



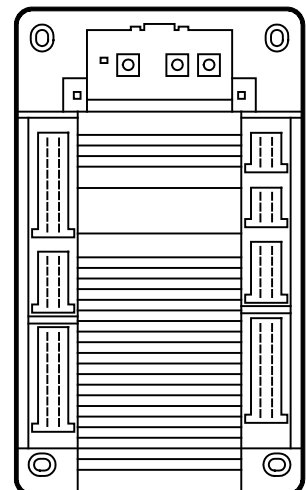
Original-Programmierhandbuch
CabinetController

ecomat100
CR0303



Laufzeitsystem V05
CODESYS® V2.3

Deutsch



Inhaltsverzeichnis

1	Über diese Anleitung	5
1.1	Copyright.....	5
1.2	Übersicht: Dokumentations-Module für ecomatmobile-Geräte.....	6
1.3	CODESYS-Programmierhandbuch.....	6
1.4	Was bedeuten die Symbole und Formatierungen?	7
1.5	Wie ist diese Dokumentation aufgebaut?	8
1.6	Historie der Anleitung (CR030n).....	9
2	Sicherheitshinweise	10
2.1	Beachten!	10
2.2	Welche Vorkenntnisse sind notwendig?	11
2.3	Anlaufverhalten der Steuerung	11
3	Systembeschreibung	12
3.1	Angaben zum Gerät.....	12
3.2	Hardware-Beschreibung	12
3.2.1	Hardware-Aufbau	13
3.2.2	Überwachungskonzept.....	15
3.2.3	Eingänge (Technologie)	17
3.2.4	Ausgänge (Technologie)	21
3.2.5	Hinweise zur Anschlussbelegung.....	24
3.2.6	Sicherheitshinweise zu Reed-Relais	24
3.2.7	Status-LED	25
3.3	Schnittstellen-Beschreibung	26
3.3.1	Serielle Schnittstelle	26
3.3.2	CAN-Schnittstellen	27
3.4	Software	28
3.4.1	Software-Module für das Gerät	28
3.4.2	Programmierhinweise für CODESYS-Projekte.....	31
3.4.3	Betriebszustände.....	35
3.4.4	Betriebsmodi	39
3.4.5	Leistungsgrenzen des Geräts.....	40
4	Konfigurationen	41
4.1	Laufzeitsystem einrichten	41
4.1.1	Laufzeitsystem neu installieren	42
4.1.2	Laufzeitsystem aktualisieren	43
4.1.3	Installation verifizieren.....	43
4.2	Programmiersystem einrichten	44
4.2.1	Programmiersystem manuell einrichten	44
4.2.2	Programmiersystem über Templates einrichten	47
4.3	Funktionskonfiguration, allgemein	48
4.3.1	Konfiguration der Ein- und Ausgänge (Voreinstellung).....	48
4.3.2	Systemvariablen.....	48
4.4	Funktionskonfiguration der Ein- und Ausgänge.....	49
4.4.1	Konfiguration der Ein- und Ausgänge (Voreinstellung).....	49
4.4.2	Eingänge konfigurieren.....	50
4.4.3	Ausgänge konfigurieren.....	53
4.5	Variablen	55
4.5.1	Retain-Variablen.....	56
4.5.2	Netzwerkvariablen.....	56

5	ifm-Funktionselemente	57
5.1	ifm-Bibliotheken für das Gerät CR0303	57
5.1.1	Bibliothek ifm_CR0303_V05yyzz.LIB	58
5.1.2	Bibliothek ifm_CAN1_EXT_Vxxyzz.LIB	60
5.1.3	Bibliothek ifm_CR0303_CANOpenMaster_V04yynn.LIB	60
5.1.4	Bibliothek ifm_CR0303_CANOpenSlave_V04yynn.LIB	61
5.1.5	Bibliothek ifm_J1939_x_Vxxyzz.LIB	61
5.1.6	Bibliothek ifm_hydraulic_CR0303_V01yyzz.LIB	62
5.2	ifm-Bausteine für das Gerät CR0303	63
5.2.1	Bausteine: CAN Layer 2	63
5.2.2	Bausteine: CANOpen-Master	80
5.2.3	Bausteine: CANOpen-Slave	89
5.2.4	Bausteine: CANOpen SDOs	97
5.2.5	Bausteine: SAE J1939	102
5.2.6	Bausteine: serielle Schnittstelle	114
5.2.7	Bausteine: SPS-Zyklus optimieren	119
5.2.8	Bausteine: Eingangswerte verarbeiten	126
5.2.9	Bausteine: analoge Werte anpassen	130
5.2.10	Bausteine: Zählerfunktionen zur Frequenz- und Periodendauermessung	133
5.2.11	Bausteine: PWM-Funktionen	143
5.2.12	Bausteine: Hydraulikregelung	150
5.2.13	Bausteine: Regler	163
5.2.14	Bausteine: Software-Reset	172
5.2.15	Bausteine: Zeit messen / setzen	174
5.2.16	Bausteine: Daten im Speicher sichern, lesen und wandeln	177
5.2.17	Bausteine: Datenzugriff und Datenprüfung	185
6	Diagnose und Fehlerbehandlung	192
6.1	Diagnose	192
6.2	Fehler	192
6.3	Reaktion im Fehlerfall	192
6.4	Reaktion auf System-Fehler	193
6.5	CAN / CANOpen: Fehler und Fehlerbehandlung	193
7	Anhang	194
7.1	Systemmerker	194
7.1.1	Systemmerker: CAN	195
7.1.2	Systemmerker: Fehlermerker	196
7.1.3	Systemmerker: LED (Standard-Seite)	197
7.1.4	Systemmerker: Spannungen	198
7.1.5	Systemmerker: 24 Eingänge und 18 Ausgänge	199
7.2	Adressbelegung und E/A-Betriebsarten	200
7.2.1	Adressbelegung Ein-/Ausgänge	200
7.2.2	Mögliche Betriebsarten Ein-/Ausgänge	205
7.3	Fehler-Tabellen	208
7.3.1	Fehlermerker	208
7.3.2	Fehler: CAN / CANOpen	208

Inhalt

8	Begriffe und Abkürzungen	209
9	Index	223
10	Notizen • Notes • Notes	227
11	ifm weltweit • ifm worldwide • ifm à l'échelle internationale	231

1 Über diese Anleitung

Inhalt

Copyright	5
Übersicht: Dokumentations-Module für ecomatmobile-Geräte	6
CODESYS-Programmierhandbuch	6
Was bedeuten die Symbole und Formatierungen?	7
Wie ist diese Dokumentation aufgebaut?	8
Historie der Anleitung (CR030n)	9

202

1.1 Copyright

6088

© Alle Rechte bei **ifm electronic gmbh**. Vervielfältigung und Verwertung dieser Anleitung, auch auszugsweise, nur mit Zustimmung der **ifm electronic gmbh**.

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

- AS-i ist Eigentum der AS-International Association, (→ www.as-interface.net)
- CAN ist Eigentum der CiA (CAN in Automation e.V.), Deutschland (→ www.can-cia.org)
- CODESYS™ ist Eigentum der 3S – Smart Software Solutions GmbH, Deutschland (→ www.codesys.com)
- DeviceNet™ ist Eigentum der ODVA™ (Open DeviceNet Vendor Association), USA (→ www.odva.org)
- EtherNet/IP® ist Eigentum der →ODVA™
- IO-Link® (→ www.io-link.com) ist Eigentum der →PROFIBUS Nutzerorganisation e.V., Deutschland
- Microsoft® ist Eigentum der Microsoft Corporation, USA (→ www.microsoft.com)
- PROFIBUS® ist Eigentum der PROFIBUS Nutzerorganisation e.V., Deutschland (→ www.profibus.com)
- PROFINET® ist Eigentum der →PROFIBUS Nutzerorganisation e.V., Deutschland
- Windows® ist Eigentum der →Microsoft Corporation, USA

1.2 Übersicht: Dokumentations-Module für ecomatmobile-Geräte

17405

Die Dokumentation für **ecomatmobile**-Geräte besteht aus folgenden Modulen:

1.	Datenblatt
Inhalt:	Technische Daten in Tabellenform
Quelle:	→ www.ifm.com > Land wählen > [Datenblattsuche] > CR0303 > [Technische Daten im PDF-Format]
2.	Montageanleitung / Betriebsanleitung
Inhalt:	Anleitung für Montage, elektrische Installation, (Inbetriebnahme*), Technische Daten
Quelle:	Anleitung wird mit dem Gerät mitgeliefert Auch zu finden auf der ifm -Homepage: → www.ifm.com > Land wählen > [Datenblattsuche] > CR0303 > [Betriebsanleitungen]
3.	Programmierhandbuch + Online-Hilfe
Inhalt:	Beschreibung der Konfiguration und der Funktionen der Geräte-Software
Quelle:	→ www.ifm.com > Land wählen > [Datenblattsuche] > CR0303 > [Betriebsanleitungen]
4.	Systemhandbuch "Know-How ecomatmobile"
Inhalt:	Hintergrundwissen zu folgenden Themen: • Übersicht Templates und Demo-Programme • CAN, CANopen • Ausgänge steuern • User-Flash-Speicher • Visualisierungen • Übersicht Dateien und Bibliotheken
Quelle:	→ www.ifm.com > Land wählen > [Datenblattsuche] > CR0303 > [Betriebsanleitungen]

*) Die in Klammern gesetzten Beschreibungen sind nur in den Anleitungen bestimmter Geräte enthalten.

1.3 CODESYS-Programmierhandbuch

17542

Im ergänzenden "Programmierhandbuch CODESYS V2.3" der 3S GmbH erhalten Sie weitergehende Informationen über die Nutzung des Programmiersystems.

Dieses Handbuch steht auf der **ifm**-Homepage als kostenloser Download zur Verfügung:

→ www.ifm.com > Land wählen > [Service] > [Download] > [Systeme für mobile Arbeitsmaschinen]

Handbücher und Online-Hilfen für **ecomatmobile** finden Sie auch hier:

→ **ecomatmobile**-DVD "Software, tools and documentation"

1.4 Was bedeuten die Symbole und Formatierungen?

203

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

 WARNUNG	
Tod oder schwere irreversible Verletzungen sind möglich.	
 VORSICHT	
Leichte reversible Verletzungen sind möglich.	
ACHTUNG	
Sachschaden ist zu erwarten oder möglich.	
	Wichtige Hinweise auf Fehlfunktionen oder Störungen
	Weitere Hinweise
► ...	Handlungsaufforderung
> ...	Reaktion, Ergebnis
→ ...	"siehe"
abc	Querverweis
123	Dezimalzahl
0x123	Hexadezimalzahl
0b010	Binärzahl
[...]	Bezeichnung von Tasten, Schaltflächen oder Anzeigen

1.5 Wie ist diese Dokumentation aufgebaut?

204
1508

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

Und so finden Sie sich zurecht:

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

Bei Fehlfunktionen oder Unklarheiten setzen Sie sich bitte mit dem Hersteller in Verbindung:

→ 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 > [Datenblattsuche] > (Artikel-Nr.) > [Betriebsanleitungen]

1.6 Historie der Anleitung (CR030n)

9181

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

Datum	Thema	Änderung
2010-09-09	PID2 (FB)	Parameter der Eingänge korrigiert
2010-11-10	Abschlusswiderstände	Korrektur in Topic 1244
2011-02-14	TIMER_READ_US (FB)	Umrechnung max. Zählwert korrigiert
2011-04-05	Speicherbausteine FRAMREAD, FRAMWRITE, FLASHREAD, FLASHWRITE	zulässige Werte der Parameter SRC, LEN, DST
2011-04-13	CANopen Übersicht	neu: CANopen-Tabellen im Anhang
2011-04-14	CR0303 diverse Korrekturen:	• Gerät hat eine eigene Hydraulik-Bibliothek • einzelne Systemmerker nicht vorhanden • IEC-Adressen von Ein- und Ausgängen • Konfiguration der Eingänge • Status-LED im Anwendungsprogramm setzen
2011-05-24	CR0303: Speicherbausteine FRAMREAD, FRAMWRITE	zulässige Werte der Parameter SRC, DST korrigiert
2012-01-09	Speicherbausteine FRAMREAD, FRAMWRITE	vertauschte Parameter SRC, DST in der Tabelle "Zulässige Werte"
2012-10-04	diverse	Korrekturen
2013-06-24	diverse	neue Dokumentenstruktur
2014-04-28	diverse FBs	Beschreibung FB-Eingang CHANNEL präzisiert
2014-06-30	Name der Dokumentation	"Systemhandbuch" umbenannt zu "Programmierhandbuch"
2014-07-18	CR0303: Fehlermerker	falsch: ERROR_A_INx richtig: ERROR_Ix
2014-07-31	FB PHASE	Beschreibung Parameter der Ausgänge C, ET korrigiert
2014-08-26	Beschreibung Eingänge, Ausgänge	highside / lowside ersetzt durch plusschaltend / minusschaltend
2015-01-13	Dokumentationsstruktur Fehlercodes, Systemmerker	• Fehlermerker: nur noch im Anhang, Kapitel <i>Systemmerker</i> • CAN / CANopen Fehler und Fehlerbehandlung: nur noch im Systemhandbuch "Know-How" • Fehlercodes, EMCY-Codes: nun im Anhang, Kapitel <i>Fehler-Tabellen</i>
2015-03-10	Verfügbarer Speicher	Darstellung verbessert
2015-05-26	FB J1939_x_GLOBAL_REQUEST	Beschreibung präzisiert
2015-06-10	diverse FBs	Beschreibung FB-Eingang CHANNEL korrigiert

2 Sicherheitshinweise

Inhalt

Beachten!	10
Welche Vorkenntnisse sind notwendig?.....	11
Anlaufverhalten der Steuerung.....	11

213

2.1 Beachten!

214
11212

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

- ▶ Die Sicherheit der Maschine/Anlage muss auf jeden Fall eigenverantwortlich durch den Hersteller der Maschine/Anlage gewährleistet werden.
- ▶ Beachten Sie die nationalen Vorschriften des Landes, in welchem die Maschine/Anlage in Verkehr gebracht werden soll!



WARNUNG

Bei Nichtbeachten der Hinweise in dieser Anleitung sind Sach- oder Körperschäden möglich! 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.
- ▶ Die handelnde Person muss für die auszuführende Arbeit über die erforderliche Ausbildung und Qualifikation verfügen.
- ▶ 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]
- ▶ Beachten Sie die Korrekturen und Hinweise in den "Release-Notes" zur vorhandenen Hardware, Software und Dokumentation auf der **ifm**-Homepage:
→ www.ifm.com > Land wählen > [Datenblattsuche] > (Artikel-Nr.) > [Betriebsanleitungen]

5020

ACHTUNG

Der Treiberbaustein der seriellen Schnittstelle kann beschädigt werden!

Beim Trennen oder Verbinden 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 oder verbinden!

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.

Zum Programmieren der SPS 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.

2.3 Anlaufverhalten der Steuerung

15233

11575



WARNUNG

Gefahr durch unbeabsichtigtes und gefährliches Anlaufen von Maschinen- oder Anlagenteilen!

- ▶ Der Programmierer muss bei der Programmerstellung verhindern, dass nach Auftreten eines Fehlers (z.B. NOT-HALT) und der anschließenden Fehlerbeseitigung unbeabsichtigt Maschinen- oder Anlagenteile gefährlich anlaufen können!
⇒ Wiederanlaufsperr realisieren!
- ▶ Dazu im Fehlerfall die in Frage kommenden Ausgänge im Programm logisch abschalten!

Ein Wiederanlauf kann z.B. verursacht werden durch:

- Spannungswiederkehr nach Spannungsausfall
- Reset nach Watchdog-Ansprechen wegen zu langer Zykluszeit
- Fehlerbeseitigung nach NOT-HALT

So erreichen Sie sicheres Verhalten der Steuerung:

- ▶ Spannungsversorgung im Anwendungsprogramm überwachen.
- ▶ Im Fehlerfall alle relevanten Ausgänge im Anwendungsprogramm ausschalten.
- ▶ Aktuatoren, die zu gefahrbringenden Bewegungen führen können, zusätzlich im Anwendungsprogramm überwachen (Feedback).
- ▶ Relaiskontakte, die zu gefahrbringenden Bewegungen führen können, zusätzlich im Anwendungsprogramm überwachen (Feedback).
- ▶ Bei Bedarf im Anwendungsprojekt sicherstellen, dass verschweißte Relaiskontakte keine gefahrbringenden Bewegungen auslösen oder fortführen können.

