: CR108n

6.6.1 Technisches zu CANopen

Inhalt

7773

CANopen-Tabellen (→ Seite 312) zur Übersicht finden Sie im Anhang.

CANopen Netzwerk-Konfiguration, Status- und Fehlerbehandlung

7471

Bei allen programmierbaren Geräten wird die CANopen-Schnittstelle von CoDeSys eingesetzt. Während Sie die Netzwerkkonfiguration und die Parametrierung der angeschlossenen Geräte direkt über die Programmiersoftware vornehmen, können die Fehlermeldungen nur über verschachtelte Variablenstrukturen im CANopen-Stack erreicht werden. Die nachfolgende Dokumentation zeigt Ihnen den Aufbau und die Anwendung der Netzwerkkonfiguration und beschreibt die Bausteine der ifm CANopen-Gerätebibliotheken.

Die Kapitel *CANopen-Unterstützung durch CoDeSys* (→ Seite 123), *CANopen-Master* (→ Seite 125), *CANopen-Slave* (→ Seite 146) und *CANopen-Netzwerkvariablen* (→ Seite 156) beschreiben die internen Bausteine des CoDeSys-CANopen-Stacks und ihre Anwendung. Außerdem bekommen Sie einen Einblick über die Anwendung des Netzwerkkonfigurators.

Die Kapitel über die Bibliotheken `ifm_CRnnnn_CANopenMaster_Vxxyyzz.lib` und `ifm_CRnnnn_CANopenSlave_Vxxyyzz.lib` beschreiben alle Bausteine zur Fehlerverarbeitung und zur Abfrage des Gerätestatus beim Einsatz als Master oder Slave.

HINWEIS

Unabhängig vom eingesetzten Gerät haben alle Bibliotheken den gleichen Aufbau der Funktionsschnittstellen. Die geringfügigen Unterschiede (z.B. CANOPEN_LED_STATUS) werden direkt in den jeweiligen Bausteinen beschrieben.

Es ist zwingend notwendig, dass Sie nur die jeweilige gerätespezifische Bibliothek einsetzen. Den Zusammenhang können Sie an der integrierten Geräte-Artikelnummer erkennen.
Beispiel CR0020: → `ifm_CR0020_CANopenMaster_Vxxyyzz.lib`

→ Kapitel *Target einrichten* (→ Seite 28)

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

CANopen-Unterstützung durch CoDeSys

1857

Allgemeines zu CANopen mit CoDeSys

2075

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 einigen Punkten eingeschränkt) unter CoDeSys konfigurieren.

CANopen ist als CoDeSys-Bibliothek in IEC 61131-3 implementiert. Die Bibliothek stützt sich auf sehr einfache Basis-CAN-Funktionen ab, die als CAN-Treiber bezeichnet werden.

Durch die Realisierung der CANopen-Funktionen als CoDeSys-Bibliothek ist eine einfache Skalierung des Zielsystems möglich. So verbraucht die CANopen-Funktion nur dann Zielsystem-Ressourcen, wenn die Funktion auch wirklich genutzt wird. Zur weiteren Schonung von Zielsystem-Ressourcen wird durch CoDeSys automatisch eine genau der Konfiguration entsprechende Datenbasis für die CANopen-Master-Funktion generiert.

Ab der Programmiersystemversion CoDeSys Version 2.3.6.0 kann ein **ecomat*mobile***-Controller als CANopen-Master und als CANopen-Slave genutzt werden.

HINWEIS

Für alle **ecomat*mobile***-Controller und das PDM360smart müssen Sie die CANopen-Bibliotheken mit folgendem Zusatz einsetzen:

- Für CR0032 Target-Version bis V01, alle anderen Geräte bis V04.00.05: "**OptTable**"
- Für CR0032 Target-Version ab V02, alle anderen Geräte ab V05: "**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 alle anderen programmierbaren Geräte genutzt (z.B. PDM360compact).

CANopen Begriffe und Implementation

1858

Nach der CANopen-Spezifikation gibt es keine Master und Slaves in einem CAN-Netz. Stattdessen gibt es nach CANopen einen NMT-Master (NMT = Netzwerk-Management), einen Konfigurationsmaster usw., immer mit der Vorstellung, dass alle Teilnehmer eines CAN-Netzes gleichberechtigt sind.

Die Implementierung geht davon aus, dass ein CAN-Netz als Peripherie einer CoDeSys-programmierbaren Steuerung dient. Demzufolge wird eine **ecomat*mobile***-Steuerung oder ein PDM360-Display im CAN-Konfigurator von CoDeSys als CANopen-Master bezeichnet. Dieser 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.

IDs (Adressen) in CANopen

3952

In CANopen werden diverse Arten von 'Adressen' (hier: IDs) unterschieden:

- COB-ID
  Der **C**ommunication-**Ob**ject-**Id**entifier adressiert die Nachricht (= das Kommunikationsobjekt) im Geräteverzeichnis. Ein Kommunikationsobjekt besteht aus einem oder mehreren CAN-Nachrichten mit bestimmten Aufgaben, z.B.:
  - PDO (**P**rocess **D**ata **O**bject = Nachrichten-Objekt mit Prozessdaten),
  - SDO (**S**ervice **D**ata **O**bject = Nachrichten-Objekt mit Servicedaten),
  - Emergency (Nachrichten-Objekt mit Notfalldaten),
  - Time (Nachrichten-Objekt mit Zeitangaben) oder
  - Error Control (Nachrichten-Objekt mit Fehlermeldungen).
- CAN-ID
  Der **CAN-Id**entifier definiert netzwerkweit CAN-Nachrichten. Der CAN-ID ist Hauptbestandteil des Arbitration-Feldes eines CAN-Datenübertragungsblocks. Je niedriger der CAN-ID, desto höher die Priorität der Meldung.
- Download-ID
  Der Download-ID bezeichnet den Node-ID für Service-Kommunikation per SDO für den Programm-Download und das Debuggen.
- Node-ID
  Der **Node-Id**entifier ist ein eindeutiger Bezeichner für CANopen-Geräte (Devices) im CAN-Netzwerk. Der Node-ID ist auch Bestandteil einiger vordefinierter Verbindungssätze (→ *Funktions-Code / Predefined Connectionset* (→ Seite 315)).

Vergleich Download-ID vs. COB-ID:

Controller Programm-Download		CANopen	
Download-ID	**COB-ID SDO**	**Node-ID**	**COB-ID SDO**
1…127	TX: 580_{16} + Download-ID	1…127	TX: 580_{16} + Node-ID
	RX: 600_{16} + Download-ID		RX: 600_{16} + Node-ID

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

CANopen-Master

Inhalt

1859

Abgrenzung zu anderen CANopen-Bibliotheken

1990

Die von 3S (Smart Software Solutions) realisierte CANopen-Bibliothek grenzt sich in verschiedenen Punkten von auf dem Markt befindlichen Systemen ab. Sie wurde nicht entwickelt, um andere Bibliotheken namhafter Hersteller überflüssig zu machen, sondern ist bewusst für den Einsatz mit dem CoDeSys-Programmier- und Laufzeitsystem optimiert.

Die Bibliotheken wurden nach der Spezifikation der CiA DS301, V402 erstellt.

Für Sie als Anwender der CoDeSys-CANopen-Bibliothek ergeben sich folgende Vorteile:

- Die Implementierung ist unabhängig vom Zielsystem und damit praktisch auf jeder mit CoDeSys-programmierbaren Steuerung direkt verwendbar.
- Das komplette System beinhaltet den CANopen-Konfigurator und die Einbindung in das Entwicklungssystem.
- Die CANopen-Funktionalität ist nachladbar. Das bedeutet, dass die CANopen-Funktionen ohne Änderung des Betriebssystems geladen und aktualisiert werden können.
- Die Ressourcen des Zielsystems werden geschont, da nicht die Ressourcen für eine Maximalkonfiguration vorgehalten werden.
- Automatisches Aktualisieren der Ein- und Ausgänge ohne zusätzliche Maßnahmen.

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

- **PDOs Senden:** Master sendet zu den Slaves (Slave = Knoten, Device)
  Senden ereignisgesteuert (d.h. bei Änderung), zeitgesteuert (RepeatTimer) oder 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.
  Applikationsgesteuertes 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.
- **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 Applikation gesetzt werden.

Folgende in CANopen definierten Funktionen werden von der 3S (Smart Software Solutions) CANopen-Bibliothek derzeit **nicht** unterstützt:

- Dynamische Identifier-Zuordnung
- Dynamische SDO-Verbindungen
- Blockweiser SDO-Transfer, segmentierter SDO-Transfer (die Funktionalität kann mit *CANx_SDO_READ* (→ Seite 185) und *CANx_SDO_WRITE* (→ Seite 187) in der jeweiligen **ifm**-Gerätebibliothek realisiert werden).
- Alle oben nicht genannten Möglichkeiten des CANopen Protokolls.

Ein CANopen-Projekt erstellen

1860

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 → CoDeSys-Handbuch unter [Ressourcen] > [Steuerungskonfiguration] und in der Online-Hilfe.

► Nach der Neuanlage eines Projektes (→ Kapitel *Target einrichten* (→ Seite 28)) in der Steuerungskonfiguration über [Einfügen] > [Unterelement anhängen] den CANopen-Master einfügen.

> Bei Steuerungen mit 2 oder mehr CAN-Schnittstellen wird automatisch Schnittstelle 1 für den Master konfiguriert.

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

- 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 `SysLibSem.LIB` und `SysLibCallback.LIB`.

► Um die vorbereiteten Netzwerkdiagnose-, Status- und EMCY-Funktion zu nutzen, die Bibliothek `ifm_CRnnnn_CANopenMaster_Vxxyyzz.LIB` manuell im Bibliotheksverwalter einfügen. Ohne diese Bibliothek müssen Sie die Netzwerkinformationen direkt aus den verschachtelten Strukturen der CoDeSys-CANopen-Bibliotheken auslesen.

► Zusätzlich die folgenden Bibliotheken und Software-Module einbinden:

- die Gerätebibliothek für die jeweilige Hardware, z.B. `ifm_CR0020_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 (→ Kapitel *Programmiersystem über Templates einrichten* (→ Seite 32)).
  Für die EDS-Dateien von Fremd-Knoten ist der jeweilige Hersteller verantwortlich.

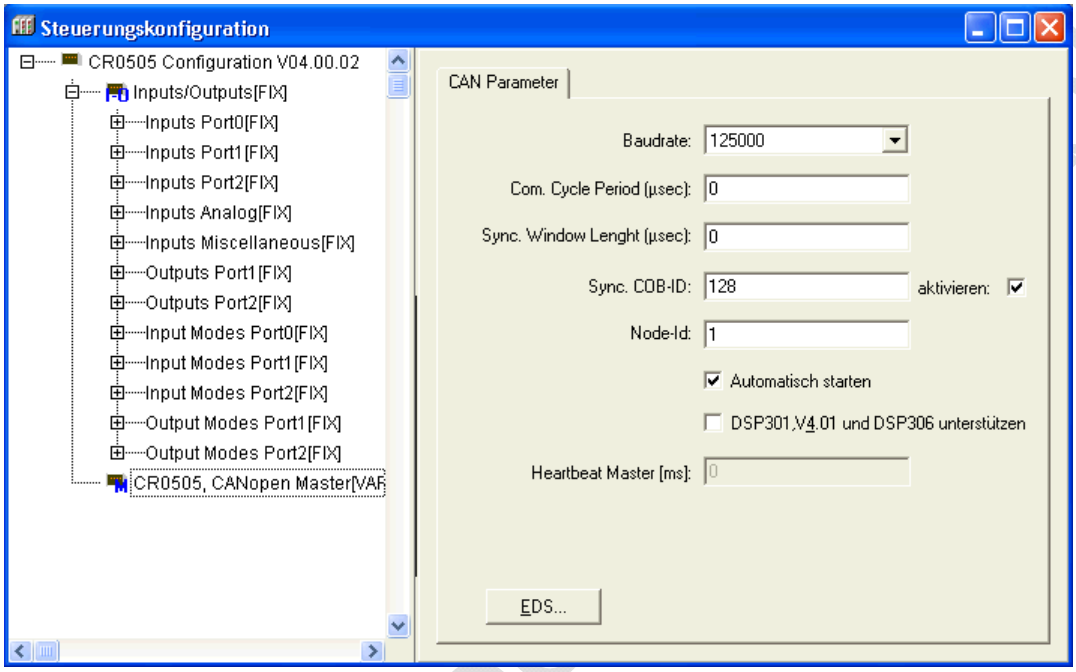
CANopen-Master: Register [CAN-Parameter]

1967

In diesem Dialogfenster können für den Master die wichtigsten Parameter eingestellt werden. Bei Bedarf kann über die Schaltfläche [EDS...] der Inhalt der Master-EDS-Datei angesehen werden.

Diese Schaltfläche wird nur angezeigt, wenn die EDS-Datei (z.B. `CR0020MasterODEntry.EDS`) im Verzeichnis `...\CoDeSys V2.3\Library\PLCConf` vorhanden ist.

Aus dieser EDS-Datei wird bei der Übersetzung des Applikations-Programms automatisch das Objektverzeichnis des Masters erzeugt.



Beispiel: Steuerungskonfiguration für CR0505 CANopen-Master

CAN Parameter: Baudrate

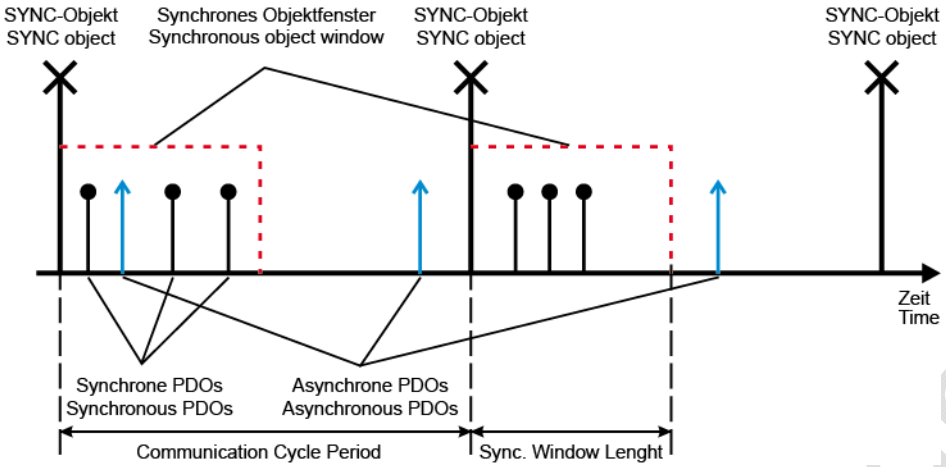
10028

Wählen Sie an dieser Stelle bitte die Baudrate für den Master aus.
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 Applikationen 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 den Download-ID!) des Masters ein.
Die Knotennummer darf im Netzwerk nur einmal vorkommen, andernfalls kommt es zu Kommunikationsstörungen.

CAN Parameter: Automatisch starten

10032

Das Netzwerk und die angeschlossenen Knoten werden nach einer erfolgreichen Konfiguration in den Zustand "operational" gesetzt und damit gestartet.
Ist das Optionsfeld nicht angewählt, muss das Netzwerk manuell gestartet werden.

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.

CANopen-Slaves einfügen und konfigurieren

Inhalt

1861

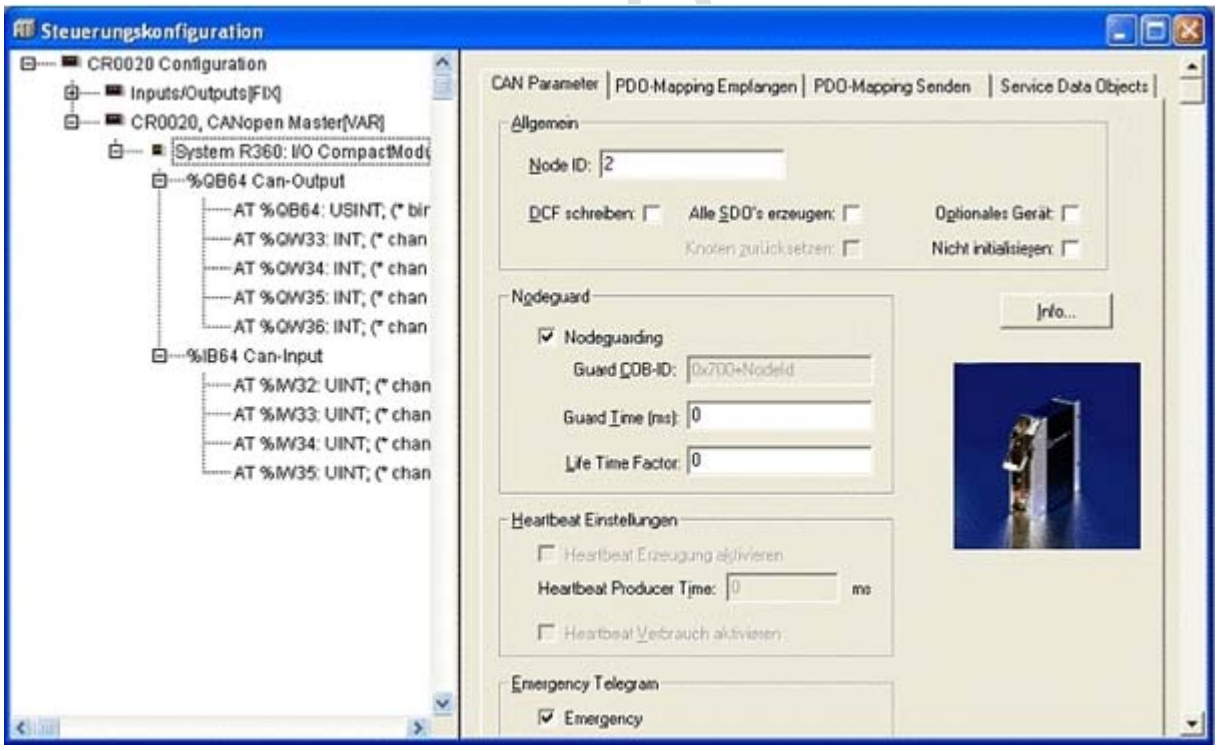
Als nächstes können Sie nun die CANopen-Slaves einfügen. Dazu müssen Sie erneut den Dialog in der Steuerungskonfiguration [Einfügen] > [Unterelement anhängen] aufrufen. Es steht Ihnen eine Liste der im Verzeichnis PLC_CONF gespeicherten CANopen-Gerätebeschreibungen (EDS-Dateien) zur Verfügung. Durch Auswahl des entsprechenden Gerätes wird dieses direkt in den Baum der Steuerungskonfiguration eingefügt.

HINWEIS

Wird ein Slave über den Konfigurationsdialog in CoDeSys hinzugefügt, wird für jeden Knoten dynamisch Quellcode in das Applikations-Programm integriert. Gleichzeitig verlängert jeder zusätzlich hinzugefügte Slave die Zykluszeit des Applikations-Programms. 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.

Bitte beachten Sie, dass die maximale Zeit für einen SPS-Zyklus von ca. 50 ms nicht überschritten werden sollte (Watchdog-Zeit: 100 ms).



Beispiel: Steuerungskonfiguration für CR0020 CANopen-Master mit angeschlossenem I/O CompactModul

CANopen-Slave: Register [CAN Parameter]

1968

CAN Parameter: Node-ID

10036

Der Node-ID dient zur eindeutigen Identifizierung des CAN-Moduls und entspricht der am Modul eingestellten Nummer zwischen 1 und 127.

Der ID wird dezimal eingegeben und wird automatisch um eins erhöht, wenn Sie ein neues Modul hinzufügen.

CAN Parameter: DCF schreiben

10037

Ist [DCF schreiben] aktiviert, wird nach dem Einfügen einer EDS-Datei im eingestellten Verzeichnis für Übersetzungsdateien eine DCF-Datei erstellt, deren Namen sich zusammensetzt aus dem Namen der EDS-Datei und dem angehängten Node-ID.

CAN Parameter: Alle SDOs erzeugen

10038

Ist diese Option aktiviert, werden für alle Kommunikationsobjekte SDOs erzeugt.
Default-Werte werden nicht erneut geschrieben!

CAN Parameter: Knoten zurücksetzen

10039

Der Slave wird zurückgesetzt ("load"), sobald die Konfiguration in die Steuerung geladen wird.

CAN Parameter: Optionales Gerät

10040

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

Wird der Slave zu einem späteren Zeitpunkt an das Netzwerk angeschlossen und erkannt, wird er automatisch gestartet. Dazu müssen Sie die Option [Automatisch starten] in den CAN-Parametern des Masters angewählt haben.

CAN Parameter: Nicht initialisieren

10041

Wird diese Option aktiviert, nimmt der Master 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.

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

CAN Parameter: Emergency Telegram

10043

Die Option ist im Normalfall angewählt. Die EMCY-Nachrichten werden mit dem angegebenen Identifier übertragen.

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 ggf. experimentell ermittelt werden.

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

Register [PDO-Mapping empfangen] und [PDO-Mapping senden]

1969

Die Registerkarten [PDO-Mapping empfangen] und [PDO-Mapping senden] im Konfigurationsdialog eines CAN-Moduls ermöglichen es, dass in der EDS-Datei beschriebene "Mapping" (Zuordnung zwischen lokalem Objektverzeichnis und PDOs vom/zum CANopen-Slave) des Moduls zu verändern (wenn es vom CAN-Modul unterstützt wird).

Auf der linken Seite stehen alle "mapbaren" Objekte der EDS-Datei zur Verfügung und können zu den PDOs (Process Data Objects) der rechten Seite hinzugefügt oder 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 einen eindeutigen 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 zwischen zwei Übertragungen des PDOs liegen soll.

Register [Service Data Objects]

1970

Index, Name, Wert, Typ und Default

Hier werden alle Objekte der EDS- oder DCF-Datei aufgelistet, die im Bereich von Index 2000_{16} bis $9FFF_{16}$ liegen und als beschreibbar definiert sind. Zu jedem Objekt werden Index, Name, Wert, Typ und Default angegeben. Der Wert kann verändert werden. Markieren Sie den Wert und drücken Sie die [Leertaste]. Nach Änderung können Sie den Wert durch die Taste [Eingabe] bestätigen oder mit [ESC] verwerfen.

Bei der Initialisierung des CAN-Buses werden die eingestellten Werte in Form von SDOs (Service Data Object) an die CAN-Module übertragen und haben damit direkten Einfluss auf das Objektverzeichnis des CANopen-Slaves. Sie werden im Normalfall bei jedem Start des Applikations-Programms neu geschrieben – unabhängig davon, ob sie im CANopen-Slave dauerhaft gespeichert werden.

Der Master zur Laufzeit

Inhalt

8569

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

Die CANopen-Master-Bibliothek stellt der CoDeSys-Applikation implizite Dienste zur Verfügung, die für die meisten Applikationen ausreichend sind. Diese Dienste werden für den Anwender transparent integriert und stehen in der Applikation ohne zusätzliche Aufrufe zur Verfügung. In der nachfolgenden Beschreibung wird davon ausgegangen, dass Sie zur Nutzung der Netzwerkdiagnose-, Status- und EMCY-Funktionen die Bibliothek `ifm_CRnnnn_CANopenMaster_Vxxyyzz.LIB` manuell im Bibliotheksverwalter eingefügt haben.

Zu den Diensten der CANopen-Master-Bibliothek zählen:

Reset aller konfigurierten Slaves am Bus beim Systemstart

8570

Um die Slaves zurückzusetzen, wird standardmäßig das NMT-Kommando "Reset Remote Node" benutzt, explizit für jeden Slave einzeln. (NMT steht nach CANopen für **N**etwork **M**anagment**t**. Die einzelnen Kommandos sind im CAN-Dokument DSP301 beschrieben.) Um Slaves mit weniger leistungsstarken CAN-Controllern nicht zu überlasten, ist es sinnvoll, die Slaves mit einem Kommando "All Remote Nodes" zurückzusetzen.
Der Dienst wird für **alle** konfigurierten Slaves ausgeführt mit *CANx_MASTER_STATUS* (→ Seite 169) mit GLOBAL_START=TRUE. Sollen die Slaves **einzeln** zurückgesetzt werden, muss dieser Eingang auf FALSE gesetzt werden.

Abfrage des Slave-Gerätetyps

8021

Abfrage des Slave-Gerätetyps mittels SDO (Abfrage des Objekts 1000_{16}) und Vergleich mit dem konfigurierten Slave-ID:

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

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

8574

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

Zum Starten der Slaves wird normalerweise das NMT-Kommando "Start remote node" benutzt. Wie beim "Reset" kann dieses Kommando durch "Start All Remote Nodes" ersetzt werden.

Der Dienst ist mittels CANx_Master_STATUS mit GLOBAL_START=TRUE aufrufbar.

Zyklisches Senden der SYNC-Message

8025

Dieser Wert ist nur bei der Konfiguration einstellbar.

Nodeguarding mit Lifetime-Überwachung

8576

Nodeguarding mit Lifetime-Überwachung für jeden Slave einstellbar:

Der Fehlerstatus kann für max. 8 Slaves mittels *CANx_MASTER_STATUS* (→ Seite 169) mit ERROR_CONTROL=TRUE überwacht werden.

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

Heartbeat vom Master an die Slaves

8577

Der Fehlerstatus kann für max. 8 Slaves mittels CANx_MASTER_STATUS mit ERROR_CONTROL=TRUE überwacht werden.

Empfangen von Emergency-Messages

8578

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

Die Fehlernachrichten können mittels CANx_MASTER_STATUS mit EMERGENCY_OBJECT_SLAVES=TRUE ausgelesen werden.

Zusätzlich liefert der FB die zuletzt erzeugte EMCY-Message am Ausgang GET_EMERGENCY.

Netzwerk starten

1863

Hier lesen Sie über das Starten des CANopen-Netzwerks.

Nach einem Download des Projekts auf die Steuerung oder einem Reset der Applikation 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" (81_{16}), jeweils mit dem Node-ID des Slaves. Wurde mit *CANx_MASTER_STATUS* (→ Seite 169) das Flag GLOBAL_START gesetzt, wird zum Hochfahren des Netzes das Kommando einmal mit Node-ID 0 benutzt.
- Alle Slaves werden konfiguriert. Dazu wird zunächst das Objekt 1000_{16} des Slaves abgefragt.
  - Wenn der Slave innerhalb der Überwachungszeit von 0,5 Sekunden antwortet, wird das jeweils nächste Konfigurations-SDO gesendet.
  - Ist ein Slave als [optional] markiert und antwortet nicht innerhalb der Überwachungszeit auf die Abfrage des Objekts 1000_{16}, wird er als nicht vorhanden markiert und keine weiteren SDOs werden an ihn geschickt.
  - Wenn ein Slave auf die Abfrage des Objekts 1000_{16} mit einem anderen Typ als dem konfigurierten (in den unteren 16 Bit) antwortet, wird er zwar konfiguriert, aber als falscher Typ markiert.
- Alle SDOs werden jeweils solange wiederholt, bis innerhalb einer Überwachungszeit eine Antwort des Slaves gesehen wurde. Hier kann die Applikation den Hochlauf der einzelnen Slaves überwachen und ggf. durch Setzen des Flags SET_TIMEOUT_STATE im NODE_STATE_SLAVE-Array des FB CANx_MASTER_STATUS reagieren.
- Wenn der Master eine Heartbeat-Zeit ungleich 0 konfiguriert hat, beginnt die Erzeugung des Hearbeats sofort nach dem Starten der Mastersteuerung.
- Nachdem alle Slaves ihre Konfigurations-SDOs erhalten haben, beginnt für Slaves mit konfiguriertem Nodeguarding das Guarding.
- Wenn der Master auf [automatisch starten] konfiguriert wurde, werden jetzt alle Slaves einzeln vom Master gestartet. Dazu wird das NMT-Kommando "Start Remote Node" (01_{16}) benutzt. Wurde mittels CANx_MASTER_STATUS das Flag GLOBAL_START gesetzt, dann wird das Kommando mit Node-ID 0 genutzt und somit alle Slaves 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 wurde, müssen die Slaves einzeln über das Flag START_NODE im NODE_STATE_SLAVE-Array oder über den Eingang GLOBAL_START von CANx_MASTER_STATUS gestartet werden.

Netzwerkzustände

Inhalt

1864

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

Beim *Netzwerk starten* (→ Seite 138) des CANopen Netzwerks und während des Betriebs durchlaufen die einzelnen Funktionsblöcke 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 *CANx_MASTER_STATUS* (→ Seite 169) aus der Bibliothek `ifm_CRnnnn_CANopenMaster_Vxxyyzz.LIB` zur Verfügung.

Hochlauf des CANopen-Masters

1971

Während des Hochlaufs des CAN-Netzwerks durchläuft der Master verschiedene Zustände, die Sie über den Ausgang NODE_STATE des FB *CANx_MASTER_STATUS* (→ Seite 169) ablesen können. (Netzwerk-Status des Masters → nächstes Kapitel)

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 1000_{16} 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.

NMT-Status für CANopen-Master

9964

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 1000_{16} 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: - Der Master sendet an den Slave ein SDO-Read-Request (Index 1000_{16}). - Die generierten SDOs werden in ein SDO-Array gepackt. - Der Slave kennt seine erste SDO und die Anzahl seiner SDOs.
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 Funktionsblock	hier steht dieser Knoten-Status
CANx_MASTER_STATUS CANx_SLAVE_STATUS	Ausgang NODE_STATE
CANOPEN_GETSTATE	Ausgang NODESTATE

Hochlauf der CANopen-Slaves

1972

Die Status eines Slaves können Sie über das Array NODE_STATE_SLAVE des FB *CANx_MASTER_STATUS* (→ Seite 169) auslesen.
(Netzwerk-Status der Slaves → nächstes Kapitel)

NMT-Status für CANopen-Slave

9965

Status hex | dez		Beschreibung
FF	-1	Der Slave wird durch die NMT-Nachricht [Reset Node] zurückgesetzt und wechselt selbständig in den Status 1.
00	0	nicht definiert
01	1	Status = Warten auf BOOTUP Der Slave wechselt nach einer maximalen Zeit von 2 s oder sofort nach Empfang seiner Bootup-Message in den Status 2.
02	2	Status = BOOTUP Der Slave wechselt nach einer Verzögerungszeit von 0,5 s automatisch in den Status 3.
03	3	Status = PREPARED Im Status 3 wird der Slave konfiguriert. Der Slave bleibt solange im Status 3, bis er alle vom Konfigurator erzeugten SDOs erhalten hat. Dabei spielt es keine Rolle, ob während der Konfiguration vom Slave SDO-Transfers mit Abort (Fehler) oder ob alle fehlerfrei beantwortet wurden. Nur die vom Slave erhaltene Antwort als solche ist wichtig – nicht ihr Inhalt. Wenn im Konfigurator die Option [Knoten zurücksetzen] aktiviert wurde, wird nach dem Senden des Objekts 1011_{16} Subindex 1, der dann den Wert "load" enthält, ein erneuter Reset des Slaves durchgeführt. Der Slave wird dann wieder mit dem Upload des Objekts 1000_{16} angefragt. Slaves, bei denen während der Konfigurationsphase ein Problem auftritt, bleiben im Status 3 oder wechseln nach der Konfigurationsphase direkt in einen Fehlerstatus (Status > 5).
04	4	Status = PRE-OPERATIONAL Ein Knoten wechselt immer in den Status 4, außer: • es handelt sich um einen "optionalen" Slave und er wurde als nicht am Bus verfügbar detektiert (Abfrage Objekt 1000_{16}) ODER: • der Slave ist zwar vorhanden, aber hat auf die Abfrage des Objekts 1000_{16} mit einem anderen Typ in den unteren 16 Bits reagiert, als der Konfigurator erwartet hat.
05	5	Status = OPERATIONAL Im Status 5 findet der normale Datenaustausch statt: "Normal Operation". Wenn der Master auf [Automatisch starten] konfiguriert wurde, wird der Slave im Status 4 gestartet (d.h. es wird eine "Start Node"-NMT-Nachricht erzeugt) und der Slave wechselt automatisch nach Status 5. Wurde GLOBAL_START gesetzt, dann wird gewartet, bis sich alle Slaves im Status 4 befinden. Anschließend werden alle Slaves mit dem NMT-Kommando [Start All Nodes] gestartet.
61	97	Ein Knoten wechselt in den Status 97, wenn er optional ist (optionales Gerät in der CAN-Konfiguration) und nicht auf die SDO-Anfrage nach dem Objekt 1000_{16} reagiert hat. Wird der Slave zu einem späteren Zeitpunkt an das Netzwerk angeschlossen und erkannt, wird er automatisch gestartet. Dazu müssen Sie aber die Option [Automatisch starten] in den CAN-Parametern des Masters angewählt haben.
62	98	Ein Knoten wechselt in den Status 98, wenn der Gerätetyp (Objekt 1000_{16}) nicht dem konfigurierten Typ entspricht.
63	99	Im Falle eines Nodeguarding-Timeouts wird der Slave auf Status 99 gesetzt. Sobald der Slave wieder auf NodeGuard-Anfragen reagiert und die Option [Automatisch starten] eingeschaltet ist, wird er automatisch vom Master gestartet. Dabei wird der Knoten abhängig von seinem Status, der in der Antwort auf die Nodeguard-Anfragen enthalten ist, neu konfiguriert oder nur gestartet. Um den Slave manuell zu starten, genügt es, die Methode [NodeStart] zu benutzen.

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 Funktionsblock	hier steht dieser Knoten-Status
CANx_MASTER_STATUS CANx_SLAVE_STATUS	Ausgang NODE_STATE
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	Knoten hat die BOOTUP-Nachricht erhalten.
04	4	PREPARED	Knoten wird per SDOs konfiguriert.
05	5	OPERATIONAL	Knoten nimmt am normalen Datenaustausch teil.
7F	127	PRE-OPERATIONAL	Knoten sendet keine Daten, ist aber vom Master konfigurierbar.

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

Knoten-Status aus FB lesen:

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

Hochlauf des Netzwerks ohne [Automatisch starten]

Inhalt

8583

Manchmal ist es notwendig, dass die Applikation 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 die Applikation für das Starten der Slaves zuständig.

Starten des Netzwerks mit GLOBAL_START

1974

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.

► Dazu von *CANx_MASTER_STATUS* (→ Seite 169) den Eingang GLOBAL_START auf TRUE setzen (mit [Automatisch starten]).

> 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

1975

Wird das Netzwerk nicht automatisch mit GLOBAL_START des FB *CANx_MASTER_STATUS* (→ Seite 169) gestartet, kann es jederzeit gestartet werden, d.h. jeder Knoten einzeln nacheinander. Ist das nicht gewünscht, besteht folgende Möglichkeit:

► Von CANx_MASTER_STATUS den Eingang START_ALL_NODES auf TRUE setzen. START_ALL_NODES wird typisch zur Laufzeit durch das Applikations-Programm 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

1976

Aus den selben Gründen, die für den Befehl START_ALL_NODES sprechen, gibt es Fälle, in denen Sie besser das NMT-Kommando RESET_ALL_NODES (anstelle RESET_NODES für jeden einzelnen Knoten) einsetzen.

► Dazu müssen Sie von *CANx_MASTER_STATUS* (→ Seite 169) den Eingang RESET_ALL_NODES auf TRUE setzen.

> Dadurch werden einmalig alle Knoten gleichzeitig zurückgesetzt.

Zugriff auf den Status des CANopen-Masters

1977

Damit der Applikations-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:

Variablendeklaration

```
VAR
    FB_MasterStatus := CR0020_MASTER_STATUS;
    :
END_VAR
```

Programmcode

```
IF   FB_MasterStatus.NODE_STATE = 5 THEN
     <Applikationscode>
END_IF
```

Durch Setzen des Flags TIME_OUT_STATE im Array NODE_STATE_SLAVE des FB *CANx_MASTER_STATUS* (→ Seite 169) kann die Applikation reagieren und zum Beispiel den nicht konfigurierbaren Knoten überspringen.

Das Objektverzeichnis des CANopen-Masters

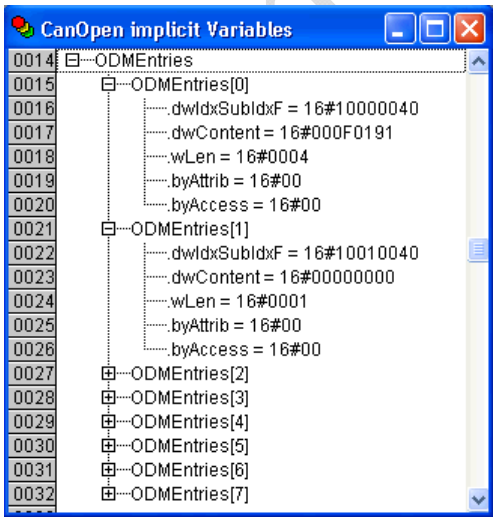
1978

In manchen Fällen ist es hilfreich, wenn der CANopen-Master über ein eigenes Objektverzeichnis verfügt. Das ermöglicht z.B. den Datenaustausch der Applikation 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.

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



Strukturelement	Beschreibung
.dwIdxSubIdxF	Die Struktur der Komponente iiiissff$_{16}$ 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 von der Applikation des Masters beliebig genutzt werden.
.byAccess	Früher Zugriffsberechtigung. Kann von der Applikation 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 41$_{16}$ 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 der Applikation über das Flag OD_CHANGED in *CANx_MASTER_STATUS* (→ Seite 169) angezeigt. Nach der Auswertung müssen Sie dieses Flag wieder zurücksetzen.

Die Applikation 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.

CANopen-Slave

Inhalt

1865

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

Funktionalität der CANopen-Slave-Bibliothek

1979

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 Änderung des PDO-Mappings erfolgt während der Konfigurationsphase durch den Master. Durch das Mapping können IEC-Variablen der Applikation in PDOs gemappt werden. D.h. den PDOs werden IEC-Variable zugeordnet, um sie im Applikations-Programm 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.
- Die Bibliothek verwaltet die Zugriffe auf das Objektverzeichnis, tritt also am Bus als SDO-Server auf.
- Die Bibliothek überwacht das Nodeguarding und die Heartbeat-Consumer-Zeit (immer nur von einem Producer) und setzt entsprechende Fehlerflags für die Applikation.
- 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.