6827

3 Systembeschreibung

Inhalt	
Angaben zum Gerät	12
Hardware-Beschreibung.....	12
Schnittstellen-Beschreibung	26
Software	28

975

3.1 Angaben zum Gerät

19966

Diese Anleitung beschreibt aus der Gerätefamilie für den mobilen Einsatz, **ecomatmobile** der **ifm electronic gmbh**:

- CabinetController: CR0303

3.2 Hardware-Beschreibung

Inhalt	
Hardware-Aufbau	13
Überwachungskonzept.....	15
Eingänge (Technologie)	17
Ausgänge (Technologie)	21
Hinweise zur Anschlussbelegung.....	24
Sicherheitshinweise zu Reed-Relais	24
Status-LED	25

14081

3.2.1 Hardware-Aufbau

Inhalt

Startvoraussetzung	13
Prinzipschaltung	13
Verfügbare Speicher	14

15332

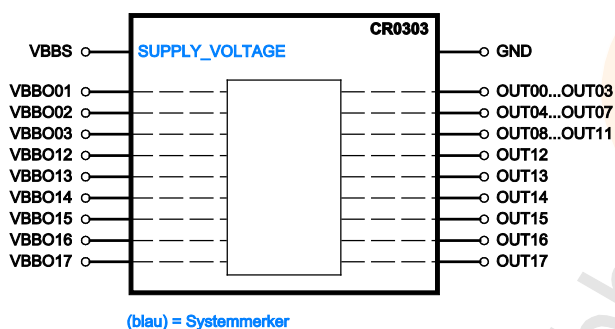
Startvoraussetzung

19971

Das Gerät startet erst, wenn am Versorgungsanschluss VBBS eine ausreichende Spannung anliegt.
Als ausreichende Spannung gilt > 10 V
Zulässige Betriebsspannung → Datenblatt

Prinzipschaltung

19969



Grafik: Prinzipaufbau der Versorgung

Verfügbarer Speicher

13736

FLASH-Speicher

9901

FLASH-Speicher (nichtflüchtiger, langsamer Speicher) insgesamt im Gerät vorhanden	1 MByte
--	---------

Davon sind folgende Speicherbereiche reserviert für ...

maximale Größe für das Anwendungsprogramm	576 kByte
Daten außerhalb des Anwendungsprogramms Anwender kann Daten speichern, z.B. Files, Bitmaps, Fonts	176 kByte
Daten außerhalb des Anwendungsprogramms Daten mit FLASHREAD (→ Seite 179) lesen oder mit FLASHWRITE (→ Seite 180) schreiben (bei Files: abzüglich 128 Byte für Header)	16 kByte

Der verbleibende Speicher ist reserviert für system-interne Zwecke.

SRAM

6754

SRAM (flüchtiger, schneller Speicher) insgesamt im Gerät vorhanden SRAM steht hier allgemein für alle Arten von flüchtigen, schnellen Speichern.	256 kByte
--	-----------

Davon sind folgende Speicherbereiche reserviert für ...

vom Anwendungsprogramm reservierte Daten	80 kByte
--	----------

Der verbleibende Speicher ist reserviert für system-interne Zwecke.

FRAM

12342

FRAM (nichtflüchtiger, schneller Speicher) insgesamt im Gerät vorhanden FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.	2 kByte
--	---------

Davon sind folgende Speicherbereiche reserviert für ...

im Anwendungsprogramm als VAR_RETAIN deklarierte Variablen	256 Byte
Vom Anwender frei verfügbarer remanenter Speicher Zugriff erfolgt über FRAMREAD (→ Seite 182) und FRAMWRITE (→ Seite 183)	1536 Byte

Der verbleibende Speicher ist reserviert für system-interne Zwecke.

3.2.2 Überwachungskonzept

19973

Die Steuerung überwacht die Versorgungsspannungen und die System-Fehlermerker. Je nach Zustand ...

- die Steuerung schaltet vollständig ab
 - > das Programm stoppt
 - > die Ausgänge werden stromlos und gehen auf logisch "0"
 - > die Status-LED erlischt

Überwachungs- und Sicherungsmechanismen

Inhalt	
Nach Einschalten der Versorgungsspannung	15
Wenn Laufzeitsystem / Anwendungsprogramm läuft.....	16
Wenn TEST-Pin nicht aktiv	16
Einmalige Mechanismen	16

3926

Für diese Geräte laufen automatisch folgende Überwachungen ab:

Nach Einschalten der Versorgungsspannung

3927

Nach dem Einschalten der Versorgungsspannung (Steuerung ist im Bootloader) laufen im Gerät folgende Tests ab:

- > RAM-Test (einmalig)
- > Versorgungsspannung
- > Systemdaten-Konsistenz
- > CRC des Bootloaders
- > wenn vorhanden und gestartet: CRC des Laufzeitsystems
- > wenn vorhanden und gestartet: CRC des Anwendungsprogramms
- > Speicherfehler:
 - Wenn Test aktiv: Merker ERROR_MEMORY = TRUE (kann ab dem ersten Zyklus ausgewertet werden).
 - Wenn Test nicht aktiv: rote LED leuchtet.

Wenn Laufzeitsystem / Anwendungsprogramm läuft

3928

Dann laufen zyklisch folgende Tests ab:

- > Watchdog triggern (100 ms)
anschließend kontinuierliche Ablaufkontrolle Watchdog
- > Kontinuierliche Temperaturkontrolle
Im Fehlerfall: Systemmerker ERROR_TEMPERATURE = TRUE
- > Kontinuierliche Spannungsüberwachung
Im Fehlerfall: Systemmerker ERROR_POWER = TRUE oder ERROR_VBBR = TRUE
- > Kontinuierliche CAN-Bus-Überwachung
- > Kontinuierliche Systemdaten-Überwachung:
 - Programm geladen,
 - Betriebsart RUN / STOP,
 - Laufzeitsystem geladen,
 - Node-ID,
 - Baudrate von CAN und RS232.
- > In Betriebsart RUN:
Zyklische E/A-Diagnose:
 - Kurzschluss,
 - Leiterbruch,
 - Überlast (Strom) der Ein- und Ausgänge,
 - Querschuss (nur bei SafetyController).

Wenn TEST-Pin nicht aktiv

3929

- > Schreibschutz für Systemdaten im FRAM ¹⁾, z.B.:
 - Laufzeitsystem geladen,
 - Kalibrierdaten.
 Realisiert über Hard- und Software.
- > Schreibschutz für Anwendungsprogramm (im Flash-Speicher)
- > DEBUG-Modus

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

Einmalige Mechanismen

3930

- > CRC-Überwachung bei Download oder Upload.
- > Überprüfung der Gerätezugehörigkeit von Laufzeitsystem und Anwendungsprogramm.

3.2.3 Eingänge (Technologie)

14090

Analog-Eingänge

2426

Die Analog-Eingänge können über das Anwendungsprogramm konfiguriert werden. Der Messbereich kann zwischen folgenden Bereichen umgeschaltet werden:

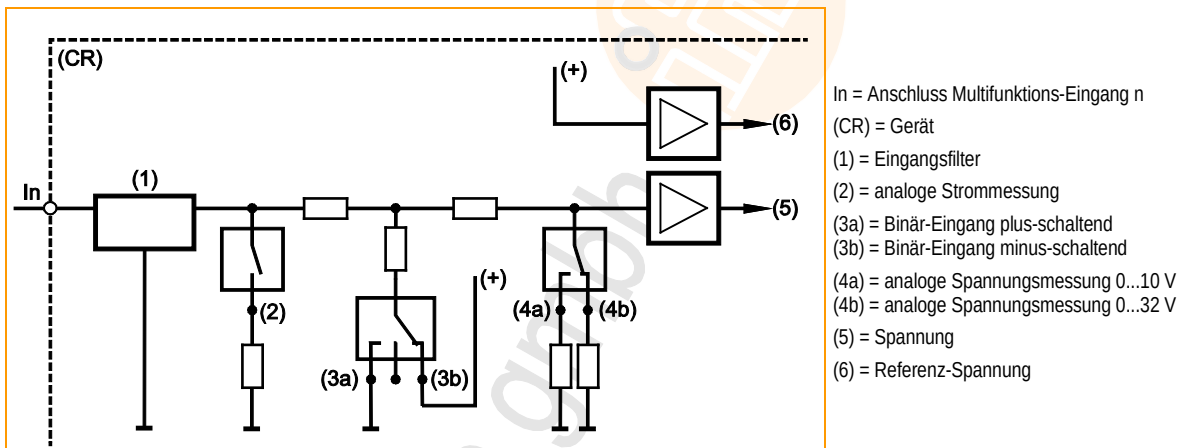
- Stromeingang 0...20 mA
- Spannungseingang 0...10 V
- Spannungseingang 0...32 V

Die Spannungsmessung kann auch ratiometrisch erfolgen (0...1000 ‰, über FBs einstellbar). Das bedeutet, ohne zusätzliche Referenzspannung können Potentiometer oder Joysticks ausgewertet werden. Ein Schwanken der Versorgungsspannung hat auf diesen Messwert keinen Einfluss.

Alternativ kann ein Analog-Kanal auch binär ausgewertet werden.

! Bei ratiometrischer Messung müssen die angeschlossenen Sensoren mit VBBS des Geräts versorgt werden. Dadurch werden Fehlmessungen durch Spannungsverschiebungen vermieden.

8971



Grafik: Prinzipschaltung Multifunktions-Eingang

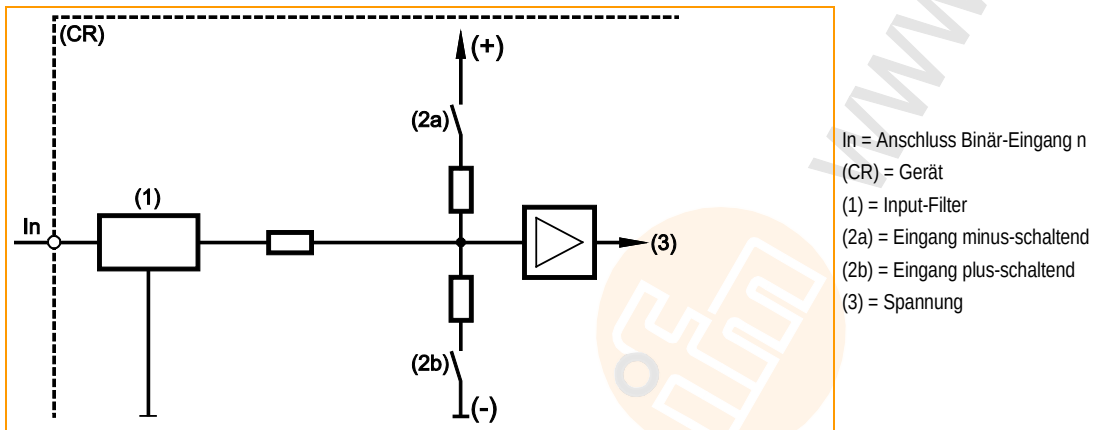
Binär-Eingänge

1015
7345

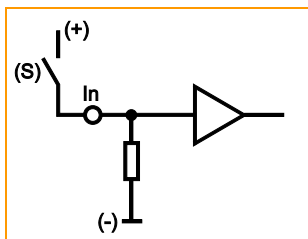
Der Binär-Eingang kann in folgenden Modi betrieben werden:

- binärer Eingang plus-schaltend (BL) für positives Gebersignal
- binärer Eingang, minus-schaltend (BH) für negatives Gebersignal

Je nach Gerät können auch die Binär-Eingänge unterschiedlich konfiguriert werden. Neben den Schutzmechanismen gegen Störungen werden die Binär-Eingänge intern über eine Analogstufe ausgewertet. Das ermöglicht die Diagnose der Eingangssignale. Im Anwendungsprogramm steht das Schaltsignal aber direkt als Bit-Information zur Verfügung.

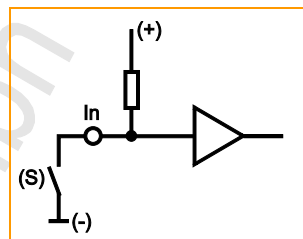


Grafik: Prinzipschaltung Binär-Eingang minus-schaltend / plus-schaltend für negative und positive Gebersignale



In = Anschluss Binär-Eingang n
(S) = Sensor

Prinzipische Schaltung Binär-Eingang plus-schaltend (BL)
für positives Sensorsignal:
Eingang = offen $\Rightarrow$ Signal = Low (Supply)



In = Anschluss Binär-Eingang n
(S) = Sensor

Prinzipische Schaltung Binär-Eingang minus-schaltend (BH)
für negatives Sensorsignal:
Eingang = offen $\Rightarrow$ Signal = High (GND)

Bei einem Teil dieser Eingänge ($\rightarrow$ Datenblatt) kann das Potential gewählt werden, gegen das geschaltet wird.

Eingangsgruppe I0 (IN00...07)

19976

Bei diesen Eingängen handelt es sich um eine Gruppe von Multifunktionskanälen.

Jeder einzelne dieser Eingänge ist wahlweise wie folgt konfigurierbar:

- binärer Eingang plus-schaltend (BL) für positives Gebersignal (mit/ohne Diagnose)

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))

- Die Konfiguration jedes einzelnen Eingangs erfolgt über das Anwendungsprogramm:
 - Konfigurationsbyte INxx_MODE

Diagnosefähige Sensoren nach NAMUR können ausgewertet werden.

Eingangsgruppe I1 (IN08...11 / FRQ00...03)

19979

Bei diesen Eingängen handelt es sich um eine Gruppe von Multifunktionskanälen.

Jeder einzelne dieser Eingänge ist wahlweise wie folgt konfigurierbar:

- binärer Eingang plus-schaltend (BL) für positives Gebersignal (mit/ohne Diagnose)
- schneller Eingang für z.B. Inkrementalgeber und Frequenz- oder Periodendauermessung

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))

Diagnosefähige Sensoren nach NAMUR können ausgewertet werden.

- Die Konfiguration jedes einzelnen Eingangs erfolgt über das Anwendungsprogramm:
 - Konfigurationsbyte INxx_MODE
 - schnelle Eingänge mit folgenden FBs:

<i>FAST_COUNT</i> (→ Seite 134)	Zählerbaustein für schnelle Eingangsimpulse
<i>FREQUENCY</i> (→ Seite 135)	misst die Frequenz des am gewählten Kanal ankommenden Signals
<i>INC_ENCODER</i> (→ Seite 136)	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
<i>PERIOD</i> (→ Seite 138)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
<i>PERIOD_RATIO</i> (→ Seite 140)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] über die angegebenen Perioden am angegebenen Kanal. Zusätzlich wird das Puls-/Periodenverhältnis in [%] angegeben.
<i>PHASE</i> (→ Seite 142)	liest ein Kanalpaar mit schnellen Eingängen ein und vergleicht die Phasenlage der Signale

Eingangsgruppe I2 (IN12...15)

19982

Bei diesen Eingängen handelt es sich um eine Gruppe von Multifunktionskanälen.

Jeder einzelne dieser Eingänge ist wahlweise wie folgt konfigurierbar:

- binärer Eingang plus-schaltend (BL) für positives Gebersignal
- binärer Eingang, minus-schaltend (BH) für negatives Gebersignal

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))

Diagnosefähige Sensoren nach NAMUR können ausgewertet werden.

Alle Eingänge zeigen das gleiche Verhalten bei Funktion und Diagnose.

 Detaillierte Beschreibung → Kapitel *Adressbelegung Ein-/Ausgänge* (→ Seite [200](#))

- Die Konfiguration jedes einzelnen Eingangs erfolgt über das Anwendungsprogramm:
 - Konfigurationsbyte INxx_MODE

Eingangsgruppe I3 (A_IN16...23)

19984

Bei diesen Eingängen handelt es sich um eine Gruppe von Multifunktionskanälen.

Jeder einzelne dieser Eingänge ist wahlweise wie folgt konfigurierbar:

- analoger Eingang 0...20 mA
- analoger Eingang 0...10 V
- analoger Eingang 0...32 V
- Spannungsmessung ratiometrisch 0...1000 ‰ von 32 V
- binärer Eingang plus-schaltend (BL) für positives Gebersignal (mit/ohne Diagnose)

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))

Alle Eingänge zeigen das gleiche Verhalten bei Funktion und Diagnose.

- Die Konfiguration jedes einzelnen Eingangs erfolgt über das Anwendungsprogramm:
 - Konfigurationsbyte A_INxx_MODE
 - FB *INPUT_ANALOG* (→ Seite [127](#)) > Eingang MODE
- > Werden die Analogeingänge auf Strommessung konfiguriert, wird bei Überschreiten des Endwertes (> 23 mA) in den sicheren Spannungsmessbereich (0...32 V DC) geschaltet und das jeweilige Fehlerbit im Merkerbyte ERROR_Ix gesetzt. Sinkt der Wert wieder unter den Grenzwert, schaltet der Eingang selbsttätig auf den Strommessbereich zurück.

3.2.4 Ausgänge (Technologie)

Inhalt	
Binär-Ausgänge.....	21
PWM-Ausgänge	21
Ausgangsgruppe Q0Q1 (OUT00...07).....	22
Ausgangsgruppe Q2 (OUT08...11)	22
Ausgangsgruppe Q3 (OUT12...17)	23

14093

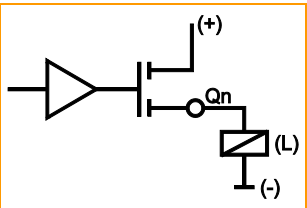
Binär-Ausgänge

19986

Bei den Geräte-Ausgängen sind folgende Betriebsarten möglich (→ Datenblatt):

- binärer Ausgang, plus-schaltend (BH), kurzschlussfest, überlastfest

15451



Qn = Anschluss Ausgang n
(L) = Last

Prinzipschaltung Binär-Ausgang plus-schaltend (BH)
für positives Ausgangssignal

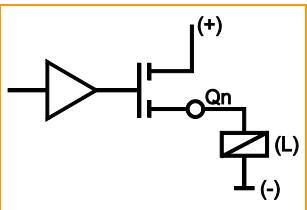
PWM-Ausgänge

14095

Bei den Geräte-Ausgängen sind folgende Betriebsarten möglich (→ Datenblatt):

- PWM-Ausgang, plus-schaltend (BH) ohne Diagnosefunktion

15451



Qn = Anschluss Ausgang n
(L) = Last

Prinzipschaltung Binär-Ausgang plus-schaltend (BH)
für positives Ausgangssignal

Ausgangsgruppe Q0Q1 (OUT00...07)

19988

Bei diesen Ausgängen handelt es sich um eine Gruppe von Multifunktionskanälen.

Jeder einzelne dieser Ausgänge ist wahlweise wie folgt konfigurierbar:

- binärer Ausgang, plus-schaltend (BH), kurzschlussfest, überlastfest
- analoger Ausgang mit Pulsweitenmodulation (PWM)

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))

- ▶ Die Konfiguration jedes einzelnen Ausgangs erfolgt über das Anwendungsprogramm:

PWM-Ausgang: → FB *PWM1000* (→ Seite [148](#))

- ▶  Zu den Grenzwerten unbedingt das Datenblatt beachten!

Ausgangsgruppe Q2 (OUT08...11)

19990

Bei diesen Ausgängen handelt es sich um eine Gruppe von Kanälen mit fest eingestellter Funktion.

Diese Ausgänge sind fix eingestellt wie folgt:

- binärer Ausgang, plus-schaltend (BH), kurzschlussfest, überlastfest

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))

- ▶  Zu den Grenzwerten unbedingt das Datenblatt beachten!

Ausgangsgruppe Q3 (OUT12...17)

19992

Bei diesen Ausgängen handelt es sich um eine Gruppe von Kanälen mit fest eingestellter Funktion.

Diese Ausgänge sind fix eingestellt wie folgt:

- binärer Ausgang, plus-schaltend (BH) mit Diagnosefunktion und Protection, kurzschlussfest, überlastfest

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))

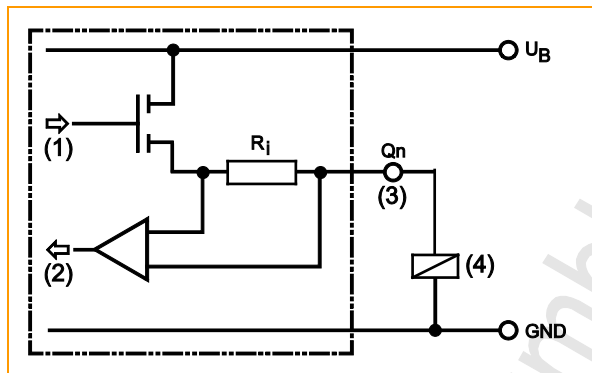
Schaltstrom	$\leq 10\text{ A}$
Überstrom Warnung ⇒ Systemmerker WARNING_OVERCURRENT_OUTxx	10...16,5 A (typisch 12 A)
Überstrom Abschaltung	13...21,5 A (typisch 16 A)

► **!** Zu den Grenzwerten unbedingt das Datenblatt beachten!

Diagnose: binäre Ausgänge (via Strommessung)

19398
19396

Die Diagnose dieser Ausgänge erfolgt über eine interne Strommessung im Ausgang:



Grafik: Prinzipschaltung

- (1) Ausgangskanal
- (2) Rücklesekanal für Diagnose
- (3) Anschluss Ausgang
- (4) Last

Diagnose: Überlast (via Strommessung)

19437
15249

Überlast kann nur an einem Ausgang mit Strommessung erkannt werden.

Überlast ist definiert als ...

"nominaler Maximalstrom laut Datenblatt + 12,5 %".

Diagnose: Leiterbruch (via Strommessung)

19400

Eine Leiterbruch-Erkennung erfolgt über den Rücklesekanal. Bei geschaltetem Ausgang ($Q_n = \text{TRUE}$) wird dann ein Leiterbruch erkannt, wenn über den Widerstand R_i kein Strom fließt (keine Spannung abfällt). Ohne den Leiterbruch fließt durch den Längswiderstand R_i der Laststrom und erzeugt damit einen Spannungsabfall, der über den Rücklesekanal ausgewertet wird.

Diagnose: Kurzschluss (via Strommessung)

19401

Eine Kurzschluss-Erkennung erfolgt über den Rücklesekanal. Bei geschaltetem Ausgang ($Q_n = \text{TRUE}$) wird dann ein Kurzschluss gegen GND erkannt, wenn über den Längswiderstand R_i die Versorgungsspannung abfällt.

3.2.5 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.

Die einzelnen Kürzel haben folgende Bedeutung:

A	Analog-Eingang
BH	Binärer highside-Eingang: minus-schaltend für negatives Sensorsignal Binärer highside-Ausgang: plus-schaltend für positives Ausgangssignal
BL	Binärer lowside-Eingang: plus-schaltend für positives Sensorsignal Binärer lowside-Ausgang: minus-schaltend für negatives Ausgangssignal
CYL	Eingang Periodendauermessung
ENC	Eingang Drehgebersignale
FRQ	Frequenzeingang
H-Bridge	Ausgang mit H-Brücken-Funktion
PWM	Pulsweiten-moduliertes Signal
PWMI	PWM-Ausgang mit Strommessung
IH	Impuls-/Zählereingang, highside, minus-schaltend für negatives Sensorsignal
IL	Impuls-/Zählereingang, lowside, plus-schaltend für positives Sensorsignal
R	Rücklesekanal für einen Ausgang

Zuordnung der Ein-/Ausgangskanäle: → Katalog, Montageanleitung oder Datenblatt

3.2.6 Sicherheitshinweise zu Reed-Relais

7348

Beim Einsatz von nichtelektronischen Schaltern Folgendes beachten:

! 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

3.2.7 Status-LED

1430

Die Betriebszustände werden durch die integrierte Status-LED (Default-Einstellung) angezeigt.

LED-Farbe	Blinkfrequenz	Beschreibung
aus	konstant aus	keine Betriebsspannung
Grün / schwarz	5 Hz	kein Laufzeitsystem geladen
Grün / schwarz	2 Hz	Anwendung RUN
Grün	konstant ein	Anwendung STOP
Rot / schwarz	2 Hz	Anwendung RUN mit Fehler
Rot	kurzzeitig ein	Fatal Error
Rot	konstant ein	Fatal Error (bei TEST-Eingang aktiv) ERROR STOP / SYSTEM STOP
Orange	kurzzeitig ein	Initialisierung oder Reset Checks

Die Betriebszustände STOP und RUN können vom Programmiersystem geändert werden.

LED im Anwendungsprogramm steuern

9989

Bei diesem Gerät kann die Status-LED auch durch das Anwendungsprogramm gesetzt werden. Dazu dienen folgende Systemvariablen (→ **Systemmarker** (→ Seite [194](#))):

LED	LED-Farbe für "aktiv" (für "Ein")
LED_X	LED-Farbe für "Pause" (für "Aus" oder andere Farbe)
---	Farbkonstante aus der Datenstruktur "LED_COLOR". Zulässige Einträge: LED_GREEN LED_BLUE LED_RED LED_WHITE LED_MAGENTA LED_CYAN LED_YELLOW LED_ORANGE LED_BLACK (= LED aus)
LED_MODE	Blinkfrequenz aus der Datenstruktur "LED_MODES". Zulässige Einträge: LED_2HZ LED_1HZ LED_05HZ (= 0,5 Hz) LED_0HZ (= konstant)

! HINWEIS

- Im Anwendungsprogramm NICHT die LED-Farbe ROT verwenden.
- > Im Fehlerfall wird die LED-Farbe ROT durch das Laufzeitsystem gesetzt.
ABER: Werden die Farben und/oder Blinkmodi im Anwendungsprogramm geändert, gilt die obige Tabelle der Voreinstellung nicht mehr.

3.3 Schnittstellen-Beschreibung

Inhalt

Serielle Schnittstelle	26
CAN-Schnittstellen	27

14098

3.3.1 Serielle Schnittstelle

14099

Dieses Gerät bietet eine serielle Schnittstelle.

Grundsätzlich kann die serielle Schnittstelle mit folgenden Funktionen genutzt werden:

- Programm-Download
- Debugging
- freie Nutzung in der Anwendung

HINWEIS

Voreingestellt steht die serielle Schnittstelle dem Anwender nicht zur Verfügung, da sie für den Programm-Download und das Debugging genutzt wird.

Setzt der Anwender das Systemmerkerbit `SERIAL_MODE=TRUE`, dann kann die Schnittstelle frei genutzt werden. Ein Debugging des Anwendungsprogramms ist dann nur noch über eine der 4 CAN-Schnittstellen oder über USB möglich.

Anschlüsse und Daten → Datenblatt

3.3.2 CAN-Schnittstellen

Inhalt

CAN: Schnittstellen und Protokolle	27
--	----

14101

Anschlüsse und Daten → Datenblatt

CAN: Schnittstellen und Protokolle

19523
14587

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

- Layer 2: CAN auf Ebene 2 (→ Kapitel *Bausteine: CAN Layer 2* (→ Seite [63](#)))
- CANopen-Master (→ Kapitel *Bausteine: CANopen-Master* (→ Seite [80](#)))
- CANopen-Slave (→ Kapitel *Bausteine: CANopen-Slave* (→ Seite [89](#)))
- CANopen-Netzwerkvariablen (via CODESYS)
- SAE J1939 (für Antriebsmanagement, → Kapitel *Bausteine: SAE J1939* (→ Seite [102](#)))
- Buslast-Erkennung
- Errorframe-Zähler
- Download-Schnittstelle
- 100 % Buslast ohne Paketverlust

11796

In diesem *ecomatmobile*-Gerät sind folgende CAN-Schnittstellen und CAN-Protokolle verfügbar:

CAN-Schnittstelle	CAN 1	CAN 2	CAN 3	CAN 4
voreingestellte Download-ID	ID 127	ID 126	ID 125	ID 124
CAN-Protokolle	CAN Layer 2	CAN Layer 2	Schnittstelle nicht vorhanden	Schnittstelle nicht vorhanden
	CANopen	---		
	SAE J1939	SAE J1939		

Standard-Baudrate = 125 kBit/s

3.4 Software

Inhalt	
Software-Module für das Gerät	28
Programmierhinweise für CODESYS-Projekte	31
Betriebszustände	35
Betriebsmodi	39
Leistungsgrenzen des Geräts	40

14107

3.4.1 Software-Module für das Gerät

Inhalt	
Bootloader	29
Laufzeitsystem	29
Anwendungsprogramm	29
Bibliotheken	30

14110

Die Software in diesem Gerät setzt wie folgt auf der Hardware auf:

Software-Modul	Anwender kann das Modul ändern?	womit?
Anwendungsprogramm mit Bibliotheken	ja	CODESYS, MaintenanceTool
Laufzeitsystem (LZS) *)	Upgrade ja Downgrade ja	MaintenanceTool
Bootloader	nein	---
(Hardware)	nein	---

*) Die Laufzeitsystem-Versionsnummer muss der Target-Versionsnummer in der CODESYS-Zielsystemeinstellung entsprechen!
→ Kapitel *Target einrichten* (→ Seite [45](#))

Nachfolgend beschreiben wir diese Software-Module:

Bootloader

14111

Im Auslieferungszustand enthalten **ecomatmobile**-Controller nur den Bootloader.

Der Bootloader ist ein Startprogramm, mit dem das Laufzeitsystem und das Anwendungsprogramm auf dem Gerät nachgeladen werden können.

Der Bootloader enthält Grundroutinen...

- zur Kommunikation der Hardware-Module untereinander,
- zum Nachladen des Laufzeitsystems.

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

Laufzeitsystem

14112

Grundprogramm im Gerät, stellt die Verbindung her zwischen der Hardware des Gerätes und dem Anwendungsprogramm.

→ Kapitel **Software-Module für das Gerät** (→ Seite [28](#))

Im Auslieferungszustand ist im Normalfall kein Laufzeitsystem im Controller geladen (LED blinkt grün mit 5 Hz). In diesem Betriebszustand ist nur der Bootloader aktiv. Dieser stellt die minimalen Funktionen für den Laufzeitsystem-Ladevorgang zur Verfügung, u.a. die Unterstützung der Schnittstellen (z.B. CAN).

Der Laufzeitsystem-Download muss im Normalfall nur einmalig durchgeführt werden. Das Anwendungsprogramm kann anschließend (auch mehrmals) in den Controller geladen werden, ohne das Laufzeitsystem zu beeinflussen.

Das Laufzeitsystem wird zusammen mit dieser Dokumentation auf einem separaten Datenträger zur Verfügung gestellt. Zusätzlich kann auch die aktuelle Version von der Homepage der **ifm electronic gmbh** heruntergeladen werden:

→ www.ifm.com > Land wählen > [Service] > [Download]

Anwendungsprogramm

14118

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

8340



WARNUNG

Für die sichere Funktion der Anwendungsprogramme, 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.

Bibliotheken

19527

ifm electronic bietet passend für jedes Gerät eine Reihe von Bibliotheken (*.LIB) an, die Programmmodule für das Anwendungsprogramm enthalten. Beispiele:

Bibliothek	Verwendung
ifm_CR0303_Vxxyzz.LIB	gerätespezifische Bibliothek Muss immer im Anwendungsprogramm enthalten sein!
ifm_CAN1_EXT_Vxxyzz.LIB	(optional) wenn die CAN-Schnittstelle 1 des Geräts auf 29 Bit arbeiten soll
ifm_CR0303_CANopenMaster_Vxxyzz.LIB	(optional) wenn die CAN-Schnittstelle 1 des Geräts als CANopen-Master betrieben werden soll
ifm_CR0303_CANopenSlave_Vxxyzz.LIB	(optional) wenn die CAN-Schnittstelle 1 des Geräts als CANopen-Slave betrieben werden soll
ifm_CR0303_J1939_x_Vxxyzz.LIB x = 1...2 = Nummer der CAN-Schnittstelle	(optional) wenn eine CAN-Schnittstelle des Geräts mit einer Motorsteuerung kommunizieren soll

→ Kapitel *ifm-Bibliotheken für das Gerät CR0303* (→ Seite [57](#))

3.4.2 Programmierhinweise für CODESYS-Projekte

Inhalt	
FB, FUN, PRG in CODESYS	32
Zykluszeit beachten!.....	32
Anwendungsprogramm erstellen.....	33
Boot-Projekt speichern	34
ifm-Downloader nutzen	34
ifm-Maintenance-Tool nutzen	34

7426

Hier erhalten Sie Tipps zum Programmieren des Geräts.

- Beachten Sie die Hinweise im CODESYS-Programmierhandbuch
 - www.ifm.com > Land wählen > [Datenblattsuche] > CR0303 > [Betriebsanleitungen],
 - **ecomatmobile**-DVD "Software, tools and documentation".