Die CANopen-Slave-Bibliothek 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 mittels *CANx_SLAVE_EMCY_HANDLER* (→ Seite 176) und *CANx_SLAVE_SEND_EMERGENCY* (→ Seite 178) von der Applikation erzeugt werden. Die Bibliothek `ifm_CRnnnn_CANopenSlave_Vxxyyzz.LIB` stellt Ihnen dazu diese FBs zur Verfügung.
- 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

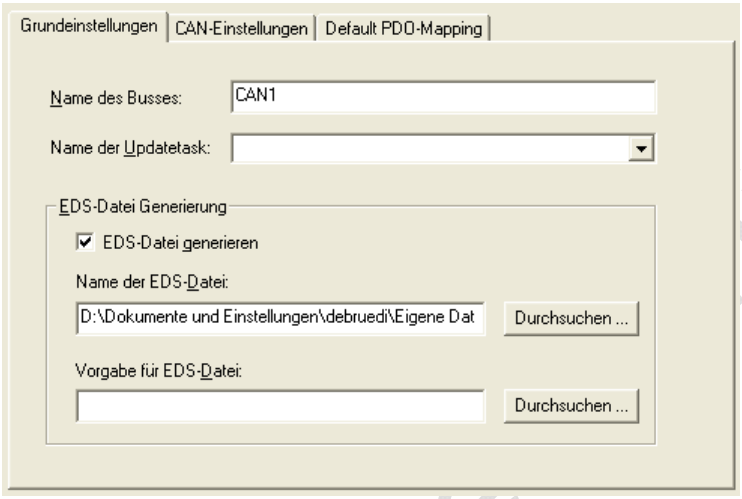
Inhalt

1980

Um die Steuerung als CANopen-Slave zu nutzen, muss zunächst in der Steuerungskonfiguration über [Einfügen] > [Unterelement anhängen] der CANopen-Slave eingefügt werden. Bei Steuerungen mit 2 oder mehr CAN-Schnittstellen wird automatisch CAN-Schnittstelle 1 als Slave konfiguriert. Alle notwendigen 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

Soll aus den Einstellungen hier eine EDS-Datei erzeugt werden, um den CANopen-Slave in eine beliebigen Masterkonfiguration einfügen zu können, muss hier die Option [EDS-Datei generieren] aktiviert werden und der Name einer Datei angegeben werden. Optional kann auch noch eine Vorlagendatei angeben werden, deren Einträge zum EDS-File 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
```

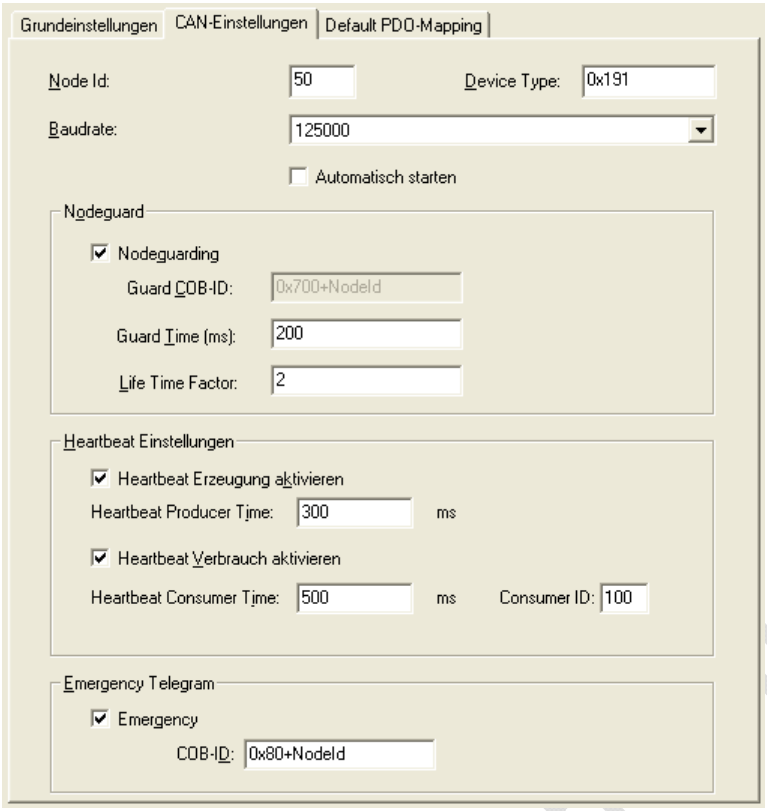
Bedeutung der einzelnen Objekte entnehmen Sie bitte der CANopen-Spezifikation DS301.

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 $16xx_{16}$ oder $1Axx_{16}$ vorhanden, aber u.U. im Subindex 0 nicht berücksichtigt. **HINWEIS:** 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 1000_{16}, der im EDS eingetragen wird) wird mit 191_{16} (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

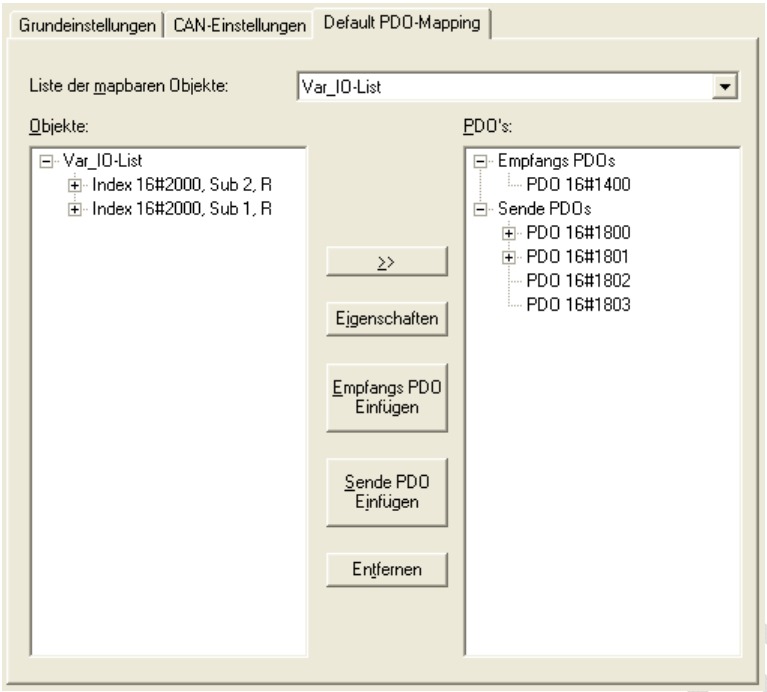
Beim Verwenden von Guarding oder Heartbeat UND
wenn Sie ein EDS-File erzeugen, das bei einem CANopen-Master eingebunden werden soll:

► 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



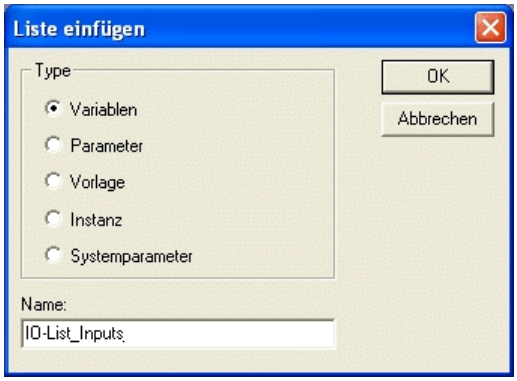
In diesem Register können Sie die Zuordnung zwischen lokalem Objektverzeichnis (OD-Editor) und den PDOs festlegen, die vom CANopen-Slave gesendet/empfangen werden. Eine solche Zuordnung wird als "Mapping" bezeichnet.

In den verwendeten Objektverzeichniseinträgen (Variablen OD) wird zwischen Objektindex/Subindex die Verbindung zu Variablen der Applikation 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 Applikations-Variablen verknüpfen wollen. Für die Parameterliste ist zurzeit nur der Index 1006_{16} / 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.

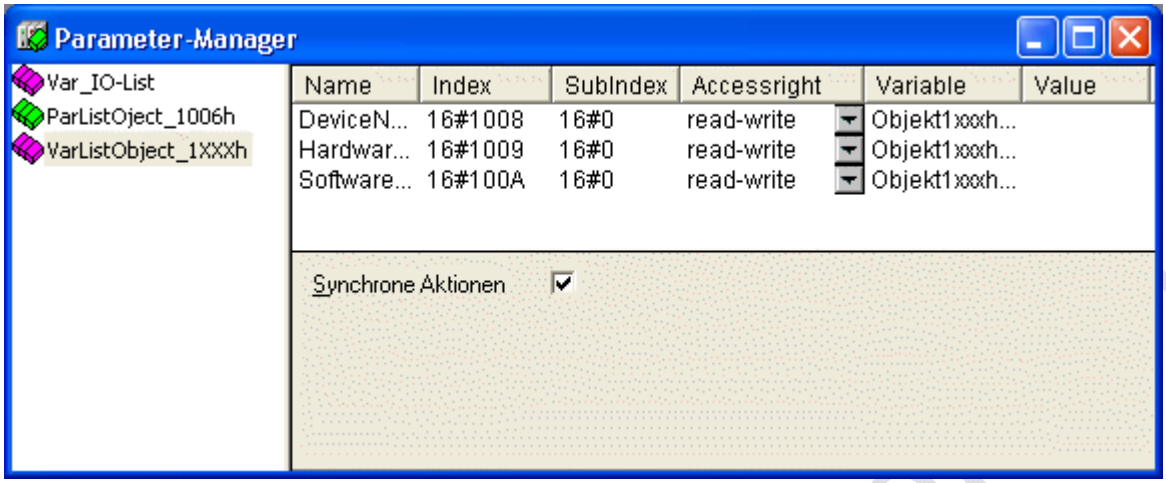
HINWEIS

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-Applikation 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* (→ Seite 131) 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.

HINWEIS

Werden mehr als 8 Datenbytes in ein PDO gemappt, werden automatisch die nächsten freien Identifier dafür genutzt, bis alle Datenbytes übertragen werden können.

Um eine klare Struktur der verwendeten Identifier zu erhalten, sollten Sie die richtige Zahl der Empfangs- und Sende-PDOs einfügen und diesen die Variablen-Bytes aus der Liste zuordnen.

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 die Regel, dass der CANopen-Slave nicht in der Lage ist, Objektverzeichniseinträge neu anzulegen, die nicht 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 von 1400_{16} (Empfangs-PDO-Kommunikationsparameter) oder 1800_{16} (Sende-PDO-Kommunikationsparameter) beginnend angelegt sein und lückenlos aufeinander folgen.

Zugriff auf den CANopen-Slave zur Laufzeit

1985

Einstellen der Knotennummer und der Baud-Rate eines CANopen-Slaves

1986

Beim CANopen-Slave kann zur Laufzeit des Applikations-Programms die Knotennummer und die Baudrate eingestellt werden.

- Zum Einstellen der **Knotennummer** wird *CANx_SLAVE_NODEID* (→ Seite 175) der Bibliothek `ifm_CRnnnn_CANopenSlave_Vxxyyzz.lib` genutzt.
- Zum Einstellen der **Baud-Rate** wird bei den Controllern und beim PDM360smart *CAN1_BAUDRATE* (→ Seite 82) oder *CAN1_EXT* (→ Seite 95) oder *CANx* der jeweiligen Gerätebibliothek benutzt. Beim PDM360 oder PDM360compact steht hierfür *CANx_SLAVE_BAUDRATE* über die Bibliothek `ifm_CRnnnn_CANopenSlave_Vxxyyzz.lib` zur Verfügung.

Zugriff auf die OD-Einträge vom Applikations-Programm

1987

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 mittels *CANx_SLAVE_STATUS* (→ Seite 181) in der Bibliothek `ifm_CRnnnn_CANopenSlave_Vxxyyzz.LIB` zur Verfügung.

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 Applikations-Programm senden

1989

Um eine Emergency-Message durch das Applikations-Programm zu versenden, können Sie *CANx_SLAVE_EMCY_HANDLER* (→ Seite 176) und *CANx_SLAVE_SEND_EMERGENCY* (→ Seite 178) einsetzen. Die Bibliothek `ifm_CRnnnn_CANopenSlave_Vxxyyzz.LIB` stellt ihnen dazu diese FBs zur Verfügung.

CANopen-Netzwerkvariablen

Inhalt

1868

Allgemeine Informationen

2076

Netzwerkvariablen

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 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. Diese Dienste sind vom Protokoll her nicht bestätigte Dienste, d.h. es gibt keine Kontrolle, ob die Nachricht auch beim Empfänger ankommt. Netzwerkvariablen-Austausch entspricht einer "1-zu-n-Verbindung" (1 Sender zu n Empfängern).

Objektverzeichnis

Das 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 Funktionsblöcken aus dem Applikations-Programm.
→ Kapitel *Das Objektverzeichnis des CANopen-Masters* (→ Seite 144)

CANopen-Netzwerkvariablen konfigurieren

Inhalt

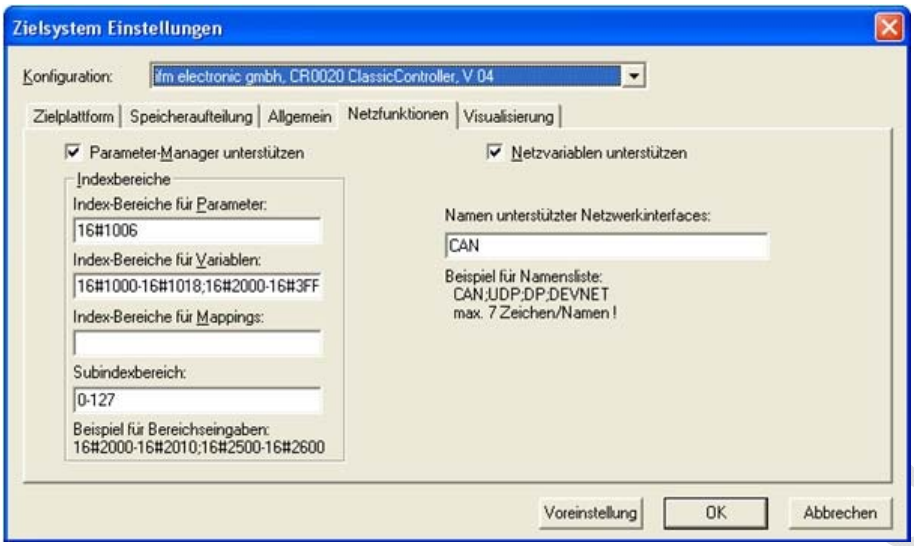
1869

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 -ende.

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* (→ Seite 161)

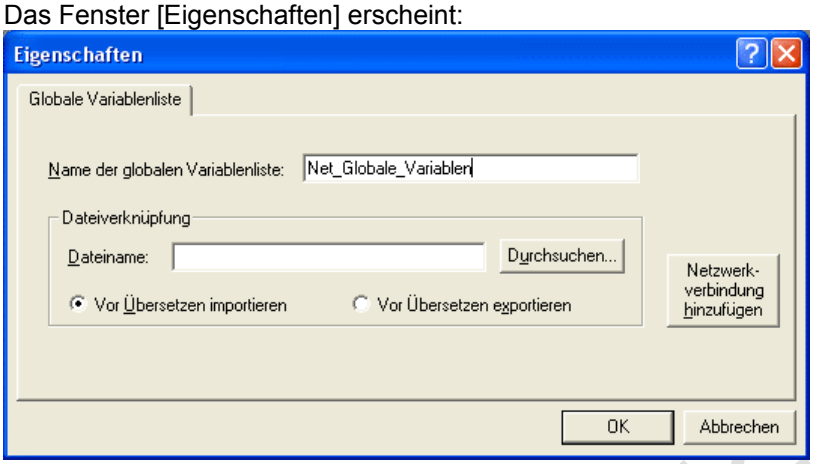
Einstellungen in den globalen Variablenlisten

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...].

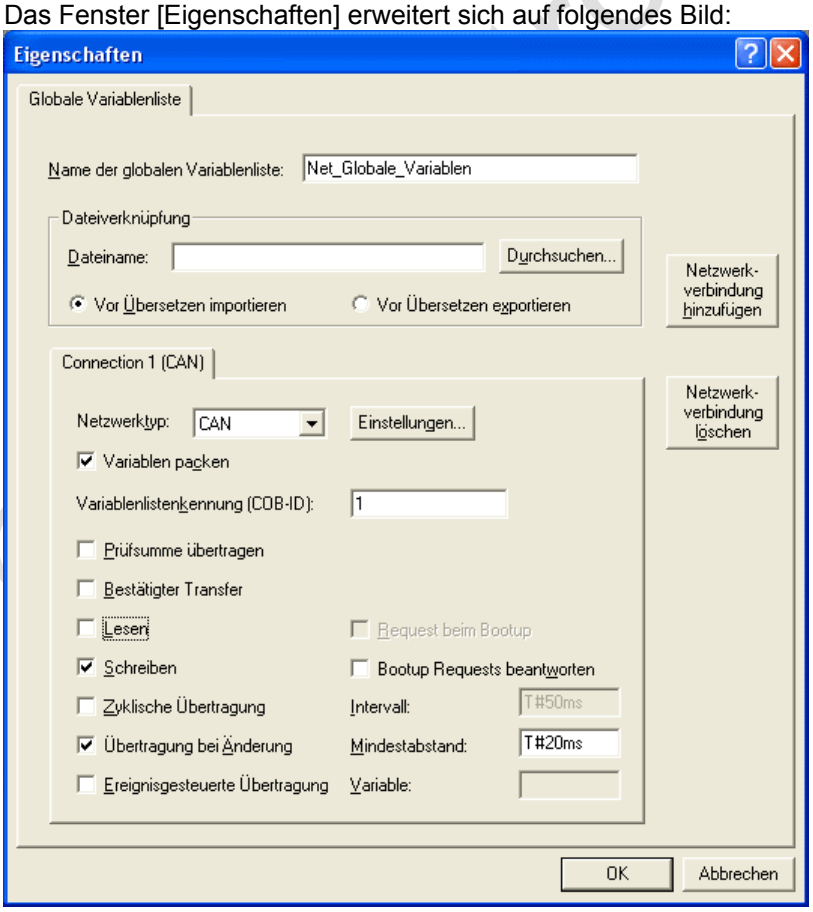
> Das Fenster [Eigenschaften] erscheint:



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

Wenn diese Option mit [**v**] aktiviert ist, werden die Variablen 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.

Wenn [Übertragung bei Änderung] konfiguriert ist, wird für jede Übertragungseinheit getrennt geprüft, ob sie geändert ist und gesendet werden muss.

Globale Variablenliste: Variablenlistenkennung (COB-ID)

10057

Der Basis-Identifier wird als eindeutige Kennung benutzt, um Variablenlisten verschiedener Projekte auszutauschen. Variablenlisten mit gleichem Basis-Identifier werden ausgetauscht. Es ist darauf zu achten, dass die Definitionen der Variablenlisten mit gleichem Basis-Identifier in den verschiedenen Projekten übereinstimmen.

HINWEIS

Der Basis-Identifier wird in CAN-Netzwerken direkt als COB-ID 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.

Es dürfen keine Strings über Netzwerkvariablen verschickt werden, da es sonst zu einem Laufzeitfehler kommt und der Watchdog aktiviert wird.

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, die selbe Liste für alle Teilnehmer zu verwenden.

Globale Variablenliste: Zyklische Übertragung

10062

Nur gültig, wenn [Schreiben] aktiviert. Die Werte werden im angegebenen [Intervall] gesendet, unabhängig davon, ob sie sich geändert haben.

Globale Variablenliste: Übertragung bei Änderung

10063

Die Variablenwerte werden nur gesendet, wenn sich einer der Werte geändert hat. Mit [Mindestabstand] (Wert > 0) kann eine Mindestzeit zwischen den Nachrichtenpaketen festgelegt werden.

Globale Variablenliste: Ereignisgesteuerte Übertragung

10064

Wenn diese Option gewählt ist, wird die CAN-Nachricht 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.

Besonderheiten bei Netzwerkvariablen

1992

Gerät	Beschreibung
ClassicController: CR0020, CR0505 ExtendedController: CR0200 SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506	Netzwerkvariablen werden nur auf CAN-Schnittstelle 1 (Wert = 0 eintragen) unterstützt. **CANopen-Master** Sende- und Empfangslisten werden direkt verarbeitet. Sie brauchen nur die oben beschriebenen Einstellungen vornehmen. **CANopen-Slave** Sendelisten werden direkt verarbeitet. Für Empfangslisten müssen Sie zusätzlich noch den Bereich der Identifier im Objektverzeichnis auf Empfangs-PDOs mappen. Es ist ausreichend, wenn Sie nur zwei Empfangs-PDOs anlegen und dem ersten Objekt den ersten Identifier und dem zweiten Objekt den letzen Identifier zuweisen. Werden die Netzwerkvariablen nur auf einem Identifier übertragen, müssen Sie nur ein Empfangs-PDO mit diesem Identifier anlegen. **Wichtig!** Bitte beachten Sie, dass die Identifier der Netzwerkvariablen und der Empfangs-PDOs als **dezimale** Werte eingegeben werden müssen.
ClassicController: CR0032, CR0033 ExtendedController: CR0232, CR0233	Netzwerkvariablen werden auf allen CAN-Schnittstellen unterstützt. (Alle anderen Angaben wie oben)
BasicController: CR0403	Netzwerkvariablen werden auf allen CAN-Schnittstellen unterstützt. (Alle anderen Angaben wie oben)
BasicDisplay: CR0451	Es steht nur eine CAN-Schnittstelle zur Verfügung (Wert = 0 eintragen). (Alle anderen Angaben wie oben)
PDM360smart: CR1070, CR1071	Es steht nur eine CAN-Schnittstelle zur Verfügung (Wert = 0 eintragen). **CANopen-Master** Sende- und Empfangslisten werden direkt verarbeitet. Sie brauchen nur die oben beschriebenen Einstellungen vornehmen. **CANopen-Slave** Sendelisten werden direkt verarbeitet. Für Empfangslisten müssen Sie zusätzlich noch den Bereich der Identifier im Objektverzeichnis auf Empfangs-PDOs mappen. Es ist ausreichend, wenn Sie nur zwei Empfangs-PDOs anlegen und dem ersten Objekt den ersten Identifier und dem zweiten Objekt den letzen Identifier zuweisen. Werden die Netzwerkvariablen nur auf einem Identifier übertragen, müssen Sie nur ein Empfangs-PDO mit diesem Identifier anlegen. **Wichtig!** Bitte beachten Sie, dass die Identifier der Netzwerkvariablen und der Empfangs-PDOs als **dezimale** Werte eingegeben werden müssen.
PDM360: CR1050, CR1051 PDM360compact: CR1052, CR1053, CR1055, CR1056	Netzwerkvariablen werden auf den CAN-Schnittstellen 1 (Wert = 0) und 2 (Wert = 1) unterstützt. **CANopen-Master** Sende- und Empfangslisten werden direkt verarbeitet. Sie brauchen nur die oben beschriebenen Einstellungen vornehmen. **CANopen-Slave** Sende- und Empfangslisten werden direkt verarbeitet. Sie brauchen nur die oben beschriebenen Einstellungen vornehmen. **Wichtig!** Wird [Netzvariablen unterstützen] im PDM360 oder PDM360compact angewählt, müssen Sie mindestens eine Variable in der Globalen Varablenliste anlegen und diese einmalig im Applikations-Programm aufgerufen. Andernfalls wird die folgende Fehlermeldung bei der Programmübersetzung generiert: Fehler 4601: Netzwerkvariablen 'CAN' : Es ist keine zyklische oder freilaufende Task zum Netwerkvariablenaustausch vorhanden.

Gerät	Beschreibung
PDM360NG: CR108n	Netzwerkvariablen werden auf allen CAN-Schnittstellen unterstützt. (Alle anderen Angaben wie oben)

6.6.2 Bibliotheken für CANopen

Inhalt

8587

ifm-Bibliothek für den CANopen-Master

Inhalt

1870

Für den CANopen-Master stellt die Bibliothek `ifm_CRnnnn_CANopenMaster_Vxxyyzz.LIB` eine Reihe von Bausteinen zur Verfügung, die im Folgenden erklärt werden.

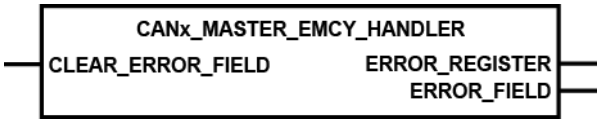
CANx_MASTER_EMCY_HANDLER

2006

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_MASTER_EMCY_HANDLER

9411

Enthalten in Bibliothek: `ifm_CRnnnn_CANopenMaster_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360: CR1050, CR1051
- PDM360compact: CR1052, CR1053, CR1055, CR1056
- PDM360smart: CR1070, CR1071

Beschreibung

2009

CANx_MASTER_EMCY_HANDLER überwacht den geräteeigenen Fehlerstatus des Masters. Der FB muss in folgenden Fällen aufgerufen werden:

- der Fehlerstatus soll ins Netzwerk übertragen werden und
- die Fehlermeldungen der Applikation sollen im Objektverzeichnis gespeichert werden.

HINWEIS

Sollen applikations-spezifische Fehlernachrichten im Objektverzeichnis gespeichert werden, muss CANx_MASTER_EMCY_HANDLER **nach** dem (mehrfachen) Bearbeiten von *CANx_MASTER_SEND_EMERGENCY* (→ Seite 166) aufgerufen werden.

Parameter der Eingänge

2010

Parameter	Datentyp	Beschreibung
CLEAR_ERROR_FIELD	BOOL	TRUE: Löscht den Inhalt des Arrays ERROR_FIELD FALSE: diese Funktion wird nicht ausgeführt

Parameter der Ausgänge

2011

Parameter	Datentyp	Beschreibung
ERROR_REGISTER	BYTE	Zeigt den Inhalt des OBV Index 1001_{16} (Error Register)
ERROR_FIELD	ARRAY[0...5] OF WORD	Das Array[0...5] zeigt den Inhalt des OBV Index 1003_{16} (Error Field). ERROR_FIELD[0]: Anzahl der gespeicherten Fehler ERROR_FIELD[1...5]: gespeicherte Fehler, der jüngste Fehler steht im Index [1]

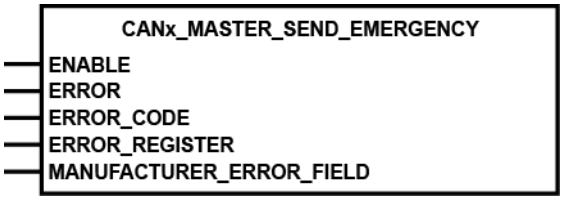
CANx_MASTER_SEND_EMERGENCY

2012

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_MASTER_SEND_EMERGENCY

9430

Enthalten in Bibliothek: `ifm_CRnnnn_CANopenMaster_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360: CR1050, CR1051
- PDM360compact: CR1052, CR1053, CR1055, CR1056
- PDM360smart: CR1070, CR1071

Beschreibung

2015

CANx_MASTER_SEND_EMERGENCY versendet applikations-spezifische Fehlerstatus. Der FB wird aufgerufen, wenn der Fehlerstatus an andere Geräte im Netzwerkverbund übertragen werden soll.

HINWEIS

Sollen applikations-spezifische Fehlernachrichten im Objektverzeichnis gespeichert werden, muss *CANx_MASTER_EMCY_HANDLER* (→ Seite 164) **nach** dem (mehrfachen) Bearbeiten von CANx_MASTER_SEND_EMERGENCY aufgerufen werden.

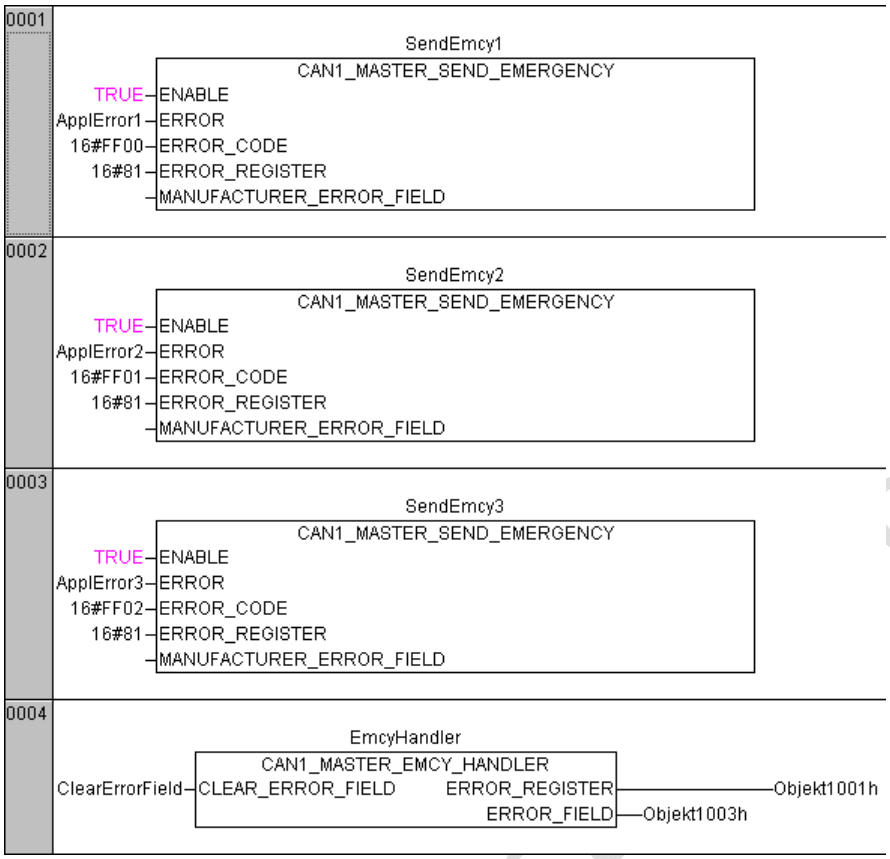
Parameter der Eingänge

2016

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
ERROR	BOOL	FALSE ⇨ TRUE (Flanke): sendet den anstehenden Fehlercode TRUE ⇨ FALSE (Flanke) UND Fehler steht nicht mehr an: Nach Verzögerung von ca. 1 s wird Null-Fehlermeldung gesendet sonst: diese Funktion wird nicht ausgeführt
ERROR_CODE	WORD	Der Error-Code gibt detailliert Auskunft über den erkannten Fehler. Die Werte sollten gemäß der CANopen-Spezifikation eingetragen werden. → Kapitel *Übersicht CANopen Error-Codes* (→ Seite 194)
ERROR_REGISTER	BYTE	Dieses Objekt spiegelt den allgemeinen Fehlerzustand des CANopen-Netzwerkteilnehmers wider. Die Werte sollten gemäß der CANopen-Spezifikation eingetragen werden.
MANUFACTURER_ERROR_FIELD	ARRAY[0...4] OF BYTE	Hier können bis zu 5 Bytes applikations-spezifische Fehlerinformationen eingetragen werden. Das Format ist dabei frei wählbar.

Beispiel: CANx_MASTER_SEND_EMERGENCY

2018



In diesem Beispiel werden nacheinander 3 Fehlermeldungen generiert:

1. ApplError1, Code = $FF00_{16}$ im Fehlerregister 81_{16}
2. ApplError2, Code = $FF01_{16}$ im Fehlerregister 81_{16}
3. ApplError3, Code = $FF02_{16}$ im Fehlerregister 81_{16}

Der FB CAN1_MASTER_EMCY_HANDLER sendet die Fehlermeldungen an das Fehler-Register "Objekt 1001_{16}" im Fehler-Array "Objekt 1003_{16}".

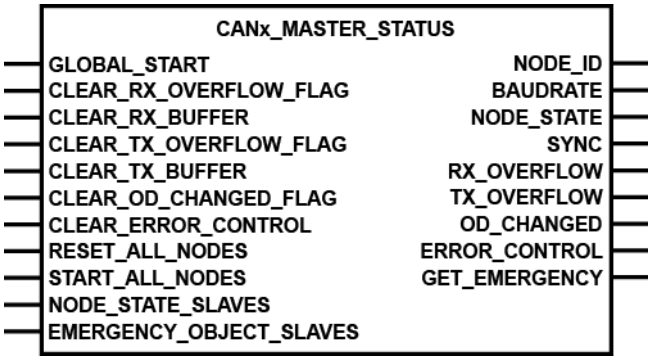
CANx_MASTER_STATUS

9933

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_MASTER_STATUS

9435

Enthalten in Bibliothek:	Für folgende Geräte verfügbar:
`ifm_CRnnnn_CAN1openMaster_Vxxyyzz.LIB`	- PDM360: CR1050, CR1051
`ifm_CRnnnn_CANopenMaster_Vxxyyzz.LIB`	- PDM360compact: CR1052, CR1053, CR1055, CR1056 - PDM360smart: CR1070, CR1071

Beschreibung

2024

Status-Anzeige des als CANopen-Master eingesetzten Gerätes

Der FB zeigt den Status des als CANopen-Master eingesetzten Gerätes an. Außerdem kann der Status des Netzwerks und der angeschlossenen Slaves überwacht werden.

Der FB vereinfacht die Anwendung der CoDeSys-CANopen-Master-Bibliotheken. Wir empfehlen dringend, die Auswertung des Netzwerkstatus und der Fehlermeldungen über diesen FB vorzunehmen.

Parameter der Eingänge

2695

Parameter	Datentyp	Beschreibung
GLOBAL_START	BOOL	TRUE: Alle angeschlossenen Netzwerkteilnehmer (Slaves) werden gleichzeitig bei der Netzwerkinitialisierung gestartet. FALSE: Die angeschlossenen Netzwerkteilnehmer werden einzeln nacheinander gestartet. Weitere Informationen → Kapitel *Starten des Netzwerks mit GLOBAL_START* (→ Seite 143)
CLEAR_RX_OVERFLOW_FLAG	BOOL	FALSE ⇨ TRUE (Flanke): Fehlerflag "Empfangspuffer-Überlauf" löschen FALSE: diese Funktion wird nicht ausgeführt
CLEAR_RX_BUFFER	BOOL	FALSE ⇨ TRUE (Flanke): Daten im Empfangspuffer löschen FALSE: diese Funktion wird nicht ausgeführt
CLEAR_TX_OVERFLOW_FLAG	BOOL	FALSE ⇨ TRUE (Flanke): Fehlerflag "Sendepuffer-Überlauf" löschen FALSE: diese Funktion wird nicht ausgeführt
CLEAR_TX_BUFFER	BOOL	FALSE ⇨ TRUE (Flanke): Daten im Sendepuffer löschen FALSE: diese Funktion wird nicht ausgeführt
CLEAR_OD_CHANGED_FLAG	BOOL	FALSE ⇨ TRUE (Flanke): Flag "Daten im Objektverzeichnis geändert" löschen FALSE: diese Funktion wird nicht ausgeführt
CLEAR_ERROR_CONTROL	BOOL	FALSE ⇨ TRUE (Flanke): Die Guard-Fehlerliste (ERROR_CONTROL) löschen FALSE: diese Funktion wird nicht ausgeführt
RESET_ALL_NODES	BOOL	FALSE ⇨ TRUE (Flanke): Alle Knoten zurücksetzen FALSE: diese Funktion wird nicht ausgeführt
START_ALL_NODES	BOOL	TRUE: Alle angeschlossenen Netzwerkteilnehmer (Slaves) werden gleichzeitig zur Laufzeit des Applikations-Programms gestartet FALSE: Die angeschlossenen Netzwerkteilnehmer müssen einzeln nacheinander gestartet werden Weitere Informationen → Kapitel *Starten des Netzwerks mit START_ALL_NODES* (→ Seite 143)
NODE_STATE_SLAVES	DWORD	Zeigt den Status aller Netzwerkknoten. Beispiel-Code → Kapitel *Beispiel: CANx_MASTER_STATUS* (→ Seite 172) Weitere Informationen → Kapitel *Der Master zur Laufzeit* (→ Seite 136)
EMERGENCY_OBJECT_SLAVES	DWORD	Zeigt die zuletzt aufgetretenen Fehlermeldungen aller Netzwerkknoten. Weitere Informationen → Kapitel *Zugriff auf die Strukturen zur Laufzeit der Applikation* (→ Seite 173)

Parameter der Ausgänge

9935

Parameter	Datentyp	Beschreibung
NODE_ID	BYTE	Node-ID des Masters
BAUDRATE	WORD	Baudrate des Masters
NODE_STATE	INT	aktueller Status des Masters.
SYNC	BOOL	SYNC-Signal des Masters. Dieses wird in Abhängigkeit der eingestellten Zeit Com. Cycle Period im *CANopen-Master: Register [CAN-Parameter]* (→ Seite 128) des Masters eingestellt.
RX_OVERFLOW	BOOL	Fehlerflag "Empfangspuffer-Überlauf"
TX_OVERFLOW	BOOL	Fehlerflag "Sendepuffer-Überlauf"
OD_CHANGED	BOOL	Flag "Objektverzeichnis Master wurde geändert"
ERROR_CONTROL	ARRAY [0..7] OF BYTE	Das Array enthält die Liste (max. 8) der fehlenden Netzwerkknoten (Guard- oder Heartbeat-Fehler) Weitere Informationen → Kapitel *Zugriff auf die Strukturen zur Laufzeit der Applikation* (→ Seite 173)
GET_EMERGENCY	STRUCT EMERGENY_MESSAGE	Am Ausgang stehen die Daten für die Struktur EMERGENY_MESSAGE zur Verfügung. Es wird immer die letzte Fehlermeldung eines Netzwerkknotens angezeigt. Um eine Liste aller aufgetretenen Fehler zu erhalten, muss das Array "EMERGENCY_OBJECT_SLAVES" ausgewertet werden.

Parameter der internen Strukturen

2698

Hier sehen Sie die Strukturen der in diesem Baustein genutzten Arrays.

Parameter	Datentyp	Beschreibung
CANx_EMERGENY_MESSAGE	STRUCT	NODE_ID: BYTE ERROR_CODE: WORD ERROR_REGISTER: BYTE MANUFACTURER_ERROR_FIELD: ARRAY[0..4] OF BYTE Die Struktur ist in den globalen Variablen der Bibliothek `ifm_CRnnnn_CANopenMaster_Vxxyyzz.LIB` angelegt.
CANx_NODE_STATE	STRUCT	NODE_ID: BYTE NODE_STATE: BYTE LAST_STATE: BYTE RESET_NODE: BOOL START_NODE: BOOL PREOP_NODE: BOOL SET_TIMEOUT_STATE: BOOL SET_NODE_STATE: BOOL Die Struktur ist in den globalen Variablen der Bibliothek `ifm_CRnnnn_CANopenMaster_Vxxyyzz.LIB` angelegt.

Ausführliche Beschreibung der Funktionalitäten des CANopen-Masters und der Mechanismen → Kapitel *CANopen-Master* (→ Seite 125).

Die folgenden Code-Fragmente zeigen Ihnen am Beispiel des Controllers CR0032 die Anwendung des FB *CANx_MASTER_STATUS* (→ Seite 169).

Beispiel: CANx_MASTER_STATUS

2031

Slave-Informationen

2699

Damit Sie auf die Informationen der einzelnen CANopen-Knoten zugreifen können, müssen Sie ein Array über die jeweilige Struktur bilden. Die Strukturen sind in der Bibliothek enthalten. Sie können Sie im Bibliotheksverwalter unter [Datentypen] sehen.

Die Anzahl der Array-Elemente wird bestimmt durch die Globale Variable MAX_NODEINDEX, die automatisch vom CANopen-Stack angelegt wird. Sie enthält die Anzahl der im Netzwerkkonfigurator angegebenen Slaves minus 1.

HINWEIS

Die Nummern der Array-Elemente entsprechen **nicht** dem Node-ID. Der Identifier kann aus der jeweiligen Struktur unter NODE_ID ausgelesen werden.

```
PROGRAM MasterStatus
VAR
    Status: CR0032_MASTER_STATUS;
    StartAllNodes: BOOL:= TRUE;
    ClearRxOverflowFlag: BOOL;
    ClearRxBuffer: BOOL;
    ClearTxOverflowFlag: BOOL;
    ClearTxBuffer: BOOL;
    ClearOdChanged: BOOL;
    ClearErrorControl: BOOL;
    ResetAllNodes: BOOL;
    ResetSingleNodeArray: ARRAY[0..MAX_NODEINDEX] OF RESET_NODE;
    NodeStateSlavesArray: ARRAY [0..MAX_NODEINDEX] OF NODE_STATE;
    EmergencyObjectSlavesArray: ARRAY[0..MAX_NODEINDEX] OF EMERGENCY_MESSAGE;
    node_id: BYTE;
    baudrate: WORD;
    node_state: INT;
    Sync: BOOL;
    RxOverflow: BOOL;
    TxOverflow: BOOL;
    OdChanged: BOOL;
    GuardHeartbeatErrorArray: ARRAY[0..7] OF BYTE;
    GetEmergency: EMERGENCY_MESSAGE;
END_VAR
```

Struktur Knoten-Status

2034

```
TYPE CAN1_NODE_STATE :
STRUCT
    NODE_ID: BYTE;
    NODE_STATE: BYTE;
    LAST_STATE: BYTE;
    RESET_NODE: BOOL;
    START_NODE: BOOL;
    PREOP_NODE: BOOL;
    SET_TIMEOUT_STATE: BOOL;
    SET_NODE_STATE: BOOL;
END_STRUCT
END_TYPE
```

Struktur Emergency_Message

2035

```
TYPE CAN1_EMERGENCY_MESSAGE :
STRUCT
    NODE_ID: BYTE;
    ERROR_CODE: WORD;
    ERROR_REGISTER: BYTE;
    MANUFACTURER_ERROR_FIELD: ARRAY[0..4] OF BYTE;
END_STRUCT
END_TYPE
```

Zugriff auf die Strukturen zur Laufzeit der Applikation

2036

Zur Laufzeit können Sie auf das jeweilige Array-Element über die globalen Variablen der Bibliothek zugreifen und so den Status oder die EMCY-Nachrichten auslesen oder den Knoten zurücksetzen.