FB, FUN, PRG in CODESYS

8473

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

FB = function block = Funktionsbaustein

- 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

Funktionsbausteine dürfen NICHT in Funktionen aufgerufen werden!
Sonst: Bei der Ausführung stürzt das Anwendungsprogramm ab.

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

Eine IEC-Anwendung darf maximal 8000 Bausteine (POU) enthalten!

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:

- jedesmal erfolgt ein zusätzlicher Initialisierungsaufruf und
- die Daten des letzten Aufrufs gehen verloren.

Zykluszeit beachten!

8006

Bei den frei programmierbaren Geräten aus der Controller-Familie **ecomatmobile** stehen in einem großen Umfang Bausteine zur Verfügung, die den Einsatz der Geräte in den unterschiedlichsten Anwendungen 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 tragem Verhalten des Geräts!

Zykluszeit darf nicht zu lang werden!

- ▶ Beim Erstellen des Anwendungsprogramms die oben aufgeführten Empfehlungen beachten und durch Austesten überprüfen.
- ▶ Bei Bedarf durch Neustrukturieren der Software und des Systemaufbaus die Zykluszeit vermindern.

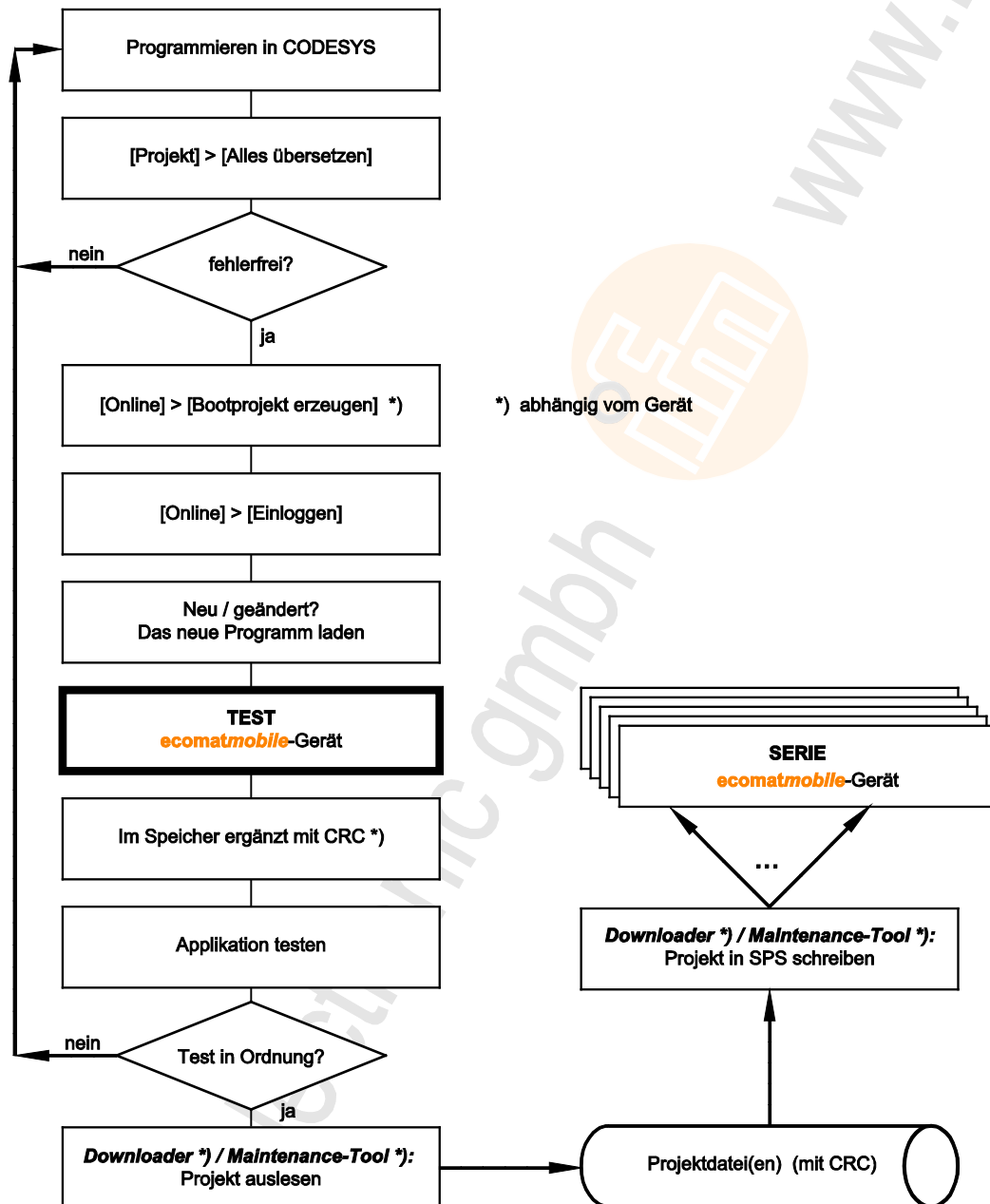
Anwendungsprogramm erstellen

8007

Das Anwendungsprogramm wird mit dem Programmiersystem CODESYS erstellt und während der Programmentwicklung mehrfach zum Testen in die Steuerung geladen:

In CODESYS: [Online] > [Einloggen] > das neue Programm laden.

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.



Grafik: Erstellen und Verteilen der Software

Boot-Projekt speichern

7430

❗ Speichern Sie im Gerät zusammen mit Ihrem Anwendungsprogramm immer auch das zugehörige Boot-Projekt! Nur so ist das Anwendungsprogramm auch nach einem Spannungsausfall im Gerät verfügbar.

❗ HINWEIS

Beachten: das Boot-Projekt ist etwas größer als das eigentliche Programm.

Jedoch: das Speichern des Boot-Projekts im Gerät wird scheitern, wenn das Boot-Projekt größer wird als der vorhandene IEC-Code-Speicherbereich. Nach Power-On-Reset ist das Boot-Projekt wieder gelöscht oder ungültig.

- ▶ CODESYS-Menü [Online] > [Bootprojekt erzeugen]
Dies muss auch nach jeder Änderung erneut erfolgen!
- > Nach einem Neustart startet das Gerät mit dem zuletzt gespeicherten Boot-Projekt.
- > Falls noch KEIN Boot-Projekt gespeichert wurde:
 - das Gerät bleibt nach dem Neustart im STOP-Betrieb
 - das Anwendungsprogramm ist nicht (mehr) vorhanden
 - die LED leuchtet grün.

ifm-Downloader nutzen

8008

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

Hier finden Sie den aktuellen **ifm**-Downloader (min. V06.18.26):

→ www.ifm.com > Land wählen > [Service] > [Download] > [Systeme für mobile Arbeitsmaschinen]
ecomatmobile-DVD "Software, tools and documentation" im Register "R360 tools [D/E]"

ifm-Maintenance-Tool nutzen

8492

Das **ifm**-Maintenance-Tool dient dem einfachen Übertragen des Programmcodes vom Programmierplatz in das Gerät. Grundsätzlich kann jedes Anwendungsprogramm mit dem **ifm**-Maintenance-Tool auf die Geräte kopiert werden. Vorteil: Dazu ist kein Programmiersystem mit einer CODESYS-Lizenz erforderlich.

Hier finden Sie das aktuelle **ifm**-Maintenance-Tool:

→ www.ifm.com > Land wählen > [Service] > [Download] > [Systeme für mobile Arbeitsmaschinen]
→ **ecomatmobile**-DVD "Software, tools and documentation" im Register "R360 tools [D/E]"

3.4.3 Betriebszustände

Inhalt	
Betriebszustände.....	35
Betriebszustände: Anwendungsprogramm nicht verfügbar	36
Betriebszustände: Anwendungsprogramm verfügbar	37
Bootloader-Zustand	38
INIT-Zustand (Reset)	38
STOP-Zustand.....	38
RUN-Zustand.....	38
SYSTEM-STOP-Zustand	38

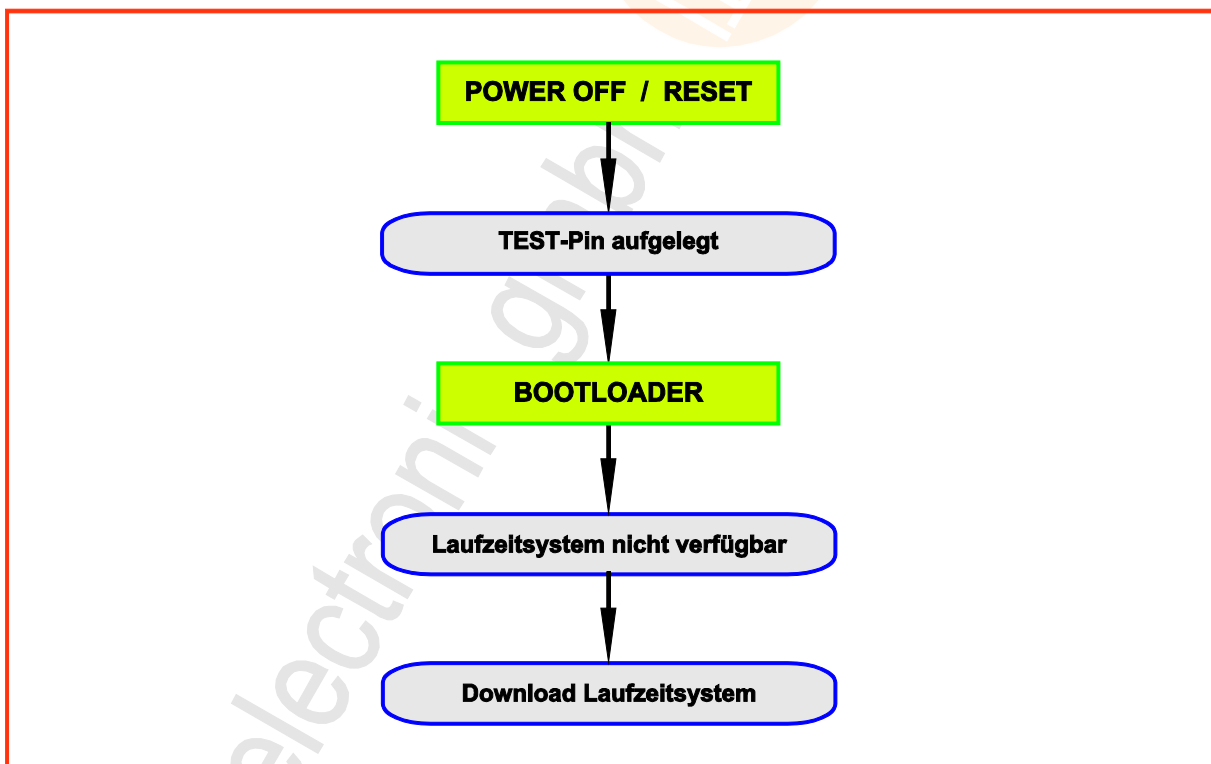
14120

Nach Anlegen der Versorgungsspannung kann sich das *ecomatmobile*-Gerät in einem von fünf möglichen Betriebszuständen befinden:

- BOOTLOADER
- INIT
- STOP
- RUN
- SYSTEM STOP (nach ERROR STOP)

Betriebszustände

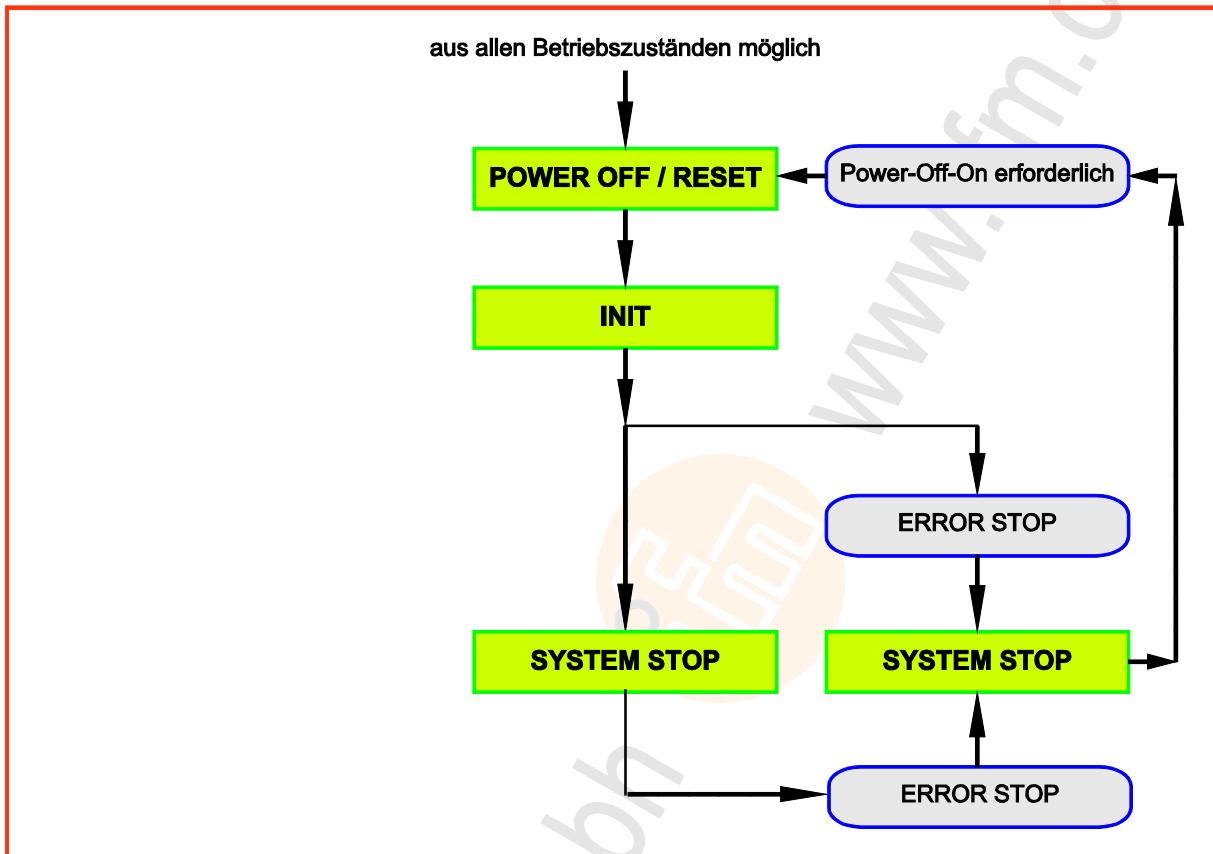
19217



Grafik: Betriebszustände (hier: Laufzeitsystem ist nicht verfügbar)

Betriebszustände: Anwendungsprogramm nicht verfügbar

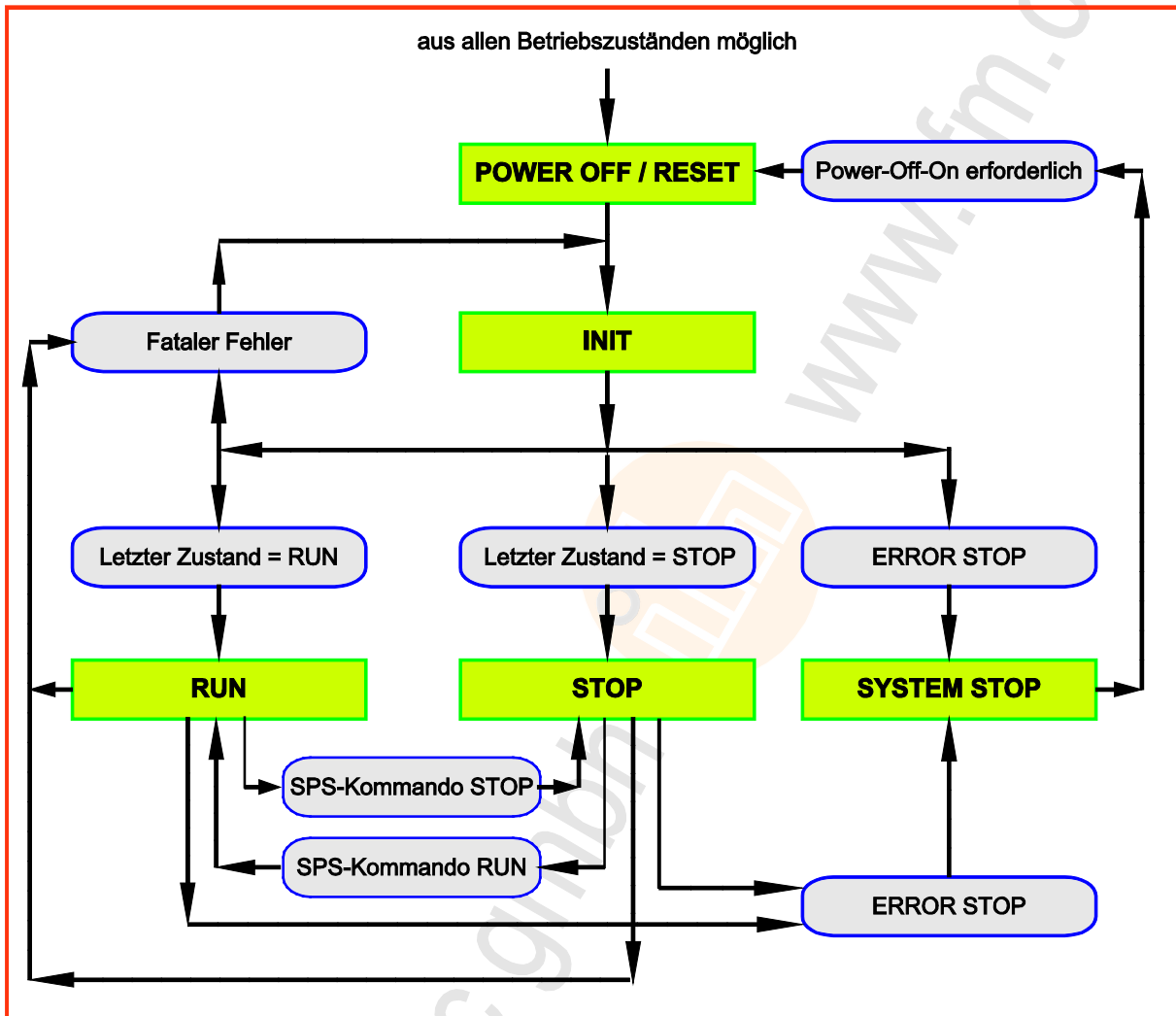
19218



Grafik: Betriebszustände (hier: Anwendungsprogramm ist nicht verfügbar)

Betriebszustände: Anwendungsprogramm verfügbar

19219



Grafik: Betriebszustände (hier: Anwendungsprogramm ist verfügbar)

Bootloader-Zustand

1080

Es wurde kein Laufzeitsystem geladen. Der **ecomatmobile**-Controller befindet sich im Bootloader-Zustand. Vor dem Laden des Anwendungsprogramms muss ein Laufzeitsystem-Download durchgeführt werden.

- > Die LED blinkt grün (5 Hz).

INIT-Zustand (Reset)

1076

Voraussetzung: ein gültiges Laufzeitsystem ist installiert.

Dieser Zustand wird nach jedem Power-On-Reset durchlaufen:

- > Das Laufzeitsystem wird initialisiert.
- > Verschiedene Checks werden durchgeführt, z.B. Warten auf gültige Versorgungsspannung.
- > Dieser nur temporäre Zustand wird vom RUN- oder STOP-Zustand abgelöst.
- > Die LED leuchtet orange.

Wechsel aus diesem Zustand in einen der folgenden Zustände möglich:

- RUN
- STOP

STOP-Zustand

1078

Dieser Zustand wird in folgenden Fällen erreicht:

- Aus dem RESET-Zustand, wenn:
 - kein Anwendungsprogramm ist geladen oder
 - der letzte Zustand vor dem RESET-Zustand war der STOP-Zustand
- Aus dem RUN-Zustand durch das STOP-Kommando
 - nur bei Betriebsmodus = TEST (→ Kapitel **TEST-Betrieb** (→ Seite [39](#)))
- > Die LED leuchtet grün.

RUN-Zustand

1077

Dieser Zustand wird in folgenden Fällen erreicht:

- Aus dem RESET-Zustand, wenn:
 - der letzte Zustand vor dem RESET-Zustand war der RUN-Zustand
- Aus dem STOP-Zustand durch das RUN-Kommando
 - nur bei Betriebsmodus = TEST (→ Kapitel **TEST-Betrieb** (→ Seite [39](#)))
- > Die LED blinkt grün (2 Hz).

SYSTEM-STOP-Zustand

19222

In diesen Zustand fällt der **ecomatmobile**-Controller, wenn ein nicht tolerierbarer Fehler (ERROR STOP) festgestellt wurde. Dieser Zustand kann nur durch einen Power-Off-On-Reset verlassen werden.

- > Die LED leuchtet rot.

3.4.4 Betriebsmodi

1083

Unabhängig von den Betriebszuständen kann der Controller in verschiedenen Betriebsmodi betrieben werden.

TEST-Betrieb

1084

ACHTUNG

Verlust der gespeicherten Software möglich!

Im Test-Betrieb besteht kein Schutz der gespeicherten Laufzeitsystem- und Anwendungs-Software.

! HINWEIS

- ▶ Erst NACH dem Anschließen des OPC-Client den TEST-Anschluss mit der Versorgungsspannung verbinden!
- > Ansonsten tritt ein fataler Fehler auf.

Dieser Betriebsmodus wird durch Anlegen von Versorgungsspannung am Test-Eingang erreicht (→ Montageanleitung > Kapitel "Technische Daten" > Kapitel "Anschlussbelegung").

Jetzt kann der Controller im RUN- oder STOP-Zustand Kommandos über eine der Schnittstellen entgegennehmen und z.B. mit dem Programmiersystem kommunizieren.

Nur im TEST-Betrieb ist ein Software-Download im Controller möglich.

Über den Merker TEST kann der Zustand vom Anwendungsprogramm abgefragt werden.

i Zusammenfassung Test-Eingang ist aktiv:

- Programmiermodus ist freigegeben
- Software-Download ist möglich
- Zustand des Anwendungsprogramms ist abfragbar
- kein Schutz der gespeicherten Software möglich

SERIAL_MODE

1085

Die serielle Schnittstelle steht für den Datenaustausch in der Anwendung zur Verfügung. Ein Debugging des Anwendungsprogramms ist dann nur noch über die CAN-Schnittstelle möglich.

Diese Funktion ist standardmäßig abgeschaltet (FALSE). Über den Merker SERIAL_MODE kann der Zustand über das Anwendungsprogramm oder das Programmiersystem gesteuert und abgefragt werden.

(→ Kapitel *Bausteine: serielle Schnittstelle* (→ Seite [114](#)))

DEBUG-Modus

1086

Wird der Eingang DEBUG von *SET_DEBUG* (→ Seite [189](#)) auf TRUE gesetzt, kann z.B. das Programmiersystem oder der Downloader mit dem Gerät kommunizieren und Systemkommandos ausführen (z.B. für Servicefunktionen über das GSM-Modem CANremote).

Ein Software-Download ist in dieser Betriebsart nicht möglich, da der Test-Eingang (→ Kapitel *TEST-Betrieb* (→ Seite [39](#))) nicht mit Versorgungsspannung verbunden wird.

3.4.5 Leistungsgrenzen des Geräts

7358



Leistungsgrenzen des Geräts beachten! → Datenblatt

Überdurchschnittliche Belastungen

1480
19994

Folgende Bausteine z.B. belasten die Systemressourcen überdurchschnittlich:

Baustein	Überdurchschnittliche Belastung
FREQUENCY PERIOD, PERIOD_RATIO, PHASE	Einsatz mehrerer Messkanäle mit einer hohen Eingangsfrequenz
CAN-Schnittstelle	Hohe Baudrate (> 250 kBit) mit einer hohen Buslast
PWM, PWM1000	Viele PWM-Kanäle gleichzeitig. Es sind besonders die Kanäle ab 4 deutlich zeitkritischer
INC_ENCODER	Viele Encoder-Kanäle gleichzeitig

Die oben exemplarisch aufgeführten Bausteine lösen System-Interrupts aus. Das bedeutet: Jeder Aufruf verlängert die Zykluszeit des Anwendungsprogramms.

1509

ACHTUNG

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

- Bei der Erstellung des Anwendungsprogramms müssen die oben aufgeführten Empfehlungen beachtet und durch Austesten überprüft werden. Bei Bedarf muss durch Neustrukturierung der Software und des Systemaufbaus die Zykluszeit optimiert werden.

Verhalten des Watchdog

1490

Ein Watchdog überwacht in diesem Gerät die Programmlaufzeit der CODESYS-Anwendung.

Wird die maximale Watchdog-Zeit (100...200 ms) überschritten:

> das Gerät führt einen Reset durch und startet neu.

4 Konfigurationen

Inhalt

Laufzeitsystem einrichten	41
Programmiersystem einrichten	44
Funktionskonfiguration, allgemein	48
Funktionskonfiguration der Ein- und Ausgänge	49
Variablen	55

1016

Die in den jeweiligen Montage- und Installationsanweisungen oder dem **Anhang** (→ Seite [194](#)) dieser Dokumentation beschriebenen Gerätekonfigurationen stehen als Standardgeräte (Lagerware) zur Verfügung. Diese decken bei den meisten Anwendungen die geforderten Spezifikationen ab.

Entsprechend den Kundenanforderungen bei Serieneinsatz ist es aber auch möglich, dass andere Gerätekonfigurationen z.B. hinsichtlich der Zusammenstellung der Ein- und Ausgänge und der Ausführung der Analogkanäle eingesetzt werden.

4.1 Laufzeitsystem einrichten

Inhalt

Laufzeitsystem neu installieren	42
Laufzeitsystem aktualisieren	43
Installation verifizieren	43

14091

4.1.1 Laufzeitsystem neu installieren

14092
2733

Im Auslieferungszustand ist im Normalfall kein Laufzeitsystem im Gerät geladen (LED blinkt grün mit 5 Hz). In diesem Betriebszustand ist nur der Bootloader aktiv. Dieser stellt die minimalen Funktionen für den Laufzeitsystem-Ladevorgang zur Verfügung, u.a. die Unterstützung der Schnittstellen (z.B. RS232, CAN).

Der Laufzeitsystem-Download muss im Normalfall nur einmalig durchgeführt werden. Das Anwendungsprogramm kann anschließend (auch mehrmals) in das Gerät geladen werden, ohne das Laufzeitsystem zu beeinflussen.

Das Laufzeitsystem wird zusammen mit dieser Dokumentation auf einem separaten Datenträger zur Verfügung gestellt. Zusätzlich kann auch die aktuelle Version von der Homepage der **ifm electronic gmbh** heruntergeladen werden:

→ www.ifm.com > Land wählen > [Service] > [Download]

2689

! HINWEIS

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

- des Laufzeitsystems (ifm_CR0303_Vxxyyyz.H86),
- der Steuerungskonfiguration (ifm_CR0303_Vxx.CFG),
- der Gerätebibliothek (ifm_CR0303_Vxxyyyz.LIB) und
- der weiteren Dateien

V	Version
xx: 00...99	Versionsnummer
yy: 00...99	Release-Nummer
zz: 00...99	Patch-Nummer

Dabei müssen der Basisdateiname (z.B. "CR0303") 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.

4368

! Folgende Dateien müssen ebenfalls geladen sein:

- die zum Projekt erforderlichen internen Bibliotheken (in IEC 61131 erstellt),
- die Konfigurationsdateien (*.CFG)
- und die Target-Dateien (*.TRG).

i Es kann vorkommen, dass das Zielsystem mit Ihrer aktuell installierten Version von CODESYS nicht oder nur teilweise programmiert werden kann. Im diesem Fall wenden Sie sich bitte an den technischen Support der **ifm electronic gmbh**.

Das Laufzeitsystem wird mit dem eigenständigen Programm "ifm-Downloader" in das Gerät übertragen. (Der ifm-Downloader und dessen Dokumentation befindet sich auf der *ecomatmobile*-DVD "Software, tools and documentation" oder kann bei Bedarf von der ifm-Homepage heruntergeladen werden: → www.ifm.com > Land wählen > [Service] > [Download]).

Das Anwendungsprogramm wird im Normalfall über das Programmiersystem in das Gerät geladen. Es kann aber ebenfalls mit dem ifm-Downloader geladen werden, wenn es zuvor aus dem Gerät ausgelesen wurde (→ Upload).

4.1.2 Laufzeitsystem aktualisieren

13269

Auf dem Gerät ist bereits ein älteres Laufzeitsystem installiert. Nun möchten Sie das Laufzeitsystem auf dem Gerät aktualisieren?

14158

ACHTUNG

Gefahr von Datenverlust!

Beim Löschen oder Aktualisieren des Laufzeitsystems werden alle Daten und Programme auf dem Gerät gelöscht.

- ▶ Alle erforderlichen Daten und Programme sichern, bevor das Laufzeitsystem gelöscht oder aktualisiert wird!

3084

Immer, wenn es zu wesentlichen Verbesserungen in der Betriebssystem-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.

Prinzipiell gelten für diesen Vorgang die gleichen Hinweise, wie zuvor im Kapitel 'Laufzeitsystem neu installieren' gegeben wurden.

4.1.3 Installation verifizieren

14512

- ▶ Nach dem Laden des Laufzeitsystems in die Steuerung:
 - Prüfen, ob das Laufzeitsystem korrekt übertragen wurde!
 - Prüfen, ob sich das richtige Laufzeitsystem auf der Steuerung befindet!
- ▶ 1. Prüfung:

mit dem ifm-Downloader oder mit dem Maintenance-Tool prüfen, ob die richtige Laufzeitsystem-Version geladen wurde:

 - Name, Version und die CRC des Laufzeitsystems im Gerät auslesen!
 - Diese Daten manuell mit den Soll-Daten vergleichen!
- ▶ 2. Prüfung (optional):

Im Anwendungsprogramm prüfen, ob die richtige Laufzeitsystem-Version geladen wurde:

 - Name und die Version des Laufzeitsystems im Gerät auslesen!
 - Diese Daten mit fest vorgegebenen Werten vergleichen!

Zum Auslesen der Daten dient folgender FB:

`GET_IDENTITY` (→ Seite [188](#))

liest die im Gerät gespeicherten spezifischen Kennungen:

- Hardware-Name und Hardware-Version des Geräts
- Name des Laufzeitsystems im Gerät
- Version und Ausgabe des Laufzeitsystems im Gerät
- Name der Anwendung (wurde zuvor mit `SET_IDENTITY` (→ Seite [190](#)) gespeichert)

4.2 Programmiersystem einrichten

Inhalt	
Programmierensystem manuell einrichten.....	44
Programmierensystem über Templates einrichten	47

3968

4.2.1 Programmiersystem manuell einrichten

Inhalt	
Target einrichten.....	45
Steuerungskonfiguration aktivieren (z.B. CR0033)	46

3963

Target einrichten

2687
11379

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

- ▶ Im Dialog-Fenster [Zielsystem Einstellungen] im Menü [Konfiguration] die gewünschte Target-Datei wählen.
- > 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 im Fenster [Zielsystem Einstellungen] > Reiter [Netzfunktionen] > [Parameter-Manager unterstützen] und / oder [Netzvariablen unterstützen] aktivieren.
- ▶ Bei Bedarf geladene (3S-)Bibliotheken wieder entfernen oder durch weitere (ifm-)Bibliotheken ergänzen.
- ▶ Immer die passende Geräte-Bibliothek `ifm_CR0303_Vxxyyyz.LIB` manuell ergänzen!

2689

HINWEIS

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

- des Laufzeitsystems (`ifm_CR0303_Vxxyyyz.H86`),
- der Steuerungskonfiguration (`ifm_CR0303_Vxx.CFG`),
- der Gerätebibliothek (`ifm_CR0303_Vxxyyyz.LIB`) und
- der weiteren Dateien

V	Version
xx: 00...99	Versionsnummer
yy: 00...99	Release-Nummer
zz: 00...99	Patch-Nummer

Dabei müssen der Basisdateiname (z.B. "CR0303") 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.

4368

 Folgende Dateien müssen ebenfalls geladen sein:

- die zum Projekt erforderlichen internen Bibliotheken (in IEC 61131 erstellt),
- die Konfigurationsdateien (*.CFG)
- und die Target-Dateien (*.TRG).

 Es kann vorkommen, dass das Zielsystem mit Ihrer aktuell installierten Version von CODESYS nicht oder nur teilweise programmiert werden kann. Im diesem Fall wenden Sie sich bitte an den technischen Support der **ifm electronic gmbh**.