```
0001 NodeStateList
0002     NodeStateList[0]
0003         .NODE_ID = 16#02
0004         .NODE_STATE = 16#04
0005         .LAST_STATE = 16#00
0006         .RESET_NODE = FALSE <= TRUE>
0007         .START_NODE = FALSE
0008         .PREOP_NODE = FALSE
0009         .SET_TIMEOUT_STATE = FALSE
0010         .SET_NODE_STATE = FALSE
0011     NodeStateList[1]
0012         .NODE_ID = 16#03
0013         .NODE_STATE = 16#03
0014         .LAST_STATE = 16#00
0015         .RESET_NODE = FALSE
0016         .START_NODE = FALSE
0017         .PREOP_NODE = FALSE
0018         .SET_TIMEOUT_STATE = FALSE
0019         .SET_NODE_STATE = FALSE
0020 NodeEmergencyList
0021     NodeEmergencyList[0]
0022         .NODE_ID = 16#02
0023         .ERROR_CODE = 16#0000
0024         .ERROR_REGISTER = 16#00
0025         .MANUFACTURER_ERROR_FIELD
0026             .MANUFACTURER_ERROR_FIELD[0] = 16#00
0027             .MANUFACTURER_ERROR_FIELD[1] = 16#00
0028             .MANUFACTURER_ERROR_FIELD[2] = 16#00
0029             .MANUFACTURER_ERROR_FIELD[3] = 16#00
0030             .MANUFACTURER_ERROR_FIELD[4] = 16#00
0031     NodeEmergencyList[1]
0032         .NODE_ID = 16#03
0033         .ERROR_CODE = 16#0000
0034         .ERROR_REGISTER = 16#00
0035         .MANUFACTURER_ERROR_FIELD
0036             .MANUFACTURER_ERROR_FIELD[0] = 16#00
0037             .MANUFACTURER_ERROR_FIELD[1] = 16#00
0038             .MANUFACTURER_ERROR_FIELD[2] = 16#00
0039             .MANUFACTURER_ERROR_FIELD[3] = 16#00
0040             .MANUFACTURER_ERROR_FIELD[4] = 16#00
```

Setzen Sie im obigen Beispiel ResetSingleNodeArray[0].RESET_NODE kurzzeitig auf TRUE, wird der erste Knoten im Konfigurationsbaum zurückgesetzt.

Weitere Informationen zu den möglichen Fehler-Codes → Kapitel *CAN-Fehler und Fehlerbehandlung* (→ Seite 189).

ifm-Bibliothek für den CANopen-Slave

Inhalt

1874

Für den CANopen-Slave stellt die Bibliothek `ifm_CRnnnn_CANopenSlave_Vxxyyzz.LIB` eine Reihe von Bausteinen zur Verfügung, die im Folgenden erklärt werden.

CANx_SLAVE_NODEID

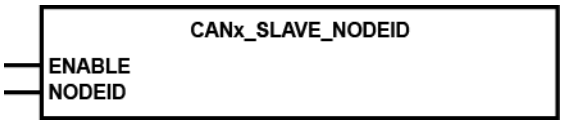
2044

= CANx Slave Node-ID

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_SLAVE_NODEID

9499

Enthalten in Bibliothek: `ifm_CRnnnn_CANopenSlave_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360: CR1050, CR1051
- PDM360compact: CR1052, CR1053, CR1055, CR1056
- PDM360smart: CR1070, CR1071

Beschreibung

2049

CANx_SLAVE_NODEID ermöglicht das Einstellen des Node-ID eines CANopen-Slaves zur Laufzeit des Applikations-Programms.

Der FB wird im Normalfall bei der Initialisierung der Steuerung einmalig, im ersten Zyklus, aufgerufen. Anschließend wird der Eingang ENABLE wieder auf FALSE gesetzt.

Parameter der Eingänge

2047

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	FALSE ⇨ TRUE (Flanke): NodeID setzen FALSE: Baustein wird nicht ausgeführt
NODEID	BYTE	Wert der neuen Knotennummer

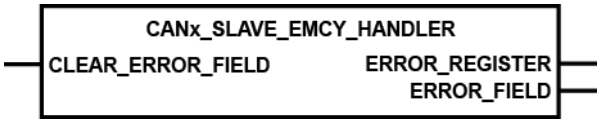
CANx_SLAVE_EMCY_HANDLER

2050

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_SLAVE_EMCY_HANDLER

9493

Enthalten in Bibliothek: `ifm_CRnnnn_CANopenSlave_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360: CR1050, CR1051
- PDM360compact: CR1052, CR1053, CR1055, CR1056
- PDM360smart: CR1070, CR1071

Beschreibung

2053

CANx_SLAVE_EMCY_HANDLER überwacht den geräteeigenen Fehlerstatus (Gerät wird als Slave betrieben).

Der FB muss in folgenden Fällen aufgerufen werden:

- der Fehlerstatus soll ins CAN-Netzwerk übertragen werden und
- die Fehlermeldungen der Applikation sollen im Objektverzeichnis gespeichert werden.

HINWEIS

Sollen applikations-spezifische Fehlernachrichten im Objektverzeichnis gespeichert werden, muss CANx_SLAVE_EMCY_HANDLER **nach** dem (mehrfachen) Bearbeiten von *CANx_SLAVE_SEND_EMERGENCY* (→ Seite 178) aufgerufen werden.

Parameter der Eingänge

2054

Parameter	Datentyp	Beschreibung
CLEAR_ERROR_FIELD	BOOL	FALSE ⇨ TRUE (Flanke): ERROR-FIELD löschen FALSE: diese Funktion wird nicht ausgeführt

Parameter der Ausgänge

2055

Parameter	Datentyp	Beschreibung
ERROR_REGISTER	BYTE	Zeigt den Inhalt des OBV Index 1001_{16} (Error Register).
ERROR_FIELD	ARRAY[0...5] OF WORD	Das Array[0...5] zeigt den Inhalt des OBV Index 1003_{16} (Error Field): ERROR_FIELD[0]: Anzahl der gespeicherten Fehler ERROR_FIELD[1...5]: gespeicherte Fehler, der jüngste Fehler steht im Index [1].

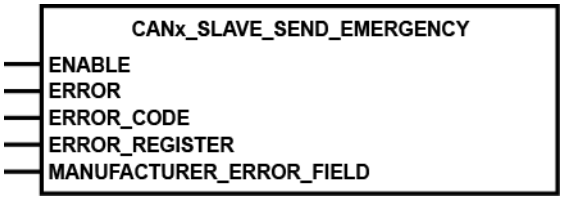
CANx_SLAVE_SEND_EMERGENCY

2056

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_SLAVE_SEND_EMERGENCY

9505

Enthalten in Bibliothek: `ifm_CRnnnn_CANopenSlave_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360: CR1050, CR1051
- PDM360compact: CR1052, CR1053, CR1055, CR1056
- PDM360smart: CR1070, CR1071

Beschreibung

2059

Mit CANx_SLAVE_SEND_EMERGENCY werden applikations-spezifische Fehlerstatus versendet. Das sind Fehlernachrichten, die zusätzlich zu den geräteinternen Fehlernachrichten (z.B. Kurzschluss am Ausgang) gesendet werden sollen.

Der FB wird aufgerufen, wenn der Fehlerstatus an andere Geräte im Netzwerkverbund übertragen werden soll.

HINWEIS

Sollen applikations-spezifische Fehlermachrichten im Objektverzeichnis gespeichert werden, muss *CANx_SLAVE_EMCY_HANDLER* (→ Seite 176) **nach** dem (mehrfachen) Bearbeiten von CANx_SLAVE_SEND_EMERGENCY aufgerufen werden.

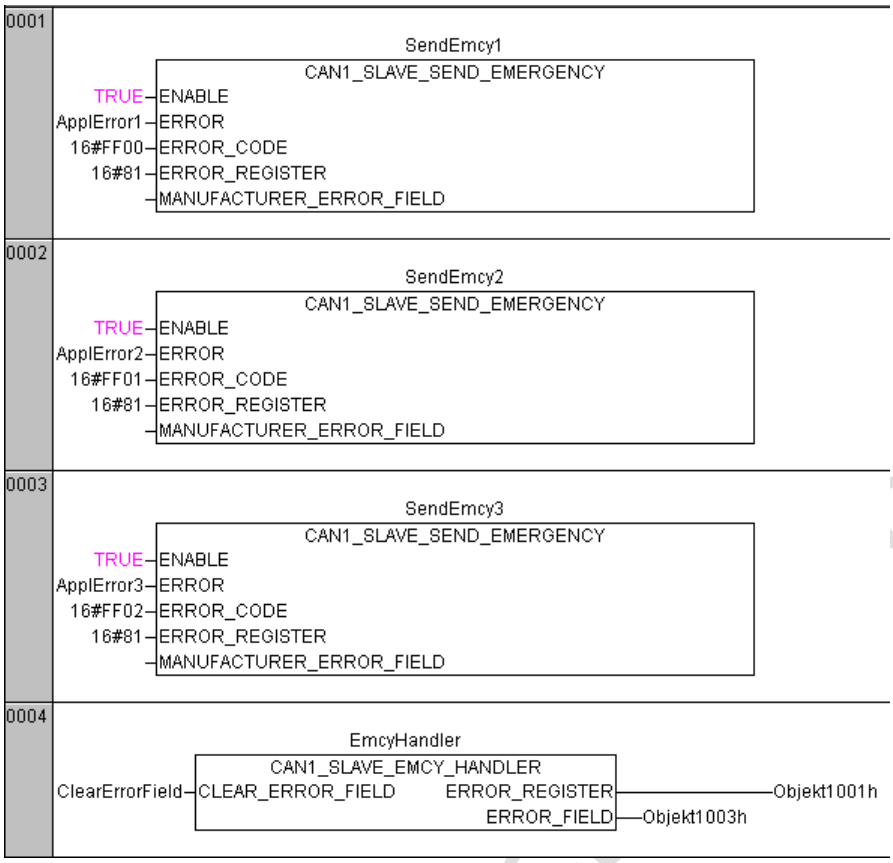
Parameter der Eingänge

2060

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
ERROR	BOOL	FALSE ⇨ TRUE (Flanke): sendet den anstehenden Fehlercode TRUE ⇨ FALSE (Flanke) UND Fehler steht nicht mehr an: Nach Verzögerung von ca. 1 s wird Null-Fehlermeldung gesendet sonst: diese Funktion wird nicht ausgeführt
ERROR_CODE	WORD	Der Error-Code gibt detailliert Auskunft über den erkannten Fehler. Die Werte sollten gemäß der CANopen-Spezifikation eingetragen werden. → Kapitel *Übersicht CANopen Error-Codes* (→ Seite 194)
ERROR_REGISTER	BYTE	Dieses Objekt spiegelt den allgemeinen Fehlerzustand des CANopen-Netzwerkteilnehmers wider. Die Werte sollten gemäß der CANopen-Spezifikation eingetragen werden.
MANUFACTURER_ERROR_FIELD	ARRAY[0...4] OF BYTE	Hier können bis zu 5 Bytes applikations-spezifische Fehlerinformationen eingetragen werden. Das Format ist dabei frei wählbar.

Beispiel: CANx_SLAVE_SEND_EMERGENCY

2062



In diesem Beispiel werden nacheinander 3 Fehlermeldungen generiert:

1. ApplError1, Code = $FF00_{16}$ im Fehlerregister 81_{16}
2. ApplError2, Code = $FF01_{16}$ im Fehlerregister 81_{16}
3. ApplError3, Code = $FF02_{16}$ im Fehlerregister 81_{16}

Der FB CAN1_SLAVE_EMCY_HANDLER sendet die Fehlermeldungen an das Fehler-Register "Objekt 1001_{16}" im Fehler-Array "Objekt 1003_{16}".

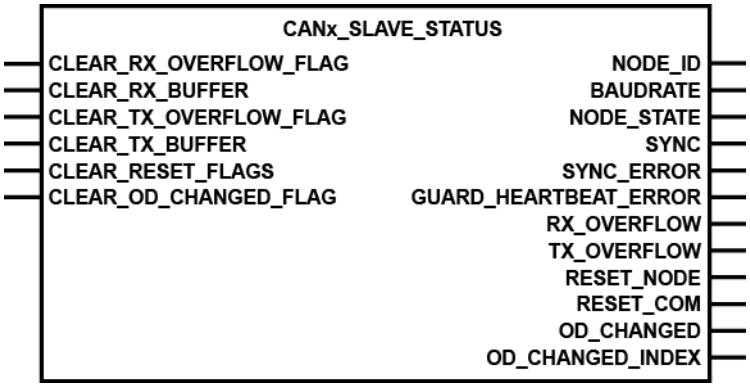
CANx_SLAVE_STATUS

2706

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_SLAVE_STATUS

9510

Enthalten in Bibliothek:	Für folgende Geräte verfügbar:
`ifm_CRnnnn_CANopen1Slave_Vxxyyzz.LIB`	- ClassicController: CR0032, CR0033 - ExtendedController: CR0232, CR0233
`ifm_CRnnnn_CAN1openSlave_Vxxyyzz.LIB`	- PDM360: CR1050, CR1051
`ifm_CRnnnn_CANopenSlave_Vxxyyzz.LIB`	- PDM360compact: CR1052, CR1053, CR1055, CR1056 - PDM360smart: CR1070, CR1071

Beschreibung

2707

CANx_SLAVE_STATUS zeigt den Status des als CANopen-Slave eingesetzten Gerätes an. Der FB vereinfacht die Anwendung der CoDeSys-CANopen-Slave-Bibliotheken. Wir empfehlen dringend, die Auswertung des Netzwerkstatus über diesen FB vorzunehmen.

Info

Eine ausführliche Beschreibung der Funktionalitäten des CANopen-Slaves und der Mechanismen → Kapitel *CANopen-Slave* (→ Seite 146).

Zur Laufzeit können Sie dann auf die einzelnen Ausgänge des Bausteins zugreifen, um eine Statusübersicht zu erhalten.

Beispiel:

```
PROGRAM SlaveStatus
VAR
    Status: CR0032_SLAVE_STATUS;
    ClearRxOverflowFlag: BOOL;
    ClearRxBuffer: BOOL;
    ClearTxOverflowFlag: BOOL;
    ClearTxBuffer: BOOL;
    ClearResetFlag: BOOL;
    ClearOdChangedFlag: BOOL;
    node_id: BYTE;
    baudrate: WORD;
    node_state: BYTE;
    Sync: BOOL;
    SyncError: BOOL;
    GuardHeartbeatError: BOOL;
    RxOverflow: BOOL;
    TxOverflow: BOOL;
    ResetNode: BOOL;
    ResetCom: BOOL;
    OdChanged: BOOL;
    OdChangedIndex: INT;
END_VAR
```

Parameter der Eingänge

2708

Parameter	Datentyp	Beschreibung
CLEAR_RX_OVERFLOW_FLAG	BOOL	FALSE ⇨ TRUE (Flanke): Fehlerflag "Empfangspuffer-Überlauf" löschen FALSE: diese Funktion wird nicht ausgeführt
CLEAR_RX_BUFFER	BOOL	FALSE ⇨ TRUE (Flanke): Daten im Empfangspuffer löschen FALSE: diese Funktion wird nicht ausgeführt
CLEAR_TX_OVERFLOW_FLAG	BOOL	FALSE ⇨ TRUE (Flanke): Fehlerflag "Sendepuffer-Überlauf" löschen FALSE: diese Funktion wird nicht ausgeführt
CLEAR_TX_BUFFER	BOOL	FALSE ⇨ TRUE (Flanke): Daten im Sendepuffer löschen FALSE: diese Funktion wird nicht ausgeführt
CLEAR_RESET_FLAG	BOOL	FALSE ⇨ TRUE (Flanke): Flag "Knoten zurückgesetzt" löschen Flag "Kommunikationsschnittstelle zurückgesetzt" löschen FALSE: diese Funktion wird nicht ausgeführt
CLEAR_OD_CHANGED_FLAG	BOOL	FALSE ⇨ TRUE (Flanke): Flag "Daten im Objektverzeichnis geändert" löschen Flag "Index-Position" löschen FALSE: diese Funktion wird nicht ausgeführt

Parameter der Ausgänge

2068

Parameter	Datentyp	Beschreibung
NODE_ID	BYTE	Node-ID des Slaves
BAUDRATE	WORD	Baudrate des Slaves
NODE_STATE	BYTE	aktueller Status des Slaves
SYNC	BOOL	Empfangenes SYNC-Signal des Masters
SYNC_ERROR	BOOL	Es wurde kein SYNC-Signal des Masters empfangen ODER: die eingestellte SYNC-Zeit (ComCyclePeriod im Master) wurde überschritten.
GUARD_HEARTBEAT_ERROR	BOOL	Es wurde kein Guard- oder Heartbeat-Signal des Masters empfangen ODER: die eingestellten Zeiten wurden überschritten
RX_OVERFLOW	BOOL	Fehlerflag "Empfangspuffer-Überlauf"
TX_OVERFLOW	BOOL	Fehlerflag "Sendepuffer-Überlauf"
RESET_NODE	BOOL	Der CAN-Stack des Slaves wurde vom Master zurückgesetzt. Dieses Flag kann von der Applikation ausgewertet und ggf. für weitere Reaktionen genutzt werden.
RESET_COM	BOOL	Das Kommunikationsinterface des CAN-Stack wurde vom Master zuückgesetzt. Dieses Flag kann von der Applikation ausgewertet und ggf. für weitere Reaktionen genutzt werden.
OD_CHANGED	BOOL	Flag "Objektverzeichnis Master wurde geändert"
OD_CHANGED_INDEX	INT	Ausgang zeigt den geänderten Index des Objektverzeichnisses

Weitere ifm-Bibliotheken zu CANopen

Inhalt

2071

Hier stellen wir Ihnen weitere **ifm**-Bausteine vor, die für CANopen sinnvolle Ergänzungen darstellen.

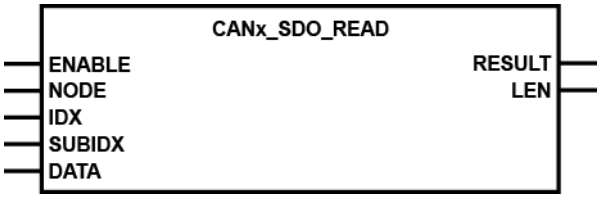
CANx_SDO_READ

621

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_SDO_READ

9442

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Beschreibung

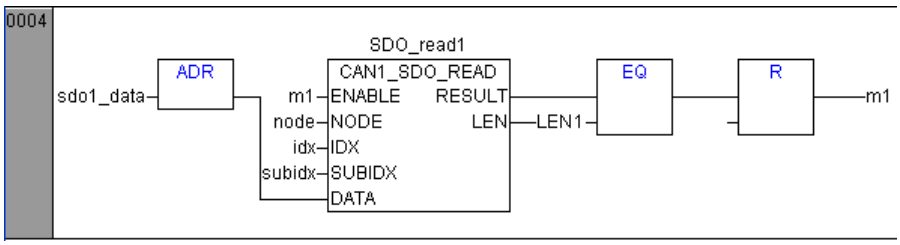
624

CANx_SDO_READ liest das SDO (→ Kapitel *Register [Service Data Objects]* (→ Seite 135)) mit den angegebenen Indizes aus dem Knoten aus.

Über diese können die Einträge im Objektverzeichnis gelesen werden. Dadurch ist es möglich, die Knotenparameter gezielt zu lesen.

- alle ecomatmobile-Controller - Platinensteuerung: CS0015 - PDM360smart: CR1070, CR1071	- PDM360: CR1050, CR1051 - PDM360compact: CR1052, CR1053, CR1055, CR1056
aus Gerätebibliothek `ifm_CRnnnn_Vxxyyzz.LIB`	aus Gerätebibliothek `ifm_CANx_SDO_Vxxyyzz.LIB`
Voraussetzung: Knoten muss sich im Zustand PRE-OPERATIONAL oder OPERATIONAL befinden.	Voraussetzung: Knoten muss sich im Modus "CANopen-Master" oder "CANopen-Slave" befinden.

Beispiel:



Parameter der Eingänge

625

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
NODE	BYTE	Nummer des Knotens
IDX	WORD	Index im Objektverzeichnis
SUBIDX	BYTE	Subindex bezogen auf den Index im Objektverzeichnis
DATA	DWORD	Adresse des Empfangsdaten-Arrays zulässige Länge = 0...255 Übergabe mit ADR-Operator

Parameter der Ausgänge

626

Parameter	Datentyp	Beschreibung
RESULT	BYTE	0 = Baustein inaktiv 1 = Baustein-Ausführung beendet 2 = Baustein ist aktiv 3 = Fehler: Baustein wurde nicht ausgeführt
LEN	WORD	Länge des Eintrags in "Anzahl der Bytes" Der Wert für LEN muss mit der Länge des Empfangs-Arrays übereinstimmen. Andernfalls treten Störungen bei der SDO-Kommunikation auf.

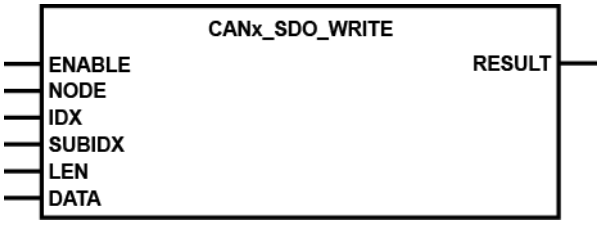
CANx_SDO_WRITE

615

Baustein-Typ = Funktionsblock (FB)

x = Nr. 1...n der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Symbol in CoDeSys:



CAN1_SDO_WRITE

9451

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR2500
- PDM360smart: CR1070, CR1071

Beschreibung

618

CANx_SDO_WRITE schreibt das SDO (→ Kapitel *Register [Service Data Objects]* (→ Seite 135)) mit den angegebenen Indizes in den Knoten.

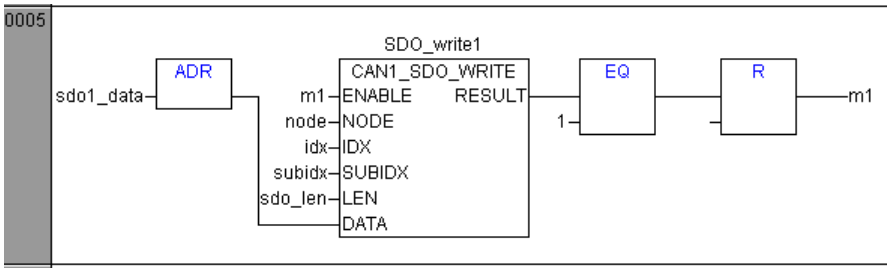
Über diesen FB können die Einträge im Objektverzeichnis geschrieben werden. Dadurch ist es möglich, die Knotenparameter gezielt zu setzen.

- alle ecomatmobile-Controller - Platinensteuerung: CS0015 - PDM360smart: CR1070, CR1071	- PDM360: CR1050, CR1051 - PDM360compact: CR1052, CR1053, CR1055, CR1056
aus Gerätebibliothek `ifm_CRnnnn_Vxxyyzz.LIB`	aus Gerätebibliothek `ifm_CANx_SDO_Vxxyyzz.LIB`
Voraussetzung: Knoten muss sich im Zustand PRE-OPERATIONAL oder OPERATIONAL befinden und im Modus "CANopen-Master".	Voraussetzung: Knoten muss sich im Modus "CANopen-Master" oder "CANopen-Slave" befinden.

HINWEIS

Der Wert für LEN muss mit der Länge des Sendearrays übereinstimmen. Andernfalls treten Störungen bei der SDO-Kommunikation auf.

Beispiel:



Parameter der Eingänge

619

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
NODE	BYTE	Nummer des Knotens
IDX	WORD	Index im Objektverzeichnis
SUBIDX	BYTE	Subindex bezogen auf den Index im Objektverzeichnis.
LEN	WORD	Länge des Eintrags in "Anzahl der Bytes" Der Wert für LEN muss mit der Länge des Sende-Arrays übereinstimmen. Andernfalls treten Störungen bei der SDO-Kommunikation auf.
DATA	DWORD	Adresse des Sendedaten-Arrays zulässige Länge = 0...255 Übergabe mit ADR-Operator

Parameter der Ausgänge

620

Parameter	Datentyp	Beschreibung
RESULT	BYTE	0 = Baustein inaktiv 1 = Bausteinausführung beendet 2 = Baustein ist aktiv 3 = Fehler: Baustein wurde nicht ausgeführt

6.7 CAN-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 Anwender sollte (je nach Applikation) auf gemeldete Fehler in der Anwendersoftware 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 in der Anwendersoftware dem Programmierer als CAN-Fehler-Bits zur Verfügung.

6.7.1 CAN-Fehler

Inhalt

8589

Fehlertelegramm

1172

Erkennt ein Busteilnehmer eine Fehlerbedingung, so sendet er sofort ein Fehlerflag 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 Fehlerflag 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

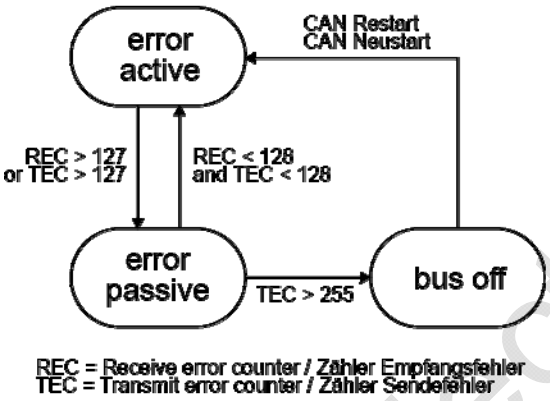
1173

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* (→ Seite 190)

error passive
→ *Teilnehmer fehlerpassiv* (→ Seite 190)

bus off
→ *Teilnehmer bus-off* (→ Seite 191)

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 Fehlerflags signalisieren. Wie bereits beschrieben, wird dadurch die übertragene Nachricht zerstört.

Teilnehmer fehlerpassiv

1175

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.

Um den Anwender über das Ansteigen des Fehlerzählers zu informieren, wird bei einem Wert des Fehlerzählers ≥ 96 die Systemvariable CANx_WARNING gesetzt. Der Teilnehmer ist in diesem Zustand noch fehleraktiv.

Teilnehmer bus-off

1176

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

Um diesen Zustand anzuzeigen, wird im Applikations-Programm der Merker CANx_BUSOFF gesetzt.

HINWEIS

Der Fehler CANx_BUSOFF wird vom Betriebssystem automatisch behandelt und zurückgesetzt. Soll eine genauere Fehlerbehandlung und Auswertung über das Applikations-Programm erfolgen, muss *CANx_ERRORHANDLER* (→ Seite 86) eingesetzt werden. Der Fehler CANx_BUSOFF muss dann explizit durch das Applikations-Programm zurückgesetzt werden.

6.7.2 Aufbau einer EMCY-Nachricht

Inhalt

8591

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 1003_{16}) abgelegt und können über einen SDO-Zugriff ausgelesen werden (→ *CANx_SDO_READ* (→ Seite 185)). Zusätzlich spiegelt sich die aktuelle Fehlersituation im Error-Register (Objekt 1001_{16}) wider.

Man unterscheidet folgende Fehler:

8046

Kommunikationsfehler

- 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

Anwendungsfehler

- Kurzschluss oder Leiterbruch
- Temperatur zu hoch

Aufbau einer Fehlernachricht

8047

Eine Fehlernachricht (EMCY Message) hat folgenden Aufbau:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
EMCY-Fehlercode, wie im Objekt 1003_{16} eingetragen		Objekt 1001_{16}	Herstellerspezifische Informationen				

Identifier

8048

Der Identifier für die Fehlernachricht besteht aus der Summe folgender Elemente:

EMCY-Default-Identifier 128 (80_{16})
+
Node-ID

EMCY-Fehlercode

8049

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

Objekt 0x1003 (Error Field)

8050

Das Objekt 1003_{16} 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-Fehlercode immer im Subindex 1_{16} 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 0_{16} wird auf die Anzahl der gespeicherten Fehler erhöht. Nachdem alle Fehler behoben sind, wird in das Fehlerfeld des Subindex 1_{16} der Wert "0" geschrieben.

Um den Fehlerspeicher zu löschen, kann der Subindex 0_{16} mit dem Wert "0" beschrieben werden. Andere Werte dürfen nicht eingetragen werden.

Gerätefehler signalisieren

1880

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. Kurzschluss an einem Ausgang), wenn *CANx_MASTER_EMCY_HANDLER* (→ Seite 164) oder *CANx_SLAVE_EMCY_HANDLER* (→ Seite 176) in das Applikations-Programm eingebunden wird.

Übersicht der automatisch verschickten EMCY-Fehlercodes für alle mit CoDeSys programmierbaren **ifm**-Geräte → Kapitel *Übersicht CANopen Error-Codes* (→ Seite 194).

Sollen zusätzlich noch applikations-spezifische Fehler durch das Applikations-Programm verschickt werden, werden *CANx_MASTER_SEND_EMERGENCY* (→ Seite 166) oder *CANx_SLAVE_SEND_EMERGENCY* (→ Seite 178) eingesetzt.

6.7.3 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 procedded 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 1001_{16} 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 "Leitungsbruch" an Kanal 2 (→ Installationsanleitung des Geräts):

COB-ID	DLC	Byte 0	Byte 1	Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
80_{16} + Node-ID		**00**	**FF**	**81**	**10**	**00**	00	00	00

Error-Code = $FF00_{16}$

Error-Register = 81_{16} = $1000\ 0001_2$, besteht also aus folgenden Fehlern:
- Generic Error (allgemeiner Fehler)
- manufacturer specific (herstellerspezifisch)

Betroffener Kanal = 0010_{16} = $0000\ 0000\ 0001\ 0000_2$ = Kanal 2

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:
  Fehlercode 2300_{16},
  im Objekt 1001_{16} wird der Wert 03_{16} ($0000\ 0011_2$) eingetragen
  (allg. Fehler und Stromfehler)

- CAN-Überlauf:
  Fehlercode 8110_{16},
  im Objekt 1001_{16} wird der Wert 13_{16} ($0001\ 0011_2$) eingetragen
  (allg. Fehler, Stromfehler und Kommunikationsfehler)

>> CAN-Überlauf bearbeitet:
Fehlercode 0000_{16},
im Objekt 1001_{16} wird der Wert 03_{16} ($0000\ 0011_2$) eingetragen
(allg. Fehler, Stromfehler, Kommunikationsfehler zurückgesetzt.)
Nur aus dieser Information kann man entnehmen, dass der Kommunikationsfehler nicht mehr anliegt.

Übersicht CANopen EMCY-Codes (CR107n)

2673

alle Angaben (hex) für 1. CAN-Schnittstelle

EMCY-Code Objekt 1003_{16}		Objekt 1001_{16}	Hersteller-spezifische Informationen					
Byte 0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Beschreibung**
00	21	03	I0	I1	I2	I3	I4	Diagnose Eingänge (nur CR1071)
00	31	05						Klemmenspannung VBBo/VBBs
00	42	09						Übertemperatur
00	61	11						Speicherfehler
00	80	11						CAN1 Monitoring SYNC-Error (nur Slave)
00	81	11						CAN1 Warngrenze (> 96)
10	81	11						CAN1 Empfangspuffer Überlauf
11	81	11						CAN1 Sendepuffer Überlauf
30	81	11						CAN1 Guard-/Heartbeat-Error (nur Slave)

7 Ein-/Ausgangs-Funktionen

Inhalt

1590

Hier zeigen wir Ihnen Funktionen zum Lesen und Bearbeiten der Signale an den Ein- und Ausgängen.

7.1 Eingangswerte verarbeiten

Inhalt

1602

Hier zeigen wir Ihnen Funktionen zum Lesen und Verarbeiten der analogen oder digitalen Signale am Geräte-Eingang.

HINWEIS

Die in der Steuerungskonfiguration von CoDeSys erscheinenden Rohwerte kommen direkt aus dem ADU. Sie sind noch nicht korrigiert!

Deshalb können in der Steuerungskonfiguration bei gleichen Geräten unterschiedliche Rohwerte erscheinen.

Erst durch die **ifm**-FBs (z.B. INPUT, INPUT_ANALOG) findet eine Fehlerkorrektur und Normierung statt. Die FBs liefern den korrigierten Wert.

7.1.1 ANALOG_RAW

9916

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- PDM360smart: CR1071

Symbol in CoDeSys:

Beschreibung

9918

ANALOG_RAW liefert das rohe Analog-Signal der Eingänge, ohne jegliche Filterung.

Parameter der Ausgänge

9919

Parameter	Datentyp	Beschreibung
P0	ARRAY [0...3] of WORD	Roh-Eingangswerte der analogen Eingänge: P0.0 für %IX0.00 P0.1 für %IX0.01 P0.2 für %IX0.02 P0.3 für %IX0.03

7.1.2 TOGGLE

3194

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek:	Für folgende Geräte verfügbar:
`ifm_PDM_UTIL_Vxxyyzz.LIB`	- PDM360: CR1050, CR1051 - PDM360compact: CR1052, CR1053, CR1055, CR1056
`ifm_PDMng_UTIL_Vxxyyzz.LIB`	- PDM360NG: CR108n
`ifm_PDMsmart_UTIL_Vxxyyzz.LIB`	- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:

Beschreibung

3304

TOGGLE ermöglicht das Setzen und Rücksetzen einer boolschen Variablen mit nur einem Eingangs-Bit.

Die erste steigende Flanke am Eingang IN setzt den Ausgang OUT auf 'TRUE'.
Die nächste steigende Flanke setzt den Ausgang wieder zurück auf 'FALSE'.
usw.

Parameter der Eingänge

3305

Parameter	Datentyp	Beschreibung
IN	BOOL	Flanke FALSE ⇨ TRUE: Setzen / Rücksetzen des Ausgangs

Parameter der Ausgänge

3306

Parameter	Datentyp	Beschreibung
OUT	BOOL	1. Flanke an IN ⇨ TRUE 2. Flanke an IN ⇨ FALSE 3. Flanke an IN ⇨ TRUE ...

7.2 Analoge Werte anpassen

Inhalt

1603

Wenn die Werte analoger Eingänge oder die Ergebnisse von analogen Funktionen angepasst werden müssen, helfen Ihnen die folgenden Funktionsblöcke.

7.2.1 NORM

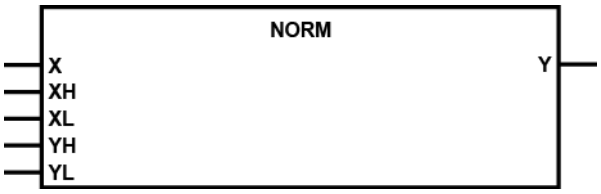
401

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

404

NORM normiert einen Wert innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen.

Der FB normiert einen Wert vom Typ WORD, der innerhalb der Grenzen XH und XL liegt, auf einen Ausgangswert innerhalb der Grenzen YH und YL. Der FB wird z.B. bei der Erzeugung von PWM-Werten aus analogen Eingangsgrößen genutzt.

HINWEIS

Der Wert für X muss sich im definierten Eingangsbereich zwischen XL und XH befinden (es findet keine interne Plausibilitätsprüfung des Wertes statt).

Bedingt durch die Rundungsfehler können Abweichungen beim normierten Wert um 1 auftreten.

Werden die Grenzen (XH/XL oder YH/YL) invertiert angegeben, erfolgt auch die Normierung invertiert.

Parameter der Eingänge

405

Parameter	Datentyp	Beschreibung
X	WORD	aktueller Eingangswert
XH	WORD	obere Grenze des Eingangswertebereich
XL	WORD	untere Grenze des Eingangswertebereich
YH	WORD	obere Grenze des Ausgangswertebereich
YL	WORD	untere Grenze des Ausgangswertebereich

Parameter der Ausgänge

406

Parameter	Datentyp	Beschreibung
Y	WORD	normierter Wert

Beispiel 1

407

unterer Grenzwert Eingang	0	XL
oberer Grenzwert Eingang	100	XH
unterer Grenzwert Ausgang	0	YL
oberer Grenzwert Ausgang	2000	YH

dann wandelt der Funktionsblock das Eingangssignal z.B. wie folgt um:

von X =	50	0	100	75
nach Y =	1000	0	2000	1500

Beispiel 2

408

unterer Grenzwert Eingang	2000	XL
oberer Grenzwert Eingang	0	XH
unterer Grenzwert Ausgang	0	YL
oberer Grenzwert Ausgang	100	YH

dann wandelt der Funktionsblock das Eingangssignal z.B. wie folgt um:

von X =	1000	0	2000	1500
nach Y =	50	100	0	25

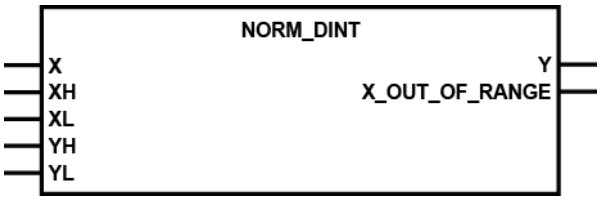
7.2.2 NORM_DINT

3200

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek:	Für folgende Geräte verfügbar:
`ifm_PDM_UTIL_Vxxyyzz.LIB`	- PDM360: CR1050, CR1051 - PDM360compact: CR1052, CR1053, CR1055, CR1056
`ifm_PDMng_UTIL_Vxxyyzz.LIB`	- PDM360NG: CR108n
`ifm_PDMsmart_UTIL_Vxxyyzz.LIB`	- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

3307

NORM_DINT normiert einen Wert innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen.

Der FB normiert einen Wert vom Typ DINT, der innerhalb der Grenzen XH und XL liegt, auf einen Ausgangswert innerhalb der Grenzen YH und YL. Der FB wird z.B. bei der Erzeugung von PWM-Werten aus analogen Eingangsgrößen genutzt.

HINWEIS

Der Wert für X muss sich im definierten Eingangsbereich zwischen XL und XH befinden (es findet keine interne Plausibilitätsprüfung des Wertes statt). Außerhalb dieses Wertebereiches wird der Ausgang X_OUT_OF_RANGE gesetzt.

Bedingt durch die Rundungsfehler können Abweichungen beim normierten Wert um 1 auftreten.

Werden die Grenzen (XH/XL oder YH/YL) invertiert angegeben, erfolgt auch die Normierung invertiert.

Parameter der Eingänge

3308

Parameter	Datentyp	Beschreibung
X	DINT	aktueller Eingangswert
XH	DINT	obere Grenze des Eingangswertebereich
XL	DINT	untere Grenze des Eingangswertebereich
YH	DINT	obere Grenze des Ausgangswertebereich
YL	DINT	untere Grenze des Ausgangswertebereich

Parameter der Ausgänge

3309

Parameter	Datentyp	Beschreibung
Y	DINT	normierter Wert
X_OUT_OF_RANGE	BOOL	Eingangswert X ist außerhalb des definierten Wertebereichs XL/XH

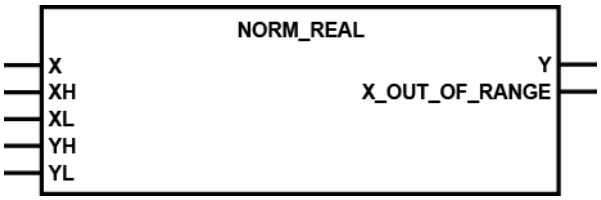
7.2.3 NORM_REAL

3202

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek:	Für folgende Geräte verfügbar:
`ifm_PDM_UTIL_Vxxyyzz.LIB`	- PDM360: CR1050, CR1051 - PDM360compact: CR1052, CR1053, CR1055, CR1056
`ifm_PDMng_UTIL_Vxxyyzz.LIB`	- PDM360NG: CR108n
`ifm_PDMsmart_UTIL_Vxxyyzz.LIB`	- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

3310

NORM_REAL normiert einen Wert innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen.

Der FB normiert einen Wert vom Typ REAL, der innerhalb der Grenzen XH und XL liegt, auf einen Ausgangswert innerhalb der Grenzen YH und YL. Der FB wird z.B. bei der Erzeugung von PWM-Werten aus analogen Eingangsgrößen genutzt.

HINWEIS

Der Wert für X muss sich im definierten Eingangsbereich zwischen XL und XH befinden (es findet keine interne Plausibilitätsprüfung des Wertes statt). Außerhalb dieses Wertebereiches wird der Ausgang X_OUT_OF_RANGE gesetzt.

Bedingt durch die Rundungsfehler können Abweichungen beim normierten Wert um 1 auftreten.

Werden die Grenzen (XH/XL oder YH/YL) invertiert angegeben, erfolgt auch die Normierung invertiert.

Parameter der Eingänge

3311

Parameter	Datentyp	Beschreibung
X	REAL	aktueller Eingangswert
XH	REAL	obere Grenze des Eingangswertebereich
XL	REAL	untere Grenze des Eingangswertebereich
YH	REAL	obere Grenze des Ausgangswertebereich
YL	REAL	untere Grenze des Ausgangswertebereich

Parameter der Ausgänge

3312

Parameter	Datentyp	Beschreibung
Y	REAL	normierter Wert
X_OUT_OF_RANGE	BOOL	Eingangswert X ist außerhalb des definierten Wertebereichs XL/XH

7.3 Zählerfunktionen zur Frequenz- und Periodendauermessung

Inhalt

1591

Je nach Controller werden bis zu 16 schnelle Eingänge unterstützt, die Eingangsfrequenzen bis zu 30 kHz verarbeiten können. Neben der reinen Frequenzmessung an den Eingängen FRQ können die Eingänge ENC auch zur Auswertung von inkrementellen Drehgebern (Zählerfunktion) mit einer maximalen Frequenz von 10 kHz eingesetzt werden. Die Eingänge CYL werden zur Periodendauermessung von langsamen Signalen eingesetzt.

Eingang	Frequenz [kHz]	Erklärung
FRQ 0 / ENC 0	30 / 10	Frequenzmessung / Drehgeber 1, Kanal A
FRQ 1 / ENC 0	30 / 10	Frequenzmessung / Drehgeber 1, Kanal B
FRQ 2 / ENC 1	30 / 10	Frequenzmessung / Drehgeber 2, Kanal A
FRQ 3 / ENC 1	30 / 10	Frequenzmessung / Drehgeber 2, Kanal B
CYL 0 / ENC 2	10	Periodendauermessung / Drehgeber 3, Kanal A
CYL 1 / ENC 2	10	Periodendauermessung / Drehgeber 3, Kanal B
CYL 2 / ENC 3	10	Periodendauermessung / Drehgeber 4, Kanal A
CYL 3 / ENC 3	10	Periodendauermessung / Drehgeber 4, Kanal B

Zur einfachen Auswertung stehen folgende Funktionsblöcke zur Verfügung:

7.3.1 Einsatzfälle

1592

Es ist zu beachten, dass – bedingt durch die unterschiedlichen Messmethoden – Fehler bei der Frequenzermittlung auftreten.

FREQUENCY (→ Seite 209) eignet sich für Frequenzen zwischen 100 Hz und 30 kHz, wobei der Fehler sich bei hohen Frequenzen verringert.

PERIOD (→ Seite 211) führt eine Periodendauermessung durch. Der FB ist damit für Frequenzen kleiner 1000 Hz geeignet. Generell kann er auch höhere Frequenzen messen. Dadurch wird aber die Zykluszeit stark belastet. Bei der Auslegung der Applikations-Software ist dies zu berücksichtigen.

7.3.2 Einsatz als Digitaleingänge

Inhalt

1593

Werden die schnellen Eingänge (FRQx / CYLx) als "normale" Digitaleingänge eingesetzt, muss die erhöhte Empfindlichkeit gegen Störimpulse beachtet werden (z.B. Kontaktprellen bei mechanischen Kontakten). Der Standard-Digitaleingang hat eine Eingangsfrequenz von 50 Hz. Das Eingangssignal muss ggf. softwaretechnisch entprellt werden.

FREQUENCY

537

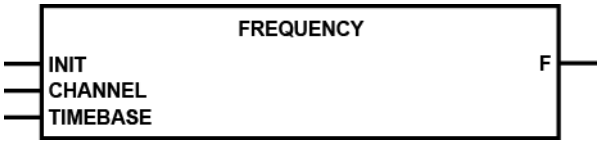
Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
  (Für Sicherheitssignale zusätzlich *SAFE_FREQUENCY_OK* zusammen mit *PERIOD* (→ Seite 211) einsetzen!)
- SmartController: CR25nn
- PDM360smart: CR1071

HINWEIS: Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CoDeSys:



Beschreibung

540

FREQUENCY misst die anstehende Signalfrequenz am angegebenen Kanal. Maximale Eingangsfrequenz → Datenblatt.

Der FB misst die Frequenz des am gewählten Kanal (CHANNEL) anstehenden Signals. Es wird dazu die positive Flanke ausgewertet. In Abhängigkeit von der Zeitbasis (TIMEBASE) können Frequenzmessungen in einem weiten Wertebereich durchgeführt werden. Hohe Frequenzen erfordern eine kurze Zeitbasis, niedrige eine entsprechend längere. Die Frequenz wird direkt in [Hz] ausgegeben.

HINWEIS

Für FREQUENCY können nur die Eingänge FRQ0...FRQ3 genutzt werden.

Parameter der Eingänge

541

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (steigende Flanke): Baustein wird initialisiert (nur 1 Zyklus lang) FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...x Wert abhängig vom Gerät, → Datenblatt)
TIMEBASE	TIME	Zeitbasis

HINWEIS

Vor dem Initialisieren kann der FB falsche Werte ausgeben.

► Ausgang erst auswerten, wenn FB initialisiert wurde!

Parameter der Ausgänge

542

Parameter	Datentyp	Beschreibung
F	REAL	Frequenz in [Hz]

PERIOD

370

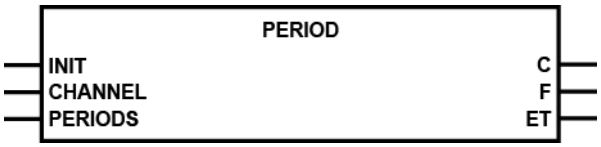
Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
  (Für Sicherheitssignale zusätzlich *SAFE_FREQUENCY_OK* zusammen mit *FREQUENCY* (→ Seite 209) einsetzen!)
- SmartController: CR25nn
- PDM360smart: CR1071

HINWEIS: Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CoDeSys:



Beschreibung

373

PERIOD misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] am angegebenen Kanal. Maximale Eingangsfrequenz → Datenblatt.

Der FB misst die Frequenz und die Zykluszeit des am gewählten Kanal (CHANNEL) anstehenden Signals. Zur Berechnung werden alle positiven Flanken ausgewertet und der Mittelwert über die Anzahl der angegebenen Perioden (PERIODS) gebildet.

Bei niedrigen Frequenzen kommt es mit FREQUENCY zu Ungenauigkeiten. Um dieses zu umgehen, kann PERIOD genutzt werden. Die Zykluszeit wird direkt in [µs] ausgegeben.

Der maximale Messbereich beträgt ca. 71 min.

HINWEIS

Für PERIOD können nur die Eingänge CYL0...CYL3 genutzt werden.
Für PDM360smart: CR1071: alle Eingänge.

Frequenzen < 0,5 Hz werden nicht mehr eindeutig angezeigt!

Parameter der Eingänge

374

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (steigende Flanke): Baustein wird initialisiert (nur 1 Zyklus lang) FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...x Wert abhängig vom Gerät, → Datenblatt)
PERIODS	BYTE	Anzahl der zu vergleichenden Perioden

HINWEIS

Vor dem Initialisieren kann der FB falsche Werte ausgeben.

► Ausgang erst auswerten, wenn FB initialisiert wurde.

Wir empfehlen dringend, alle benötigten Instanzen dieses FB zeitgleich zu initialisieren. Andernfalls können falsche Werte ausgegeben werden.

Parameter der Ausgänge

375

Parameter	Datentyp	Beschreibung
C	DWORD	Zykluszeit der erfassten Perioden in [µs]
F	REAL	Frequenz der erfassten Perioden in [Hz]
ET	TIME	Verstrichene Zeit seit Beginn der Periodendauermessung (nutzbar bei sehr langsamen Signalen)

PERIOD_RATIO

364

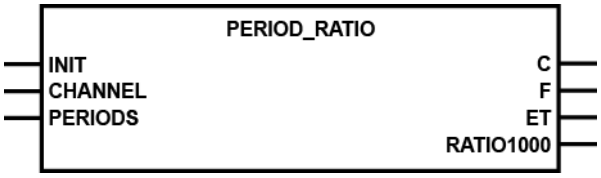
Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1071

HINWEIS: Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CoDeSys:



Beschreibung

367

PERIOD_RATIO misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] über die angegebenen Perioden am angegebenen Kanal. Zusätzlich wird das Puls-/Pausenverhältnis in [‰] angegeben. Maximale Eingangsfrequenz → Datenblatt.

Der FB misst die Frequenz und die Zykluszeit des am gewählten Kanal (CHANNEL) anstehenden Signals. Zur Berechnung werden alle positiven Flanken ausgewertet und der Mittelwert über die Anzahl der angegebenen Perioden (PERIODS) gebildet. Zusätzlich wird das Puls-/Pausenverhältnis in [‰] angegeben.

Beispiel: Bei einem Signalverhältnis von 25 ms High-Pegel und 75 ms Low-Pegel wird der Wert RATIO1000 von 250 ‰ ausgegeben.

Bei niedrigen Frequenzen kommt es mit FREQUENCY zu Ungenauigkeiten. Um dieses zu umgehen, kann PERIOD_RATIO genutzt werden. Die Zykluszeit wird direkt in [µs] ausgegeben.

Der maximale Messbereich beträgt ca. 71 min.

HINWEIS

Für PERIOD_RATIO können nur die Eingänge CYL0...CYL3 genutzt werden.
Für PDM360smart: CR1071: alle Eingänge.

Der Ausgang RATIO1000 liefert bei einen Puls/Pausenverhältnis von 100 % (Eingangssignal dauerhaft auf Versorgungsspannung) den Wert 0.

Frequenzen < 0,05 Hz werden nicht mehr eindeutig angezeigt!

Parameter der Eingänge

368

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (steigende Flanke): Baustein wird initialisiert (nur 1 Zyklus lang) FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...x Wert abhängig vom Gerät, → Datenblatt)
PERIODS	BYTE	Anzahl der zu vergleichenden Perioden

HINWEIS

Vor dem Initialisieren kann der FB falsche Werte ausgeben.

► Ausgang erst auswerten, wenn FB initialisiert wurde.

Wir empfehlen dringend, alle benötigten Instanzen dieses FB zeitgleich zu initialisieren. Andernfalls können falsche Werte ausgegeben werden.

Parameter der Ausgänge

369

Parameter	Datentyp	Beschreibung
C	DWORD	Zykluszeit der erfassten Perioden in [µs]
F	REAL	Frequenz der erfassten Perioden in [Hz]
ET	TIME	Verstrichene Zeit seit Beginn des letzten Zustandswechsels des Eingangssignals (nutzbar bei sehr langsamen Signalen)
RATIO1000	WORD	Puls-/Pause-Verhältnis in [‰]

PHASE

358

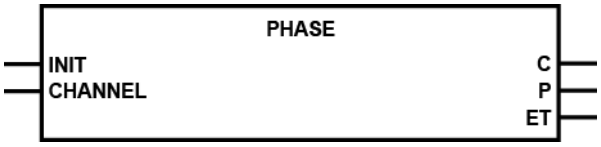
Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1071

HINWEIS: Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CoDeSys:



Beschreibung

361

PHASE liest ein Kanalpaar mit schnellen Eingängen ein und vergleicht die Phasenlage der Signale. Maximale Eingangsfrequenz → Datenblatt.

Diese Funktion fasst jeweils ein Kanalpaar mit schnellen Eingängen zusammen, so dass die Phasenlage zweier Signale zueinander ausgewertet werden kann. Es kann eine Periodendauer bis in den Sekundenbereich ausgewertet werden.

HINWEIS

Bei Frequenzen kleiner 15 Hz wird eine Periodendauer bzw. Phasenverschiebung von 0 angezeigt.

Parameter der Eingänge

362

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (steigende Flanke): Baustein wird initialisiert (nur 1 Zyklus lang) FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des Eingangskanal-Paares (0/2): 0 = Kanalpaar 0 = Eingänge 0 + 1 2 = Kanalpaar 1 = Eingänge 2 + 3

HINWEIS

Vor dem Initialisieren kann der FB falsche Werte ausgeben.

► Ausgang erst auswerten, wenn FB initialisiert wurde.

Wir empfehlen dringend, alle benötigten Instanzen dieses FB zeitgleich zu initialisieren. Andernfalls können falsche Werte ausgegeben werden.

Parameter der Ausgänge

363

Parameter	Datentyp	Beschreibung
C	DWORD	Periodendauer in [µs]
P	INT	Winkel der Phasenverschiebung (0...360 °)
ET	TIME	Verstrichene Zeit seit Beginn der Periodendauermessung (nutzbar bei sehr langsamen Signalen)

INC_ENCODER

4187

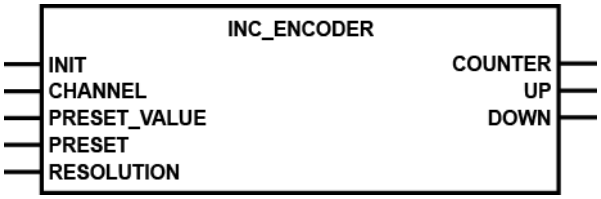
Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR25nn
- PDM360smart: CR1071

HINWEIS: Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CoDeSys:



Beschreibung

4330
2602

INC_ENCODER organisiert Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern.

Immer zwei Frequenzeingänge bilden das Eingangspaar, das über den FB ausgewertet wird. In folgender Tabelle finden Sie die zulässigen Grenzfrequenzen und die max. anschließbaren inkrementalen Drehgeber:

Gerät	Grenzfrequenz	max. Anzahl Drehgeber
BasicController: CR040n	1 kHz	2
CabinetController: CR030n	10 kHz	2
ClassicController: CR0020, CR0505	10 kHz	4
ClassicController: CR0032, CR0033	30 kHz	4
ExtendedController: CR0200	10 kHz	8
ExtendedController: CR0232, CR0233	30 kHz	8
Platinensteuerung: CS0015	0,5 kHz	2
SafetyController: CR7020, CR7021, CR7505, CR7506	10 kHz	4
SafetyController: CR7032	30 kHz	4
ExtendedSafetyController: CR7200, CR7201	10 kHz	8
ExtendedSafetyController: CR7132	30 kHz	8
SmartController: CR25nn	10 kHz	2
PDM360smart: CR1071	1 kHz	2

HINWEIS

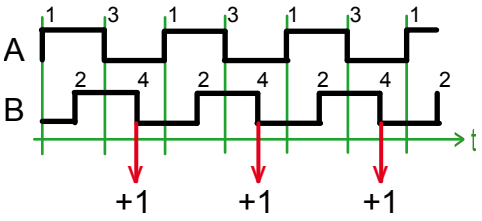
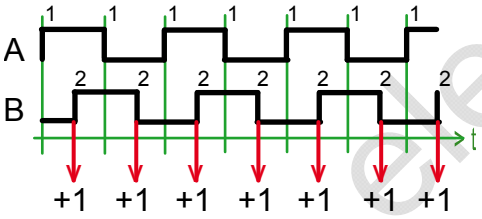
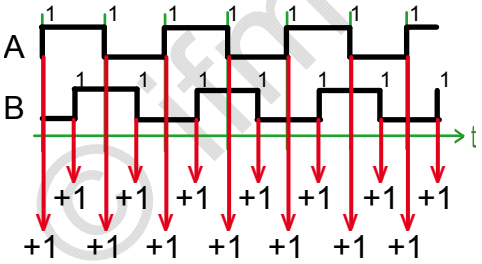
Je nach weiterer Belastung des Geräts kann die Grenzfrequenz sinken, wenn "viele" Drehgeber ausgewertet werden.

Bei zu hoher Belastung kann die Zykluszeit unzulässig lang werden (→ *Begrenzungen und Programmierhinweise* (→ Seite 55)).

Über den PRESET_VALUE kann der Zähler auf einen Voreinstellwert gesetzt werden. Der Wert wird übernommen, wenn PRESET auf TRUE gesetzt wird. Anschließend muss PRESET wieder auf FALSE gesetzt werden, damit der Zähler wieder aktiv wird.

Am Ausgang COUNTER steht der aktuelle Zählerstand an. Die Ausgänge UP und DOWN zeigen die aktuelle Zählrichtung des Zählers an. Die Ausgänge sind dann TRUE, wenn im vorangegangenen Programmzyklus der Zähler in die entsprechende Richtung gezählt hat. Bleibt der Zähler stehen, wird auch der Richtungsausgang im folgenden Programmzyklus zurückgesetzt.

Am Eingang RESOLUTION kann die Auflösung des Drehgebers vervielfacht ausgewertet werden:
1 = normale Auflösung (identisch mit der Auflösung des Drehgebers),
2 = Auflösung doppelt auswerten,
4 = Auflösung 4-fach auswerten.
Alle anderen Werte an diesem Eingang bedeuten normale Auflösung.

	RESOLUTION = 1 Bei normaler Auflösung wird nur die fallende Flanke des B-Signals ausgewertet.
	RESOLUTION = 2 Bei doppelter Auflösung werden die fallenden und die steigenden Flanken des B-Signals ausgewertet.
	RESOLUTION = 4 Bei 4-facher Auflösung werden die fallenden und die steigenden Flanken sowohl des A-Signals wie auch des B-Signals ausgewertet.

Parameter der Eingänge

4332
529

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (steigende Flanke): Baustein wird initialisiert (nur 1 Zyklus lang) FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des Eingangskanal-Paares (0...x Wert abhängig vom Gerät, → Datenblatt) 0 = Kanalpaar 0 = Eingänge 0 + 1 1 = Kanalpaar 1 = Eingänge 2 + 3 2 = Kanalpaar 2 = Eingänge 4 + 5 3 = Kanalpaar 3 = Eingänge 6 + 7
PRESET_VALUE	DINT	Voreinstellwert des Zählers
PRESET	BOOL	TRUE: (nur 1 Zyklus lang): Voreinstellwert wird übernommen FALSE: Zähler aktiv
RESOLUTION	BYTE	Faktor der Drehgeber-Auflösung (1, 2, 4): 1 = normale Auflösung 2 = doppelte Auflösung 4 = 4-fache Auflösung Alle anderen Werte zählen wie "1"

Parameter der Ausgänge

530

Parameter	Datentyp	Beschreibung
COUNTER	DINT	aktueller Zählerstand
UP	BOOL	TRUE: Zähler zählt aufwärts FALSE: Zähler steht
DOWN	BOOL	TRUE: Zähler zählt abwärts FALSE: Zähler steht

FAST_COUNT

567

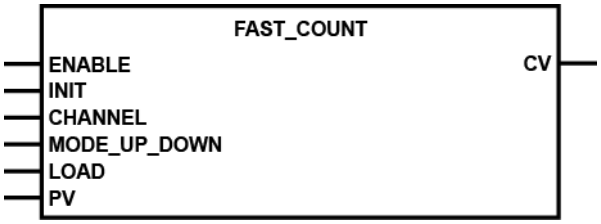
Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1071

HINWEIS: Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CoDeSys:



Beschreibung

570

FAST_COUNT arbeitet als Zählerbaustein für schnelle Eingangsimpulse.

Diese Funktion erfasst schnelle Impulse an den FRQ-Eingangskanälen 0...3. Mit dem FRQ-Eingangskanal 0 arbeitet FAST_COUNT wie der Baustein CTU. Maximale Eingangsfrequenz → Datenblatt.

> **HINWEIS**
>
> Bei den **ecomat*mobile***-Controllern kann der Kanal 0 technisch bedingt nur als Aufwärtszähler eingesetzt werden. Die Kanäle 1...3 können als Auf- und Abwärtszähler genutzt werden.

Parameter der Eingänge

571

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt beginnend vom Startwert FALSE: Baustein wird nicht ausgeführt
INIT	BOOL	TRUE (steigende Flanke): Baustein wird initialisiert (nur 1 Zyklus lang) FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...x Wert abhängig vom Gerät, → Datenblatt)
MODE_UP_DOWN	BOOL	TRUE: Zähler zählt abwärts FALSE: Zähler zählt aufwärts
LOAD	BOOL	TRUE: Startwert PV wird geladen FALSE: Startwert "0" wird geladen
PV	DWORD CR1071: WORD	Startwert (Preset value)

HINWEIS

Nach Rücksetzen des Parameters INIT zählt der Zähler vom angegebenen Startwert an.

Nach erneutem Setzen von ENABLE zählt der Zähler von dem Wert an weiter, der beim letzten Rücksetzen von ENABLE gültig war.

Parameter der Ausgänge

572

Parameter	Datentyp	Beschreibung
CV	DWORD CR1071: WORD	Ausgangswert des Zählers

7.4 PWM-Funktionen

Inhalt

2303

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

7.4.1 Verfügbarkeit von PWM

8472

PWM ist in folgenden Geräten verfügbar:

	Anzahl verfügbare PWM-Ausgänge	davon stromgeregelt (PWMi)	PWM-Frequenz [Hz]
BasicController: CR0401	8	0	20...250
BasicController: CR0403	12	2	20...250
CabinetController: CR0301	4	0	25...250
CabinetController: CR0302, CR0303	8	0	25...250
ClassicController: CR0020	12	8	25...250
ClassicController: CR0505	8	8	25...250
ClassicController: CR0032, CR0033	16	16	25...250
ExtendedController: CR0200	24	16	25...250
ExtendedController: CR0232, CR0233	32	32	25...250
Platinensteuerung: CS0015	8	0	25...250
SafetyController: CR7020, CR7021	12	8	25...250
SafetyController: CR7505, CR0506	8	8	25...250
ExtendedSafetyController: CR7200, CR7201	24	16	25...250
SmartController: CR25nn	4	4	25...250
PDM360smart: CR1071	4	0	25...250

7.4.2 PWM-Signalverarbeitung

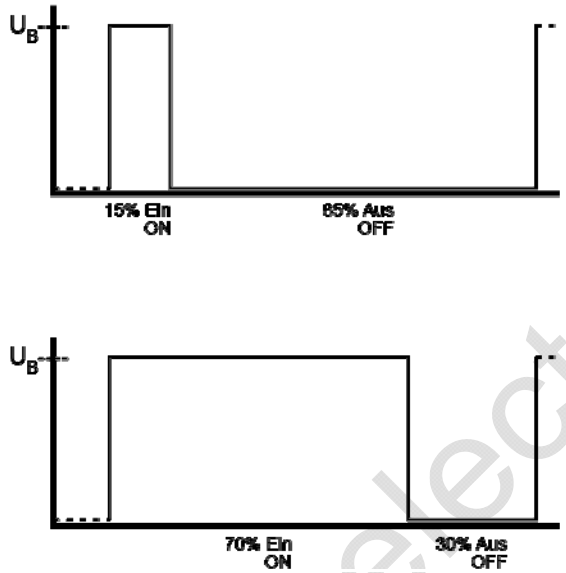
Inhalt

1526

PWM – Einführung

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 Applikations-Programm initialisiert und entsprechend dem gewünschten Ausgangssignal parametriert werden.

PWM-Funktionen und deren Parameter

Inhalt

1527

PWM / PWM1000

1528

Je nach Einsatzfall und gewünschter Auflösung kann bei der Applikations-Programmierung zwischen PWM und PWM1000 gewählt werden. Bei Einsatz der Reglerfunktionen wird eine hohe Genauigkeit und damit Auflösung benötigt. Daher wird in diesem Fall die mehr technische PWM-Funktion genutzt.

Soll der Aufwand bei der Implementierung gering gehalten und soll keine hohen Anforderungen an die Genauigkeit gestellt werden, kann *PWM1000* (→ Seite 233) eingesetzt werden. Bei diesem FB können die PWM-Frequenz direkt in [Hz] und das Puls-Pausen-Verhältnis in 1‰-Schritten eingegeben werden.

PWM-Frequenz

1529

Abhängig vom Ventiltyp wird eine entsprechende PWM-Frequenz benötigt. Die PWM-Frequenz wird bei der PWM-Funktion über den Reload-Wert (Funktion PWM) oder direkt als Zahlenwert in Hz (Funktion PWM1000) übergeben. Je nach R360-Controller unterscheiden sich die PWM-Ausgänge in ihrer Arbeits-, aber nicht in ihrer Wirkungsweise.

Mittels eines intern ablaufenden Zählers, abgeleitet vom CPU-Takt, wird die PWM-Frequenz realisiert. Mit der Initialisierung der Funktion PWM wird dieser Zähler gestartet. Je nach PWM-Ausgangsgruppe (0...3 und/oder 4...7 oder 4...11) zählt dieser dann von $FFFF_{16}$ rückwärts bzw. von 0000_{16} aufwärts. Bei Erreichen eines übergebenen Vergleichswertes (VALUE) wird der Ausgang gesetzt. Mit Überlauf des Zählers (Zählerstandwechsel von 0000_{16} nach $FFFF_{16}$ oder von $FFFF_{16}$ nach 0000_{16}) wird der Ausgang wieder zurückgesetzt und der Vorgang neu gestartet.

Soll dieser interne Zähler nicht zwischen 0000_{16} und $FFFF_{16}$ laufen, kann ein anderer Preset-Wert (RELOAD) für den internen Zähler übergeben werden. Dadurch steigt die PWM-Frequenz. Der Vergleichswert muss innerhalb des nun festgelegten Bereiches liegen.

PWM-Kanäle 0...3

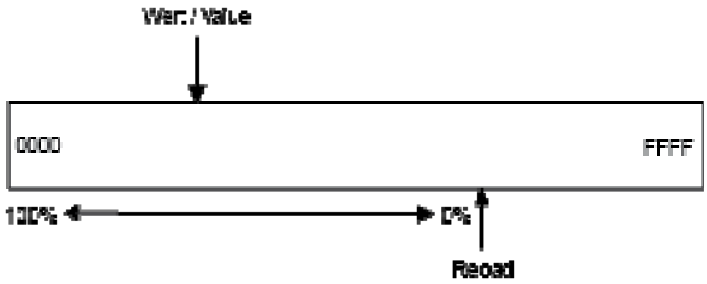
1530

Diese 4 PWM-Kanäle bieten die größte Flexibilität bei der Parametrierung. Die PWM-Kanäle 0...3 sind in allen Controller-Varianten vorhanden, je nach Geräteausführung mit oder ohne Stromregelung.

Für jeden Kanal kann eine eigene PWM-Frequenz (RELOAD-Wert) eingestellt werden. Zwischen *PWM* (→ Seite 229) und *PWM1000* (→ Seite 233) kann frei gewählt werden.

Berechnung des RELOAD-Wertes

1531



Grafik: RELOAD-Wert für PWM-Kanäle 0...3

Der RELOAD-Wert des internen PWM-Zählers berechnet sich in Abhängigkeit des Parameters DIV64 und der CPU-Frequenz wie folgt:

	- CabinetController: CR0303 - ClassicController: CR0020, CR0505 - ExtendedController: CR0200 - SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506	- CabinetController: CR0301, CR0302 - SmartController: CR25nn - Platinensteuerung: CS0015 - PDM360smart: CR1071
DIV64 = 0	RELOAD = 20 MHz / f_{PWM}	RELOAD = 10 MHz / f_{PWM}
DIV64 = 1	RELOAD = 312,5 kHz / f_{PWM}	RELOAD = 156,25 kHz / f_{PWM}

Je nachdem, ob eine hohe oder niedrige PWM-Frequenz benötigt wird, muss der Eingang DIV64 auf FALSE (0) oder TRUE (1) gesetzt werden. Bei PWM-Frequenzen unter 305 Hz oder 152 Hz (je nach Controller) muss DIV64 auf "1" gesetzt werden, damit der Reload-Wert nicht größer als $FFFF_{16}$ wird.

Berechnungsbeispiele RELOAD-Wert

1532

- CabinetController: CR0303 - ClassicController: CR0020, CR0505 - ExtendedController: CR0200 - SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506		- CabinetController: CR0301, CR0302 - SmartController: CR25nn - Platinensteuerung: CS0015 - PDM360smart: CR1071
Die PWM-Frequenz soll 400 Hz betragen. 20 MHz _____ = 50000_{10} = $C350_{16}$ = RELOAD 400 Hz Der zulässige Bereich des PWM-Wertes ist damit der Bereich von 0000_{16} bis $C350_{16}$. Der Vergleichswert, bei dem der Ausgang durchschaltet, muss dann zwischen 0000_{16} und $C350_{16}$ liegen.		Die PWM-Frequenz soll 200 Hz betragen. 10 MHz _____ = 50000_{10} = $C350_{16}$ = RELOAD 200 Hz Der zulässige Bereich des PWM-Wertes ist damit der Bereich von 0000_{16} bis $C350_{16}$. Der Vergleichswert, bei dem der Ausgang durchschaltet, muss dann zwischen 0000_{16} und $C350_{16}$ liegen.

Daraus ergeben sich folgende Puls-Pausen-Verhältnisse:

Puls-Pausen-Verhältnis	Einschaltdauer	Wert für Puls-Pausen-Verhältnis
Minimal	0 %	$C350_{16}$
Maximal	100 %	0000_{16}

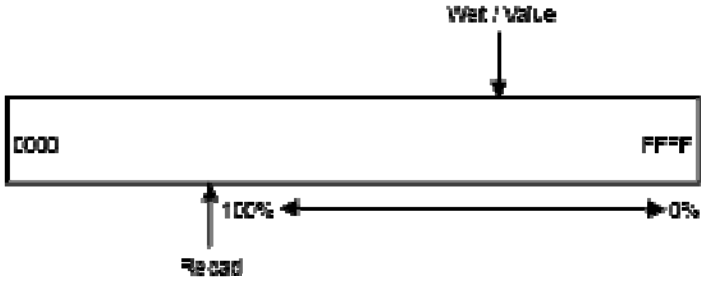
Zwischen minimaler und maximaler Ansteuerung sind 50000 Zwischenwerte (PWM-Werte) möglich.

PWM-Kanäle 4...7 / 8...11 (wenn vorhanden)

1533

Gilt nur für folgende Geräte:
- CabinetController: CR0303
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506

Diese 4/8 PWM-Kanäle können nur auf eine gemeinsame PWM-Frequenz eingestellt werden. Bei der Programmierung dürfen PWM und PWM1000 nicht gemischt eingesetzt werden.



Grafik: RELOAD-Wert für PWM-Kanäle 4...7 / 8...11

Der RELOAD-Wert des internen PWM-Zählers berechnet sich (für alle Controller) in Abhängigkeit des Parameters DIV64 und der CPU-Frequenz wie folgt:

DIV64 = 0	RELOAD = 10000_{16} – (2,5 MHz / f_{PWM})
DIV64 = 1	RELOAD = 10000_{16} – (312,5 kHz / f_{PWM})

Je nachdem, ob eine hohe oder niedrige PWM-Frequenz benötigt wird, muss der Eingang DIV64 auf FALSE (0) oder TRUE (1) gesetzt werden. Bei PWM-Frequenzen unter 39 Hz muss DIV64 auf "1" gesetzt werden, damit der RELOAD-Wert nicht kleiner als 0000_{16} wird.

Beispiel:

Die PWM-Frequenz soll 200 Hz betragen.

$$\frac{2{,}5\ \text{MHz}}{200\ \text{Hz}} = 12500_{10} = 30D4_{16}$$

RELOAD-Wert = 10000_{16} - $30D4_{16}$ = $CF2C_{16}$

Der zulässige Bereich des PWM-Wertes ist damit der Bereich von $CF2C_{16}$ bis $FFFF_{16}$

Der Vergleichswert, bei dem der Ausgang durchschaltet, muss dann zwischen $CF2C_{16}$ und $FFFF_{16}$ liegen.

HINWEIS

Die PWM-Frequenz ist für alle PWM-Ausgänge (4...7 oder 4...11) gleich.

PWM und PWM1000 dürfen nicht gemischt werden.

Daraus ergeben sich folgende Puls-Pausen-Verhältnisse:

Puls-Pausen-Verhältnis	Einschaltdauer	Wert für Puls-Pausen-Verhältnis
Minimal	0 %	$FFFF_{16}$
Maximal	100 %	$CF2C_{16}$

Zwischen minimaler und maximaler Ansteuerung sind 12500 Zwischenwerte (PWM-Werte) möglich.

HINWEIS

für ClassicController und ExtendedController gilt:

Werden die PWM-Ausgänge 4...7 eingesetzt (unabhängig ob stromgeregelt oder über einen der PWM-Funktionsblöcke), muss auch bei den Ausgängen 8...11 die gleiche Frequenz und der entsprechende Reload-Wert eingestellt werden. Daraus folgt: es müssen bei diesen Ausgängen die gleichen Funktionsblöcke eingesetzt werden.

PWM-Dither

1534

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 von einer Schwebung mit der Dither-Frequenz und der Amplitude DITHER_VALUE überlagert. Die Dither-Frequenz wird als Verhältnis (Teiler, DITHER_DIVIDER * 2) der PWM-Frequenz angegeben.

Rampenfunktion

1535

Soll der Wechsel von einem PWM-Wert zum nächsten nicht hart erfolgen, z.B. von 15 % Ein auf 70 % Ein (→ Grafik in Kapitel *PWM – Einführung* (→ Seite 223)), kann z.B. durch Nutzung von PT1 ein verzögerter Anstieg realisiert werden. Die für PWM genutzte Rampenfunktion basiert auf der CoDeSys-Bibliothek `UTIL.LIB`. Auf diese Weise können dann z.B. Hydrauliksysteme im Sanftanlauf betrieben werden.

HINWEIS

Beim Installieren der **ecomat*mobile***-DVD "Software, tools and documentation" wurden auch Projekte mit Beispielen auf Ihrem Computer im Programmverzeichnis abgelegt:
…`\ifm electronic\CoDeSys V…\Projects\DEMO_PLC_CDV`… (für Controller) oder
…`\ifm electronic\CoDeSys V…\Projects\DEMO_PDM_CDV`… (für PDMs)

Dort finden Sie auch Projekte mit Beispielen zu diesem Thema. Es wird dringend empfohlen, dem gezeigten Schema zu folgen.
→ Kapitel *ifm-Demo-Programme* (→ Seite 42)

HINWEIS

Die PWM-Funktion der Controller ist eine vom Prozessor zur Verfügung gestellte Hardware-Funktion. Die PWM-Funktion bleibt solange gesetzt, bis am Controller ein Hardware-Reset (Aus- und Einschalten der Versorgungsspannung) durchgeführt wurde.

PWM

320

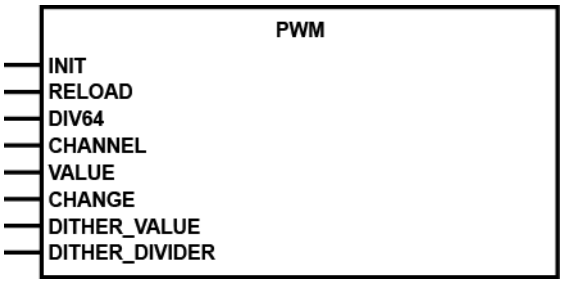
Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR25nn
- PDM360smart: CR1071

HINWEIS: Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CoDeSys:



Beschreibung

323

PWM wird zum Initialisieren und Parametrieren der PWM-Ausgänge genutzt.

Der FB hat einen mehr technischen Hintergrund. Durch seinen Aufbau können die PWM-Werte sehr fein abgestuft ausgegeben werden. Damit eignet sich dieser FB zum Aufbau von Reglern.

Der FB wird einmalig für jeden Kanal in der Initialisierung des Applikations-Programms aufgerufen. Dabei muss der Eingang INIT auf TRUE gesetzt sein. Bei der Initialisierung wird auch der Parameter RELOAD übergeben.

HINWEIS

Der Wert RELOAD muss für die Kanäle 4...7 gleich sein.
Aber beim ClassicController oder ExtendedController: für die Kanäle 4...11
Aber beim PDM360smart: CR1071: für die Kanäle 0...3

Bei diesen Kanälen dürfen PWM und *PWM1000* (→ Seite 233) nicht gemischt werden.

Die PWM-Frequenz (und damit der RELAOD-Wert) ist intern auf 5 kHz begrenzt.

Je nachdem, ob eine hohe oder niedrige PWM-Frequenz benötigt wird, muss der Eingang DIV64 auf FALSE (0) oder TRUE (1) gesetzt werden.

Während des zyklischen Programmablaufes ist INIT auf FALSE gesetzt. Der FB wird aufgerufen und dabei der neue PWM-Wert übergeben. Der Wert wird übernommen, wenn der Eingang CHANGE = TRUE ist.