Steuerungskonfiguration aktivieren (z.B. CR0033)

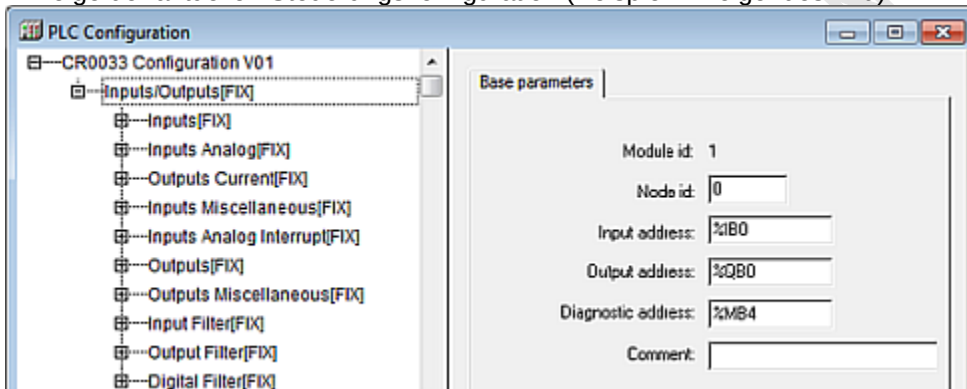
15824

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

- ▶ Den Punkt [Steuerungskonfiguration] erreicht man über den Reiter [Ressourcen].
Mit Doppelklick auf den Punkt [Steuerungskonfiguration] öffnet sich das entsprechende Fenster.
- ▶ In CODESYS den Reiter [Ressourcen] klicken:

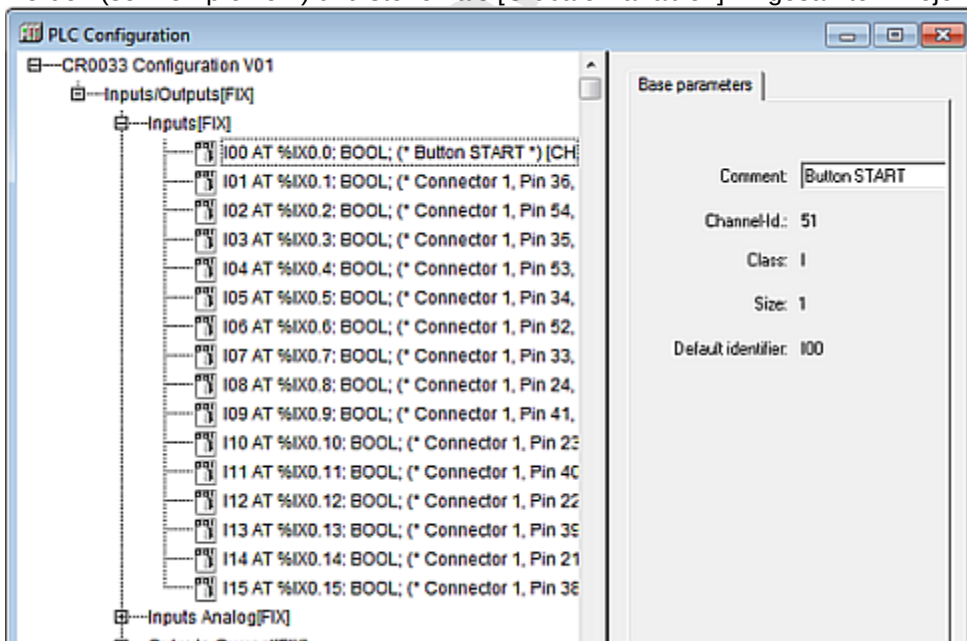


- ▶ In der linken Spalte Doppelklick auf [Steuerungskonfiguration]
- > Anzeige der aktuellen Steuerungskonfiguration (Beispiel → folgendes Bild):



Durch die Konfiguration ist für den Anwender in der Programmumgebung Folgendes verfügbar:

- alle wichtigen System- und Fehlermerker
Je nach Anwendung und Anwendungsprogramm 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.



4.2.2 Programmiersystem über Templates einrichten

13745

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

970

-  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_DVD_V...
- ▶ Die gewünschte dort gespeicherte Vorlage in CODESYS öffnen mit:
[Datei] > [Neu aus Vorlage...]
 - > CODESYS legt ein neues Projekt an, dem der prinzipielle Programmaufbau entnommen werden kann. Es wird dringend empfohlen, dem gezeigten Schema zu folgen.

4.3 Funktionskonfiguration, allgemein

Inhalt

Konfiguration der Ein- und Ausgänge (Voreinstellung)	48
Systemvariablen	48

3971

4.3.1 Konfiguration der Ein- und Ausgänge (Voreinstellung)

2249

- Alle Ein-/Ausgänge sind im Auslieferungszustand im Binär-Modus (plus-schaltend!).
- Die Diagnosefunktion ist nicht aktiv.
- Der Überlastschutz ist aktiv.

4.3.2 Systemvariablen

2252
13519
15576

Alle Systemvariablen (→ Kapitel *Systemmerker* (→ Seite [194](#))) liegen auf festen, nicht verschiebbaren Adressen.

- > Zur Anzeige und Verarbeitung eines Watchdog-Fehlers oder Ursachen eines Neustarts wird die Systemvariable LAST_RESET gesetzt.
- > Anzeige der gewählten E/A-Konfiguration über Mode-Bytes

4.4 Funktionskonfiguration der Ein- und Ausgänge

Inhalt	
Konfiguration der Ein- und Ausgänge (Voreinstellung)	49
Eingänge konfigurieren	50
Ausgänge konfigurieren	53

1394

Bei bestimmten Ein- und Ausgängen sind zusätzliche Diagnosefunktionen aktivierbar. Damit kann das jeweilige Ein- und Ausgangssignal überwacht werden und im Fehlerfall kann das Anwendungsprogramm darauf reagieren.

Je nach Ein- und Ausgang müssen bei der Nutzung der Diagnose bestimmte Randbedingungen beachtet werden:

- Anhand des Datenblattes prüfen, für welche Ein- und Ausgänge des Geräts welche Diagnosemöglichkeit zur Verfügung steht!
- Zur Konfiguration der Ein- und Ausgänge sind in den Gerätebibliotheken (ifm_CR0303_Vxxyzz.LIB) Konstanten vordefiniert (z.B. IN_DIGITAL_H). Ausführliche Angaben → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#)).

4.4.1 Konfiguration der Ein- und Ausgänge (Voreinstellung)

2249

- Alle Ein-/Ausgänge sind im Auslieferungszustand im Binär-Modus (plus-schaltend!).
- Die Diagnosefunktion ist nicht aktiv.
- Der Überlastschutz ist aktiv.

4.4.2 Eingänge konfigurieren

Inhalt

Sicherheitshinweise zu Reed-Relais	50
Analogeingänge: Konfiguration und Diagnose	51
Binäreingänge: Konfiguration und Diagnose	52
Schnelle Eingänge	52

3973

Zulässige Betriebsarten → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))

Sicherheitshinweise zu Reed-Relais

7348

Beim Einsatz von nichtelektronischen Schaltern Folgendes beachten:

! 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

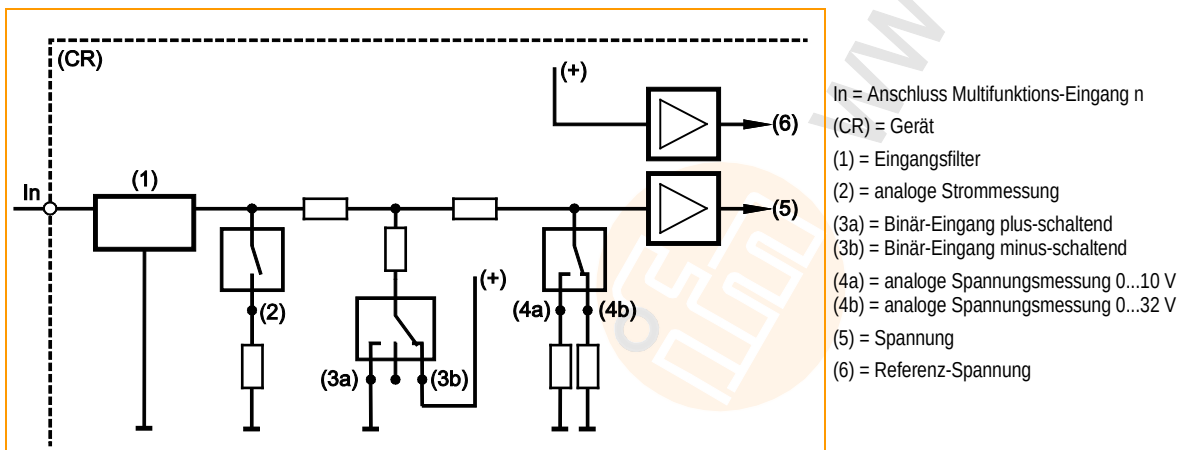
Analogeingänge: Konfiguration und Diagnose

19998

- ▶ Die Konfiguration jedes einzelnen Eingangs erfolgt über das Anwendungsprogramm:
 - Konfigurationsbyte A_INxx_MODE
→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))
- ▶ Werden die Analogeingänge auf Strommessung konfiguriert, wird bei Überschreiten des Endwertes ($> 23 \text{ mA}$) in den sicheren Spannungsmessbereich ($0...32 \text{ V DC}$) geschaltet und das jeweilige Fehlerbit im Merkerbyte ERROR_Ix gesetzt. Sinkt der Wert wieder unter den Grenzwert, schaltet der Eingang selbsttätig auf den Strommessbereich zurück.

Alternativ kann ein Analog-Kanal auch binär ausgewertet werden.

8971



Grafik: Prinzipschaltung Multifunktions-Eingang

Binäreingänge: Konfiguration und Diagnose

20001

- Die Konfiguration jedes einzelnen Eingangs erfolgt über das Anwendungsprogramm:

- Konfigurationsbyte INxx_MODE

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))

Schnelle Eingänge

19996

Die Geräte verfügen über schnelle Zähl-/Impulseingänge für eine Eingangsfrequenz bis 30 kHz (→ Datenblatt).

Geeignete Funktionsbausteine sind z.B.:

<i>FAST_COUNT</i> (→ Seite 134)	Zählerbaustein für schnelle Eingangsimpulse
<i>FREQUENCY</i> (→ Seite 135)	misst die Frequenz des am gewählten Kanal ankommenden Signals
<i>INC_ENCODER</i> (→ Seite 136)	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
<i>PERIOD</i> (→ Seite 138)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
<i>PERIOD_RATIO</i> (→ Seite 140)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] über die angegebenen Perioden am angegebenen Kanal. Zusätzlich wird das Puls-/Periodenverhältnis in [%] angegeben.
<i>PHASE</i> (→ Seite 142)	liest ein Kanalpaar mit schnellen Eingängen ein und vergleicht die Phasenlage der Signale

 Bei Einsatz dieser Bausteine werden automatisch die dort parametrierten Ein-/Ausgänge konfiguriert. Der Programmierer der Anwendung ist hiervon entlastet.

Einsatz als Binäreingänge

3804

Durch die zulässigen hohen Eingangsfrequenzen können auch Fehlsignale erkannt werden, z.B. prellende Kontakte mechanischer Schalter.

- Bei Bedarf die Fehlsignale im Anwendungsprogramm unterdrücken!

4.4.3 Ausgänge konfigurieren

Inhalt

Binärausgänge: Konfiguration und Diagnose.....	53
PWM-Ausgänge	54

3976

Zulässige Betriebsarten → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))

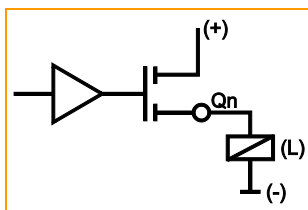
Binärausgänge: Konfiguration und Diagnose

20004

Diese Ausgänge sind fix eingestellt wie folgt:

- binärer Ausgang, plus-schaltend (BH), kurzschlussfest, überlastfest

15451



Qn = Anschluss Ausgang n
(L) = Last

Prinzipschaltung Binär-Ausgang plus-schaltend (BH)
für positives Ausgangssignal

13975



WARNUNG

Gefährlicher Wiederanlauf möglich!

Gefahr von Personenschaden! Gefahr von Sachschaden an der Maschine/Anlage!

Wird ein Ausgang im Fehlerfall hardwaremäßig abgeschaltet, ändert sich der durch das Anwendungsprogramm erzeugte logische Zustand dadurch nicht.

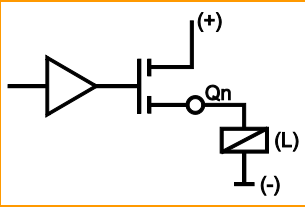
► Abhilfe:

- Die Ausgänge zunächst im Anwendungsprogramm logisch zurücksetzen!
- Fehler beseitigen!
- Ausgänge situationsabhängig wieder setzen.

PWM-Ausgänge

Bei den Geräte-Ausgängen sind folgende Betriebsarten möglich (→ Datenblatt):

- PWM-Ausgang, plus-schaltend (BH) ohne Diagnosefunktion



Qn = Anschluss Ausgang n
(L) = Last

Prinzipschaltung Binär-Ausgang plus-schaltend (BH)
für positives Ausgangssignal

 **WARNUNG**

Sach- oder Körperschäden möglich durch Fehlfunktionen!
Für Ausgänge im PWM-Modus gilt:

- es gibt keine Diagnosefunktionen
- der Überlastschutz OUT_OVERLOAD_PROTECTION ist NICHT aktiv

 **HINWEIS**

PWM-Ausgänge dürfen NICHT parallel betrieben werden, um z.B. den max. Ausgangsstrom zu erhöhen. Die Ausgänge arbeiten nicht synchron.
Andernfalls kann die komplette Last über nur einen Ausgang gehen. Die Strommessung funktioniert dann nicht mehr.

Verfügbarkeit von PWM

Gerät	Anzahl verfügbare PWM-Ausgänge	davon stromgeregelt (PWMi)	PWM-Frequenz [Hz]
CabinetController: CR0303	8	---	20...250

FBs für PWM-Funktionen

Für die PWM-Funktion der Ausgänge stehen folgende Funktionsbausteine zur Verfügung:

<i>PWM</i> (→ Seite 144)	initialisiert und parametrisiert einen PWM-fähigen Ausgangskanal Festlegung der PWM-Frequenz über RELOAD
<i>PWM1000</i> (→ Seite 148)	initialisiert und parametrisiert einen PWM-fähigen Ausgangskanal das Puls-Pausen-Verhältnis kann in 1 %-Schritten angegeben werden

4.5 Variablen

Inhalt

Retain-Variablen.....	56
Netzwerkvariablen.....	56

3130

In diesem Kapitel erfahren Sie mehr über den Umgang mit Variablen.

14486

Das Gerät unterstützt folgende Variablentypen:

Variable	Deklarationsort	Gültigkeitsbereich	Speicherverhalten
lokal	im Deklarationsteil des Bausteins	gilt nur im Baustein (POU), in dem sie konfiguriert wurde	flüchtig
lokal Retain			nicht flüchtig
global	in [Ressourcen] > [Globale Variablen] > [Globale_Variablen]	gilt in allen Bausteinen (POUs) dieses CODESYS-Projekts	flüchtig
global Retain			nicht flüchtig
Netzwerk	in [Ressourcen] > [Globale Variablen] > Deklarationsliste	Werte stehen allen CODESYS-Projekten im gesamten Netzwerk zur Verfügung, wenn die Variable in ihren Deklarationslisten enthalten ist.	flüchtig
Netzwerk Retain			nicht flüchtig



→ CODESYS-Programmierhandbuch

→ *ecomatmobile*-DVD "Software, tools and documentation"

4.5.1 Retain-Variablen

3131

Als RETAIN deklarierte Variablen erzeugen remanente Daten. Retain-Variablen behalten beim Aus-/Einschalten des Geräts oder einem Online-Reset die in ihnen gespeicherten Werte.

14166

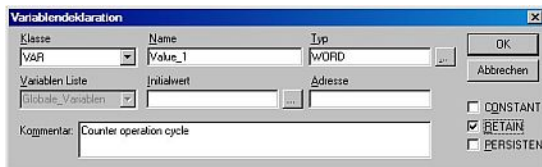
Typische Einsätze für Retain-Variablen sind z.B.:

- Betriebsstunden, die zur Laufzeit der Maschine fortgeschrieben werden,
- Positionswerte von Inkrementalgebern,
- im Bildschirmgerät eingetragene Sollwerte,
- Maschinenparameter,

also alle Variablen, deren Werte beim Ausschalten des Geräts nicht verloren gehen dürfen.

Als Retain können alle Variablentypen, auch komplexe Stukturen (z.B. Timer), gekennzeichnet werden.

► Dazu in der Variablen-Deklaration das Kontrollfeld [RETAIN] aktivieren (→ Bild).



4.5.2 Netzwerkvariablen

9856

Globale Netzwerkvariablen dienen dem Datenaustausch zwischen Controllern im Netzwerk. Die Werte von globalen Netzwerkvariablen stehen allen CODESYS-Projekten im gesamten Netzwerk zur Verfügung, wenn die Variablen in deren Deklarationslisten enthalten sind.

► Dazu folgende Bibliothek(en) in das CODESYS-Projekt einbinden:

- 3S_CANopenNetVar.lib

5 ifm-Funktionselemente

Inhalt

ifm-Bibliotheken für das Gerät CR0303.....	57
ifm-Bausteine für das Gerät CR0303	63

13586

Alle CODESYS-Funktionselemente (FBs, PRGs, FUNs) sind in Bibliotheken zusammengefasst. Nachfolgend zeigen wir Ihnen alle **ifm**-Bibliotheken, die Sie zusammen mit diesem Gerät nutzen können.

Anschließend finden Sie eine thematisch gegliederte Beschreibung der Funktionselemente.

5.1 ifm-Bibliotheken für das Gerät CR0303

Inhalt

Bibliothek ifm_CR0303_V05yyzz.LIB.....	58
Bibliothek ifm_CAN1_EXT_Vxxyzz.LIB.....	60
Bibliothek ifm_CR0303_CANopenMaster_V04yynn.LIB.....	60
Bibliothek ifm_CR0303_CANopenSlave_V04yynn.LIB.....	61
Bibliothek ifm_J1939_x_Vxxyzz.LIB.....	61
Bibliothek ifm_hydraulic_CR0303_V01yyzz.LIB	62

14235

Legende für ..._Vxxyzz.LIB:

V	Version
xx: 00...99	Versionsnummer
yy: 00...99	Release-Nummer
zz: 00...99	Patch-Nummer

Hier finden Sie die für dieses Gerät passenden **ifm**-Funktionselemente aufgelistet, nach CODESYS-Bibliotheken sortiert.

5.1.1 Bibliothek ifm_CR0303_V05yyzz.LIB

18572

> Dies ist die Geräte-Bibliothek.

Diese **ifm**-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
CAN1_BAUDRATE (→ Seite 64)	stellt die Übertragungsrate für den Busteilnehmer an der CAN-Schnittstelle 1 ein
CAN1_DOWNLOADID (→ Seite 65)	stellt den Download-Identifizier für die CAN-Schnittstelle 1 ein
CAN2 (→ Seite 71)	initialisiert die CAN-Schnittstelle 2 Modus und Baudrate einstellen
CANx_ERRORHANDLER (→ Seite 72)	führt ein "manuelles" Bus-Recover auf der CAN-Schnittstelle x durch x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_EXT_RECEIVE_ALL (→ Seite 73)	CAN-Schnittstelle x: konfiguriert alle Datenempfangsobjekte und liest den Empfangspuffer der Datenobjekte aus x = 2 = Nummer der CAN-Schnittstelle
CANx_RECEIVE (→ Seite 75)	CAN-Schnittstelle x: konfiguriert ein Datenempfangsobjekt und liest den Empfangspuffer des Datenobjektes aus x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_RECEIVE_RANGE (→ Seite 77)	CAN-Schnittstelle x: konfiguriert eine Folge von Datenempfangsobjekten und liest den Empfangspuffer der Datenobjekte aus x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SDO_READ (→ Seite 98)	CAN-Schnittstelle x: liest das SDO mit den angegebenen Indizes aus dem Knoten aus x = 1 = Nummer der CAN-Schnittstelle
CANx_SDO_WRITE (→ Seite 100)	CAN-Schnittstelle x: schreibt das SDO mit den angegebenen Indizes in den Knoten x = 1 = Nummer der CAN-Schnittstelle
CANx_TRANSMIT (→ Seite 79)	übergibt in jedem Aufruf ein CAN-Datenobjekt (Message) an die CAN-Schnittstelle x zur Übertragung x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CHECK_DATA (→ Seite 186)	erzeugt über einen konfigurierbaren Speicherbereich eine Prüfsumme (CRC) und prüft die Daten des Speicherbereichs auf ungewollte Veränderung
DELAY (→ Seite 164)	verzögert die Ausgabe des Eingangswertes um die Zeit T (Totzeit-Glied)
FAST_COUNT (→ Seite 134)	Zählerbaustein für schnelle Eingangsimpulse
FLASHREAD (→ Seite 179)	liest unterschiedliche Datentypen direkt aus dem Flash-Speicher in den RAM
FLASHWRITE (→ Seite 180)	schreibt unterschiedliche Datentypen direkt in den Flash-Speicher
FRAMREAD (→ Seite 182)	liest unterschiedliche Datentypen direkt aus dem FRAM-Speicher in den RAM FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.
FRAMWRITE (→ Seite 183)	schreibt unterschiedliche Datentypen direkt in den FRAM-Speicher FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.
FREQUENCY (→ Seite 135)	misst die Frequenz des am gewählten Kanal ankommenden Signals
GET_IDENTITY (→ Seite 188)	liest die im Gerät gespeicherten spezifischen Kennungen: • Hardware-Name und Hardware-Version des Geräts • Name des Laufzeitsystems im Gerät • Version und Ausgabe des Laufzeitsystems im Gerät • Name der Anwendung (wurde zuvor mit SET_IDENTITY (→ Seite 190) gespeichert)
GLR (→ Seite 165)	Der Gleichlaufregler ist ein Regler mit PID-Verhalten
INC_ENCODER (→ Seite 136)	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
INPUT_ANALOG (→ Seite 127)	Strom- und Spannungsmessung am analogen Eingangskanal
INPUT_CURRENT (→ Seite 128)	Strommessung am analogen Eingangskanal
INPUT_VOLTAGE (→ Seite 129)	Spannungsmessung am analogen Eingangskanal
MEMCPY (→ Seite 184)	schreibt und liest unterschiedliche Datentypen direkt in den Speicher
NORM (→ Seite 131)	normiert einen Wert [WORD] innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen

Baustein	Kurzbeschreibung
PERIOD (→ Seite 138)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_RATIO (→ Seite 140)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] über die angegebenen Perioden am angegebenen Kanal. Zusätzlich wird das Puls-/Periodenverhältnis in [%] angegeben.
PHASE (→ Seite 142)	liest ein Kanalpaar mit schnellen Eingängen ein und vergleicht die Phasenlage der Signale
PID1 (→ Seite 167)	PID-Regler
PID2 (→ Seite 169)	PID-Regler
PT1 (→ Seite 171)	Regelstrecke mit Verzögerung 1. Ordnung
PWM (→ Seite 144)	initialisiert und parametrisiert einen PWM-fähigen Ausgangskanal Festlegung der PWM-Frequenz über RELOAD
PWM1000 (→ Seite 148)	initialisiert und parametrisiert einen PWM-fähigen Ausgangskanal das Puls-Pausen-Verhältnis kann in 1 %-Schritten angegeben werden
SERIAL_PENDING (→ Seite 115)	ermittelt die Anzahl der im seriellen Empfangspuffer gespeicherten Datenbytes
SERIAL_RX (→ Seite 116)	liest mit jedem Aufruf ein empfangenes Datenbyte aus dem seriellen Empfangspuffer aus
SERIAL_SETUP (→ Seite 117)	initialisiert die serielle RS232-Schnittstelle
SERIAL_TX (→ Seite 118)	überträgt ein Datenbyte über die serielle RS232-Schnittstelle
SET_DEBUG (→ Seite 189)	organisiert (abhängig vom TEST-Eingang) den DEBUG-Modus oder den Monitoring-Modus
SET_IDENTITY (→ Seite 190)	setzt eine anwendungsspezifische Programmkennung
SET_INTERRUPT_I (→ Seite 120)	bedingtes Ausführen eines Programnteils nach einer Interrupt-Anforderung über einen Eingangskanal
SET_INTERRUPT_XMS (→ Seite 123)	bedingtes Ausführen eines Programnteils im Intervall von x Millisekunden
SET_PASSWORD (→ Seite 191)	setzt Benutzerkennung für Zugangskontrolle bei Programm- und Speicher-Upload
SOFTRESET (→ Seite 173)	führt einen kompletten Neustart des Geräts aus
TIMER_READ (→ Seite 175)	liest die aktuelle Systemzeit in [ms] aus Max-Wert = 49d 17h 2min 47s 295ms
TIMER_READ_US (→ Seite 176)	liest die aktuelle Systemzeit in [µs] aus Max-Wert = 1h 11min 34s 967ms 295µs

5.1.2 Bibliothek ifm_CAN1_EXT_Vxxyzz.LIB

18732

Diese Bibliothek enthält die Ergänzungs-Bausteine zur Motorsteuerung auf der 1. CAN-Schnittstelle. Bibliothek ist nur zulässig für die 1. CAN-Schnittstelle.

Diese **ifm**-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
CAN1_EXT (→ Seite 66)	initialisiert die CAN-Schnittstelle 1 auch für den Extended-Mode Modus und Baudrate einstellen
CAN1_EXT_ERRORHANDLER (→ Seite 67)	führt ein "manuelles" Bus-Recover auf der CAN-Schnittstelle 1 durch
CAN1_EXT_RECEIVE (→ Seite 68)	CAN-Schnittstelle 1: konfiguriert ein Datenempfangsobjekt und liest den Empfangspuffer des Datenobjektes aus
CANx_EXT_RECEIVE_ALL (→ Seite 73)	CAN-Schnittstelle x: konfiguriert alle Datenempfangsobjekte und liest den Empfangspuffer der Datenobjekte aus x = 1 = Nummer der CAN-Schnittstelle
CAN1_EXT_TRANSMIT (→ Seite 70)	übergibt in jedem Aufruf ein CAN-Datenobjekt (Message) an die CAN-Schnittstelle 1 zur Übertragung

5.1.3 Bibliothek ifm_CR0303_CANopenMaster_V04yynn.LIB

18714

Diese Bibliothek enthält die Bausteine für den Betrieb des Geräts als CANopen-Master. Bibliothek ist nur zulässig für die 1. CAN-Schnittstelle.

x = 1 = Nummer der CAN-Schnittstelle

Diese **ifm**-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
CANx_MASTER_EMCY_HANDLER (→ Seite 81)	verwaltet den geräteeigenen Fehlerstatus des CANopen-Masters an der CAN-Schnittstelle x x = 1 = Nummer der CAN-Schnittstelle
CANx_MASTER_SEND_EMERGENCY (→ Seite 82)	versendet anwendungsspezifische Fehlerstatus des CANopen-Masters an der CAN-Schnittstelle x x = 1 = Nummer der CAN-Schnittstelle
CANx_MASTER_STATUS (→ Seite 84)	Status-Anzeige an der CAN-Schnittstelle x des als CANopen-Master eingesetzten Gerätes x = 1 = Nummer der CAN-Schnittstelle

5.1.4 Bibliothek ifm_CR0303_CANOpenSlave_V04yynn.LIB

18719

Diese Bibliothek enthält die Bausteine für den Betrieb des Geräts als CANopen-Slave. Bibliothek ist nur zulässig für die 1. CAN-Schnittstelle.

x = 1 = Nummer der CAN-Schnittstelle

Diese **ifm**-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
CANx_SLAVE_EMCY_HANDLER (→ Seite 90)	verwaltet den geräteeigenen Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x: • Error Register (Index 0x1001) und • Error Field (Index 0x1003) des CANopen Objektverzeichnis x = 1 = Nummer der CAN-Schnittstelle
CANx_SLAVE_NODEID (→ Seite 91)	ermöglicht das Einstellen der Node-ID eines CANopen-Slaves an der CAN-Schnittstelle x zur Laufzeit des Anwendungsprogramms x = 1 = Nummer der CAN-Schnittstelle
CANx_SLAVE_SEND_EMERGENCY (→ Seite 92)	versendet anwendungsspezifische Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x x = 1 = Nummer der CAN-Schnittstelle
CANx_SLAVE_STATUS (→ Seite 94)	zeigt den Status des an der CAN-Schnittstelle x als CANopen-Slave eingesetzten Gerätes x = 1 = Nummer der CAN-Schnittstelle

5.1.5 Bibliothek ifm_J1939_x_Vxxyzz.LIB

18722

Diese Bibliothek enthält die Bausteine zur Motorsteuerung.

x = 1...2 = Nummer der CAN-Schnittstelle

Diese **ifm**-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
J1939_x (→ Seite 103)	CAN-Schnittstelle x: Protokoll-Handler für das Kommunikationsprofil SAE J1939 x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
J1939_x_GLOBAL_REQUEST (→ Seite 104)	CAN-Schnittstelle x: organisiert globales Anfordern und Empfangen von Daten der J1939-Netzwerkteilnehmer x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
J1939_x_RECEIVE (→ Seite 106)	CAN-Schnittstelle x: empfängt eine einzelne Nachricht oder einen Nachrichtenblock x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
J1939_x_RESPONSE (→ Seite 108)	CAN-Schnittstelle x: organisiert die automatische Antwort auf ein Request-Telegramm x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
J1939_x_SPECIFIC_REQUEST (→ Seite 110)	CAN-Schnittstelle x: automatisches Anfordern einzelner Nachrichten von einem bestimmten (specific) J1939-Netzwerkteilnehmer x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
J1939_x_TRANSMIT (→ Seite 112)	CAN-Schnittstelle x: versendet einzelne Nachrichten oder Nachrichtenblocks x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

5.1.6 Bibliothek ifm_hydraulic_CR0303_V01yyzz.LIB

18716

Diese Bibliothek enthält die Bausteine für Hydraulik-Steuerungen.

Diese **ifm**-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
JOYSTICK_0 (→ Seite 151)	skaliert Signale [INT] aus einem Joystick auf fest definierte Kennlinien, normiert auf 0...1000
JOYSTICK_1 (→ Seite 154)	skaliert Signale [INT] aus einem Joystick auf parametrierbare Kennlinien, normiert auf 0...1000
JOYSTICK_2 (→ Seite 158)	skaliert Signale [INT] aus einem Joystick auf einen parametrierbaren Kennlinien-Verlauf; die Normierung ist frei bestimmbar
NORM_HYDRAULIC (→ Seite 161)	normiert einen Wert [DINT] innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen



5.2 ifm-Bausteine für das Gerät CR0303

Inhalt	
Bausteine: CAN Layer 2	63
Bausteine: CANopen-Master	80
Bausteine: CANopen-Slave	89
Bausteine: CANopen SDOs	97
Bausteine: SAE J1939	102
Bausteine: serielle Schnittstelle	114
Bausteine: SPS-Zyklus optimieren	119
Bausteine: Eingangswerte verarbeiten	126
Bausteine: analoge Werte anpassen	130
Bausteine: Zählerfunktionen zur Frequenz- und Periodendauermessung	133
Bausteine: PWM-Funktionen	143
Bausteine: Hydraulikregelung	150
Bausteine: Regler	163
Bausteine: Software-Reset	172
Bausteine: Zeit messen / setzen	174
Bausteine: Daten im Speicher sichern, lesen und wandeln	177
Bausteine: Datenzugriff und Datenprüfung	185

13988
3826

Hier finden Sie die Beschreibung der für dieses Gerät passenden **ifm**-Funktionselemente, nach Thema sortiert.

5.2.1 Bausteine: CAN Layer 2

Inhalt	
CAN1_BAUDRATE	64
CAN1_DOWNLOADID	65
CAN1_EXT	66
CAN1_EXT_ERRORHANDLER	67
CAN1_EXT_RECEIVE	68
CAN1_EXT_TRANSMIT	70
CAN2	71
CANx_ERRORHANDLER	72
CANx_EXT_RECEIVE_ALL	73
CANx_RECEIVE	75
CANx_RECEIVE_RANGE	77
CANx_TRANSMIT	79

13754

Hier werden die CAN-Funktionsbausteine (Layer 2) zur Nutzung im Anwendungsprogramm beschrieben.

CAN1_BAUDRATE

651

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

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.

ACHTUNG

Für CR250n, CR0301, CR0302, CS0015 beachten:

Das EEPROM-Speichermodule 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 Baudrate wird erst nach einem RESET gültig (Spannung Aus/Ein oder Soft-Reset).