Eine Strommessung für den initialisierten PWM-Kanal kann realisiert werden:

- mit *OUTPUT_CURRENT* *)
  *) Gilt nur für folgende Geräte:
  - ClassicController: CR0020, CR0032, CR0033, CR0505
  - ExtendedController: CR0200, CR0232, CR0233
  - SafetyController: CR7nnn
  - SmartController: CR25nn
- oder z.B. mit **ifm**-Gerät EC2049 (Vorschaltgerät zur Strommessung).

PWM_DITHER wird einmalig für jeden Kanal in der Initialisierung des Applikations-Programms aufgerufen. Dabei muss der Eingang INIT auf TRUE gesetzt sein. Bei der Initialisierung werden der DIVIDER (Divisor) zur Bildung der Dither-Frequenz und der Wert (VALUE) übergeben.

Info

Die Parameter DITHER_FREQUENCY und DITHER_VALUE können für jeden Kanal individuell eingestellt werden.

Parameter der Eingänge

324

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (steigende Flanke): Baustein wird initialisiert (nur 1 Zyklus lang) FALSE: im weiteren Programmablauf
RELOAD	WORD	Wert zur Festlegung der PWM-Frequenz (→ Kapitel *Berechnung des RELOAD-Wertes* (→ Seite 225))
DIV64	BOOL	CPU-Takt / 64
CHANNEL	BYTE	aktueller PWM-Kanal / -Ausgang
VALUE	WORD	aktueller PWM-Wert
CHANGE	BOOL	TRUE: neuer PWM-Wert wird übernommen FALSE: geänderter PWM-Wert hat keinen Einfluss auf den Ausgang
DITHER_VALUE	WORD	Amplitude des Dither-Wertes (→ Kapitel *PWM-Dither* (→ Seite 227))
DITHER_DIVIDER	WORD	Dither-Frequenz = PWM-Frequenz / DIVIDER * 2

PWM100

332

Baustein-Typ = Funktionsblock (FB)

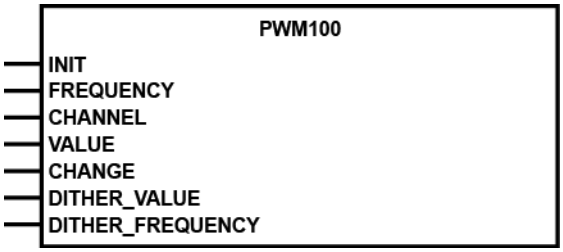
WICHTIG: Neue **ecomat*mobile***-Controller unterstützen nur noch *PWM1000* (→ Seite 233).

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7200, CR7505
- SmartController: CR25nn
- PDM360smart: CR1071

HINWEIS: Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CoDeSys:



Beschreibung

335

PWM100 organisiert die Initialisierung und Parametrierung der PWM-Ausgänge.

Der FB ermöglicht eine einfache Anwendung der PWM-Funktion im Gerät. Die PWM-Frequenz kann direkt in [Hz] und das Puls-Pausen-Verhältnis in 1 %-Schritten angegeben werden. Zum Aufbau von Reglern ist dieser Baustein durch die relativ grobe Abstufung **nicht** geeignet.

Der FB wird einmalig für jeden Kanal in der Initialisierung des Applikations-Programms aufgerufen. Dabei muss der Eingang INIT auf TRUE gesetzt sein. Bei der Initialisierung wird auch der Parameter FREQUENCY übergeben.

HINWEIS

Der Wert FREQUENCY muss für die Kanäle 4...7 gleich sein.
Aber beim ClassicController oder ExtendedController: für die Kanäle 4...11
Aber beim PDM360smart: CR1071: für die Kanäle 0...3

Bei diesen Kanälen dürfen *PWM* (→ Seite 229) und PWM100 nicht gemischt werden.

Die PWM-Frequenz ist intern auf 5 kHz begrenzt.

Während des zyklischen Programmablaufes ist INIT auf FALSE gesetzt. Der FB wird aufgerufen und dabei der neue PWM-Wert übergeben. Der Wert wird übernommen, wenn der Eingang CHANGE = TRUE ist.

Eine Strommessung für den initialisierten PWM-Kanal kann realisiert werden:

- über *OUTPUT_CURRENT* *)
  *) Gilt nur für folgende Geräte:
  - ClassicController: CR0020, CR0032, CR0033, CR0505
  - ExtendedController: CR0200, CR0232, CR0233
  - SafetyController: CR7nnn
  - SmartController: CR25nn
- oder z.B. mit **ifm**-Gerät EC 2049 (Vorschaltgerät zur Strommessung).

DITHER wird einmalig für jeden Kanal in der Initialisierung des Applikations-Programms aufgerufen. Dabei muss der Eingang INIT auf TRUE gesetzt sein. Bei der Initialisierung werden der Wert FREQUENCY zur Bildung der Dither-Frequenz und der Dither-Wert (VALUE) übergeben.

Info

Die Parameter DITHER_FREQUENCY und DITHER_VALUE können für jeden Kanal individuell eingestellt werden.

Parameter der Eingänge

336

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (steigende Flanke): Baustein wird initialisiert (nur 1 Zyklus lang) FALSE: im weiteren Programmablauf
FREQUENCY	WORD	PWM-Frequenz in [Hz]
CHANNEL	BYTE	aktueller PWM-Kanal / -Ausgang
VALUE	BYTE	aktueller PWM-Wert
CHANGE	BOOL	TRUE: neuer PWM-Wert wird übernommen FALSE: geänderter PWM-Wert hat keinen Einfluss auf den Ausgang
DITHER_VALUE	BYTE	Amplitude des Dither-Wertes in [%]
DITHER_FREQUENCY	WORD	Dither-Frequenz in [Hz]

PWM1000

326

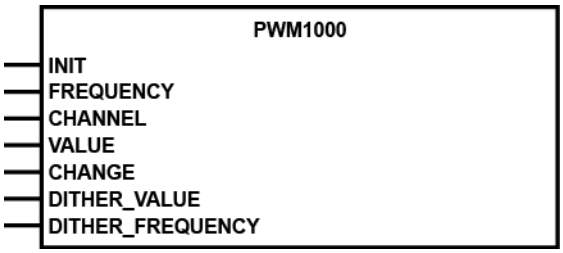
Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1071

HINWEIS: Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CoDeSys:



Beschreibung

329

PWM1000 organisiert die Initialisierung und Parametrierung der PWM-Ausgänge.

Der FB ermöglicht eine einfache Anwendung der PWM-Funktion im Gerät. Die PWM-Frequenz kann direkt in [Hz] und das Puls-Pausen-Verhältnis in 1 ‰-Schritten angegeben werden.

Der FB wird einmalig für jeden Kanal in der Initialisierung des Applikations-Programms aufgerufen. Dabei muss der Eingang INIT auf TRUE gesetzt sein. Bei der Initialisierung wird auch der Parameter FREQUENCY übergeben.

HINWEIS

Der Wert FREQUENCY muss für die Kanäle 4...7 gleich sein.
Aber beim ClassicController oder ExtendedController: für die Kanäle 4...11
Aber beim PDM360smart: CR1071: für die Kanäle 0...3

Bei diesen Kanälen dürfen *PWM* (→ Seite 229) und PWM1000 nicht gemischt werden.

Die PWM-Frequenz ist intern auf 5 kHz begrenzt.

Während des zyklischen Programmablaufes ist INIT auf FALSE gesetzt. Der FB wird aufgerufen und dabei der neue PWM-Wert übergeben. Der Wert wird übernommen, wenn der Eingang CHANGE = TRUE ist.

Eine Strommessung für den initialisierten PWM-Kanal kann realisiert werden:

- mit *OUTPUT_CURRENT* *)
  *) Gilt nur für folgende Geräte:
  - ClassicController: CR0020, CR0032, CR0033, CR0505
  - ExtendedController: CR0200, CR0232, CR0233
  - SafetyController: CR7nnn
  - SmartController: CR25nn
- oder z.B. mit **ifm**-Gerät EC2049 (Vorschaltgerät zur Strommessung).

DITHER wird einmalig für jeden Kanal in der Initialisierung des Applikations-Programms aufgerufen. Dabei muss der Eingang INIT auf TRUE gesetzt sein. Bei der Initialisierung werden der Wert FREQUENCY zur Bildung der Dither-Frequenz und der Dither-Wert (VALUE) übergeben.

Info

Die Parameter DITHER_FREQUENCY und DITHER_VALUE können für jeden Kanal individuell eingestellt werden.

Parameter der Eingänge

330

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (steigende Flanke): Baustein wird initialisiert (nur 1 Zyklus lang) FALSE: im weiteren Programmablauf
FREQUENCY	WORD	PWM-Frequenz in [Hz]
CHANNEL	BYTE	aktueller PWM-Kanal / -Ausgang
VALUE	WORD	aktueller PWM-Wert
CHANGE	BOOL	TRUE: neuer PWM-Wert wird übernommen FALSE: geänderter PWM-Wert hat keinen Einfluss auf den Ausgang
DITHER_VALUE	WORD	Amplitude des Dither-Wertes in [%]
DITHER_FREQUENCY	WORD	Dither-Frequenz in [Hz]

7.5 Regler-Funktionen

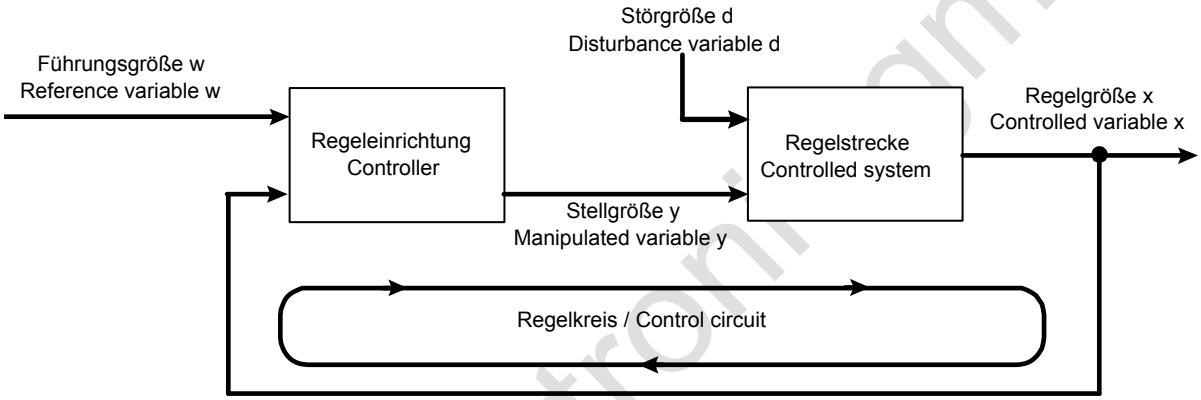
Inhalt

1622

7.5.1 Allgemeines

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

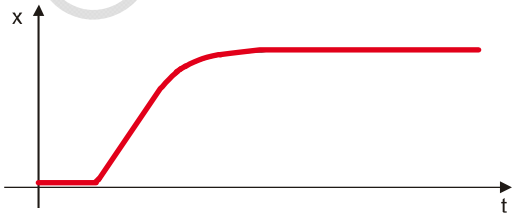
Die Auswahl einer geeigneten Regeleinrichtung und deren optimale Einstellung setzt genaue Angaben über das Beharrungsverhalten und das dynamische Verhalten der Regelstrecke voraus. In den meisten Fällen können diese Kennwerte aber nur experimentell ermittelt werden und sind kaum beeinflussbar.

Man kann drei Typen von Regelstrecken unterscheiden:

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).

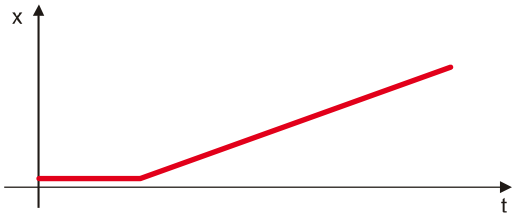


Grafik: P-Regler = Regelstrecke mit Ausgleich

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).

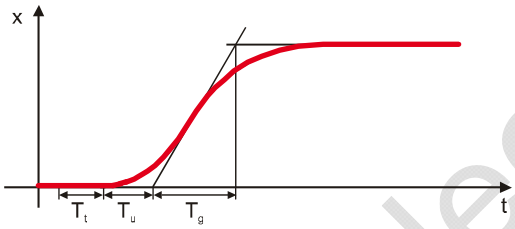


Grafik: I-Regler = Regelstrecke ohne Ausgleich

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.

7.5.2 Einstellregel für einen Regler

1627

Für Regelstrecken, deren Zeitkonstanten nicht bekannt sind, ist das Einstellverfahren nach Ziegler und Nickols im geschlossenen Regelkreis vorteilhaft:

Einstellregel

1628

Die Regeleinrichtung wird zunächst als eine reine P-Regeleinrichtung betrieben. Dazu wird die Vorhaltezeit T_V auf 0 und die Nachstellzeit T_N auf einen sehr großen Wert (ideal auf unendlich) für eine träge Strecke eingestellt. Bei einer schnellen Regelstrecke sollte ein kleines T_N gewählt werden.

Der Proportionalbeiwert KP wird anschließend solange vergrößert, bis die Regel- und die Stellabweichung bei KP = $KP_{kritisch}$ Dauerschwingungen mit konstanter Amplitude ausführen. Es ist damit die Stabilitätsgrenze erreicht.

Anschließend muss die Periodendauer $T_{kritisch}$ der Dauerschwingung ermittelt werden.

Nur bei Bedarf einen D-Anteil hinzufügen.

T_V sollte ca. 2...10-mal kleiner sein als T_N

KP sollte gleich groß wie KD gewählt werden.

Idealisiert ist die Regelstrecke wie folgt einzustellen:

Regeleinrichtung	KP = KD	TN	TV
P	2,0 * $KP_{kritisch}$	—	—
PI	2,2 * $KP_{kritisch}$	0,83 * $T_{kritisch}$	—
PID	1,7 * $KP_{kritisch}$	0,50 * $T_{kritisch}$	0,125 * $T_{kritisch}$

HINWEIS

Bei diesem Einstellverfahren darauf achten, dass die Regelstrecke durch die auftretenden Schwingungen keinen Schaden nimmt. Bei empfindlichen Regelstrecken darf KP nur bis zu einem Wert erhöht werden, bei dem sicher noch keine Schwingungen auftreten.

Dämpfung von Überschwingungen

1629

Um Überschwingungen zu dämpfen, kann *PT1* (→ Seite 240) (Tiefpass) eingesetzt werden. Dazu wird der Sollwert XS durch das PT1-Glied gedämpft, bevor er der Reglerfunktion zugeführt wird.

Die Einstellgröße T1 sollte ca. 4...5-mal größer sein als TN (des PID- oder GLR-Reglers).

7.5.3 Funktionsblöcke für Regler

Inhalt

1634

Der nachfolgende Abschnitt beschreibt im Detail die Bausteine, die zum Aufbau von Software-Reglern im **ecomat*mobile***-Gerät bereitgestellt werden. Die Bausteine können auch als Basis für die Entwicklung von eigenen Regelungsfunktionen genutzt werden.

DELAY

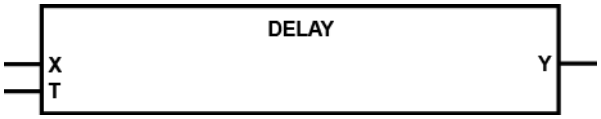
585

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

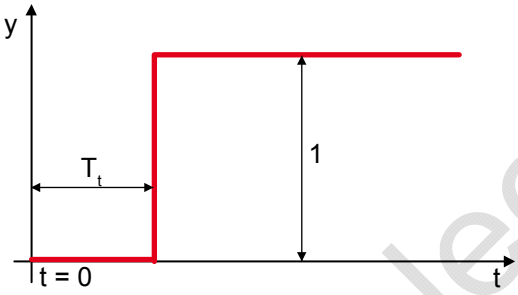
Symbol in CoDeSys:



Beschreibung

588

DELAY verzögert die Ausgabe des Eingangswertes um die Zeit T (Totzeit-Glied).



Grafik: Zeitlicher Verlauf von DELAY

HINWEIS

Damit der FB einwandfrei arbeitet, muss er in jedem Zyklus aufgerufen werden.

Parameter der Eingänge

589

Parameter	Datentyp	Beschreibung
X	WORD	Eingangswert
T	TIME	Verzögerungszeit (Totzeit)

Parameter der Ausgänge

590

Parameter	Datentyp	Beschreibung
Y	WORD	Eingangswert, verzögert um die Zeit T

PT1

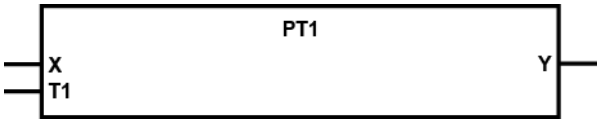
338

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1071

Symbol in CoDeSys:



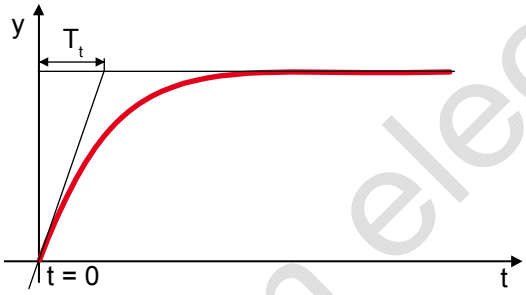
Beschreibung

341

PT1 organisiert eine Regelstrecke mit Verzögerung 1. Ordnung.

Bei der Funktion handelt es sich um eine proportionale Regelstrecke mit Verzögerung. Sie wird z.B. zur Bildung von Rampen bei Einsatz der PWM-Funktionen genutzt.

Die Ausgangsvariable Y des Tiefpassfilters hat folgenden zeitlichen Verlauf (Einheitssprungfunktion):



Grafik: Zeitlicher Verlauf bei PT1

Parameter der Eingänge

342

Parameter	Datentyp	Beschreibung
X	INT	Eingangswert
T1	TIME	Verzögerungszeit (Zeitkonstante)

Parameter der Ausgänge

343

Parameter	Datentyp	Beschreibung
Y	INT	Ausgangsvariable

PID1

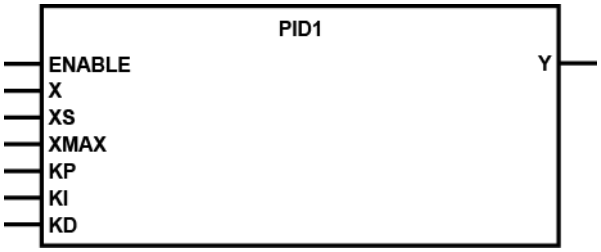
351

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1071

Symbol in CoDeSys:



Beschreibung

354

PID1 organisiert einen PID-Regler.

Die Änderung der Stellgröße eines PID-Reglers setzt sich aus einem proportionalen, integralen und differentialen Anteil zusammen. Die Stellgröße ändert sich zunächst um einen von der Änderungsgeschwindigkeit der Eingangsgröße abhängigen Betrag (D-Anteil). Nach Ablauf der Vorhaltezeit geht die Stellgröße auf den dem Proportionalbereich entsprechenden Wert zurück und ändert sich dann entsprechend der Nachstellzeit.

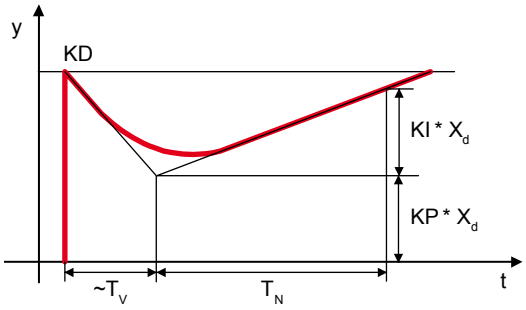
HINWEIS

Die Stellgröße Y ist bereits auf die PWM-Funktion normiert (RELOAD-Wert = 65 535). Beachten Sie dabei die umgekehrte Logik:
65 535 = minimaler Wert
0 = maximaler Wert.

Beachten Sie, dass die Eingangsgrößen KI und KD zykluszeitabhängig sind. Um ein stabiles, reproduzierbares Regelverhalten zu bekommen, sollte der FB zeitgesteuert aufgerufen werden.

Wenn X > XS, dann wird die Stellgröße erhöht.
Wenn X < XS, dann wird die Stellgröße reduziert.

Die Stellgröße Y hat folgenden zeitlichen Verlauf:



Grafik: Typische Sprungantwort eines PID-Reglers

Parameter der Eingänge

355

Parameter	Datentyp	Beschreibung
X	WORD	Istwert
XS	WORD	Sollwert
XMAX	WORD	Maximalwert des Sollwertes
KP	BYTE	Konstante des Proportional-Anteils
KI	BYTE	Integral-Anteil
KD	BYTE	Proportionalanteil des Differential-Anteils

Parameter der Ausgänge

356

Parameter	Datentyp	Beschreibung
Y	WORD	Stellgröße

Einstellempfehlung

357

KP = 50
KI = 30
KD = 5

Bei den oben angegebenen Werten arbeitet der Regler sehr schnell und stabil. Der Regler schwingt bei dieser Einstellung nicht.

► Um den Regler zu optimieren, können die Werte anschließend schrittweise verändert werden.

PID2

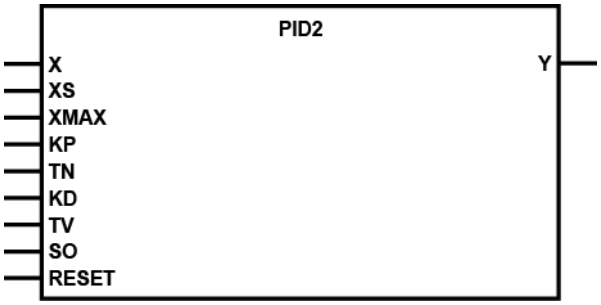
9167

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1071

Symbol in CoDeSys:



Beschreibung

347

PID2 organisiert einen PID-Regler mit Selbstoptimierung.

Die Änderung der Stellgröße eines PID-Reglers setzt sich aus einem proportionalen, integralen und differentialen Anteil zusammen. Die Stellgröße ändert sich zunächst um einen von der Änderungsgeschwindigkeit der Eingangsgröße abhängigen Betrag (**D**ifferential-Anteil). Nach Ablauf der Vorhaltezeit TV geht die Stellgröße auf den dem Proportionalbereich entsprechenden Wert zurück und ändert sich dann entsprechend der Nachstellzeit TN.

Die an den Eingängen KP und KD eingegebenen Werte werden intern durch 10 geteilt. Damit kann eine feinere Abstufung erreicht werden (z.B: KP = 17, das entspricht 1,7).

HINWEIS

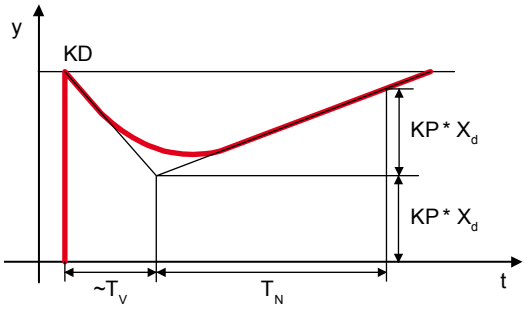
Die Stellgröße Y ist bereits auf die PWM-Funktion normiert (RELOAD-Wert = 65 535). Beachten Sie dabei die umgekehrte Logik:
65 535 = minimaler Wert
0 = maximaler Wert.

Beachten Sie, dass die Eingangsgröße KD zykluszeitabhängig ist. Um ein stabiles, reproduzierbares Regelverhalten zu bekommen, sollte der FB zeitgesteuert aufgerufen werden.

Wenn X > XS, dann wird die Stellgröße erhöht.
Wenn X < XS, dann wird die Stellgröße reduziert.

Eine Führungsgröße wird intern zur Stellgröße hinzuaddiert:
Y = Y + 65 536 - (XS / XMAX * 65 536).

Die Stellgröße Y hat folgenden zeitlichen Verlauf.



Grafik: Typische Sprungantwort eines PID-Reglers

Parameter der Eingänge

348

Parameter	Datentyp	Beschreibung
X	WORD	Istwert
XS	WORD	Sollwert
XMAX	WORD	Maximalwert des Sollwertes
KP	BYTE	Konstante des Proportional-Anteils (/10)
TN	TIME	Nachstellzeit (Integral-Anteil)
KD	BYTE	Proportionalanteil des Differential-Anteils (/10)
TV	TIME	Vorhaltezeit (Differential-Anteil)
SO	BOOL	Selbstoptimierung
RESET	BOOL	Reset

Parameter der Ausgänge

349

Parameter	Datentyp	Beschreibung
Y	WORD	Stellgröße

Einstellempfehlung

9127
350

► TN gemäß des Zeitverhaltens der Strecke wählen
(schnelle Strecke = kleines TN, träge Strecke = großes TN)

► KP langsam, schrittweise erhöhen bis zu einem Wert, bei dem sicher noch kein Schwingen auftritt.

► TN bei Bedarf nachjustieren

► Nur bei Bedarf D-Anteil hinzufügen:
TV ca. 2...10-mal kleiner als TN wählen.
KD etwa gleich groß wie KP wählen.

Beachten Sie, dass die maximale Regelabweichung + 127 beträgt. Für ein gutes Regelverhalten sollte dieser Bereich einerseits nicht überschritten, andererseits aber möglichst ausgenutzt werden.

Durch den Funktionseingang SO (Selbstoptimierung) werden die Regeleigenschaften deutlich verbessert. Voraussetzungen, dass die gewünschten Eigenschaften erreicht werden, sind:

- Der Regler wird mit I-Anteil betrieben (TN ≥ 50 ms)
- Die Parameter KP und insbesondere TN sind bereits gut an die reale Regelstrecke angepasst.
- Der Regelbereich (X - XS) von ± 127 wird ausgenutzt (bei Bedarf durch Multiplikation von X, XS und XMAX den Regelbereich vergrößern).

► Nach Abschluss der Parametereinstellungen kann SO = TRUE gesetzt werden.

> Die Regeleigenschaften werden dann merklich verbessert. Insbesondere Überschwingungen werden reduziert.

GLR

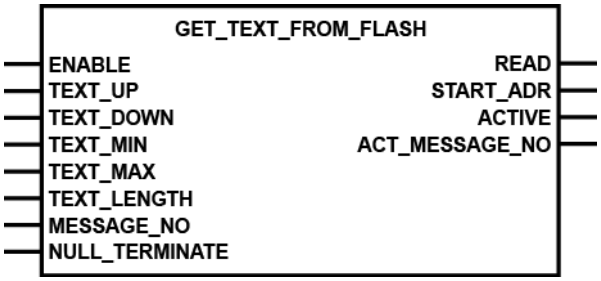
531

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
- SmartController: CR25nn
- PDM360smart: CR1071

Symbol in CoDeSys:



Beschreibung

534

GLR organisiert einen Gleichlauf-Regler.

Bei dem Gleichlaufregler handelt es sich um einen Regler mit PID-Verhalten.

Die am Funktionseingang KP und KD eingegebenen Werte werden intern durch 10 geteilt. Damit kann eine feinere Abstufung erreicht werden (z.B: KP = 17, das entspricht 1,7).

Die Stellgröße bezüglich des größeren Istwerts wird jeweils erhöht.
Die Stellgröße bezüglich des kleineren Istwerts entspricht der Führungsgröße.
Führungsgröße = 65 536 - (XS / XMAX * 65 536).

HINWEIS

Die Stellgrößen Y1 und Y2 sind bereits auf die PWM-Funktion normiert (RELOAD-Wert = 65 535).
Beachten Sie dabei die umgekehrte Logik:
65 535 = minimaler Wert
0 = maximaler Wert.

Beachten Sie, dass die Eingangsgröße KD zykluszeitabhängig ist. Um ein stabiles, reproduzierbares Regelverhalten zu bekommen, sollte die Funktion zeitgesteuert aufgerufen werden.

Parameter der Eingänge

535

Parameter	Datentyp	Beschreibung
X1	WORD	Istwert Kanal 1
X2	WORD	Istwert Kanal 2
XS	WORD	Sollwert = Führungsgröße
XMAX	WORD	Maximalwert des Sollwertes
KP	BYTE	Konstante des Poportional-Anteils (/10)
TN	TIME	Nachstellzeit (Integral-Anteil)
KD	BYTE	Proportionalanteil des Differential-Anteils (/10)
TV	TIME	Vorhaltezeit (Differential-Anteil)
RESET	BOOL	Reset

Parameter der Ausgänge

536

Parameter	Datentyp	Beschreibung
Y1	WORD	Stellgröße Kanal 1
Y2	WORD	Stellgröße Kanal 2

8 Kommunikation über Schnittstellen

Inhalt

8602

Hier zeigen wir Ihnen Funktionen, die der Kommunikation über Schnittstellen dienen.

8.1 Nutzung der seriellen Schnittstelle

Inhalt

1600

HINWEIS

Grundsätzlich steht die serielle Schnittstelle dem Anwender nicht zur Verfügung, da sie für den Programm-Download und das Debugging genutzt wird.

Setzt der Anwender das Systemmerker-Bit SERIAL_MODE auf TRUE, dann kann die Schnittstelle frei genutzt werden. Der Programm-Download und das Debugging sind dann jedoch nur noch über die CAN-Schnittstelle möglich.
Für CRxx32 gilt: Ein Debugging der Applikations-Software ist dann nur noch über alle 4 CAN-Schnittstellen oder über USB möglich.

Mit den folgend aufgeführten Bausteinen kann die serielle Schnittstelle im Applikations-Programm genutzt werden.

8.1.1 SERIAL_SETUP

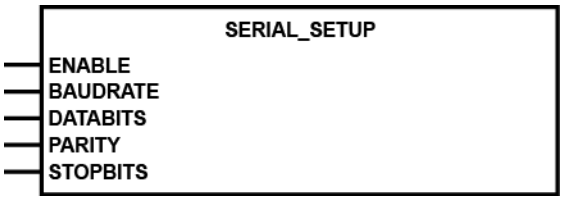
302

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

305

SERIAL_SETUP initialisiert die serielle RS232-Schnittstelle.

Der FB setzt die serielle Schnittstelle auf die angegebenen Parameter. Mit dem Eingang ENABLE wird der FB für einen Zyklus aktiviert.

Die SERIAL-Bausteine bilden die Grundlage für die Erstellung eines anwenderspezifischen Protokolls für die serielle Schnittstelle.

HINWEIS

Grundsätzlich steht die serielle Schnittstelle dem Anwender nicht zur Verfügung, da sie für den Programmdownload und das Debugging genutzt wird.

Setzt der Anwender das Systemmerkerbit SERIAL_MODE auf TRUE, dann kann die Schnittstelle frei genutzt werden. Der Programm-Download und das Debugging sind dann jedoch nur noch über die CAN-Schnittstelle möglich.
Für CRnn32 gilt: Ein Debugging der Applikations-Software ist dann nur noch über alle 4 CAN-Schnittstellen oder über USB möglich.

HINWEIS

Ein Teil der Ein- und Ausgänge des SafetyControllers ist für sichere Applikationen zugelassen ...
- bis zu PL d nach ISO 13849,
- bis zu SIL CL 2 nach IEC 62061.

Voraussetzung dafür ist, dass die Ein- und Ausgänge des SafetyController (wie in Kapitel *Konfigurationen* (→ Seite 14) beschrieben) verschaltet und durch das Applikations-Programm ausgewertet werden.

Parameter der Eingänge

306

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (nur 1 Zyklus lang): Schnittstelle wird initialisiert FALSE: im weiteren Programmablauf
BAUDRATE	WORD	Baud-Rate (zulässige Werte = 9 600, 19 200, 28 800, (57 600)) Voreinstellwert → Datenblatt
DATABITS	BYTE	Daten-Bits (zulässige Werte: 7 oder 8) Voreinstellwert = 8
PARITY	BYTE	Parität (zulässige Werte: 0=keine, 1=gerade, 2=ungerade) Voreinstellwert = 0
STOPBITS	BYTE	Stopp-Bits (zulässige Werte: 1 oder 2) Voreinstellwert = 1

8.1.2 SERIAL_TX

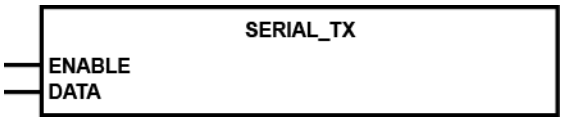
296

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

299

SERIAL_TX überträgt ein Datenbyte über die serielle RS232-Schnittstelle.

Mit dem Eingang ENABLE kann die Übertragung freigegeben oder gesperrt werden.

Die SERIAL-Bausteine bilden die Grundlage für die Erstellung eines anwenderspezifischen Protokolls für die serielle Schnittstelle.

HINWEIS

Grundsätzlich steht die serielle Schnittstelle dem Anwender nicht zur Verfügung, da sie für den Programmdownload und das Debugging genutzt wird.

Setzt der Anwender das Systemmerkerbit SERIAL_MODE auf TRUE, dann kann die Schnittstelle frei genutzt werden. Der Programm-Download und das Debugging sind dann jedoch nur noch über die CAN-Schnittstelle möglich.
Für CRnn32 gilt: Ein Debugging der Applikations-Software ist dann nur noch über alle 4 CAN-Schnittstellen oder über USB möglich.

Parameter der Eingänge

300

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Übertragung freigegeben FALSE: Übertragung gesperrt
DATA	BYTE	zu übertragendes Byte

8.1.3 SERIAL_RX

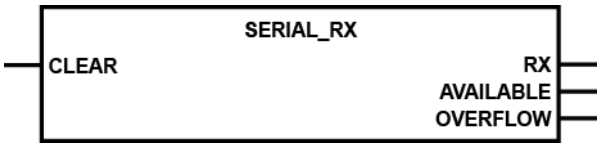
308

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

311

SERIAL_RX liest mit jedem Aufruf ein empfangenes Datenbyte aus dem seriellen Empfangspuffer aus.

Anschließend wird der Wert von AVAILABLE um 1 dekrementiert.

Gehen mehr als 1000 Datenbytes ein, läuft der Puffer über und es gehen Daten verloren. Dieses wird durch das Bit OVERFLOW angezeigt.

Die SERIAL-Bausteine bilden die Grundlage für die Erstellung eines anwenderspezifischen Protokolls für die serielle Schnittstelle.

HINWEIS

Grundsätzlich steht die serielle Schnittstelle dem Anwender nicht zur Verfügung, da sie für den Programmdownload und das Debugging genutzt wird.

Setzt der Anwender das Systemmerkerbit SERIAL_MODE auf TRUE, dann kann die Schnittstelle frei genutzt werden. Der Programm-Download und das Debugging sind dann jedoch nur noch über die CAN-Schnittstelle möglich.
Für CRnn32 gilt: Ein Debugging der Applikations-Software ist dann nur noch über alle 4 CAN-Schnittstellen oder über USB möglich.

Parameter der Eingänge

312

Parameter	Datentyp	Beschreibung
CLEAR	BOOL	TRUE: Empfangspuffer wird gelöscht FALSE: diese Funktion wird nicht ausgeführt

Parameter der Ausgänge

313

Parameter	Datentyp	Beschreibung
RX	BYTE	empfangene Byte-Daten aus dem Empfangspuffer
AVAILABLE	WORD	Anzahl der empfangenen Datenbytes 0 = keine gültigen Daten vorhanden
OVERFLOW	BOOL	Überlauf des Datenpuffers, Datenverlust!

Beispiel:

Es werden 3 Bytes empfangen:

1. Aufruf von SERIAL_RX
   1 gültiger Wert am Ausgang RX
   → AVAILABLE = 3
2. Aufruf von SERIAL_RX
   1 gültiger Wert am Ausgang RX
   → AVAILABLE = 2
3. Aufruf von SERIAL_RX
   1 gültiger Wert am Ausgang RX
   → AVAILABLE = 1
4. Aufruf von SERIAL_RX
   ungültiger Wert am Ausgang RX
   → AVAILABLE = 0

Wenn AVAILABLE = 0 ist, kann der Baustein im Programmablauf übersprungen werden.

8.1.4 SERIAL_PENDING

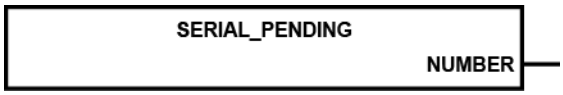
314

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

317

SERIAL_PENDING ermittelt die Anzahl der im seriellen Empfangspuffer gespeicherten Datenbytes.

Im Gegensatz zu *SERIAL_RX* (→ Seite 252) bleibt der Inhalt des Puffers nach Aufruf dieser Funktion unverändert.

Die SERIAL-Bausteine bilden die Grundlage für die Erstellung eines anwenderspezifischen Protokolls für die serielle Schnittstelle.

HINWEIS

Grundsätzlich steht die serielle Schnittstelle dem Anwender nicht zur Verfügung, da sie für den Programmdownload und das Debugging genutzt wird.

Setzt der Anwender das Systemmerkerbit SERIAL_MODE auf TRUE, dann kann die Schnittstelle frei genutzt werden. Der Programm-Download und das Debugging sind dann jedoch nur noch über die CAN-Schnittstelle möglich.
Für CRnn32 gilt: Ein Debugging der Applikations-Software ist dann nur noch über alle 4 CAN-Schnittstellen oder über USB möglich.

Parameter der Ausgänge

319

Parameter	Datentyp	Beschreibung
NUMBER	WORD	Anzahl der empfangenen Datenbytes

9 Daten verwalten

Inhalt

8606

Hier zeigen wir Ihnen Funktionen, mit denen Sie Daten im Gerät lesen und verarbeiten können.

9.1 Software-Reset

Inhalt

1594

Hiermit kann die Steuerung per Kommando im Applikations-Programm neu gestartet werden.

9.1.1 SOFTRESET

260

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:

Beschreibung

263

SOFTRESET führt einen kompletten Neustart des Controllers aus.

Die Funktion kann z.B. in Verbindung mit CANopen genutzt werden, wenn ein Node-Reset ausgeführt werden soll. Das Verhalten des Controllers nach einem SOFTRESET entspricht dem nach Aus- und Einschalten der Versorgungsspannung.

HINWEIS

Bei einer laufenden Kommunikation muss die lange Reset-Phase beachtet werden, da andernfalls Guarding-Fehler gemeldet werden.

Parameter der Eingänge

264

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv

9.2 Systemzeit lesen / schreiben

Inhalt

1601

Mit folgenden Bausteinen der **ifm electronic gmbh** können Sie die kontinuierlich laufende Systemzeit des Controllers lesen und im Applikations-Programm auswerten oder bei Bedarf ändern.

9.2.1 TIMER_READ

236

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:

Beschreibung

239

TIMER_READ liest die aktuelle Systemzeit aus.

Mit Anlegen der Versorgungsspannung bildet der Controller einen Zeittakt, der in einem Register aufwärts gezählt wird. Dieses Register kann mittels des Funktionsaufrufes ausgelesen und z.B. zur Zeitmessung genutzt werden.

HINWEIS

Der System-Timer läuft maximal bis FFFF FFFF$_{16}$ (entspricht ca. 49,7 Tage) und startet anschließend wieder bei 0.

Parameter der Ausgänge

241

Parameter	Datentyp	Beschreibung
T	TIME	Aktuelle Systemzeit (Auflösung [ms])

9.2.2 TIMER_READ_US

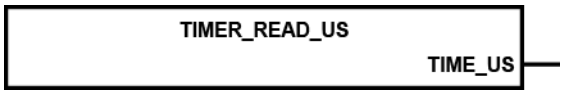
657

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

660

TIMER_READ_US liest die aktuelle Systemzeit in [µs] aus.

Mit Anlegen der Versorgungsspannung bildet das Gerät einen Zeittakt, der in einem Register aufwärts gezählt wird. Dieses Register kann mittels des FB-Aufrufes ausgelesen werden und z.B. zur Zeitmessung genutzt werden.

Info

Der System-Timer läuft maximal bis zum Zählerwert 4294967295 (µs) und startet anschließend wieder bei 0.

4 294 967 295 µs = 4 295 s = 71,6 min = 1,2 h

Parameter der Ausgänge

662

Parameter	Datentyp	Beschreibung
TIME_US	DWORD	Aktuelle Systemzeit (Auflösung [µs])

9.3 Gerätetemperatur auslesen

Inhalt

2364

9.3.1 TEMPERATURE

2216

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- ClassicController: CR0032, CR0033
- ExtendedController: CR0232, CR0233
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:

Beschreibung

2365

TEMPERATURE liest die aktuelle Temperatur im Gerät aus.

Der FB kann zyklisch aufgerufen werden und zeigt am Ausgang die aktuelle Gerätetemperatur an.

Parameter der Eingänge

2366

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv

Parameter der Ausgänge

2367

Parameter	Datentyp	Beschreibung
TEMPERATURE	INT	Aktuelle Geräteinnentemperatur [°C]

9.4 Daten im Speicher sichern, lesen und wandeln

Inhalt

1595

9.4.1 Manuelle Datensicherung

Inhalt

1597

Neben der Möglichkeit, die Daten automatisch zu sichern, können über FB-Aufrufe Anwenderdaten manuell in integrierte Speicher gesichert und von dort wieder gelesen werden.

Je nach Gerät stehen folgende Speicher zur Verfügung:

Speicher / Gerät	Eigenschaften
EEPROM-Speicher Für folgende Geräte verfügbar: - CabinetController: CR0301, CR0302 - Platinensteuerung: CS0015 - SmartController: CR25nn	Langsames Schreiben und Lesen. Begrenzte Schreib-/Lesehäufigkeit. Beliebige Speicherbereiche wählbar. Daten sichern mit E2WRITE. Daten lesen mit E2READ
FRAM-Speicher [1]) Für folgende Geräte verfügbar: - CabinetController: CR0303 - ClassicController: CR0020, CR0032, CR0033, CR0505 - ExtendedController: CR0200, CR0232, CR0233 - SafetyController: CR7nnn - PDM360smart: CR1070, CR1071	Schnelles Schreiben und Lesen. Unbegrenzte Schreib-/Lesehäufigkeit. Beliebige Speicherbereiche wählbar. Daten sichern mit FRAMWRITE. Daten lesen mit FRAMREAD.
Flash-Speicher Für alle Geräte	Schnelles Schreiben und Lesen. Begrenzte Schreib-/Lesehäufigkeit. Nur zum Speichern großer Datenmengen sinnvoll einsetzbar. Vor dem erneuten Schreiben muss Speicherinhalt gelöscht werden. Daten sichern mit FLASHWRITE. Daten lesen mit FLASHREAD.

[1]) FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.

Info

Der Programmierer kann sich anhand der Speicheraufteilung (→ Datenblatt oder Betriebsanleitung) darüber informieren, welcher Speicherbereich frei zur Verfügung steht.

GET_TEXT_FROM_FLASH

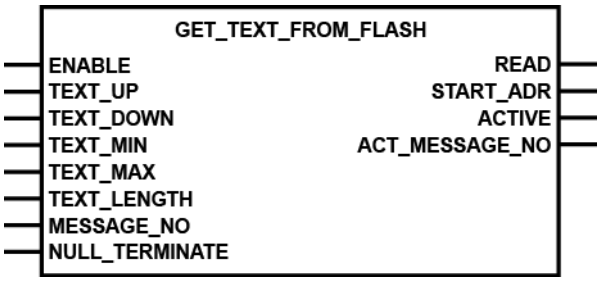
3196

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_PDMsmart_UTIL_Vxxyyzz.Lib`

Für folgende Geräte verfügbar:
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

3301

GET_TEXT_FROM_FLASH steuert *FLASHREAD* (→ Seite 268) oder *FRAMREAD* (→ Seite 272) an, um Texte vom Typ STRING direkt auszulesen.

Im Gegensatz zum PDM360 und PDM360compact hat das PDM360smart kein Dateisystem. Daher bieten sich hier Flash-Speicher oder FRAM-Speicher [1]) zum Ablegen von Textmeldungen an. Zum Auslesen dieser Speicherbereiche werden FLASHREAD oder FRAMREAD benötigt.

Um nun gezielt einen oder auch mehrere Texte auszulesen, muss die Startadresse des Textes im Speicher berechnet werden. Diese Berechnung und auch das Setzen/Rücksetzen des ENABLE-Eingangs erfolgt in GET_TEXT_FROM_FLASH.

Die Organisation der Texte im Speicher muss nach folgenden Regeln erfolgen:

Textlänge
Die Textlänge sollte für alle Texte gleich sein und ist wegen der Displaygröße des PDM360smart auf jeweils maximal 20 Zeichen begrenzt.

Erstellung der Texte
Die Texte sollten mit einem Tabellenkalkulationsprogramm (z.B. Excel) erstellt und im CSV-Format gespeichert werden. Diese CSV-Datei kann mit dem **ifm**-Downloader direkt in den gewünschten Speicherbereich geladen werden.
→ auf der **ecomat*mobile***-DVD "Software, tools and documentation":
- DE: Beschreibung "`Batchverarbeitung_ifm.pdf`" (→ `\doku_d`),
- UK: Beschreibung "`Batchmode_ifm.pdf`" (→ `\doku_gb`).

Ein STRING wird automatisch vom Programmiersystem mit einem NULL-Byte abgeschlossen. Daher belegt ein Text mit 20 Zeichen 21 Bytes im Speicher. Der FB berücksichtigt das bei der Berechnung. Bei einer Textlänge von 20 Zeichen können 16394/21 = 780 Texte im Flash-Speicher gespeichert werden.

[1]) FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.

Parameter der Eingänge

3302

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
TEXT_UP	BOOL	Flanke FALSE ⇨ TRUE: nächsten Text lesen
TEXT_DOWN	BOOL	Flanke FALSE ⇨ TRUE: vorherigen Text lesen
TEXT_MIN	WORD	untere Grenze für MESSAGE_NO
TEXT_MAX	WORD	obere Grenze für MESSAGE_NO
TEXT_LENGTH	BYTE	Textlänge
MESSAGE_NO	WORD	Textnummer
NULL_TERMINATE	BOOL	TRUE: String hat Null-Terminierung FALSE: String hat keine Null-Terminierung

Parameter der Ausgänge

3303

Parameter	Datentyp	Beschreibung
READ	BOOL	Kommando Lesen ► Dieses Signal auf den Eingang ENABLE von FLASHREAD oder FRAMREAD legen!
START_ADR	WORD	berechnete Startadresse ► Dieses Signal auf den Eingang SCR von FLASHREAD oder FRAMREAD legen!
ACTIV	BOOL	ist TRUE, wenn Eingang ENABLE = 1
ACT_MESSAGE_NO	WORD	aktuelle Textnummer

MEMCPY

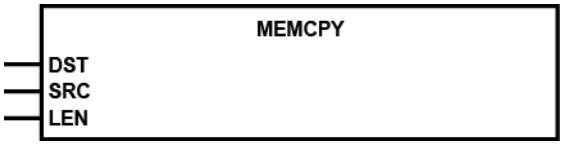
409

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

412

MEMCPY ermöglicht das Schreiben und Lesen unterschiedlicher Datentypen direkt in den Speicher.

Der FB schreibt den Inhalt der Adresse von SRC an die Adresse DST.

► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

> Dabei werden genau so viele Bytes übertragen, wie diese unter LEN angegeben wurden. Dadurch ist es auch möglich, genau ein Byte einer Wortdatei zu übertragen.

Parameter der Eingänge

413

Parameter	Datentyp	Beschreibung
DST	DWORD	Adresse der Zielvariablen ► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
SRC	DWORD	Adresse der Quellvariablen ► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
LEN	WORD	Anzahl der Datenbytes

FLASHWRITE

555

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:

Beschreibung

558

⚠ WARNUNG

Gefahr durch unkontrollierten Prozessablauf!

Der Zustand der Ein-/Ausgänge wird während der Ausführung von FLASHWRITE "eingefroren".

► Diesen Funktionsblock nicht bei laufender Maschine ausführen!

FLASHWRITE ermöglicht das Schreiben unterschiedlicher Datentypen direkt in den Flash-Speicher.

Der FB schreibt den Inhalt der Adresse SRC in den Flash-Speicher. Dabei werden genau so viele Bytes übertragen, wie diese unter LEN angegeben sind.

► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Bevor der Speicher erneut beschrieben wird, muss vorher ein Löschvorgang durchgeführt werden. Dies geschieht mit dem Beschreiben der Adresse "0" mit beliebigem Inhalt.

Info

Mit diesem FB sollen während der Inbetriebnahme große Datenmengen gesichert werden, auf die im Prozess nur lesend zugegriffen wird.

Parameter der Eingänge

559

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
DST	WORD CR0301, CR0302, CS0015: INT	Relative Anfangsadresse im Speicher (→ Tabelle unten)
LEN	WORD CR0301, CR0302, CS0015: INT	Anzahl der Datenbytes (→ Tabelle unten)
SRC	DWORD CR0301, CR0302, CS0015: DINT	Adresse der Quellvariablen ► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Gerät	zulässige Werte für DST dez | hex		zulässige Werte für LEN dez | hex	
CabinetController: CR030n	0...16 383	0...3FFF	0...16 383	0...3FFF
ClassicController: CR0020, CR0505	0...65 535	0...FFFF	0...65 535	0...FFFF
ExtendedController: CR0200	0...65 535	0...FFFF	0...65 535	0...FFFF
Platinensteuerung: CS0015	0...16 383	0...3FFF	0...16 383	0...3FFF
SafetyController: CR7021, CR7201, CR7506	0...65 535	0...FFFF	0...65 535	0...FFFF
SmartController: CR25nn	0...65 535	0...FFFF	0...65 535	0...FFFF
PDM360smart: CR1070, CR1071	0...16 383	0...3FFF	0...16 384	0...4000

FLASHREAD

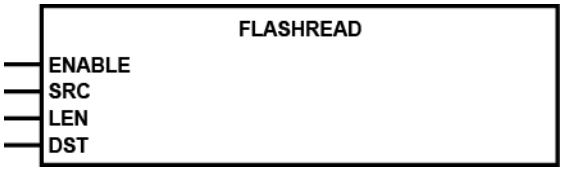
561

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

564

FLASHREAD ermöglicht das Lesen unterschiedlicher Datentypen direkt aus dem Flash-Speicher.

Der FB liest den Inhalt ab der Adresse von SRC aus dem Flash-Speicher. Dabei werden genau so viele Bytes übertragen, wie diese unter LEN angegeben sind.

► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Parameter der Eingänge

565

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
SRC	WORD CR0301, CR0302, CS0015: INT	Relative Anfangsadresse im Speicher (→ Tabelle unten)
LEN	WORD CR0301, CR0302, CS0015: INT	Anzahl der Datenbytes (→ Tabelle unten)
DST	DWORD CR0301, CR0302, CS0015: DINT	Adresse der Zielvariablen ► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Gerät	zulässige Werte für SRC dez | hex		zulässige Werte für LEN dez | hex	
CabinetController: CR030n	0...16 383	0...3FFF	0...16 383	0...3FFF
ClassicController: CR0020, CR0505	0...65 535	0...FFFF	0...65 535	0...FFFF
ExtendedController: CR0200	0...65 535	0...FFFF	0...65 535	0...FFFF
Platinensteuerung: CS0015	0...16 383	0...3FFF	0...16 383	0...3FFF
SafetyController: CR7021, CR7201, CR7506	0...65 535	0...FFFF	0...65 535	0...FFFF
SmartController: CR25nn	0...16 383	0...3FFF	0...16 383	0...3FFF
PDM360smart: CR1070, CR1071	0...16 383	0...3FFF	0...16 384	0...4000

FRAMWRITE

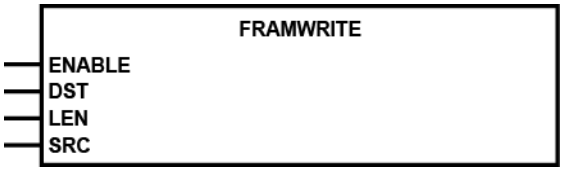
543

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR0303
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- SafetyController: CR7nnn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

546

FRAMWRITE ermöglicht das schnelle Schreiben unterschiedlicher Datentypen direkt in den FRAM-Speicher [1]).

Der FB schreibt den Inhalt der Adresse SRC in den spannungsausfallsicheren FRAM-Speicher. Dabei werden genau so viele Bytes übertragen, wie diese über LEN angegeben sind.

► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Der FRAM-Speicher kann in mehreren unabhängigen Teilsegmenten beschrieben werden. Die Überwachung der Speichersegmente muss im Applikationsprogramm erfolgen.

[1]) FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.

Parameter der Eingänge

547

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
DST	WORD CR0303: INT	Relative Anfangsadresse im Speicher (→ Tabelle unten)
LEN	WORD CR0303: INT	Anzahl der Datenbytes (→ Tabelle unten)
SRC	DWORD CR0303: DINT	Adresse der Quellvariablen ► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Gerät	zulässige Werte für DST dez | hex		zulässige Werte für LEN dez | hex	
CabinetController: CR0303	512...2 047	200...7FF	0...128	0...80
ClassicController: CR0020, CR0505	0...1 023	0...3FF		
ExtendedController: CR0200	0...1 023	0...3FF		
SafetyController: CR7021, CR7201, CR7506	0...1 023	0...3FF		
PDM360smart: CR1070, CR1071	512...2 047	200...7FF	0...128	0...80

FRAMREAD

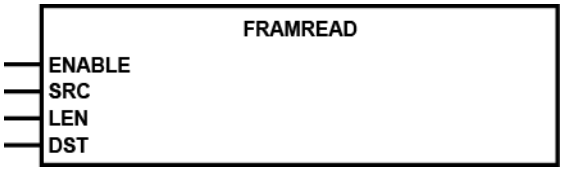
549

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR0303
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- SafetyController: CR7nnn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

552

FRAMREAD ermöglicht das schnelle Lesen unterschiedlicher Datentypen direkt aus dem FRAM-Speicher [1]).

Der FB liest den Inhalt ab der Adresse von SRC aus dem FRAM-Speicher. Dabei werden genau so viele Bytes übertragen, wie diese unter LEN angegeben sind.

► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Der FRAM-Speicher kann in mehreren unabhängigen Teilsegmenten ausgelesen werden. Die Überwachung der Speichersegmente muss im Applikationsprogramm erfolgen.

[1]) FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.

Parameter der Eingänge

553

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
SRC	WORD CR0303: INT	Relative Anfangsadresse im Speicher (→ Tabelle unten)
LEN	WORD CR0303: INT	Anzahl der Datenbytes (→ Tabelle unten)
DST	DWORD CR0303: DINT	Adresse der Zielvariablen ► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Gerät	zulässige Werte für SRC dez | hex		zulässige Werte für LEN dez | hex	
CabinetController: CR0303	0...2 047	0...7FF	0...128	0...80
ClassicController: CR0020, CR0505	0...1 023	0...3FF		
ExtendedController: CR0200	0...1 023	0...3FF		
SafetyController: CR7021, CR7201, CR7506	0...1 023	0...3FF		
PDM360smart: CR1070, CR1071	0...2 047	0...7FF	0...128	0...80

9.5 Datenzugriff und Datenprüfung

Inhalt

1598

Die Bausteine in diesem Kapitel steuern den Datenzugriff und ermöglichen ein Prüfen der Daten.

9.5.1 SET_IDENTITY

284

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:

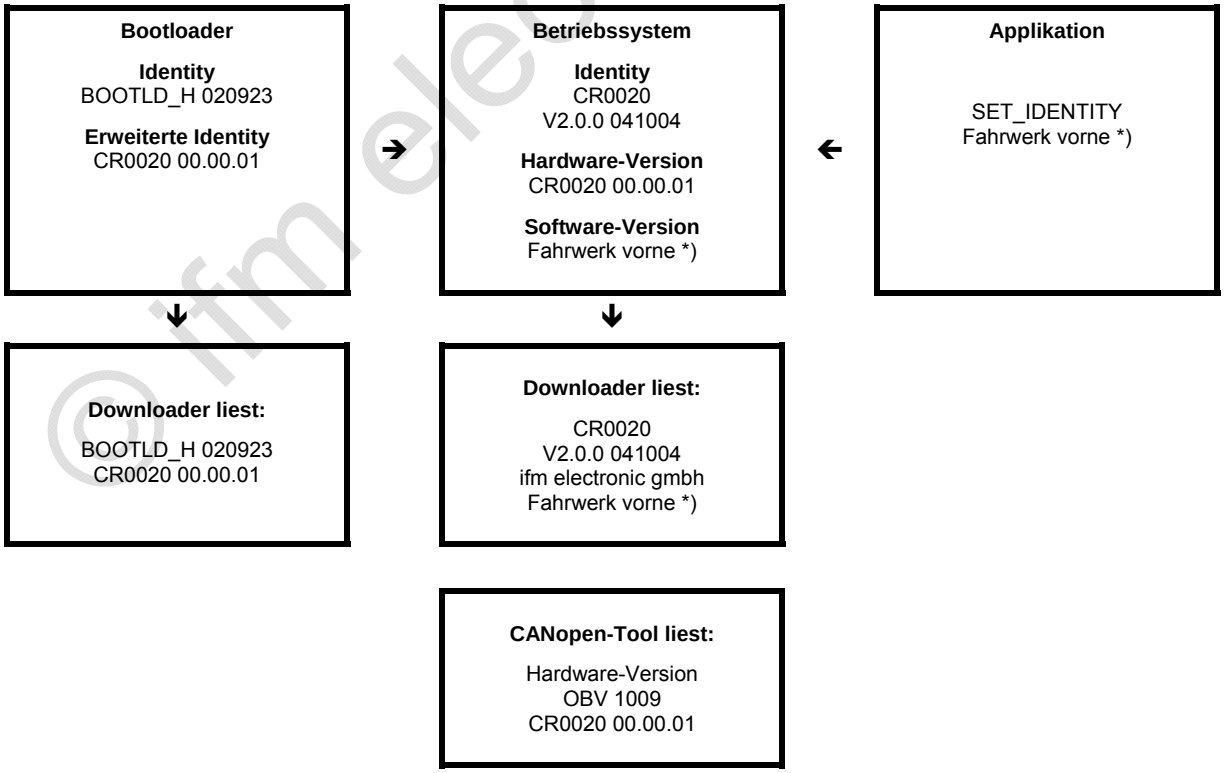
Beschreibung

287

SET_IDENTITY setzt eine applikationsspezifische Programmkennung.

Mit dem FB kann durch das Applikations-Programm eine Programmkennung erzeugt werden. Diese Kennung kann zur Identifizierung des geladenen Programms über das Software-Tool DOWNLOADER.EXE als Software-Version ausgelesen werden.

Die nachfolgende Grafik zeigt die Zusammenhänge der unterschiedlichen Kennungen, wie sie mit den unterschiedlichen Software-Tools angezeigt werden. (Beispiel: ClassicController CR0020):



*) **HINWEIS:** 'Fahrwerk vorne' steht hier stellvertretend für einen kundenspezifischen Text.

Parameter der Eingänge

288

Parameter	Datentyp	Beschreibung
ID	STRING(80)	Beliebiger String mit einer maximalen Länge von 80 Zeichen

9.5.2 GET_IDENTITY

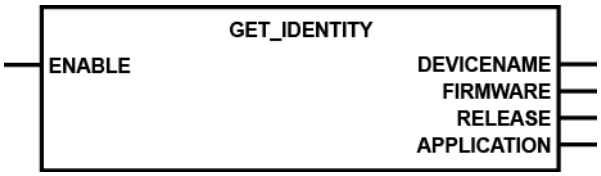
2212

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

2344

GET_IDENTITY liest die im Gerät gespeicherten Geräte- und applikations-spezifischen Kennungen.

Der Name der Applikation kann mit *SET_IDENTITY* (→ Seite 275) verändert werden.

Parameter der Eingänge

2609

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv

Parameter der Ausgänge

2610

Parameter	Datentyp	Beschreibung
DEVICENAME	STRING(31)	Hardware-Name und Version als String von max. 31 Zeichen z.B.: "CR0032 00.00.01"
FIRMWARE	STRING(31)	Name des Laufzeitsystems als String von max. 31 Zeichen z.B.: "CR0032"
RELEASE	STRING(31)	Version und Build des Laufzeitsystems als String von max. 31 Zeichen z.B.: "V00.00.01 071128"
APPLICATION	STRING(79)	Name der Applikation als String von max. 79 Zeichen z.B.: "Crane1704"

9.5.3 SET_PASSWORD

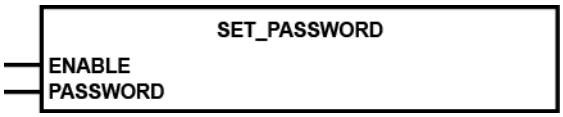
266

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

269

SET_PASSWORD setzt Benutzerkennung für Programm- und Speicher-Upload mit dem DOWNLOADER.

Ist die Benutzerkennung aktiv, kann durch das Software-Tool DOWNLOADER das Applikations-Programm oder der Datenspeicher nur ausgelesen werden, wenn das richtige Password eingegeben wurde.

Wird an den Eingang PASSWORD ein Leer-String (Default-Zustand) übergeben, ist ein Upload der Applikations-Software oder des Datenspeichers jederzeit möglich.

ACHTUNG

Für CR250n, CR0301, CR0302, CS0015 beachten:

Das EEPROM-Speichermodul kann bei Dauerbetrieb dieser Funktion zerstört werden!

► Diesen Baustein nur **einmalig** bei der Initialisierung im ersten Programmzyklus ausführen! Anschließend den Baustein wieder sperren (ENABLE = "FALSE")!

HINWEIS

Beim Laden eines neuen Applikations-Programms wird die Kennung wieder zurückgesetzt.

Parameter der Eingänge

270

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (nur 1 Zyklus lang): Kennung wird gesetzt FALSE: Baustein wird nicht ausgeführt
PASSWORD	STRING(16)	Benutzerkennung (maximale String-Länge 16)

9.5.4 CHECK_DATA

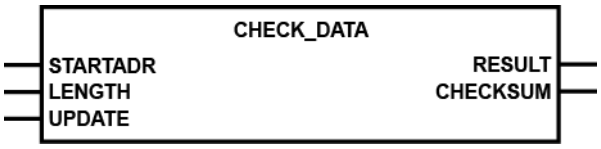
603

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SafetyController: CR7nnn
- SmartController: CR25nn
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

606

CHECK_DATA sichert die Daten im Applikations-Datenspeicher über einen CRC-Code.

Der FB dient dazu, in sicherheitsrelevanten Applikationen einen Bereich des Datenspeichers (mögliche Adressen ab %MW0) auf eine nicht gewollte Datenänderung zu überwachen. Der FB bildet dazu über den angegebenen Datenbereich eine CRC-Checksumme.

Wenn Eingang UPDATE = FALSE und Daten im Speicher sich ungewollt verändern, wird RESULT = FALSE. Das Ergebnis kann dann für weitere Aktionen (z.B. Abschalten der Ausgänge) genutzt werden.

Nur wenn der Eingang UPDATE auf TRUE gesetzt ist, sind Datenänderungen im Speicher (z.B. vom Applikations-Programm oder **ecomat*mobile***-Gerät) zulässig. Der Wert der Prüfsumme wird dann neu berechnet. Der Ausgang RESULT ist wieder permanent TRUE.

► Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

► Zusätzlich die Anzahl der Datenbytes LENGTH (Länge ab der STARTADR) angeben.

HINWEIS

Bei dem FB handelt es sich um eine Sicherheitsfunktion. Dennoch wird durch Einsatz dieses FB der Controller nicht automatisch zur Sicherheitssteuerung. Als Sicherheitssteuerung kann nur eine geprüfte, zugelassene und mit einem speziellen Betriebssystem versehene Steuerung genutzt werden.

Parameter der Eingänge

607

Parameter	Datentyp	Beschreibung
STARTADR	DINT	Startadresse des überwachten Datenspeichers (WORD-Adresse ab %MW0)
LENGTH	WORD	Länge des überwachten Datenspeichers in [Byte]
UPDATE	BOOL	TRUE: Datenänderungen zulässig FALSE: Datenänderungen nicht zulässig

Parameter der Ausgänge

608

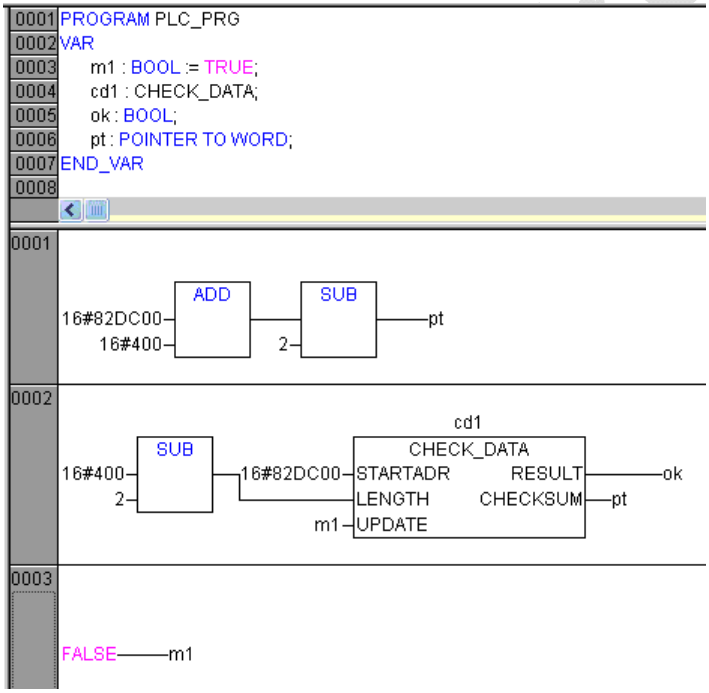
Parameter	Datentyp	Beschreibung
RESULT	BOOL	TRUE: CRC-Checksumme in Ordnung FALSE: CRC-Checksumme fehlerhaft (Daten wurden geändert)
CHECKSUM	WORD	Ergebnis der CRC-Prüfsummenbildung

Beispiel: CHECK_DATA

4168

Im folgenden Beispiel ermittelt das Programm die Prüfsumme und legt sie über den Pointer pt im RAM ab:

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     m1 : BOOL := TRUE;
0004     cd1 : CHECK_DATA;
0005     ok : BOOL;
0006     pt : POINTER TO WORD;
0007 END_VAR
0008
```



HINWEIS: Das hier gezeigte Verfahren ist für den Flash-Speicher nicht geeignet.

10 SPS-Zyklus optimieren

Inhalt

8609

Hier zeigen wir Ihnen Funktionen zum Optimieren des SPS-Zyklus.

10.1 Interrupts verarbeiten

Inhalt

1599

Die SPS arbeitet das gespeicherte Applikations-Programm zyklisch in voller Länge ab. Von z.B. äußeren Ereignissen abhängige Verzweigungen im Programm (= bedingte Sprünge) lassen die Zykluszeit variieren. Für bestimmte Funktionen kann dieses Verhalten nachteilig sein.

Mit Hilfe gezielter Unterbrechungen (= Interrupts) des zyklischen Programmablaufs können zeitkritische Abläufe unabhängig vom Zyklus in festen Zeitrastern oder bei bestimmten Ereignissen aufgerufen werden.

Für SafetyController sind Interrupt-Funktionen grundsätzlich nicht zulässig und deshalb nicht verfügbar.

10.1.1 SET_INTERRUPT_XMS

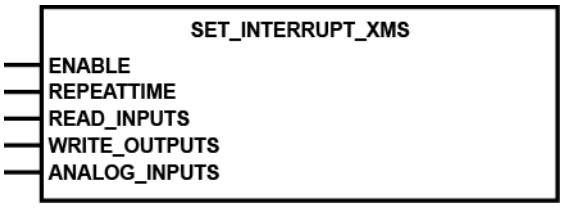
272

Baustein-Typ = Funktionsblock (FB)

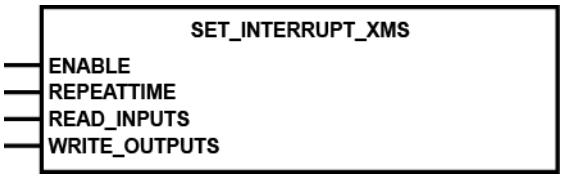
Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0032, CR0033, CR0505
- ExtendedController: CR0200, CR0232, CR0233
- Platinensteuerung: CS0015
- SmartController: CR25nn
- PDM360smart: CR1071

Symbol in CoDeSys:



(nur bei Geräten mit Analog-Kanälen)



(bei Geräten ohne Analog-Kanäle)

Beschreibung

275

SET_INTERRUPT_XMS organisiert das Ausführen eines Programmteils im Intervall von x ms.

In der klassischen SPS ist die Zykluszeit das Maß der Dinge für Echtzeitbetrachtungen. Gegenüber kundenspezifischen Steuerungen ist die SPS damit im Nachteil. Auch ein "Echtzeit-Betriebssystem" ändert nichts an dieser Tatsache, wenn das gesamte Applikationsprogramm in einem einzigen unveränderlichen Block abläuft.

Ein möglicher Lösungsansatz wäre, die Zykluszeit kurz zu halten. Dieser Weg führt oft dazu, die Applikation auf mehrere Steuerungszyklen zu verteilen. Die Programmierung wird dadurch jedoch unübersichtlich und schwierig.

Eine andere Möglichkeit besteht darin, einen bestimmten Programmteil in festen Zeitabständen (alle x ms) unabhängig vom Steuerungszyklus aufzurufen.

Der zeitkritische Teil der Applikation wird vom Anwender in einen Baustein vom Type PROGRAMM (PRG) zusammengefasst. Dieser Baustein wird zur Interrupt-Routine deklariert, indem einmalig (zur Initialisierungszeit) SET_INTERRUPT_XMS aufgerufen wird. Das hat zur Folge, dass dieser Programmteil immer nach Ablauf der REPEATTIME (alle x ms) abgearbeitet wird. Werden Ein- und Ausgänge in diesem Programmteil genutzt, werden diese ebenfalls im festgelegten Takt gelesen oder beschrieben. Über die Eingänge READ_INPUTS, WRITE_OUTPUTS oder ANALOG_INPUTS kann das Lesen oder Schreiben unterbunden werden.

Innerhalb des Programmteils können also alle zeitkritischen Ereignisse bearbeitet werden, indem Eingänge oder globale Variablen verknüpft und Ausgänge beschrieben werden. So können auch Zeitglieder genauer überwacht werden, als es in einem "normalen" Zyklus möglich ist.

HINWEIS

Damit der per Interrupt aufgerufene Programmteil nicht zusätzlich zyklisch aufgerufen wird, sollte er (mit Ausnahme des Initialisierungsaufrufes) im Zyklus übersprungen werden.

Es können mehrere Timer-Interrupt-Blöcke aktiv sein. Der Zeitbedarf der Interrupt-Funktionen muss so berechnet werden, dass alle aufgerufenen Bausteine ausgeführt werden können. Das gilt besonders bei Berechnungen, Gleitkomma-Arithmetik und Regler-Funktionen.

Bitte beachten: Bei einer hohen CAN-Busaktivität kann die eingestellte REPEATTIME schwanken.

HINWEIS

Die Eindeutigkeit der Ein- und Ausgänge im Zyklus wird durch die Interrupt-Routine aufgehoben. Deshalb wird nur ein Teil der Ein- und Ausgänge bedient. Wurden sie im Interrupt-Programm initialisiert, werden folgende Ein- und Ausgänge gelesen oder geschrieben.

Eingänge, digital:

%IX0.0...%IX0.7 (CRnn32)

%IX0.12...%IX0.15, %IX1.4...%IX1.8 (übrige ClassicController, ExtendedController, SafetyController)

%IX0.0, %IX0.8 (SmartController)

IN08...IN11 (CabinetController)

IN0...IN3 (Platinensteuerung)

Eingänge, analog:

%IX0.0...%IX0.7 (CRnn32)

alle Kanäle (Auswahl bitcodiert) (alle übrigen Controller)

Ausgänge, digital:

%QX0.0...%QX0.7 (ClassicController, ExtendedController, SafetyController)

%QX0.0, %QX0.8 (SmartController)

OUT00...OUT03 (CabinetController)

OUT0...OUT7 (Platinensteuerung)

Auch globale Variablen verlieren ihre Eindeutigkeit, wenn auf sie quasi gleichzeitig im Zyklus und durch die Interrupt-Routine zugegriffen wird. Insbesondere größere Datentypen (z.B. DINT) sind von dieser Problematik betroffen.

Alle anderen Ein- und Ausgänge werden, wie üblich, einmalig im Zyklus bearbeitet.

Parameter der Eingänge

276

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (nur 1 Zyklus): Datenänderungen zulässig FALSE: Datenänderungen nicht zulässig (während des Programmablaufs)
REPEATTIME	TIME	Zeitfenster, in dem der Interrupt ausgelöst wird
READ_INPUTS	BOOL	TRUE: in die Routine eingebundene Eingänge werden gelesen (Eingänge ggf. auf IN_FAST setzen) FALSE: diese Funktion wird nicht ausgeführt
WRITE_OUTPUTS	BOOL	TRUE: in die Routine eingebundene Ausgänge werden geschrieben FALSE: diese Funktion wird nicht ausgeführt
ANALOG_INPUTS	BYTE	(gilt nur bei Geräten mit Analogkanälen) TRUE: in die Routine eingebundene Analog-Eingänge werden gelesen und der Rohwert der Spannung an die Systemmerker ANALOG_IRQxx ausgegeben FALSE: diese Funktion wird nicht ausgeführt

10.1.2 SET_INTERRUPT_I

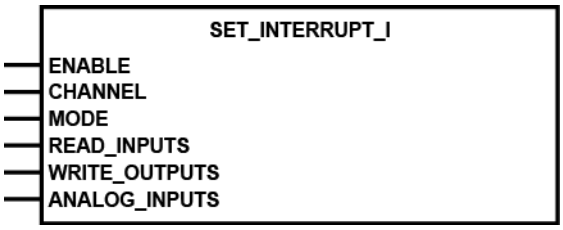
278

Baustein-Typ = Funktionsblock (FB)

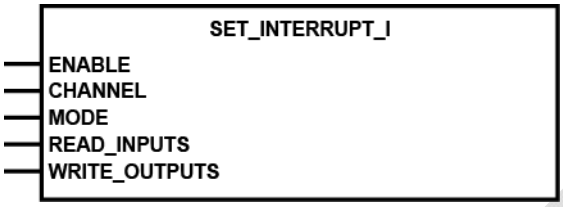
Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- CabinetController: CR030n
- ClassicController: CR0020, CR0505
- ExtendedController: CR0200
- Platinensteuerung: CS0015
- SmartController: CR25nn
- PDM360smart: CR1071

Symbol in CoDeSys:



(nur bei Geräten mit Analog-Kanälen)



(bei Geräten ohne Analog-Kanäle)

Beschreibung

281

SET_INTERRUPT_I organisiert das Ausführen eines Programmteils durch eine Interrupt-Anforderung über einen Eingangskanal.

In der klassischen SPS ist die Zykluszeit das Maß der Dinge für Echtzeitbetrachtungen. Gegenüber kundenspezifischen Steuerungen ist die SPS damit im Nachteil. Auch ein "Echtzeit-Betriebssystem" ändert nichts an dieser Tatsache, wenn das gesamte Applikationsprogramm in einem einzigen unveränderlichen Block abläuft.

Ein möglicher Lösungsansatz wäre, die Zykluszeit kurz zu halten. Dieser Weg führt oft dazu, die Applikation auf mehrere Steuerungszyklen zu verteilen. Die Programmierung wird dadurch jedoch unübersichtlich und schwierig.

Eine andere Möglichkeit besteht darin, einen bestimmten Programmteil nur auf Anforderung durch einen Eingangsimpuls unabhängig vom Steuerungszyklus aufzurufen.

Der zeitkritische Teil der Applikation wird vom Anwender in einen Baustein vom Type PROGRAMM (PRG) zusammengefasst. Dieser Baustein wird zur Interrupt-Routine deklariert, indem einmalig (zur Initialisierungszeit) SET_INTERRUPT_I aufgerufen wird. Das hat zur Folge, dass dieser Programmteil immer dann ausgeführt wird, wenn eine Flanke am Eingang CHANNEL erkannt wird. Werden Ein- und Ausgänge in diesem Programmteil genutzt, werden diese ebenfalls in der Interrupt-Routine, ausgelöst durch die Eingangs-Flanke, gelesen oder beschrieben. Über die Eingänge READ_INPUTS, WRITE_OUTPUTS oder ANALOG_INPUTS kann das Lesen oder Schreiben unterbunden werden.

Innerhalb des Programmteils können also alle zeitkritischen Ereignisse bearbeitet werden, indem Eingänge oder globale Variablen verknüpft und Ausgänge beschrieben werden. So können auch

Bausteine nur genau dann ausgeführt werden, wenn sie durch ein Eingangssignal angefordert werden.

HINWEIS

Damit der per Interrupt aufgerufene Programmteil nicht zusätzlich zyklisch aufgerufen wird, sollte er (mit Ausnahme des Initialisierungsaufrufes) im Zyklus übersprungen werden.

Der Eingang (CHANNEL), der zum Auslösen des Interrupt überwacht wird, kann in der Interrupt-Routine nicht initialisiert und weiter verarbeitet werden.

Die Eingänge müssen in der Betriebsart IN_FAST sein, sonst können die Interrupts nicht gelesen werden.

HINWEIS

Die Eindeutigkeit der Ein- und Ausgänge im Zyklus wird durch die Interrupt-Routine aufgehoben. Deshalb wird nur ein Teil der Ein- und Ausgänge bedient. Wurden sie im Interrupt-Programm initialisiert, werden folgende Ein- und Ausgänge gelesen oder geschrieben.

Eingänge, digital:

%IX0.0...%IX0.7 (CRnn32)

%IX0.12...%IX0.15, %IX1.4...%IX1.8 (übrige ClassicController, ExtendedController, SafetyController)

%IX0.0, %IX0.8 (SmartController)

IN08...IN11 (CabinetController)

IN0...IN3 (Platinensteuerung)

Eingänge, analog:

%IX0.0...%IX0.7 (CRnn32)

alle Kanäle (Auswahl bitcodiert) (alle übrigen Controller)

Ausgänge, digital:

%QX0.0...%QX0.7 (ClassicController, ExtendedController, SafetyController)

%QX0.0, %QX0.8 (SmartController)

OUT00...OUT03 (CabinetController)

OUT0...OUT7 (Platinensteuerung)

Auch globale Variablen verlieren ihre Eindeutigkeit, wenn auf sie quasi gleichzeitig im Zyklus und durch die Interrupt-Routine zugegriffen wird. Insbesondere größere Datentypen (z.B. DINT) sind von dieser Problematik betroffen.

Alle anderen Ein- und Ausgänge werden, wie üblich, einmalig im Zyklus bearbeitet.

Parameter der Eingänge

282

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (nur 1 Zyklus): Datenänderungen zulässig FALSE: Datenänderungen nicht zulässig (während des Programmablaufs)
CHANNEL	BYTE	Interrupt-Eingang Classic/ExtendedController: 0 = %IX1.4 1 = %IX1.5 2 = %IX1.6 3 = %IX1.7 SmartController: 0 = %IX0.0 1 = %IX0.8 CabinetController: 0 = IN08 (usw.) 3 = IN11 CS0015: 0 = IN0 (usw.) 3 = IN3
MODE	BYTE	Art der Flanke am Eingang CHANNEL, die den Interrupt auslöst 1 = steigende Flanke 2 = fallende Flanke 3 = steigende und fallende Flanke
READ_INPUTS	BOOL	TRUE: in die Routine eingebundene Eingänge werden gelesen (Eingänge ggf. auf IN_FAST setzen) FALSE: diese Funktion wird nicht ausgeführt
WRITE_OUTPUTS	BOOL	TRUE: in die Routine eingebundene Ausgänge werden geschrieben FALSE: diese Funktion wird nicht ausgeführt
ANALOG_INPUTS	BYTE	(gilt nur bei Geräten mit Analogkanälen) Auswahl der Eingänge bitcodiert: 0_{10} = kein Eingang gewählt 1_{10} = 1. Analogeingang gewählt ($0000\ 0001_2$) 2_{10} = 2. Analogeingang gewählt ($0000\ 0010_2$) ... 128_{10} = 8. Analogeingang gewählt ($1000\ 0000_2$) Eine Kombination der Eingänge entsteht durch ODER-Verknüpfung der Werte. Beispiel: 1. und 3. Analogeingang wählen: ($0000\ 0001_2$) ODER ($0000\ 0100_2$) = ($0000\ 0101_2$) = 5_{10}

10.2 Zykluszeit steuern

Inhalt

3142

10.2.1 PLCPRGTC

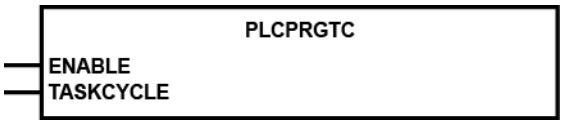
9954

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:

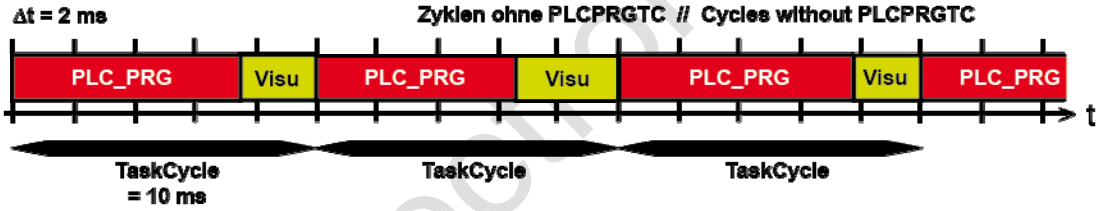


Beschreibung

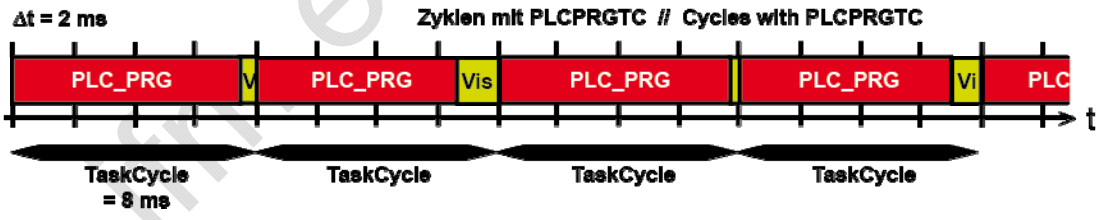
9955

PLCPRGTC ermöglich bei zeitkritischen Applikationen, die Aufrufzykluszeit für PLC_PRG zur verändern.

Wird der Baustein nicht eingebunden, wird die Aktualisierung der Visualisierung automatisch alle 10 ms unterbrochen und PLC_PRG und die daraus aufgerufenen Programmteile ausgeführt. Die übrige Zeit wird für die Aktualisierung der Visualisierung genutzt. Beispiel:



Soll PLC_PRG häufiger abgearbeitet werden (z.B. um schnelle Signale zur verarbeiten), kann mit dem FB PLCPRGTC die Zykluszeit für das Aufrufen des PLC_PRG verkürzt werden. Beispiel:



HINWEIS

Bei kürzerer Taskzeit für PLC_PRG verbleibt weniger Zeit für die Aktualisierung der Visualisierung.

Dies kann im Extremfall dazu führen, dass die Anzeige stark verzögert aufgebaut wird und Anzeigewerte verloren gehen.

Parameter der Eingänge

9957

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
TASKCYCLE	TIME	Zykluszeit für Task-Aufruf der Visualisierung

11 LED, Buzzer, Visualisierung

Inhalt

8615

Hier zeigen wir Ihnen folgende Funktionen:

- Ansteuern von LED
- Ansteuern des Buzzer
- Verwalten der Visualisierung

11.1 Visualisierung verwalten

Inhalt

8617

Hier zeigen wir Ihnen Funktionen zum Verwalten von Visualisierungen.

11.1.1 PDMsmart_MAIN

9928

Baustein-Typ = Programm (PRG)

Enthalten in Bibliothek: `ifm_PDMsmart_INIT_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:

Beschreibung

9930

PDMsmart_MAIN enthält folgende wichtige Funktionen für die Initialisierung des Geräts.

► Sie sollten PDMsmart_MAIN in eines der ersten Netzwerke des Applikations-Programms einbinden.

WICHTIG: Der Eingang INIT darf nur im ersten Programmzyklus auf TRUE gesetzt werden.

Wenn Sie prüfen wollen, ob PDMsmart_MAIN erfolgreich initialisiert ist:

► Variable PDM_FILE_OPEN_ERROR abfragen.

Parameter der Eingänge

3259

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (steigende Flanke): Baustein wird initialisiert (nur 1 Zyklus lang) FALSE: im weiteren Programmablauf

Globale Variable dieses Programms

9931

Alle Variablen dieses Programms sind in den Globalen Variablen der Bibliothek abgelegt.

Name	Datentyp	Beschreibung
SOFTKEY_F1 ... SOFTKEY_F6	BOOL	TRUE = Funktionstaste F1 gedrückt ... TRUE = Funktionstaste F6 gedrückt
SOFTKEY_ESC	BOOL	TRUE = Funktionstaste ESC gedrückt
SOFTKEY_OK	BOOL	TRUE = Funktionstaste OK gedrückt
SOFTKEY_LEFT	BOOL	TRUE = Funktionstaste LEFT gedrückt
SOFTKEY_RIGHT	BOOL	TRUE = Funktionstaste RIGHT gedrückt
SOFTKEY_DOWN	BOOL	TRUE = Funktionstaste DOWN gedrückt
SOFTKEY_UP	BOOL	TRUE = Funktionstaste UP gedrückt

Weitere Variable sind als Systemmerker in der Systemsteuerung definiert:
→ *Adressbelegung Ein-/Ausgänge* (→ Seite 310)

11.1.2 PDMsmart_MAIN_MAPPER

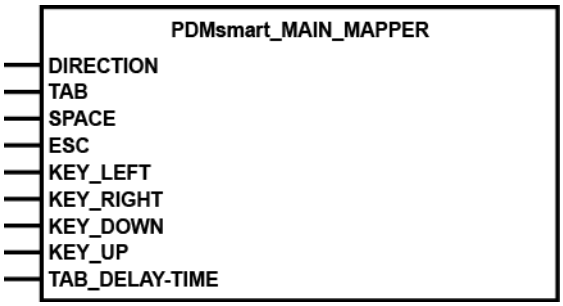
9923

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `ifm_CRnnnn_Vxxyyzz.LIB`

Für folgende Geräte verfügbar:
- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

9925

PDMsmart_MAIN_MAPPER ist die Schnittstelle zwischen CoDeSys-Tastaturkommandos für die Bedienung der Visualisierung und dem Laufzeitsystem des PDM. Durch Setzen/Rücksetzen der einzelnen Eingänge werden die Eingaben der PC-Tastatur emuliert.

Parameter der Eingänge

9926

Parameter	Datentyp	Beschreibung
DIRECTION	BOOL	Entspricht der PC-Taste [Umschalt] oder [Shift]. Sinnvoll mit dem Eingang TAB: TRUE: Die Markierung wechselt zum vorhergehenden Element FALSE: Die Markierung wechselt zum nächsten Element
TAB	BOOL	TRUE (Impuls, erstmalig): Markieren des ersten Elements der Elementliste, für das eine Eingabe konfiguriert ist TRUE (Impuls, weiterer) und DIRECTION=FALSE: Weiterschalten zum nächsten eingabefähigen Element TRUE (Impuls, weiterer) und DIRECTION=TRUE: Zurückschalten zum vorhergehenden eingabefähigen Element FALSE: diese Funktion wird nicht ausgeführt
SPACE	BOOL	TRUE (Impuls, erster): Selektiertes Visualisierungselement betätigen. Je nach gewähltem Eingabemodus kann dann im Eingabefeld navigiert werden. TRUE (Impuls, zweiter): Eingabe beenden; (neuen) Wert ins PDM schreiben FALSE: diese Funktion wird nicht ausgeführt
ESC	BOOL	TRUE (Impuls): Editiermodus abbrechen; Wert nicht verändern FALSE: diese Funktion wird nicht ausgeführt
KEY_LEFT	BOOL	TRUE (Impuls) und Eingabemodus=Position: Cursor im Eingabefeld um eine Position nach links verschieben FALSE: diese Funktion wird nicht ausgeführt
KEY_RIGHT	BOOL	TRUE (Impuls) und Eingabemodus=Position: Cursor im Eingabefeld um eine Position nach rechts verschieben FALSE: diese Funktion wird nicht ausgeführt
KEY_DOWN	BOOL	TRUE (Impuls) und Eingabemodus=Schrittweite: Wert im Eingabefeld um die angegebene Schrittweite mindern FALSE: diese Funktion wird nicht ausgeführt
KEY_UP	BOOL	TRUE (Impuls) und Eingabemodus=Schrittweite: Wert im Eingabefeld um die angegebene Schrittweite erhöhen FALSE: diese Funktion wird nicht ausgeführt
TAB_DELAY_TIME	TIME	Zeitverzögerung für den Eingang TAB Typische Werte: 250...400 ms Wert etwas größer einstellen als die Intervall-Zeit VISU_TASK

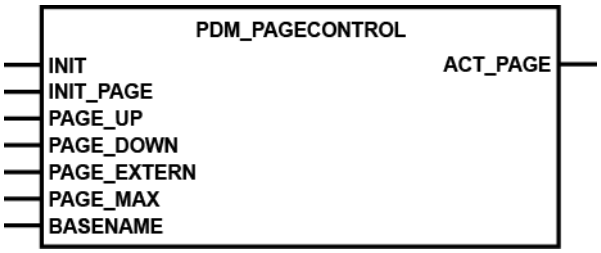
11.1.3 PDM_PAGECONTROL

3186

Baustein-Typ = Programm (PRG)

Enthalten in Bibliothek:	Für folgende Geräte verfügbar:
`ifm_PDM_UTIL_Vxxyyzz.LIB`	- PDM360: CR1050, CR1051 - PDM360compact: CR1052, CR1053, CR1055, CR1056
`ifm_PDMng_UTIL_Vxxyyzz.LIB`	- PDM360NG: CR108n
`ifm_PDMsmart_UTIL_Vxxyyzz.LIB`	- PDM360smart: CR1070, CR1071

Symbol in CoDeSys:



Beschreibung

3294

PDM_PAGECONTROL steuert den Aufruf bestimmter Visualisierungsseiten. Der Aufruf und die Rückgabe der Visualisierungsseiten erfolgt in CoDeSys über die Systemvariable CurrentVisu (vom Typ STRING[40]).

Mit dem Programm kann wahlweise eine bestimmte Visualisierungsseite aufgerufen oder schrittweise in den Visualisierungen geblättert werden.

Das Programm lässt sich optimal nutzen, wenn die Namen aller Visualisierungen dem gleichen Schema entsprechen, also einer Kombination aus einem Basisnamen, gefolgt von einer 5-stelligen Zahl (ab Bibliotheks-Version V04.00.07; davor: 3-stellig *)).

Beispiel BASENAME = `PAGE`:
Visualisierungsname = `PAGE00001`, `PAGE00002`, `PAGE00003`, usw.

Für den Basisnamen sind 1...35 Großbuchstaben (keine Sonderzeichen) zulässig. Die Nummerierung der Visualisierungen sollte lückenlos erfolgen. Das Programm setzt den endgültigen Visualisierungsnamen aus dem Parameter BASENAME und der Nummer zusammen oder liest die Nummer aus dem aktuellen Visualisierungsnamen aus und stellt sie im Ausgangsparameter ACT_PAGE zur Verfügung.

Anstatt die Visualisierungen mit Basisnamen und laufender Nummer zu benennen, kann jede Visualisierung auch individuell benannt werden, z.B.: `SERVICE1`, `MOTORDATA2`, `CONFIGURATION3`. Die Programmierung ist in diesem Fall aber aufwendiger, weil Basisname und Visualisierungsnummer einzeln zugewiesen werden müssen. Ein schrittweises Blättern ist nur noch sehr eingeschränkt möglich.

Tipp

Verwenden Sie als BASENAME den Buchstaben P, dann ist Ihr Programm kompatibel mit den **ifm**-Templates.

***) Beachten Sie die neue 5-stellige Nummerierung auch bei der Namensgebung Ihrer bereits bestehenden Visualisierungsseiten!**

Parameter der Eingänge

3293

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (nur 1 Zyklus lang): Display wird initialisiert mit der in INIT_PAGE angegebenen Initialisierung FALSE: im weiteren Programmablauf
INIT_PAGE	WORD	Visualisierungsnummer, die mit INIT aufgerufen werden soll
PAGE_UP	BOOL	Flanke FALSE ⇨ TRUE: inkrementiert die Visualisierungsnummer
PAGE_DOWN	BOOL	Flanke FALSE ⇨ TRUE: dekrementiert die Visualisierungsnummer
PAGE_EXTERN	WORD	Angegebene Visualisierungsseite wird direkt aufgerufen (unabhängig von PAGE_UP / PAGE_DOWN) Sobald PAGE_EXTERN = ACT_PAGE, dann PAGE_EXTERN wieder auf "0" setzen!
PAGE_MAX	WORD	Maximale Anzahl der anwählbaren Visualisierungsseiten
BASENAME	STRING[35]	Gemeinsamer Namensbestandteil der Visualisierungsseite. Die Nummerierung der Visualisierungsseiten erfolgt durch die Namensgebung, z.B. "P00001". Hierbei gelten: - "P" = BASENAME (nur Großbuchstaben!) - "00001" = Visualisierungsnummer (5-stellig!)

Parameter der Ausgänge

3295

Parameter	Datentyp	Beschreibung
ACT_PAGE	WORD	aktuelle Visualisierungsnummer

11.1.4 Bibliothek Instrumente

Inhalt

3354

Einbinden von fertigen Visualisierungs-Elementen

HINWEIS: Diese Bibliothek steht weiterhin zur Verfügung, um mit älteren Applikationen kompatibel zu bleiben. Wir empfehlen stattdessen – wegen deutlich besserer Darstellung – den Einsatz von Hintergrund-Bitmaps.
Siehe auch: Darstellbare → *CoDeSys-Visualisierungs-Elemente*

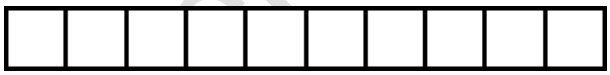
Die Bibliothek `Instrumente_x.LIB` bietet eine Anzahl von vorgefertigten Visualisierungs-Elementen. Diese können Sie direkt in Ihre Visualisierungsseiten über [Einfügen] > [Visualisierung] einbinden. Die Visualisierungs-Elemente sind so aufgebaut, dass die aktiven Elemente über Platzhalter animiert werden können. Dazu werden die Platzhalter direkt mit einer Variablen aus dem Applikations-Programm verknüpft. Weiter Informationen finden Sie in der CoDeSys-Onlinehilfe unter "Platzhalter in der Visualisierung".

Die Bibliothek enthält folgende Funktionen:

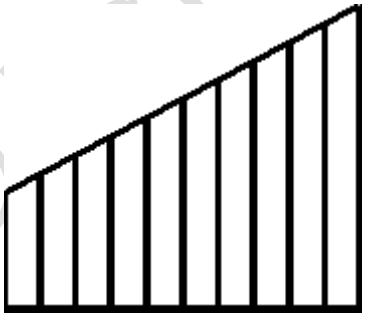
- *CONTROL_ANALOGCLOCK* (→ Seite 300)
  zeigt die aktuelle Uhrzeit auf dem Zifferblatt einer Analoguhr:

Analog_Clock

- *SCALE_LED_GRAF* (→ Seite 301)
  zeigt Eingangswerte als eine 10-stellige, werteabhängige LED-Zeile:



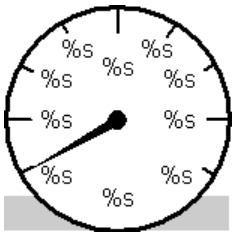
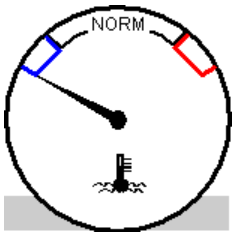
Visu = Bargraf_LED10_H



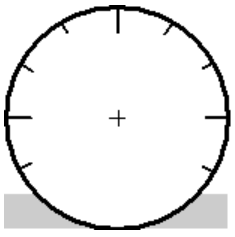
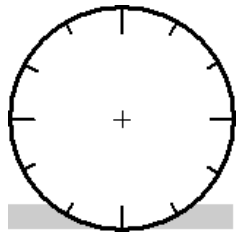
Visu = Bargraf_LED10_H2

Visu = Bargraf_LED10_V

- *SCALE_METER* (→ Seite 303)
  zeigt Eingangswerte als kreisförmige Messgeräte-Skala:

			
METER_NO = 1	METER_NO = 2	METER_NO = 3	METER_NO = 4
Visu = Meter1	Visu = Meter2	Visu = Meter3	Visu = Meter4

- Zusätzlich bietet die Bibliothek als Visualisierung 2 neutrale Skalen:

	
Visu = ClockFace1	Visu = ClockFace2

CONTROL_ANALOGCLOCK

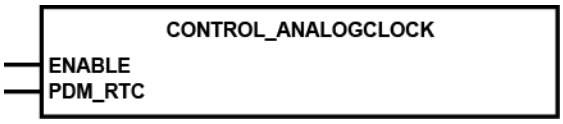
3366

Baustein-Typ = Programm (PRG)

Enthalten in Bibliothek: `Instrumente_x.LIB`

Für folgende Geräte verfügbar:
- PDM360: CR1050, CR1051
- PDM360compact: CR1052, CR1053, CR1055, CR1056
- PDM360smart: CR1070, CR1071
- PDM360NG: CR108n

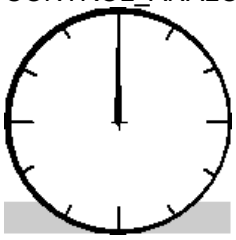
Symbol in CoDeSys:



Beschreibung

3378

CONTROL_ANALOGCLOCK zeigt die aktuelle Uhrzeit auf dem Zifferblatt einer Analoguhr:



Parameter der Eingänge

3379

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
PDM_RTC	DT	Systemzeit und Datum aus SysRtcGetTime

SCALE_LED_GRAF

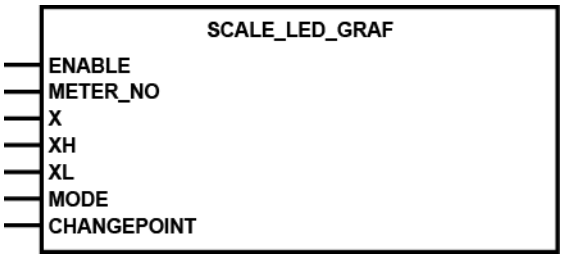
3369

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `Instrumente_x.LIB`

Für folgende Geräte verfügbar:
- PDM360: CR1050, CR1051
- PDM360compact: CR1052, CR1053, CR1055, CR1056
- PDM360smart: CR1070, CR1071
- PDM360NG: CR108n

Symbol in CoDeSys:

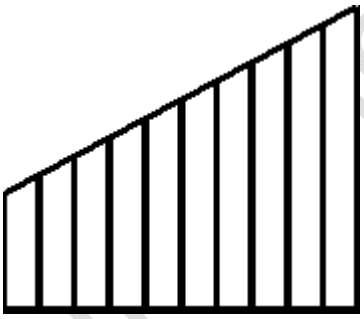


Beschreibung

3381

SCALE_LED_GRAF zeigt Eingangswerte als eine 10-stellige, werteabhängige LED-Zeile, z.B. eine der 3 Visualisierungen aus dieser Bibliothek:

Visu = Bargraf_LED10_H



Visu = Bargraf_LED10_H2

Visu = Bargraf_LED10_V

Der FB bildet einen Eingangswert relativ zu einem definierten Wertebereich ab.

Parameter der Eingänge

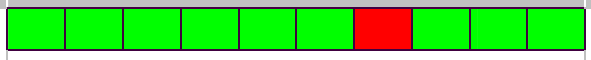
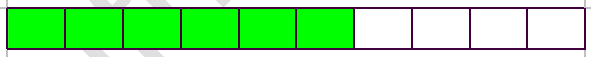
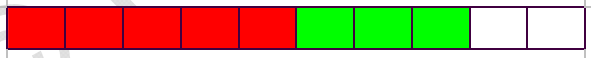
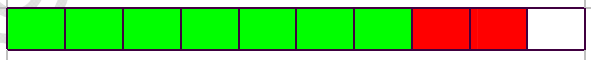
3382

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
X	INT	Eingangswert
XH	INT	Obere Grenze des Wertebereichs
XL	INT	Untere Grenze des Wertebereichs
MODE	BYTE	Betriebsart der LED-Kette Wertebereich = 0...10
CHANGEPOINT	BYTE	Farbwechsel-Punkt bei MODE = 9 oder 10 Wertebereich = 0...10

Betriebsart der LED-Kette

3383

Alle Variablen dieses Programms sind in den Globalen Variablen der Bibliothek abgelegt.

Mode	LED-Kette	Beschreibung
1		rotes Einzel-Segment auf grünem Leuchtband
2		grünes Einzel-Segment auf rotem Leuchtband
3		rotes Einzel-Segment
4		grünes Einzel-Segment
5		rote Segmentkette auf grünem Leuchtband
6		grüne Segmentkette auf rotem Leuchtband
7		rote Segmentkette
8		grüne Segmentkette
9		rote Segmentkette mit Farbwechsel-Punkt (hier CHANGEPOINT = 5)
10		grüne Segmentkette mit Farbwechsel-Punkt (hier CHANGEPOINT = 7)

SCALE_METER

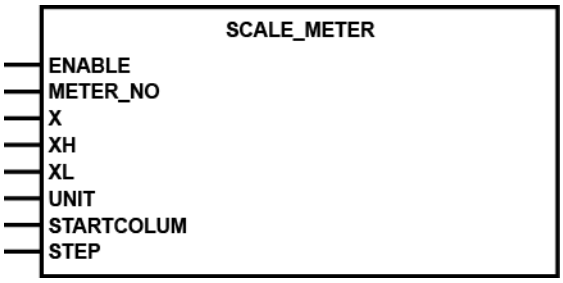
3372

Baustein-Typ = Funktionsblock (FB)

Enthalten in Bibliothek: `Instrumente_x.LIB`

Für folgende Geräte verfügbar:
- PDM360: CR1050, CR1051
- PDM360compact: CR1052, CR1053, CR1055, CR1056
- PDM360smart: CR1070, CR1071
- PDM360NG: CR108n

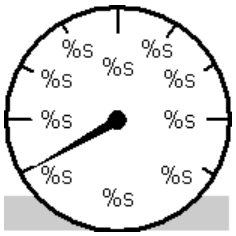
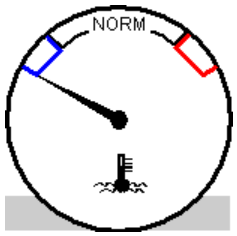
Symbol in CoDeSys:



Beschreibung

3384

SCALE_METER zeigt Eingangswerte als kreisförmige Messgeräte-Skala:

			
METER_NO = 1	METER_NO = 2	METER_NO = 3	METER_NO = 4
Visu = Meter1	Visu = Meter2	Visu = Meter3	Visu = Meter4

Der FB bildet einen Eingangswert relativ zu einem definierten Wertebereich ab.

In der Visualisierung Meter1 dient "%s" als Platzhalter für die parametrierten Werte und Einheit. In den anderen Visualisierungen gibt es keine oder keine definierbaren Skalenwerte.

Parameter der Eingänge

3385

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein wird ausgeführt FALSE: Baustein wird nicht ausgeführt > Baustein-Ein- und Ausgänge sind nicht aktiv
METER_NO	BYTE	1 = Meter1 = 270°-Skala mit definierbaren Werten und Einheit 2 = Meter2 = 270°-Skala 3 = Meter3 = Tankanzeige 4 = Meter4 = Temperaturanzeige
X	INT	Eingangswert
XH	INT	Obere Grenze des Wertebereichs
XL	INT	Untere Grenze des Wertebereichs
UNIT	STRING[6]	Für METER_NO = 1: Maßeinheit in der Skala (Text)
STARTCOLUM	INT	Startwert der Skala, z.B.: 10 = Die Skala beginnt bei 10
STEP	INT	Schrittweite der Skala, z.B.: 10 = Skalawerte für 10, 20, 30, ...

12 Anhang

Inhalt

1664

Hier stellen wir Ihnen – ergänzend zu den Angaben in den Datenblättern – zusammenfassende Tabellen zur Verfügung.

12.1 Fehler und Diagnose

9901

12.1.1 Fehler und Störungen beheben

3109

Hier zeigen wir Ihnen, wie Sie auf bestimmte Fehler und Störungen reagieren können, um das Gerät wieder nutzen zu können.

Wirkung	Ursache	Abhilfe
Nach Neustart vom Setup-Menü aus (= Soft-Reset) bleibt der Hochlauf mit leerem Bild stecken	Seltener Softwarefehler	Hard-Reset über Spannungsversorgung AUS / EIN
Sehr lange Bildwechsel-Zeiten	a) zu viele grafische Elemente im Bild b) zu viele verschiedene Zeichensätze (Fonts) c) zu viele systembelastende Bausteine d) Bausteine zu oft aufgerufen e) zu viele REAL-Variable im Bild	Empfohlene Begrenzungen einhalten! → *Begrenzungen und Programmierhinweise* (→ Seite 55)
System-Absturz (keine Reaktion)	a) falsche Platzhalter für Variable im CoDeSys-Programm	a) Platzhalter prüfen: z.B. %s (Falsch: %S)
Bildschirm bleibt dunkel	Globale Variable BACKLIGHT zu niedrig eingestellt	a) SETUP-Programm starten und wieder beenden → BACKLIGHT = 90 b) Im Applikations-Programm der BACKLIGHT-Variablen einen höheren Wert zuweisen

12.1.2 Systemmeldungen und Betriebszustände

9900

Je nach Betriebszustand werden auf dem Display des Geräts verschiedene Systemmeldungen ausgegeben. In der folgenden Liste sind alle Meldungen aufgeführt.

Systemmeldung	Betriebszustand
Bootloader...	Auslieferungszustand Es müssen das Laufzeitsystem (Betriebsystem) und ein Applikations-Programm geladen werden.
No Application	Es wurde noch kein Applikations-Programm geladen. Das Laufzeitsystem (Betriebssystem) ist im Gerät gespeichert.
Application stopped	Die Ausführung des Applikationsprogrammes wurde durch die Programmiersoftware angehalten. Erneuter Start ist nur durch das Programmiersystem möglich!
Application running	Ein Applikations-Programm ist geladen und gestartet, das keine Visualisierung enthält.
Undervoltage Application stopped	Es wurde Unterspannung erkannt. Das Applikations-Programm wurde angehalten.
Fatal Error	Ein nicht tolerierbarer Fehler wurde festgestellt (z.B. Speicher- oder CRC-Fehler). Dieser Zustand kann nur durch einen Reset (Aus-/Einschalten) verlassen werden.

12.2 Adressbelegung und E/A-Betriebsarten

Inhalt

1656

→ auch Datenblatt

12.2.1 Adressen / Variablen der E/As

Inhalt

2376

Adressen / Variablen der Eingänge

9943

IEC-Adresse	E/A-Variable	Bemerkung
%QB18 **)	I00_MODE	Konfigurations-Byte für %IX0.0
%QB19 **)	I01_MODE	Konfigurations-Byte für %IX0.1
%QB20 **)	I02_MODE	Konfigurations-Byte für %IX0.2
%QB21 **)	I03_MODE	Konfigurations-Byte für %IX0.3
%IX1.0	F1	Funktionstaste [F1]
%IX1.1	F2	Funktionstaste [F2]
%IX1.2	F3	Funktionstaste [F3]
%IX1.3	F4	Funktionstaste [F4]
%IX1.4	F5	Funktionstaste [F5]
%IX1.5	F6	Funktionstaste [F6]
%IX1.6	KEY_ESC	Funktionstaste [ESC]
%IX1.7	KEY_UP	Funktionstaste [▲]
%IX1.8	KEY_OK	Funktionstaste [OK]
%IX1.9	KEY_LEFT	Funktionstaste [◀]
%IX1.10	KEY_DOWN	Funktionstaste [▼]
%IX1.11	KEY_RIGHT	Funktionstaste [▶]
%IW2	SUPPLY_VOLTAGE	WORD Versorgungsspannung in [mV]

**) Gilt nur für folgende Geräte: PDM360smart: CR1071

Adressen / Variablen der Ausgänge

9944

IEC-Adresse	E/A-Variable	Bemerkung
%QB2	LED_F1	LED in Funktionstaste [F1] 0...100 %
%QB3	LED_F2	LED in Funktionstaste [F2] 0...100 %
%QB4	LED_F3	LED in Funktionstaste [F3] 0...100 %
%QB5	LED_F4	LED in Funktionstaste [F4] 0...100 %
%QB6	LED_F5	LED in Funktionstaste [F5] 0...100 %
%QB7	LED_F6	LED in Funktionstaste [F6] 0...100 %
%QB8	LED_ESC	LED in Funktionstaste [ESC] 0...100 %
%QB9	LED_UP	LED in Funktionstaste [▲] 0...100 %
%QB10	LED_OK	LED in Funktionstaste [OK] 0...100 %
%QB11	LED_LEFT	LED in Funktionstaste [◀] 0...100 %
%QB12	LED_DOWN	LED in Funktionstaste [▼] 0...100 %
%QB13	LED_RIGHT	LED in Funktionstaste [▶] 0...100 %
%QB14	LED_NIGHT	LED-Helligkeit im Nachtmodus aktiv
%QB15	LED_MAX_VALUE	LED-Helligkeit im Normalbetrieb 0...100 %
%QB16	LED_NIGHT_VALUE	LED-Helligkeit im Nachtbetrieb 0...100 %
%QB17	BACKLIGHT	Hintergrundbeleuchtung des Displays 0...100 %

12.2.2 Mögliche Betriebsarten Ein-/Ausgänge

9950

Gilt nur für folgende Geräte: PDM360smart: CR1071

Eingänge	Betriebsart	Konfig.-Wert	Ausgänge	Betriebsart	Konfig.-Wert
I00…I03	IN_NOMODE	0	Q00...Q03	OUT_NOMODE	0
	IN_DIGITAL_H (plus)	1		OUT_DIGITAL_H	1 (default)
	IN_VOLTAGE30	16 (default)			
	IN_FAST	128 (default)			

Mögliche Konfigurations-Kombinationen (wo zulässig) entstehen durch Addition der Werte.

12.2.3 Adressbelegung Ein-/Ausgänge

2371

Adressbelegung der Eingänge

9947

Gilt nur für folgende Geräte: PDM360smart: CR1071

Abkürzungen → Kapitel *Hinweise zur Anschlussbelegung* (→ Seite 46)

Betriebsarten der Ein- und Ausgänge → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 309)

IEC-Adresse	Name E/A-Variable	Konfiguration mit Variable	Default-Wert	mögliche Betriebsarten
%IX0.0	I00	I00_MODE	192	BL / FRQ
%IX0.1	I01	I01_MODE	192	BL / FRQ
%IX0.2	I02	I02_MODE	192	BL / FRQ
%IX0.3	I03	I03_MODE	192	BL / FRQ

Adressbelegung der Ausgänge

9948

Gilt nur für folgende Geräte: PDM360smart: CR1071

Abkürzungen → Kapitel *Hinweise zur Anschlussbelegung* (→ Seite 46)

Betriebsarten der Ein- und Ausgänge → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 309)

IEC-Adresse	Name E/A-Variable	Konfiguration mit Variable	Default-Wert	mögliche Betriebsarten
%QX0.0	Q00	Q00_MODE	1	Aus / H-digital / PWM
%QX0.1	Q01	Q01_MODE	1	Aus / H-digital / PWM
%QX0.2	Q02	Q02_MODE	1	Aus / H-digital / PWM
%QX0.3	Q03	Q03_MODE	1	Aus / H-digital / PWM

12.3 Systemmerker

9946

Systemmerker	Art	Beschreibung
CANx_BUSOFF	BOOL	CAN-Schnittstelle x: Fehler "CAN-Bus off"
CANx_LASTERROR ¹)	BYTE	CAN-Schnittstelle x: Fehlernummer der letzten CAN-Übertragung: 0= kein Fehler ≥ 0 → CAN-Spezifikation → LEC
CANx_WARNING	BOOL	CAN-Schnittstelle x: Warnschwelle erreicht (≥ 96)
ERROR	BOOL	Sammelfehlermeldung setzen, Relais *) ausschalten
ERROR_IO	BOOL	Sammelfehlermeldung Ein-/Ausgangsfehler
ERROR_MEMORY	BOOL	Speicherfehler
ERROR_POWER	BOOL	Spannungs-Fehler: SUPPLY_VOLTAGE < 10 000 mV oder > 32 000 mV
ERROR_TEMPERATUR	BOOL	Temperatur-Fehler (< - 25 °C oder > 85 °C)

CANx steht für die Nummer der CAN-Schnittstelle (CAN 1...x, abhängig vom Gerät).

¹) Der Zugriff auf diese Merker erfordert genaue Kenntnisse des CAN-Controllers und wird im Normalfall nicht benötigt.

*) Relais nur in folgenden Geräten vorhanden: CR0020, CR0032, CR0033, CR0200, CR0232, CR0233, CR0505, CR7020, CR7021, CR7200, CR7201, CR7505, CR7506

12.4 CANopen-Tabellen

Inhalt

9941

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

12.4.1 IDs (Adressen) in CANopen

3952

In CANopen werden diverse Arten von 'Adressen' (hier: IDs) unterschieden:

- COB-ID
  Der **C**ommunication-**Ob**ject-**Id**entifier adressiert die Nachricht (= das Kommunikationsobjekt) im Geräteverzeichnis. Ein Kommunikationsobjekt besteht aus einem oder mehreren CAN-Nachrichten mit bestimmten Aufgaben, z.B.:
  - PDO (**P**rocess **D**ata **O**bject = Nachrichten-Objekt mit Prozessdaten),
  - SDO (**S**ervice **D**ata **O**bject = Nachrichten-Objekt mit Servicedaten),
  - Emergency (Nachrichten-Objekt mit Notfalldaten),
  - Time (Nachrichten-Objekt mit Zeitangaben) oder
  - Error Control (Nachrichten-Objekt mit Fehlermeldungen).
- CAN-ID
  Der **CAN-Id**entifier definiert netzwerkweit CAN-Nachrichten. Der CAN-ID ist Hauptbestandteil des Arbitration-Feldes eines CAN-Datenübertragungsblocks. Je niedriger der CAN-ID, desto höher die Priorität der Meldung.
- Download-ID
  Der Download-ID bezeichnet den Node-ID für Service-Kommunikation per SDO für den Programm-Download und das Debuggen.
- Node-ID
  Der **Node-Id**entifier ist ein eindeutiger Bezeichner für CANopen-Geräte (Devices) im CAN-Netzwerk. Der Node-ID ist auch Bestandteil einiger vordefinierter Verbindungssätze (→ *Funktions-Code / Predefined Connectionset* (→ Seite 315)).

Vergleich Download-ID vs. COB-ID:

Controller Programm-Download		CANopen	
Download-ID	**COB-ID SDO**	**Node-ID**	**COB-ID SDO**
1…127	TX: 580_{16} + Download-ID	1…127	TX: 580_{16} + Node-ID
	RX: 600_{16} + Download-ID		RX: 600_{16} + Node-ID

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

12.4.2 Aufbau von CANopen-Meldungen

Inhalt

9971

Eine CANopen-Meldung besteht aus dem 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.

HINWEIS: Beachten Sie die umgekehrte Byte-Reihenfolge!

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 des COB-ID

9972

Der erste Teil einer Meldung ist der COB-ID. Aufbau des 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**						

Der COB-ID besteht aus *Funktions-Code / Predefined Connectionset* (→ Seite 315) und Node-ID.

Beispiel:
Das Kommunikations-Objekt = TPDO1 (TX)
Die Knoten-Nummer des Geräts = $20_{16} = 32_{10}$

Berechnung:
Der Funktions-Code für das Kommunikations-Objekt TPDO1 = 3_{16}
Die Wertigkeit des Funktions-Code im 11-Bit-COB-ID = $3_{16} \times 80_{16} = 180_{16}$
Dazu die Knoten-Nummer (20_{16}) addieren ⇨ der COB-ID lautet: $1A0_{16}$

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**
--	$3_{16} = 3_{10}$				$20_{16} = 32_{10}$						

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 käme.

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	Kommando	Index		Sub-Index	Daten *)			
XXX	8	Byte	Byte 0	Byte 1	Byte	Byte 0	Byte 1	Byte 2	Byte 3

*) abhängig von den zu transportierenden Daten

HINWEIS: Beachten Sie die umgekehrte Byte-Reihenfolge!

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

CANopen	
Node-ID	**COB-ID SDO**
1…127	TX: 580_{16} + Node-ID
	RX: 600_{16} + Node-ID

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

DLC (Data length code) bezeichnet die Anzahl der Daten-Bytes (bei 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* (→ Seite 317)

SDO-Abbruch-Code

9970

HINWEIS: 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)

12.4.3 Bootup-Nachricht

9961

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

	Byte 1	Byte 0
hex	700_{16} + Node-ID	NMT-Status
dez	$1\,792_{10}$ + Node-ID	NMT-Status

Somit ist der Teilnehmer im CAN-Netzwerk lauffähig.

Beispiel:
Der Node-ID des Teilnehmers ist $7D_{16} = 125_{10}$.
Dann lautet Byte 1 der Bootup-Nachricht: $77D_{16} = 1\,917_{10}$

HINWEIS: Es gibt Geräte, die kein [700_{16} + Node-ID] senden können.
Diese Geräte senden stattdessen folgende Bootup-Nachricht und ohne Status:

hex	80_{16} + Node-ID
dez	128_{10} + Node-ID

12.4.4 Netzwerk-Management (NMT)

9974

Netzwerk-Management-Kommandos

9962

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

Byte 1	Byte 2	Byte 2
COB-ID	Kommando	Node-ID

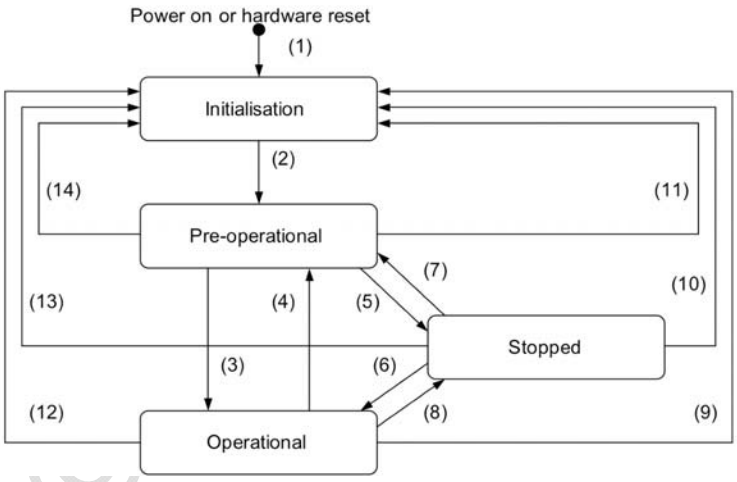
Node-ID = 00 ⇨ Kommando gilt zeitgleich für alle Knoten im Netz

COB-ID	NMT-Kommando		Beschreibung	
00	$01_{16} = 01_{10}$	Node-ID	start_remode_node	CAN-Teilnehmer starten
00	$02_{16} = 02_{10}$	Node-ID	stop_remode_node	CAN-Teilnehmer stoppen
00	$80_{16} = 128_{10}$	Node-ID	enter_pre-operational	umschalten auf Pre-Operational
00	$81_{16} = 129_{10}$	Node-ID	reset_node	CAN-Teilnehmer zurücksetzen
00	$82_{16} = 130_{10}$	Node-ID	reset_communication	CAN-Kommunikation 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 Indication"

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

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

(6) NMT Service "Start Remote Node Indication"

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

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

NMT-Status für CANopen-Master

9964

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 1000_{16} 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: - Der Master sendet an den Slave ein SDO-Read-Request (Index 1000_{16}). - Die generierten SDOs werden in ein SDO-Array gepackt. - Der Slave kennt seine erste SDO und die Anzahl seiner SDOs.
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 Funktionsblock	hier steht dieser Knoten-Status
CANx_MASTER_STATUS CANx_SLAVE_STATUS	Ausgang NODE_STATE
CANOPEN_GETSTATE	Ausgang NODESTATE

NMT-Status für CANopen-Slave

9965

Status hex | dez		Beschreibung
FF	-1	Der Slave wird durch die NMT-Nachricht [Reset Node] zurückgesetzt und wechselt selbständig in den Status 1.
00	0	nicht definiert
01	1	Status = Warten auf BOOTUP Der Slave wechselt nach einer maximalen Zeit von 2 s oder sofort nach Empfang seiner Bootup-Message in den Status 2.
02	2	Status = BOOTUP Der Slave wechselt nach einer Verzögerungszeit von 0,5 s automatisch in den Status 3.
03	3	Status = PREPARED Im Status 3 wird der Slave konfiguriert. Der Slave bleibt solange im Status 3, bis er alle vom Konfigurator erzeugten SDOs erhalten hat. Dabei spielt es keine Rolle, ob während der Konfiguration vom Slave SDO-Transfers mit Abort (Fehler) oder ob alle fehlerfrei beantwortet wurden. Nur die vom Slave erhaltene Antwort als solche ist wichtig – nicht ihr Inhalt. Wenn im Konfigurator die Option [Knoten zurücksetzen] aktiviert wurde, wird nach dem Senden des Objekts 1011_{16} Subindex 1, der dann den Wert "load" enthält, ein erneuter Reset des Slaves durchgeführt. Der Slave wird dann wieder mit dem Upload des Objekts 1000_{16} angefragt. Slaves, bei denen während der Konfigurationsphase ein Problem auftritt, bleiben im Status 3 oder wechseln nach der Konfigurationsphase direkt in einen Fehlerstatus (Status > 5).
04	4	Status = PRE-OPERATIONAL Ein Knoten wechselt immer in den Status 4, außer: • es handelt sich um einen "optionalen" Slave und er wurde als nicht am Bus verfügbar detektiert (Abfrage Objekt 1000_{16}) ODER: • der Slave ist zwar vorhanden, aber hat auf die Abfrage des Objekts 1000_{16} mit einem anderen Typ in den unteren 16 Bits reagiert, als der Konfigurator erwartet hat.
05	5	Status = OPERATIONAL Im Status 5 findet der normale Datenaustausch statt: "Normal Operation". Wenn der Master auf [Automatisch starten] konfiguriert wurde, wird der Slave im Status 4 gestartet (d.h. es wird eine "Start Node"-NMT-Nachricht erzeugt) und der Slave wechselt automatisch nach Status 5. Wurde GLOBAL_START gesetzt, dann wird gewartet, bis sich alle Slaves im Status 4 befinden. Anschließend werden alle Slaves mit dem NMT-Kommando [Start All Nodes] gestartet.
61	97	Ein Knoten wechselt in den Status 97, wenn er optional ist (optionales Gerät in der CAN-Konfiguration) und nicht auf die SDO-Anfrage nach dem Objekt 1000_{16} reagiert hat. Wird der Slave zu einem späteren Zeitpunkt an das Netzwerk angeschlossen und erkannt, wird er automatisch gestartet. Dazu müssen Sie aber die Option [Automatisch starten] in den CAN-Parametern des Masters angewählt haben.
62	98	Ein Knoten wechselt in den Status 98, wenn der Gerätetyp (Objekt 1000_{16}) nicht dem konfigurierten Typ entspricht.
63	99	Im Falle eines Nodeguarding-Timeouts wird der Slave auf Status 99 gesetzt. Sobald der Slave wieder auf NodeGuard-Anfragen reagiert und die Option [Automatisch starten] eingeschaltet ist, wird er automatisch vom Master gestartet. Dabei wird der Knoten abhängig von seinem Status, der in der Antwort auf die Nodeguard-Anfragen enthalten ist, neu konfiguriert oder nur gestartet. Um den Slave manuell zu starten, genügt es, die Methode [NodeStart] zu benutzen.

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 Funktionsblock	hier steht dieser Knoten-Status
CANx_MASTER_STATUS CANx_SLAVE_STATUS	Ausgang NODE_STATE
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	Knoten hat die BOOTUP-Nachricht erhalten.
04	4	PREPARED	Knoten wird per SDOs konfiguriert.
05	5	OPERATIONAL	Knoten nimmt am normalen Datenaustausch teil.
7F	127	PRE-OPERATIONAL	Knoten sendet keine Daten, ist aber vom Master konfigurierbar.

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

Knoten-Status aus FB lesen:

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

12.4.5 CANopen Error-Code

Inhalt

9967

Emergency-Nachrichten

9973

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

COB-ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
80_{16} + Node-ID		Error-Code		Objekt 1001_{16}	gerätespezifisch				

HINWEIS: Beachten Sie die umgekehrte Byte-Reihenfolge!

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 procedded 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 1001_{16} 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 "Leitungsbruch" an Kanal 2 (→ Installationsanleitung des Geräts):

COB-ID	DLC	Byte 0	Byte 1	Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
80_{16} + Node-ID		**00**	**FF**	**81**	**10**	**00**	00	00	00

Error-Code = $FF00_{16}$

Error-Register = 81_{16} = $1000\ 0001_2$, besteht also aus folgenden Fehlern:
- Generic Error (allgemeiner Fehler)
- manufacturer specific (herstellerspezifisch)

Betroffener Kanal = 0010_{16} = $0000\ 0000\ 0001\ 0000_2$ = Kanal 2

12.5 Visualisierungen im Gerät

Inhalt

3111

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

12.5.1 Grundsätzliches

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* (→ Seite 55)!

12.5.2 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.

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.

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

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.

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.

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!

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* (→ Seite 337) und *ISO 9241-11 _ Anforderungen an die Gebrauchstauglichkeit* (→ Seite 337) 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

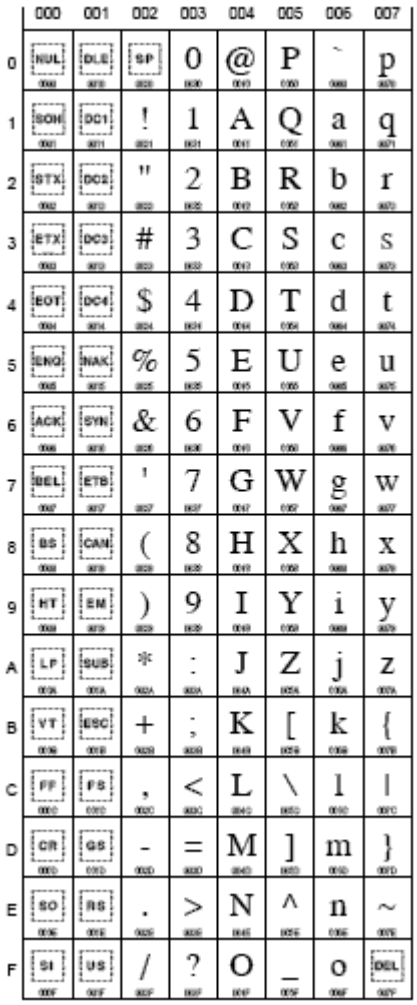
Der Universal Character Set (UCS) ist eine Zeichenkodierung, die im internationalen Standard ISO 10646 definiert ist. Für alle praktischen Belange ist dies dasselbe wie Unicode.

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.

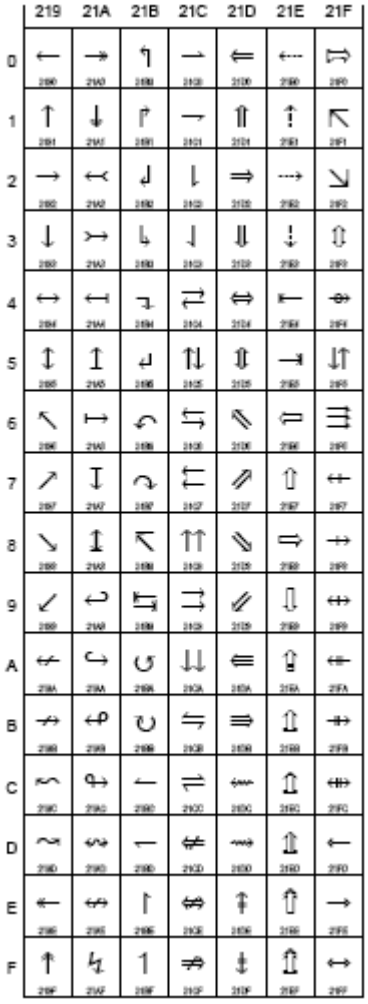
Nicht alle dieser 65 536 Zeichenadressen werden dabei standardisiert belegt: Ein nutzerdefinierter Bereich erlaubt es, ca. 2 000 Adressen mit nutzerspezifischen Zeichen zu belegen.

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.

12.5.3 Grundlegende Informationen zu Bitmap-Grafiken

Inhalt

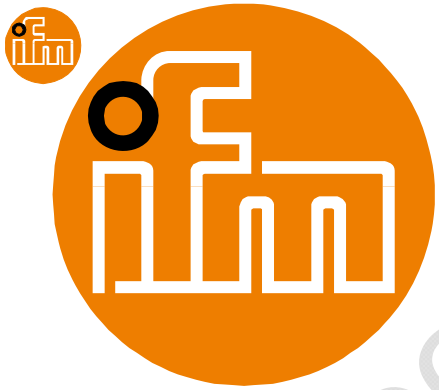
9858
3112

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)

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

9906

Ein Digitalfoto mit einer Auflösung von 5 Megapixeln hat eine Bildgröße von 2 560 x 1 920 Bildpunkten (= 4 915 200 Pixeln). Dieses Foto soll nun in einer Bildgröße von nur 128 x 64 Bildpunkten (= Monitorgröße bei diesem Gerät) dargestellt werden.

Folge nach dem Skalieren: Es sind nur noch 8 192 Bildpunkte (= 0,167 % des ursprünglichen Bildes) verblieben, die anderen 4 907 008 Pixel entfallen ersatzlos.

Oder anders ausgedrückt:
- Waagerecht wird nur jedes 20. Pixel verwendet.
- Senkrecht wird nur jedes 30. Pixel verwendet.

Daher kann ein so gewandeltes Foto nicht mehr die Qualität des Originals besitzen. Wichtige Informationen gehen verloren.

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

Bitmap-Grafiken anpassen

9996

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

12.6 Übersicht der verwendeten Dateien und Bibliotheken

Inhalt

2711

(Stand: 02.03.2011)

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.

12.6.1 Dateien und Bibliotheken im Gerät installieren

2721

Werkseinstellung: Das Gerät enthält nur den Bootloader.

► Betriebssystem (`*.H86` oder `*.RESX`) laden.

► Projekt (`*.PRO`) im PC anlegen: Target (`*.TRG`) eintragen.

► Zusätzlich je nach Gerät und Target-Version:
Steuerungskonfiguration (`*.CFG`) festlegen.

> CoDeSys bindet die zum Target zugehörenden Dateien in das Projekt ein:
`*.TRG, *.CFG, *.CHM, *.INI, *.LIB`.

► Bei Bedarf das Projekt mit weiteren Bibliotheken (`*.LIB`) ergänzen.

Bestimmte Bibliotheken binden automatisch weitere Bibliotheken in das Projekt ein:
z.B. basieren einige Funktionsblöcke in **ifm**-Bibliotheken (`ifm_*.LIB`) auf Funktionsblöcken in CoDeSys-Bibliotheken (`3S_*.LIB`).

12.6.2 Allgemeine Übersicht

2712

Dateiname	Beschreibung und Speicherort [3])
`ifm_CRnnnn_Vxxyyzz.CFG` [1]) `ifm_CRnnnn_Vxx.CFG` [2])	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`	Betriebssystem / 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 Controllers / Monitors
V*	CoDeSys-Version
Vxx	Versionsnummer der ifm-Software
yy	Release-Nummer der ifm-Software
zz	Patch-Nummer der ifm-Software

[1]) gültig für CRnn32 Target-Version bis V01, alle anderen Geräte bis V04

[2]) gültig für CRnn32 Target-Version ab V02, CR040n ab V01, alle anderen Geräte ab V05

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

HINWEIS

Es müssen immer die zum gewählten Target passenden Software-Stände zum Einsatz kommen:

- des Betriebssystems (`CRnnnn_Vxxyyzz.H86` / `CRnnnn_Vxxyyzz.RESX`),
- der Steuerungskonfiguration (`CRnnnn_Vxx.CFG`),
- der Gerätebibliothek (`ifm_CRnnnn_Vxxyyzz.LIB`)
- und der weiteren Dateien
  (→ Kapitel *Übersicht der verwendeten Dateien und Bibliotheken* (→ Seite 344)).

CRnnnn	Geräte-Artikelnummer
Vxx: 00...99	Versionsnummer
yy: 00...99	Release-Nummer
zz: 00...99	Patch-Nummer

Dabei müssen der Basisdateiname (z.B. "`CR0032`") und die Software-Versionsnummer "xx" (z.B. "`02`") überall den gleichen Wert haben! Andernfalls geht das Gerät in den STOP-Zustand

Die Werte für "yy" (Release-Nummer) und "zz" (Patch-Nummer) müssen **nicht** übereinstimmen.

WICHTIG: Folgende Dateien müssen ebenfalls geladen sein:

- die zum Projekt erforderlichen internen Bibliotheken (in IEC 1131 erstellt),
- die Konfigurationsdateien (`*.CFG`)
- und die Target-Dateien (`*.TRG`).

12.6.3 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.

Dateien für Betriebssystem / Laufzeitsystem

2714

Dateiname	Funktion	verfügbar für:
`ifm_CRnnnn_Vxxyyzz.H86` `ifm_CRnnnn_Vxxyyzz.HEX`	Betriebssystem / Laufzeitsystem	alle ecomat*mobile*-Controller BasicDisplay: CR0451 PDM: CR10nn
`ifm_Browser_CRnnnn.INI`	CoDeSys-Browser-Kommandos	alle ecomat*mobile*-Controller PDM: CR10nn
`ifm_Errors_CRnnnn.INI`	CoDeSys-Fehler-Datei	alle ecomat*mobile*-Controller PDM: CR10nn

Target-Datei

2715

Dateiname	Funktion	verfügbar für:
`ifm_CRnnnn_Vxx.TRG`	Target-Datei	alle ecomat*mobile*-Controller BasicDisplay: CR0451 PDM: CR10nn

Steuerungskonfigurations-Datei

2716

Dateiname	Funktion	verfügbar für:
`ifm_CRnnnn_Vxxyyzz.CFG`	Steuerungskonfiguration	alle ecomat*mobile*-Controller BasicDisplay: CR0451 PDM: CR10nn

ifm-Gerätebibliotheken

2717

Dateiname	Funktion	verfügbar für:
ifm_CRnnnn_Vxxyyzz.LIB	gerätespezifische Bibliothek	alle ecomat*mobile*-Controller BasicDisplay: CR0451 PDM: CR10nn
ifm_CR0200_MSTR_Vxxyyzz.LIB	Bibliothek ohne Extended-Funktionen	ExtendedController: CR0200
ifm_CR0200_SMALL_Vxxyyzz.LIB	Bibliothek ohne Extended-Funktionen, reduzierter Funktionsumfang	ExtendedController: CR0200

ifm-CANopen-Hilfsbibliotheken Master/Slave

2718

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

Dateiname	Funktion	verfügbar für:
ifm_CRnnnn_CANopenMaster_Vxxyyzz.LIB	CANopen Master Emergency- und Status-Handler	alle ecomat*mobile*-Controller *) PDM: CR10nn *)
ifm_CRnnnn_CANopenSlave_Vxxyyzz.LIB	CANopen Slave Emergency- und Status-Handler	alle ecomat*mobile*-Controller *) PDM: CR10nn *)
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 BasicDisplay: CR0451 PDM360NG: CR108n

*) jedoch NICHT für...
- BasicController: CR040n
- BasicDisplay: CR0451
- PDM360NG: CR108n

CoDeSys-CANopen-Bibliotheken

2719

Diese Bibliotheken sind für folgende Geräte NICHT verwendbar:
- BasicController: CR040n
- BasicDisplay: CR0451
- PDM360NG: CR108n

Dateiname	Funktion	verfügbar für:
3S_CanDrvOptTable.LIB [1] 3S_CanDrvOptTableEx.LIB [2]	CANopen Treiber	alle ecomat*mobile*-Controller PDM360smart: CR1070, CR1071
3S_CanDrv.LIB [3]		PDM360: CR1050, CR1051 PDM360compact: CR1052, CR1053, CR1055, CR1056
3S_CANopenDeviceOptTable.LIB [1] 3S_CANopenDeviceOptTableEx.LIB [2]	CANopen Slave-Treiber	alle ecomat*mobile*-Controller PDM360smart: CR1070, CR1071
3S_CANopenDevice.LIB [3]		PDM360: CR1050, CR1051 PDM360compact: CR1052, CR1053, CR1055, CR1056
3S_CANopenManagerOptTable.LIB [1] 3S_CANopenManagerOptTableEx.LIB [2]	CANopen Netzwerkmanager	alle ecomat*mobile*-Controller PDM360smart: CR1070, CR1071
3S_CANopenManager.LIB [3]		PDM360: CR1050, CR1051 PDM360compact: CR1053, CR1056
3S_CANopenMasterOptTable.LIB [1] 3S_CANopenMasterOptTableEx.LIB [2]	CANopen Master	alle ecomat*mobile*-Controller PDM360smart: CR1070, CR1071
3S_CANopenMaster.LIB [3]		PDM360: CR1050, CR1051 PDM360compact: CR1052, CR1053, CR1055, CR1056
3S_CANopenNetVarOptTable.LIB [1] 3S_CANopenNetVarOptTableEx.LIB [2]	Treiber für Netzwerkvariablen	alle ecomat*mobile*-Controller PDM360smart: CR1070, CR1071
3S_CANopenNetVar.LIB [3]		PDM360: CR1050, CR1051 PDM360compact: CR1052, CR1053, CR1055, CR1056

[1]) gültig für CRnn32 Target-Version bis V01, alle anderen Geräte bis V04
[2]) gültig für CRnn32 Target-Version ab V02, alle anderen Geräte ab V05
[3]) Für folgende Geräte gilt: diese Bibliothek ist funktionslos als Platzhalter enthalten:
- BasicController: CR040n
- BasicDisplay: CR0451
- PDM360NG: CR108n

spezielle ifm-Bibliotheken

2720

Dateiname	Funktion	verfügbar für:
ifm_RawCAN_NT_Vxxyyzz.LIB	CAN-Bausteine im CAN-Stack auf Basis Layer 2	BasicController: CR040n BasicDisplay: CR0451 PDM360NG: CR108n
ifm_J1939_NT_Vxxyyzz.LIB	J1939 Kommunikationsfunktionen im CAN-Stack	BasicController: CR040n BasicDisplay: CR0451 PDM360NG: CR108n
ifm_NetVarLib_NT_Vxxyyzz.LIB	zusätzlicher Treiber für Netzwerkvariablen	BasicController: CR040n BasicDisplay: CR0451 PDM360NG: CR108n
ifm_J1939_Vxxyyzz.LIB	J1939 Kommunikationsfunktionen	bis Target 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 Target V05 für: CabinetController: CR0303 ClassicController: CR0020, CR0505 ExtendedController: CR0200 SafetyController: CR7nnn SmartController: CR2500 PDM360smart: CR1070, CR1071
ifm_CRnnnn_J1939_Vxxyyzz.LIB	J1939 Kommunikationsfunktionen	ClassicController: CR0032, CR0033 ExtendedController: CR0232, CR0233
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 Target V04 für: PDM360smart: CR1070, CR1071

Dateiname	Funktion	verfügbar für:
ifm_CAN1_EXT_Vxxyyzz.LIB	Stellt den CAN-Bus von 11 Bit auf 29 Bit um	ab Target V05 für: CabinetController: CR030n ClassicController: CR0020, CR0505 ExtendedController: CR0200 Platinensteuerung: CS0015 SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506 SmartController: CR25nn 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: CR10nn
ifm_Hydraulic_16bitOS04_Vxxyyzz.LIB	Hydraulikfunktionen für Controller	bis Target V04 für: ClassicController: CR0020, CR0505 ExtendedController: CR0200 SafetyController: CR7020, CR7200, CR7505 SmartController: CR25nn
ifm_Hydraulic_16bitOS05_Vxxyyzz.LIB	Hydraulikfunktionen für Controller	ab Target V05 für: ClassicController: CR0020, CR0505 ExtendedController: CR0200 SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506 SmartController: CR25nn
ifm_Hydraulic_32bit_Vxxyyzz.LIB	Hydraulikfunktionen für Controller	ClassicController: CR0032, CR0033 ExtendedController: CR0232, CR0233
ifm_Hydraulic_CR0303_Vxxyyzz.LIB	Hydraulikfunktionen für Controller	ab Target V05 für: CabinetController: CR0303
ifm_SafetyIO_Vxxyyzz.LIB	Sicherheitsfunktionen	SafetyController: CR7nnn
ifm_PDM_UTIL_Vxxyyzz.LIB	Hilfsfunktionen PDM	PDM360: CR1050, CR1051 PDM360compact: CR1052, CR1053, CR1055, CR1056
ifm_PDMng_UTIL_Vxxyyzz.LIB	Hilfsfunktionen PDM	PDM360NG: CR108n
ifm_PDMsmart_UTIL_Vxxyyzz.LIB	Hilfsfunktionen PDM	PDM360smart: CR1070, CR1071
ifm_PDM_Input_Vxxyyzz.LIB	alternative Eingabefunktionen PDM	PDM: CR10nn
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 PDM360NG: CR108n
ifm_PDM360NG_linux_syscall_asynch.LIB	Linux-Kommandos an das System senden	PDM360NG: CR108n
ifm_PDM360NG_USB_Vxxyyzz.LIB	Geräte an der USB-Schnittstelle verwalten	PDM360NG: CR108n

Dateiname	Funktion	verfügbar für:
`ifm_PDM360NG_USB_LL_Vxxyyzz.LIB`	Hilfsbibliothek für `ifm_PDM360NG_USB_Vxxyyzz.LIB`	PDM360NG: CR108n
`Instrumente_x.LIB`	vordefinierte Anzeige-Instrumente	PDM: CR10nn
`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.

13 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.

Anforderungsrate rd

Die Anforderungsrate r_d ist die Häufigkeit je Zeiteinheit von Anforderungen an eine sicherheitsgerichtete Reaktion eines SRP/CS.

Anleitung

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

Applikations-Software

Software, die speziell für die Applikation (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.

Für sicherheitsrelevante Teile von Steuerungen (→SRP/CS) müssen spezielle Anforderungen erfüllt sein.
→ Programmiersprache, sicherheitsrelevant

Architektur

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

Ausfall

Ausfall ist die Beendigung der Fähigkeit einer Einheit, eine geforderte Funktion zu erfüllen.

Nach einem Ausfall hat die Einheit einen Fehler. Der Ausfall ist ein Ereignis, der Fehler ein Zustand.

Der so definierte Begriff kann nicht auf Einheiten angewendet werden, die nur aus Software bestehen.

Ausfall, gefahrbringend

Ein gefahrbringender Ausfall hat das Potential, das SRP/CS in einen gefährlichen Zustand oder eine Fehlfunktion zu bringen. Ob dieses Potential bemerkt werden kann oder nicht, hängt von der Architektur des Systems ab. In einem redundanten System wird ein gefährlicher Hardware-Ausfall weniger wahrscheinlich zu einem gefährlichen Ausfall des Gesamtsystems führen.

Ausfall, systematischer

Ein systematischer Ausfall ist ein Ausfall mit deterministischem (nicht zufälligem) Bezug zu einer bestimmten Ursache. Der systematische Ausfall kann nur beseitigt werden durch Änderung des Entwurfs oder des Herstellprozesses, Betriebsverfahren, Dokumentation oder zugehörigen Faktoren.

Eine Instandsetzung ohne Änderung des Systems wird den Grund des systematischen Ausfalls in der Regel nicht beseitigen.

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.

Betriebsdauer, mittlere

Mean **t**ime **b**etween **f**ailures (MTBF) = mittlere Betriebsdauer zwischen Ausfällen.
Ist der Erwartungswert der Betriebsdauer zwischen zwei aufeinanderfolgenden Ausfällen von Einheiten, die instand gesetzt werden.

HINWEIS: Für Einheiten, die NICHT instandgesetzt werden, ist der Erwartungswert (Mittelwert) der Verteilung von Lebensdauern die mittlere Lebensdauer →MTTF.

Betriebssystem

Grundprogramm im Gerät, stellt die Verbindung her zwischen der Hardware des Gerätes und der Anwender-Software.

Bootloader

Im Auslieferungszustand enthalten **ifm**-Geräte nur den Bootloader.
Der Bootloader ist ein Startprogramm, mit dem das Betriebssystem (= Laufzeitsystem) und das Applikations-Programm auf dem Gerät nachgeladen werden können.

Der Bootloader enthält Grundroutinen...
- zur Kommunikation der Hardware-Module untereinander,
- zum Nachladen des Betriebssystems.

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. Gibt es in verschiedenen Varianten z.B. als "CANopen" oder "CAN in Automation (CiA)."

CAN-Stack

CAN-Stack = Stapel von CAN-Datenübertragungs-Aufträgen.

CCF

Common **c**ause **f**ailure = Ausfall in Folge von gemeinsamer Ursache
Ausfälle verschiedener Einheiten aufgrund eines gemeinsamen Ereignisses, wobei diese Ausfälle nicht auf gegenseitige Ursachen beruhen.

CiA

CiA = CAN in Automation e.V.

Anwender- und Herstellerorganisation in Deutschland / Erlangen. Definitions- und Kontrollorgan für CAN und CAN-basierende Netzwerkprotokolle.

Homepage → http://www.can-cia.org

CiA DS 304

DS = **D**raft **S**tandard

CAN-Geräteprofil CANopen-Safety für sicherheitsgerichtete Kommunikation.

CiA DS 401

DS = **D**raft **S**tandard

CAN-Geräteprofil für digitale und analoge E/A-Baugruppen

CiA DS 402

DS = **D**raft **S**tandard

CAN-Geräteprofil für Antriebe

CiA DS 403

DS = **D**raft **S**tandard

CAN-Geräteprofil für Bediengeräte

CiA DS 404

DS = **D**raft **S**tandard

CAN-Geräteprofil für Messtechnik und Regler

CiA DS 405

DS = **D**raft **S**tandard

Spezifikation zur Schnittstelle zu programmierbaren Steuerungen (IEC 61131-3)

CiA DS 406

DS = **D**raft **S**tandard

CAN-Geräteprofil für Drehgeber / Encoder

CiA DS 407

DS = **D**raft **S**tandard

CAN-Applikations-Profil für den öffentlichen Nahverkehr

COB-ID

COB = **C**ommunication **Ob**ject = Kommunikationsobjekt
ID = **Id**entifier = Kennung

Über den COB-ID unterscheiden die Teilnehmer die verschiedenen auszutauschenden Nachrichten.

CoDeSys

CoDeSys ist eingetragene Marke der 3S – Smart Software Solutions GmbH, Deutschland

"CoDeSys for Automation Alliance" vereinigt Firmen der Automatisierungsindustrie, deren Hardwaregeräte alle mit dem weit verbreiteten IEC 61131-3 Entwicklungswerkzeug CoDeSys programmiert werden.

Homepage → http://www.3s-software.com

CRC

CRC = **C**yclic **R**edundancy **C**heck = zyklische Redundanzprüfung

CRC ist ein Verfahren aus der Informationstechnik zur Bestimmung eines Prüfwerts für Daten, um Fehler bei der Übertragung oder Duplizierung von Daten erkennen zu können.

Vor Beginn der Übertragung eines Blocks der Daten wird ein CRC-Wert berechnet. Nach Abschluss der Transaktion wird am Zielort der CRC-Wert erneut berechnet. Anschließend werden diese beiden Prüfwerte verglichen.

D

DC

Direct **C**urrent = Gleichstrom

DC

Diagnostic **C**overage = Diagnose-Deckungsgrad
Der Diagnose-Deckungsgrad ist das Maß für die Wirksamkeit der Diagnose als Verhältnis der Ausfallrate der bemerkten gefahrbringenden Ausfälle und der Ausfallrate der gesamten gefahrbringenden Ausfälle:

Formel: DC = Ausfallrate bemerkte gefahrbringende Ausfälle / Ausfallrate gesamte gefahrbringende Ausfälle

Bezeichnung	Bereich
kein	DC < 60 %
niedrig	60 % ≤ DC < 90 %
mittel	90 % ≤ DC < 99 %
hoch	99 % ≤ DC

Tabelle: Diagnose-Deckungsgrad DC

Für die in der Tabelle gezeigten Grenzwerte wird eine Genauigkeit von 5 % angenommen.

Der Diagnose-Deckungsgrad kann für das gesamte sicherheitsgerichtete System ermittelt werden oder nur für Teile des sicherheitsgerichteten Systems.

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.

Diagnose-Deckungsgrad

Diagnostic **C**overage = Diagnose-Deckungsgrad
Der Diagnose-Deckungsgrad ist das Maß für die Wirksamkeit der Diagnose als Verhältnis der Ausfallrate der bemerkten gefahrbringenden Ausfälle und der Ausfallrate der gesamten gefahrbringenden Ausfälle:

Formel: DC = Ausfallrate bemerkte gefahrbringende Ausfälle / Ausfallrate gesamte gefahrbringende Ausfälle

Bezeichnung	Bereich
kein	DC < 60 %
niedrig	60 % ≤ DC < 90 %
mittel	90 % ≤ DC < 99 %
hoch	99 % ≤ DC

Tabelle: Diagnose-Deckungsgrad DC

Für die in der Tabelle gezeigten Grenzwerte wird eine Genauigkeit von 5 % angenommen.

Der Diagnose-Deckungsgrad kann für das gesamte sicherheitsgerichtete System ermittelt werden oder nur für Teile des sicherheitsgerichteten Systems.

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.
→ Kapitel *Was ist der Dither?*

diversitär

Unter Diversität (Vielfalt) versteht man in der Technik eine Strategie zur Erhöhung der Ausfallsicherheit.

Dabei werden Systeme redundant ausgelegt, allerdings werden bewusst verschiedene Realisierungen und keine baugleichen Einzelsysteme verwendet. Man geht davon aus, dass Systeme, die das Gleiche leisten, aber unterschiedlich realisiert sind, auch gegen unterschiedliche Störungen empfindlich oder unempfindlich sind und daher möglichst nicht alle gleichzeitig ausfallen.

Die konkrete Realisierung kann je nach Einsatzgebiet und geforderter Sicherheit unterschiedlich aussehen:

- Verwendung von Bauteilen verschiedener Hersteller,
- Nutzung unterschiedlicher Protokolle zur Steuerung von Geräten,
- Verwendung komplett unterschiedlicher Technologien, beispielsweise einer elektrischen und einer pneumatischen Steuerung,
- Verwendung unterschiedlicher Messmethoden (Strom, Spannung),
- zwei Kanäle mit gegenläufigen Werteverläufen:
  Kanal A: 0...100 %
  Kanal B: 100...0 %

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
Störungen und Fehler werden 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 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 Betriebssystem.

Die Firmware stellt die Verbindung her zwischen der Hardware des Gerätes und der Anwender-Software. Diese Software wird vom Hersteller der Steuerung als Teil des Systems geliefert und kann vom Anwender nicht verändert werden.

EMCY

Abkürzung für Emergency (engl.) = Notfall

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.

Erstfehler-Eintrittszeit

Das ist die Zeit bis zum ersten Versagen eines Sicherheitselements.

Im Zeitraum von maximal 30 s überprüft das Betriebssystem mittels interner Überwachungs- und Testroutinen die Steuerung.

Diese „Testzykluszeit“ muss kleiner sein als die statistische Erstfehler-Eintrittszeit für die Applikation.

Ethernet

Das Ethernet ist eine weit verbreitete, herstellerneutrale Technologie, mit der im Netzwerk Daten mit einer Geschwindigkeit von 10 oder 100 Millionen Bit pro Sekunde (Mbps) übertragen werden können. Das 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 = „equipment under control" (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.

Fehler

Ein Fehler ist die Unfähigkeit einer Einheit, eine geforderte Funktion auszuführen.

Kein Fehler ist diese Unfähigkeit während vorbeugender Wartung oder anderer geplanter Handlungen oder aufgrund des Fehlers externer Mittel.

Ein Fehler ist oft das Resultat eines Ausfalls der Einheit selbst, kann aber ohne vorherigen Ausfall bestehen.

In der ISO 13849-1 ist mit "Fehler" der "zufällige Fehler" gemeint.

Fehler-Toleranzzeit

Das ist die maximale Zeit, die zwischen dem Entstehen eines Fehlers und der Einnahme des sicheren Zustandes in der Applikation vergehen darf, ohne dass eine Gefahr für Personen zu befürchten ist.

Dabei ist die maximale Zykluszeit des Applikations-Programms (im ungünstigsten Fall 100 ms, → *Verhalten des Watchdog* (→ Seite 56)) und die möglichen Verzögerungs- und Reaktionszeiten durch Abschaltglieder zu berücksichtigen.

Die sich daraus ergebende Gesamtzeit muss kleiner sein als die Fehler-Toleranzzeit der Applikation.

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.

Firmware

System-Software, Grundprogramm im Gerät, praktisch das Betriebssystem.

Die Firmware stellt die Verbindung her zwischen der Hardware des Gerätes und der Anwender-Software. Diese Software wird vom Hersteller der Steuerung als Teil des Systems geliefert und kann vom Anwender nicht verändert werden.

Flash-Speicher

Flash-ROM (oder Flash-EPROM oder Flash-Memory) kombiniert die Vorteile von Halbleiterspeicher und Festplatten. Wie jeder andere Halbleiterspeicher kommt Flash-Speicher ohne bewegliche Teile aus. Und die Daten bleiben wie bei einer Festplatte auch nach dem Ausschalten erhalten.

Der Flash-ROM hat sich aus dem EEPROM (**E**lectrical **E**rasable and **P**rogrammable **R**ead-**O**nly **M**emory) entwickelt. Beim Flash-ROM ist die Speicherung von Daten funktionell identisch wie beim EEPROM. 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.
- Im Vergleich zu Festplatten haben Flash-Speicher eine sehr kurze Zugriffszeit. Lese- und Schreibgeschwindigkeit sind über den gesamten Speicherbereich weitestgehend konstant.
- Die erreichbare Speichergröße ist durch die einfache und platzsparende Anordnung der Speicherzellen nach oben offen.

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.

FMEA

FMEA = **F**ailure **M**ode and **E**ffects **A**nalysis = **F**ehler-**M**öglichkeits- und **E**influss-**A**nalyse

Methode der Zuverlässigkeitstechnik, um potenzielle Schwachstellen zu finden. Im Rahmen des Qualitäts- oder Sicherheitsmanagements wird die FMEA zur Fehlervermeidung und Erhöhung der

technischen Zuverlässigkeit vorbeugend eingesetzt.

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.

Funktionale Sicherheit

Teil der Gesamtsicherheit, bezogen auf das →EUC und das EUC-Leit- oder Steuerungssystem, die von der korrekten Funktion des elektrischen oder elektronischen sicherheitsgerichteten Systems, sicherheitsgerichteten Systemen anderer Technologien und externer Einrichtungen zur Risikominderung abhängt.

G

Gebrauchsdauer Tm

Die Gebrauchsdauer T_M ist der Zeitraum, der die vorgegebene Verwendung der SRP/CS abdeckt.

Gefährdung

Mit Gefährdung bezeichnet man eine potentielle Schadensquelle.

Man unterscheidet den Ursprung der Gefährdung, z.B.:
- mechanische Gefährdung,
- elektrische Gefährdung,

oder die Art des zu erwartenden Schadens, z.B.:
- Gefährdung durch elektrischen Schlag,
- Gefährdung durch Schneiden,
- Gefährdung durch Vergiftung.

Die Gefährdung im Sinne dieser Definition ist bei der bestimmungsgemäßen Verwendung der Maschine entweder dauerhaft vorhanden, z.B.:
- Bewegung von gefährdenden beweglichen Teilen,
- Lichtbogen beim Schweißen,
- ungesunde Körperhaltung,
- Geräusch-Emission,
- hohe Temperatur,

oder die Gefährdung kann unerwartet auftreten, z.B.:
- Explosion,
- Gefährdung durch Quetschen als Folge eines unbeabsichtigten / unerwarteten Anlaufs,
- Herausschleudern als Folge eines Bruchs,
- Stürzen als Folge von Geschwindigkeitsänderung.

H

Heartbeat

Heartbeat (engl.) = Herzschlag

Die Teilnehmer senden regelmäßig kurze Signale. So können die anderen Teilnehmer prüfen, ob ein Teilnehmer ausgefallen ist. Dazu ist kein Master erforderlich.

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-User-Zyklus

IEC-User-Zyklus = SPS-Zyklus im CoDeSys-Applikations-Programm.

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"

K

Kategorie (CAT)

Einstufung der sicherheitsrelevante Teile einer Steuerung bezüglich ihres Widerstandes gegen Fehler und ihres nachfolgenden Verhaltens bei einem Fehler. Diese Sicherheit wird erreicht durch die Struktur der Anordnung der Teile, die Fehlererkennung und/oder ihre Zuverlässigkeit. (→ EN 954)

Klemme 15

Klemme 15 ist in Fahrzeugen die vom Zündschloss geschaltete Plusleitung.

L

Lebensdauer, mittlere

Mean **t**ime **t**o **d**angerous **f**ailure = erwartete mittlere Dauer bis zum gefahrbringenden Ausfall.

Bezeichnung	Bereich
niedrig	3 Jahre ≤ $MTTF_d$ < 10 Jahre
mittel	10 Jahre ≤ $MTTF_d$ < 30 Jahre
hoch	30 Jahre ≤ $MTTF_d$ ≤ 100 Jahre

Tabelle: Mittlere Zeit jedes Kanals bis zum gefahrbringenden Ausfall $MTTF_d$

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.

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* (→ Seite 359)

MRAM

MRAM bedeutet **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.

MTBF

Mean **t**ime **b**etween **f**ailures (MTBF) = mittlere Betriebsdauer zwischen Ausfällen.
Ist der Erwartungswert der Betriebsdauer zwischen zwei aufeinanderfolgenden Ausfällen von Einheiten, die instand gesetzt werden.

HINWEIS: Für Einheiten, die NICHT instandgesetzt werden, ist der Erwartungswert (Mittelwert) der Verteilung von Lebensdauern die mittlere Lebensdauer →MTTF.

MTTF

Mean **t**ime **t**o **f**ailure (MTTF) = mittlere Dauer bis zum Ausfall oder: mittlere Lebensdauer.

MTTFd

Mean **t**ime **t**o **d**angerous **f**ailure = erwartete mittlere Dauer bis zum gefahrbringenden Ausfall.

Bezeichnung	Bereich
niedrig	3 Jahre ≤ $MTTF_d$ < 10 Jahre
mittel	10 Jahre ≤ $MTTF_d$ < 30 Jahre
hoch	30 Jahre ≤ $MTTF_d$ ≤ 100 Jahre

Tabelle: Mittlere Zeit jedes Kanals bis zum gefahrbringenden Ausfall $MTTF_d$

Muting

Mit Muting bezeichnet man die vorübergehende und automatische Unterdrückung einer Sicherheitsfunktion durch das SRP/CS.

Beispiel: Der Sicherheits-Lichtvorhang ist überbrückt, wenn die schließenden Werkzeuge unter einen fingersicheren Abstand zueinander gelangt sind. Die bedienende Person kann nun gefahrlos an die Maschine herantreten und das Werkstück führen.

N

NMT

NMT = **N**etwork **M**anagemen**t** = Netzwerk-Verwaltung (hier: im CAN-Bus)

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.

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.

Diese Dienste sind vom Protokoll her nicht bestätigte Dienste, d.h. 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 Ziel- oder Quelladresse.

Performance-Level

Performance-**L**evel
Ist nach ISO 13849-1 eine Einstufung (PL a...e) der Fähigkeit von sicherheitsrelevanten Teilen einer Steuerung, eine Sicherheitsfunktion unter vorhersehbaren Bedingungen auszuführen.
→ Kapitel *Performance-Level PL*

PES

Programable **e**lectronic **s**ystem = Programmierbares elektronisches System
Ein programmierbares elektronisches System ist ein 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.

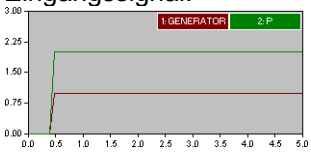
PGN

PGN = **P**arameter **G**roup **N**umber = Parameter-Gruppennummer
PGN = PDU Format (PF) + PDU Source (PS)

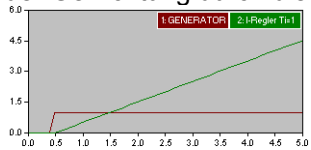
Die Parameter-Gruppennummer ist ein Begriff aus dem CAN-Protokoll SAE J1939. Sie fasst die Teiladressen PF und PS zusammen.

PID-Regler

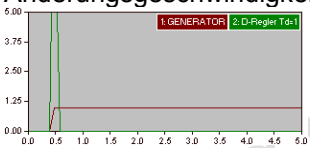
P = Proportional-Anteil
Der P-Regler besteht ausschließlich aus einem proportionalen Anteil der Verstärkung K_p. Mit seinem Ausgangssignal ist er proportional dem Eingangssignal.



I = Integral-Anteil
Ein I-Regler wirkt durch zeitliche Integration der Regelabweichung auf die Stellgröße mit der Gewichtung durch die Nachstellzeit T_N



D = Differential-Anteil
Der D-Regler reagiert nicht auf die Regelabweichung, sondern nur auf deren Änderungsgeschwindigkeit.



Piktogramm

Piktogramme sind bildhafte Symbole, die eine Information durch vereinfachte grafische Darstellung vermitteln.

→ Kapitel *Was bedeuten die Symbole und Formatierungen?* (→ Seite 7)

PL

Performance-**L**evel
Ist nach ISO 13849-1 eine Einstufung (PL a...e) der Fähigkeit von sicherheitsrelevanten Teilen einer Steuerung, eine Sicherheitsfunktion unter vorhersehbaren Bedingungen auszuführen.
→ Kapitel *Performance-Level PL*

PLr

Mit dem "erforderlichen Performance-Level" PL_r wird nach ISO 13849 die erforderliche Risikominderung für jede Sicherheitsfunktion erreicht.

Für jede gewählte Sicherheitsfunktion, die durch ein SRP/CS ausgeführt wird, muss ein PL_r festgelegt und dokumentiert werden. Die Bestimmung des PL_r ist das Ergebnis der Risikobeurteilung, bezogen auf den Anteil der Risikominderung durch die sicherheitsrelevanten Teile der Steuerung.

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.

prepared

prepared (engl.) = vorbereitet (auch: angehalten)

Betriebszustand eines CANopen-Teilnehmers. In diesem Modus werden nur NMT-Kommandos übertragen.

Programmiersprache, sicherheitsrelevant

Für sicherheitsrelevante Applikationen sollten nur folgende Programmiersprachen verwendet werden:

- Programmiersprache mit eingeschränktem Sprachumfang (LVL = **l**imited **v**ariability **l**anguage), kann vordefinierte, applikations-spezifische Bibliotheksfunktionen kombinieren.
  In CoDeSys sind das Kontaktplan KOP (Ladder Diagram LD) und Funktionsplan FUP (Function block diagram FBD).
- Programmiersprache mit nicht eingeschränktem Sprachumfang (FVL =

full **v**ariability **l**anguage), kann einen großen Bereich von Funktionen kombinieren.
Dazu gehören z.B. C, C++, Assembler. In CoDeSys ist das Strukturierter Text (ST).

► Strukturierter Text ist ausschließlich in gesonderten, zertifizierten Funktionen zu empfehlen, normalerweise in Embedded Software.

► Im "normalen" Applikations-Programm sollten nur KOP (LD) und FUP (FBD) eingesetzt werden. Damit sollen die folgenden Mindestanforderungen erfüllt werden können.

Generell werden folgende Mindestanforderungen an sicherheitsrelevante Applikations-Software (SRASW) gestellt:

► Programm modular und klar strukturieren. Folge: einfache Testbarkeit.

► Funktionen verständlich darstellen:
- für den Operator auf dem Bildschirm (Navigation),
- Lesbarkeit des späteren Dokumentationsausdrucks.

► Symbolische Variablen verwenden (keine IEC-Adressen).

► Variablennamen und Kommentare aussagekräftig formulieren.

► Einfache Funktionen verwenden (keine indirekte Adressierung, keine Variablenfelder).

► Defensiv programmieren.

► Leichtes Erweitern oder Anpassen des Programms ermöglichen.

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

Via PWM kann ein (vom Gerät dazu befähigter) digitaler Ausgang mittels regelmäßiger, schneller Impulse eine beinahe analoge Spannung ausgeben. 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.
→ Kapitel *PWM-Signalverarbeitung* (→ Seite 223)
→ Kapitel *Was macht ein PWM-Ausgang?*

R

Ratio

Ratio (lat.) = Verhältnis

Messungen können auch ratiometrisch erfolgen = Verhältnismessung. Das Eingangssignal erzeugt ein Ausgangssignal, das in einem bestimmten Verhältnis zu ihm liegt. Das bedeutet, ohne zusätzliche Referenzspannung können analoge Eingangssignale ausgewertet werden. Ein Schwanken der Versorgungsspannung hat auf diesen Messwert dann keinen Einfluss.
→ Kapitel *Zählerfunktionen zur Frequenz- und Periodendauermessung* (→ Seite 207)

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.

redundant

Redundanz ist das Vorhandensein von mehr als den notwendigen Mitteln, damit eine

Funktionseinheit eine geforderte Funktion ausführt oder damit Daten eine Information darstellen können.

Man unterscheidet verschiedene Arten der Redundanz:

- Die funktionelle Redundanz zielt darauf ab, sicherheitstechnische Systeme mehrfach parallel auszulegen, damit beim Ausfall einer Komponente die anderen den Dienst gewährleisten.
- Zusätzlich versucht man, die redundanten Systeme voneinander räumlich zu trennen. Dadurch minimiert man das Risiko, dass sie einer gemeinsamen Störung unterliegen.
- Schließlich verwendet man manchmal Bauteile unterschiedlicher Hersteller, um zu vermeiden, dass ein systematischer Fehler sämtliche redundanten Systeme ausfallen lässt (diversitäre Redundanz).

Die Software von redundanten Systemen sollte sich möglichst in den folgenden Aspekten unterscheiden:

- Spezifikation (verschiedene Teams),
- Spezifikationssprache,
- Programmierung (verschiedene Teams),
- Programmiersprache,
- Compiler.

remanent

Remanente Daten sind gegen Datenverlust bei Spannungsausfall geschützt.

Z.B. kopiert das Betriebssystem die remanenten Daten automatisch in einen Flash-Speicher, sobald die Spannungsversorgung unter einen kritischen Wert sinkt. Bei Wiederkehr der Spannungsversorgung lädt das Betriebssystem die remanenten Daten zurück in den Arbeitsspeicher.

Dagegen sind die Daten im Arbeitsspeicher einer Steuerung flüchtig und bei Unterbrechung der Spannungsversorgung normalerweise verloren.

Restrisiko

Das ist das verbleibende Risiko, nachdem Schutzmaßnahmen ergriffen wurden. Vor dem Restrisiko muss in Betriebsanleitungen und an der Maschine deutlich gewarnt werden.

Risiko

Als Risiko gilt die Kombination der Wahrscheinlichkeit des Eintritts eines Schadens und des Ausmaßes des Schadens.

Risikoanalyse

Kombination aus ...

- Festlegung der Grenzen der Maschine (Verwendungszweck, zeitliche Grenzen),
- Identifizierung der Gefährdung (Eingreifen von Personen, Betriebszustände der Maschine, vorhersehbarer Missbrauch) und
- der Risikoeinschätzung (Verletzungsgrad, Schadensumfang, Häufigkeit und Dauer der Gefahr, Eintrittswahrscheinlichkeit, Möglichkeit zur Vermeidung oder Begrenzung des Schadens).

Risikobeurteilung

Das ist die Gesamtheit des Verfahrens, das die Risikoanalyse und die Risikobewertung umfasst.

Nach Maschinenrichtlinie 2006/42/EG gilt: "Der Hersteller einer Maschine oder sein Bevollmächtigter hat dafür zu sorgen, dass eine Risikobeurteilung vorgenommen wird, um die für die Maschine geltenden Sicherheits- und Gesundheitsanforderungen zu ermitteln. Die Maschine muss dann unter Berücksichtigung der Ergebnisse der Risikobeurteilung konstruiert und gebaut werden." (→ Anhang 1, Allgemeine Grundsätze)

Risikobewertung

Das ist die auf der Risikoanalyse beruhende Beurteilung, ob die Ziele zur Risikominderung erreicht wurden.

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.

Rückstellung, manuell

Die manuelle Rückstellung ist eine interne Funktion des SRP/CS zum anuellen Wiederherstellen einer oder mehrerer Sicherheitsfunktionen. Wird vor dem Neustart einer Maschine verwendet.

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.
→ CiA DS 402

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"

Schaden

Als Schaden bezeichnet man eine physische Verletzung oder Schädigung der Gesundheit.

Schutzmaßnahme

Maßnahme zur vorgesehenen Minderung des Risikos, z.B.:
- fehlerausschließender Entwurf,
- technische Schutzmaßnahme (trennende Schutzeinrichtung),
- ergänzende Schutzmaßnahme (Benutzerinformation),
- persönliche Schutzausrüstung (Helm, Schutzbrille).

SCT

Bei CANopen-Safety überprüft die Sicherheits-Zykluszeit SCT (**S**afeguard **c**ycle **t**ime) die korrekte Funktion der periodischen Übertragung (Daten-Refresh) der SRDOs. Die Daten müssen innerhalb der eingestellten Zeit wiederholt worden sein, um gültig zu sein. Andernfalls signalisiert die empfangene Steuerung einen Fehler und geht in den sicheren Zustand (= Ausgänge abgeschaltet).

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.

SDO ist eine Spezifikation für eine herstellerunabhängige Datenstruktur zum einheitlichen Datenzugriff. Dabei fordern "Clients" die gewünschten Daten von "Servern" an. Die SDOs bestehen immer aus 8 Bytes. Längere Datenpakete werden auf mehrere Nachrichten verteilt.

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), auf dem entnommen werden kann, was getestet wurde und ob das Ergebnis positiv oder negativ ist.

Sicherheitsfunktion

Der Ausfall einer Sicherheitsfunktion einer Maschine kann zum unmittelbar erhöhten Risiko führen. Der Konstrukteur einer solchen Maschine muss daher:
- einen Ausfall der Sicherheitsfunktion sicher verhindern,
- einen Ausfall der Sicherheitsfunktion rechtzeitig sicher erkennen,
- Maschine bei einem Ausfall der Sicherheitsfunktion rechtzeitig in einen sicheren Zustand bringen.

Sicherheits-Normentypen

Sicherheitsnormen auf dem Gebiet der Maschinen sind wie folgt strukturiert:

Typ-A-Normen (Sicherheits-Grundnormen) behandeln Grundbegriffe, Entwurfsleitsätze und allgemeine Aspekte, die auf Maschinen angewendet werden können. Beispiele: Terminologie Methodik (ISO 12100-1), Technische Prinzipien (ISO 12100-2), Risikobeurteilung (ISO 14121), ...

Typ-B-Normen (Sicherheits-Fachgrundnormen)
behandeln einen Sicherheitsaspekt oder eine Art von Schutzeinrichtungen, die für eine Reihe von Maschinen verwendet werden können.

- Typ-B1-Normen für bestimmte Sicherheitsaspekte. Beispiele: Sicherheits-Abstände (EN 294), Arm-/Hand-Geschwindigkeiten (EN 999), Sicherheitsbezogene Teile von Steuerungen (ISO 13849), Temperaturen, Lärm, ...
- Typ-B2-Normen für Schutzeinrichtrungen. Beispiele: Not-Aus-Schaltungen ((ISO 13850), Zweihand-Schaltungen, trennende oder berührungslos wirkende Schutzeinrichtungen (ISO 61496), ...

Typ-C-Normen (Maschinensicherheitsnormen) behandeln detaillierte Sicherheitsanforderungen an eine bestimmte Maschine oder Maschinengruppen.

SIL

Der Sicherheits-Integritätslevel SIL ist nach IEC 62061 eine Einstufung (SIL CL 1...4) der Sicherheitsintegrität der Sicherheitsfunktionen. Er dient der Beurteilung elektrischer/elektronischer/programmierbar elektronischer (E/E/PE)-Systeme in Bezug auf die Zuverlässigkeit von Sicherheitsfunktionen. Aus dem angestrebten Level ergeben sich die sicherheitsgerichteten Konstruktionsprinzipien, die eingehalten werden müssen, damit das Risiko einer Fehlfunktion minimiert werden kann.

Slave

Passiver Teilnehmer am Bus, antwortet nur auf Anfrage des →Masters. Slaves haben im Bus eine eindeutige und einmalige →Adresse.

SRDO

Über SRDOs (**S**afety-**R**elated **D**ata **O**bjects = *Sicherheitsrelevante Datenobjekte*) werden die sicheren Daten ausgetauscht. Ein SRDO besteht immer aus zwei CAN-Nachrichten mit unterschiedlichen Identifiern:

- Nachricht 1 enthält die Originalanwenderdaten,
- Nachricht 2 enthält die gleichen Daten, die aber bitweise invertiert werden.

SRP/CS

Safety-**R**elated **P**art of a **C**ontrol **S**ystem = Sicherheitsrelevanter Teil einer Steuerung

SRP/CS ist ein Teil einer Steuerung, das auf sicherheitsgerichtete Eingangssignale reagiert und sicherheitsgerichtete Ausgangssignale erzeugt. Die Kombination sicherheitsrelevanter

Teile einer Steuerung beginnt an dem Punkt, an dem sicherheitsgerichtete Signale erzeugt werden (einschließlich Betätiger z.B. eines Positionsschalters) und endet an den Ausgängen der Leistungssteuerungselemente (einschließlich z.B. der Hauptkontakte eines Schützes).

SRVT

Die sicherheitsrelevante Objekt-Gültigkeitsdauer SRVT (**S**afety-**R**elated **O**bject **V**alidation **T**ime) sorgt bei CANopen-Safety dafür, dass die Zeit zwischen den SRDO-Nachrichten-Paaren eingehalten wird:

Nur wenn die redundante, invertierte Nachricht innerhalb der eingestellten Zeit SRVT nach der Original-Nachricht übertragen wurde, sind die damit übertragenen Daten gültig. Andernfalls signalisiert die empfangende Steuerung einen Fehler und geht in den sicheren Zustand (= Ausgänge abgeschaltet).

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.

Symbole

Piktogramme sind bildhafte Symbole, die eine Information durch vereinfachte grafische Darstellung vermitteln.

→ Kapitel *Was bedeuten die Symbole und Formatierungen?* (→ Seite 7)

Symbole und Formatierungen

Ein Link ist ein Querverweis zu einer anderen Stelle im Dokument oder auf ein externes Dokument.

Systemvariable

Variable, auf die via IEC-Adresse oder Symbolname aus der SPS zugegriffen werden kann.

T

Target

Das Target gibt das Zielsystem an, auf dem das SPS-Programm laufen soll. Im Target sind die Dateien (Treiber und ggf. spezifische Hilfedateien) enthalten, die zum Programmieren und Parametrieren erforderlich sind.

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 Applikations-Programm.

Testrate rt

Die Testrate r_t ist die Häufigkeit der automatischen Tests, um Fehler in einem SRP/CS rechtzeitig zu bemerken.

U

Überwachung

Die Überwachung ist eine Sicherheitsfunktion, die sicherstellt, dass eine Schutzmaßnahme eingeleitet wird, sobald Folgendes eintritt:

- Die Fähigkeit eines Bauteils oder eines Elements, seine Funktion auszuführen, wird vermindert.
- Die Betriebsbedingungen werden so verändert, dass das resultierende Risiko steigt.

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 Applikation 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. Diese Dienste sind vom Protokoll her nicht bestätigte Dienste, d.h. 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.

wo

wo = write only (engl.) = nur schreiben

Unidirektionale Datenübertragung: Daten können nur verändert werden, jedoch nicht gelesen.

Z

Zustand, sicher

Der Zustand einer Maschine gilt als sicher, wenn von ihr keine Gefährdung mehr ausgeht. Dies ist meist der Fall, wenn alle gefahrbringenden Bewegungsmöglichkeiten abgeschaltet sind und nicht unerwartet wieder anlaufen können.

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.

14 Index

15 ifm weltweit • ifm worldwide • ifm à l'échelle internationale

8310

Stand: 2010-10-08

http://www.ifm.com • E-Mail: info@ifm.com

Service-Hotline: 0800 16 16 16 4 (nur Deutschland, Mo...Fr, 07.00...18.00 Uhr)

ifm Niederlassungen • Sales offices • Agences

D	ifm electronic gmbh Vertrieb Deutschland Niederlassung Nord • 31135 Hildesheim • Tel. 0 51 21 / 76 67-0 Niederlassung West • 45128 Essen • Tel. 02 01 / 3 64 75 -0 Niederlassung Mitte-West • 58511 Lüdenscheid • Tel. 0 23 51 / 43 01-0 Niederlassung Süd-West • 64646 Heppenheim • Tel. 0 62 52 / 79 05-0 Niederlassung Baden-Württemberg • 73230 Kirchheim • Tel. 0 70 21 / 80 86-0 Niederlassung Bayern • 82178 Puchheim • Tel. 0 89 / 8 00 91-0 Niederlassung Ost • 07639 Tautenhain • Tel. 0 36 601 / 771-0 ifm electronic gmbh • Friedrichstraße 1 • 45128 Essen
A	ifm electronic gmbh • 1120 Wien • Tel. +43 16 17 45 00
AUS	ifm efector pty ltd. • Mulgrave Vic 3170 • Tel. +61 3 00 365 088
B, L	ifm electronic N.V. • 1731 Zellik • Tel. +32 2 / 4 81 02 20
BR	ifm electronic Ltda. • 03337-000, Sao Paulo SP • Tel. +55 11 / 2672-1730
CH	ifm electronic ag • 4 624 Härkingen • Tel. +41 62 / 388 80 30
CN	ifm electronic Co. Ltd. • 201210 Shanghai • Tel. +86 21 / 5027 8559
CND	ifm efector Canada inc. • Oakville, Ontario L6K 3V3 • Tel. +1 800-441-8246
CZ	ifm electronic spol. s.r.o. • 25243 Průhonice • Tel. +420 267 990 211
DK	ifm electronic a/s • 2605 BROENDBY • Tel. +45 70 20 11 08
E	ifm electronic s.a. • 08820 El Prat de Llobregat • Tel. +34 93 479 30 80
F	ifm electronic s.a. • 93192 Noisy-le-Grand Cedex • Tél. +33 0820 22 30 01
FIN	ifm electronic oy • 00440 Helsinki • Tel . +358 75 329 5000
GB, IRL	ifm electronic Ltd. • Hampton, Middlesex TW12 2HD • Tel. +44 208 / 213-0000
GR	ifm electronic Monoprosopi E.P.E. • 15125 Amaroussio • Tel. +30 210 / 6180090
H	ifm electronic kft. • 9028 Györ • Tel. +36 96 / 518-397
I	ifm electronic s.a. • 20041 Agrate-Brianza (MI) • Tel. +39 039 / 68.99.982
IL	Astragal Ltd. • Azur 58001 • Tel. +972 3 -559 1660
IND	ifm electronic India Branch Office • Kolhapur, 416234 • Tel. +91 231-267 27 70
J	efector co., ltd. • Togane-shi, Chiba 283-0826 • Tel. +81 475-50-3003
MAL	ifm electronic Pte. Ltd • 80250 Johor Bahru Johor • Tel. +60 7 / 331 5022
MEX	ifm efector S. de R. L. de C. V. • Monterrey, N. L. 64630 • Tel. +52 81 8040-3535
N	Sivilingeniør J. F. Knudtzen A/S • 1396 Billingstad • Tel. +47 66 / 98 33 50
NL	ifm electronic b.v. • 3843 GA Harderwijk • Tel. +31 341 / 438 438
P	ifm electronic s.a. • 4430-208 Vila Nova de Gaia • Tel. +351 223 / 71 71 08
PL	ifm electronic Sp. z o.o. • 40-524 Katowice • Tel. +48 32-608 74 54
RA, ROU	ifm electronic s.r.l. • 1107 Buenos Aires • Tel. +54 11 / 5353 3436
ROK	ifm electronic Ltd. • 140-884 Seoul • Tel. +82 2 / 790 5610
RP	Gram Industrial, Inc. • 1770 Mantilupa City • Tel. +63 2 / 850 22 18
RUS	ifm electronic • 105318 Moscow • Tel. +7 495 921-44-14
S	ifm electronic a b • 512 60 Överlida • Tel. +46 325 / 661 500
SGP	ifm electronic Pte. Ltd. • Singapore 609 916 • Tel. +65 6562 8661/2/3
SK	ifm electronic s.r.o. • 835 54 Bratislava • Tel. +421 2 / 44 87 23 29
THA	Sang Chai Meter Co., Ltd. • Bangkok 10 400 • Tel. +66 2 / 616 80 51
TR	ifm electronic Ltd. Sti. • 34381 Sisli/Istanbul • Tel. +90 212 / 210 50 80
UA	TOV ifm electronic • 02660 Kiev • Tel. +380 44 501 8543
USA	ifm efector inc. • Exton, PA 19341 • Tel. +1 610 / 5 24-2000
ZA	ifm electronic (Pty) Ltd. • 0157 Pretoria • Tel. +27 12 345 44 49

Technische Änderungen behalten wir uns ohne vorherige Ankündigung vor.
We reserve the right to make technical alterations without prior notice.
Nous nous réservons le droit de modifier les données techniques sans préavis.