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

Parameter der Eingänge

655

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (im 1. Zyklus): Parameter übernehmen und aktivieren sonst: diese Funktion wird nicht ausgeführt
BAUDRATE	WORD := 125	Baudrate [kBit/s] zulässig = 20, 50, 100, 125, 250, 500, 1000

CAN1_DOWNLOADID

645

= CAN1 Download-ID

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxyxyz.LIB

Symbol in CODESYS:



Beschreibung

648

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

Mit dem FB kann der Kommunikations-Identifizier für den Programmdownload und das Debuggen eingestellt werden. Der neue Wert wird eingetragen, wenn der Eingang ENABLE auf TRUE gesetzt wird.

! Der neue Wert wird erst nach einem RESET gültig (Spannung Aus/Ein oder Soft-Reset).

ACHTUNG

Für CR250n, CR0301, CR0302, CS0015 beachten:

Das EEPROM-Speichermodule 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")!

Parameter der Eingänge

649

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (im 1. Zyklus): Parameter übernehmen und aktivieren sonst: diese Funktion wird nicht ausgeführt
ID	BYTE	Download-ID der CAN-Schnittstelle x setzen x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt) zulässig = 1...127 voreingestellt = 127 - (x-1)

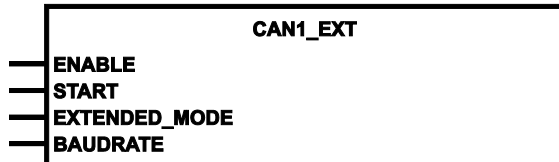
CAN1_EXT

4192

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CAN1_EXT_Vxxyzz.LIB

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 **SAE J1939** benutzt werden soll.

Eine Änderung der Baudrate wird erst gültig nach Spannung Aus/Ein.

Die Baudraten 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.

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

Parameter der Eingänge

4334

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
START	BOOL	TRUE (im 1. Zyklus): CAN-Protokoll an CAN-Schnittstelle x starten FALSE: im weiteren Programmablauf
EXTENDED_MODE	BOOL := FALSE	TRUE: Identifier der CAN-Schnittstelle arbeitet mit 29 Bits FALSE: Identifier der CAN-Schnittstelle arbeitet mit 11 Bits
BAUDRATE	WORD := 125	Baudrate [kBit/s] zulässig = 50, 100, 125, 250, 500, 800, 1000

CAN1_EXT_ERRORHANDLER

4195

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CAN1_EXT_Vxxyzz.LIB

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-aktiv (= 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 Anwendungsprogramm zurückgesetzt werden.

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

Parameter der Eingänge

2177

Parameter	Datentyp	Beschreibung
BUSOFF_RECOVER	BOOL	TRUE (nur 1 Zyklus lang): > Bus-off-Zustand beheben > Neustart der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt) FALSE: Funktion wird nicht ausgeführt

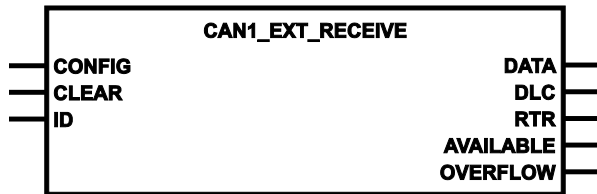
CAN1_EXT_RECEIVE

4302

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CAN1_EXT_Vxxyyyz.LIB

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.

! Soll dieser FB verwendet werden, muss zuvor mit **CAN1_EXT** (→ Seite 66) die 1. CAN-Schnittstelle für den erweiterten ID initialisiert werden.

Parameter der Eingänge

2172

Parameter	Datentyp	Beschreibung
CONFIG	BOOL	TRUE (im 1. Zyklus): Datenobjekt konfigurieren FALSE: im weiteren Programmablauf
CLEAR	BOOL	TRUE: Empfangspuffer löschen FALSE: Funktion wird nicht ausgeführt
ID	DWORD	Nummer des Datenobjekt-Identifiers: Normal Frame (2 ¹¹ IDs): 0...2 047 = 0x0000 0000...0x0000 07FF Extended Frame (2 ²⁹ IDs): 0...536 870 911 = 0x0000 0000...0x1FFF FFFF

Parameter der Ausgänge

19810

Parameter	Datentyp	Beschreibung
DATA	ARRAY [0..7] OF BYTE	empfangene Daten (1...8 Bytes)
DLC	BYTE	Anzahl der mit RDO empfangenen Bytes im Array DATA zulässig: 0...8
RTR	BOOL = FALSE	empfangene Nachricht war ein Remote Transmission Request (wird hier nicht unterstützt)
AVAILABLE	BYTE	Anzahl der verbleibenden Datenbytes zulässig = 0...16 0 = keine gültigen Daten vorhanden
OVERFLOW	BOOL	TRUE: Überlauf des Datenpuffers ⇒ Datenverlust! FALSE: Datenpuffer ist ohne Datenverlust



CAN1_EXT_TRANSMIT

4307

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CAN1_EXT_Vxxyyzz.LIB`

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.

! Soll dieser FB verwendet werden, muss zuvor mit **CAN1_EXT** (→ Seite [66](#)) die 1. CAN-Schnittstelle für den erweiterten ID initialisiert werden.

Parameter der Eingänge

4380

Parameter	Datentyp	Beschreibung
ID	DWORD	Nummer des Datenobjekt-Identifiers: Normal Frame (2 ¹¹ IDs): 0...2 047 = 0x0000 0000...0x0000 07FF Extended Frame (2 ²⁹ IDs): 0...536 870 911 = 0x0000 0000...0x1FFF FFFF
DLC	BYTE	Anzahl der mit RDO zu übertragenden Bytes aus dem Array DATA zulässig: 0...8
DATA	ARRAY [0..7] OF BYTE	zu sendende Daten (1...8 Bytes)
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert

Parameter der Ausgänge

614

Parameter	Datentyp	Beschreibung
RESULT	BOOL	TRUE (nur 1 Zyklus lang): der Baustein hat den Sendeauftrag angenommen FALSE: Sendeauftrag wurde nicht angenommen

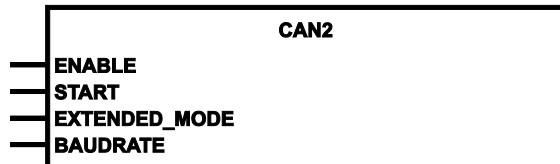
CAN2

639

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxyxyz.LIB

Symbol in CODESYS:



Beschreibung

642

CAN2 initialisiert die 2. CAN-Schnittstelle.

Der FB muss aufgerufen werden, wenn die 2. CAN-Schnittstelle benutzt werden soll.

Eine Änderung der Baudrate wird erst gültig nach Spannung Aus/Ein.

Die Baudraten 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.

Für die 2. CAN-Schnittstelle stehen u.a. Funktionsbibliotheken für *SAE J1939* und *Nutzung der CAN-Schnittstelle nach ISO 11992* zur Verfügung. Die Funktionsblöcke nach ISO 11992 sind nur verfügbar im CR2501 auf der 2. CAN-Schnittstelle.

! Der FB muss **vor** den Funktionen CAN2_... ausgeführt werden.

Parameter der Eingänge

643

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
START	BOOL	TRUE (im 1. Zyklus): CAN-Protokoll an CAN-Schnittstelle x starten FALSE: im weiteren Programmablauf
EXTENDED_MODE	BOOL := FALSE	TRUE: Identifier der CAN-Schnittstelle arbeitet mit 29 Bits FALSE: Identifier der CAN-Schnittstelle arbeitet mit 11 Bits
BAUDRATE	WORD := 125	Baudrate [kBit/s] zulässig = 50, 100, 125, 250, 500, 800, 1000

CANx_ERRORHANDLER

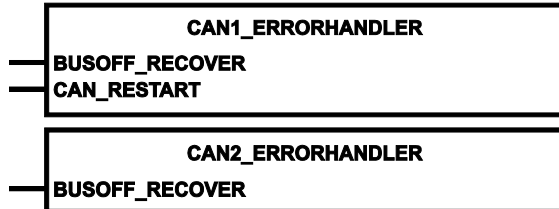
633

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

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 gearterter CAN-Fehler. Die CAN-Schnittstelle wird dadurch neu initialisiert.

Das Fehlerbit muss anschließend im Anwendungsprogramm 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** (→ Seite [71](#)) 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 instanziiert werden!

Parameter der Eingänge

637

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

CANx_EXT_RECEIVE_ALL

4183

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

Baustein-Typ = Funktionsbaustein (FB)

CAN1_EXT_RECEIVE_ALL: Baustein ist enthalten in Bibliothek ifm_CAN1_EXT_Vxxyzz.LIB

CAN2_EXT_RECEIVE_ALL: Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



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 (im 1. Zyklus): Datenobjekt konfigurieren FALSE: im weiteren Programmablauf
CLEAR	BOOL	TRUE: Empfangspuffer löschen FALSE: Funktion wird nicht ausgeführt

Parameter der Ausgänge

2292

Parameter	Datentyp	Beschreibung
ID	DWORD	Nummer des Datenobjekt-Identifiers
DATA	ARRAY [0..7] OF BYTE	empfangene Daten (1...8 Bytes)
DLC	BYTE	Anzahl der mit SRDO empfangenen Bytes im Array DATA zulässig: 0...8
AVAILABLE	BYTE	Anzahl der verbleibenden Datenbytes zulässig = 0...16 0 = keine gültigen Daten vorhanden
OVERFLOW	BOOL	TRUE: Überlauf des Datenpuffers ⇒ Datenverlust! FALSE: Datenpuffer ist ohne Datenverlust



CANx_RECEIVE

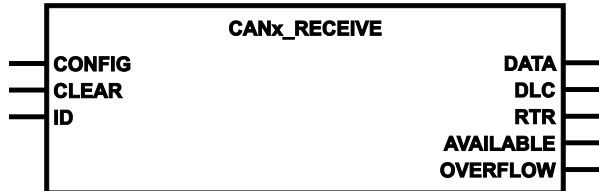
627

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



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.

! Soll CAN2_RECEIVE verwendet werden, muss zuvor mit **CAN2** (→ Seite [71](#)) die zweite CAN-Schnittstelle initialisiert werden.

Parameter der Eingänge

631

Parameter	Datentyp	Beschreibung
CONFIG	BOOL	TRUE (im 1. Zyklus): Datenobjekt konfigurieren FALSE: im weiteren Programmablauf
CLEAR	BOOL	TRUE: Empfangspuffer löschen FALSE: Funktion wird nicht ausgeführt
ID	WORD	Nummer des Datenobjekt-Identifier Zulässige Werte: 0...2 047

Parameter der Ausgänge

19810

Parameter	Datentyp	Beschreibung
DATA	ARRAY [0..7] OF BYTE	empfangene Daten (1...8 Bytes)
DLC	BYTE	Anzahl der mit RDO empfangenen Bytes im Array DATA zulässig: 0...8
RTR	BOOL = FALSE	empfangene Nachricht war ein Remote Transmission Request (wird hier nicht unterstützt)
AVAILABLE	BYTE	Anzahl der verbleibenden Datenbytes zulässig = 0...16 0 = keine gültigen Daten vorhanden
OVERFLOW	BOOL	TRUE: Überlauf des Datenpuffers ⇒ Datenverlust! FALSE: Datenpuffer ist ohne Datenverlust



CANx_RECEIVE_RANGE

4179

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB (xx ≥ 05)

Symbol in CODESYS:



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 78)).

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 dieselben IDs an denselben CAN-Schnittstellen NICHT gemischt eingesetzt werden mit **CANx_RECEIVE** (→ Seite 75) 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 (im 1. Zyklus): Datenobjekt konfigurieren FALSE: im weiteren Programmablauf
CLEAR	BOOL	TRUE: Empfangspuffer löschen FALSE: 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...536 870 911 (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...536 870 911 (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	empfangene Daten (1...8 Bytes)
DLC	BYTE	Anzahl der mit RDO empfangenen Bytes im Array DATA zulässig: 0...8
AVAILABLE	BYTE	Anzahl der verbleibenden Datenbytes zulässig = 0...16 0 = keine gültigen Daten vorhanden
OVERFLOW	BOOL	TRUE: Überlauf des Datenpuffers ⇒ Datenverlust! FALSE: Datenpuffer ist ohne Datenverlust

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
0009
0010 (* CAN2 init *)
0011 can20(ENABLE:= TRUE , START:= init, EXTENDED_MODE:= FALSE, BAUDRATE:= 125);
0012
0013 (* CAN2_RECEIVE_RANGE in mehreren Steps initialisieren *)
0014 CASE initstep OF
0015   1:
0016     cr2(CONFIG:= TRUE, CLEAR:= FALSE, FIRST_ID:= 16#100, LAST_ID:= 16#10F, ID=> , DATA=> , DLC=> , AVAILABLE=> , OVERFLOW=> );
0017     initstep := initstep + 1;
0018   2:
0019     cr2(CONFIG:= TRUE, CLEAR:= FALSE, FIRST_ID:= 16#110, LAST_ID:= 16#11F, ID=> , DATA=> , DLC=> , AVAILABLE=> , OVERFLOW=> );
0020     initstep := initstep + 1;
0021   3:
0022     cr2(CONFIG:= TRUE, CLEAR:= FALSE, FIRST_ID:= 16#120, LAST_ID:= 16#12F, ID=> , DATA=> , DLC=> , AVAILABLE=> , OVERFLOW=> );
0023     initstep := initstep + 1;
0024   4:
0025     cr2(CONFIG:= TRUE, CLEAR:= FALSE, FIRST_ID:= 16#130, LAST_ID:= 16#13F, ID=> , DATA=> , DLC=> , AVAILABLE=> , OVERFLOW=> );
0026     initstep := initstep + 1;
0027   ELSE
0028     cr2(CONFIG:= FALSE, CLEAR:= FALSE, FIRST_ID:= 16#100, LAST_ID:= 16#100, ID=> , DATA=> , DLC=> , AVAILABLE=> , OVERFLOW=> );
0029 END_CASE
0030
0031 init := FALSE;
0032
0033 (* Test *)
0034 IF cr2.available > 0 THEN
0035   cnt := cnt + 1;
0036 END_IF

```

CANx_TRANSMIT

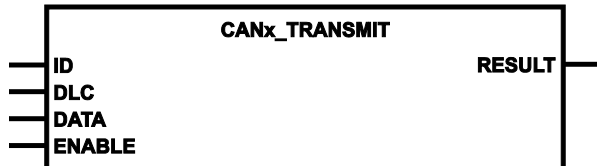
609

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



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.

! Soll CAN2_TRANSMIT verwendet werden, muss zuvor mit **CAN2** (→ Seite [71](#)) die zweite CAN-Schnittstelle initialisiert werden.

Parameter der Eingänge

613

Parameter	Datentyp	Beschreibung
ID	WORD	Nummer des Datenobjekt-Identifizier Zulässige Werte: 0...2 047
DLC	BYTE	Anzahl der mit RDO zu übertragenden Bytes aus dem Array DATA zulässig: 0...8
DATA	ARRAY [0..7] OF BYTE	zu sendende Daten (1...8 Bytes)
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert

Parameter der Ausgänge

614

Parameter	Datentyp	Beschreibung
RESULT	BOOL	TRUE (nur 1 Zyklus lang): der Baustein hat den Sendeauftrag angenommen FALSE: Sendeauftrag wurde nicht angenommen

5.2.2 Bausteine: CANopen-Master

Inhalt

CANx_MASTER_EMCY_HANDLER	81
CANx_MASTER_SEND_EMERGENCY	82
CANx_MASTER_STATUS	84

1870

Für den CANopen-Master stellt **ifm electronic** eine Reihe von Bausteinen zur Verfügung, die im Folgenden erklärt werden.

CANx_MASTER_EMCY_HANDLER

13192

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_CANopenMaster_Vxxyyzz.LIB

Symbol in CODESYS:



Beschreibung

2009

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

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

Über den FB können die aktuellen Werte aus dem Error-Register (Index 0x1001/01) und Error Field (Index 0x1003/0-5) des CANopen-Objektverzeichnis ausgelesen werden.

! Sollen anwendungsspezifische Fehlernachrichten im Objektverzeichnis gespeichert werden, muss CANx_MASTER_EMCY_HANDLER **nach** dem (mehrfachen) Bearbeiten von **CANx_MASTER_SEND_EMERGENCY** (→ Seite [82](#)) aufgerufen werden.

Parameter der Eingänge

2010

Parameter	Datentyp	Beschreibung
CLEAR_ERROR_FIELD	BOOL	FALSE ⇒ TRUE (Flanke): • Inhalt des ERROR_FIELD an FB-Ausgang ausgeben • Inhalt des ERROR_FIELD im Objektverzeichnis löschen sonst: diese Funktion wird nicht ausgeführt

Parameter der Ausgänge

2011

Parameter	Datentyp	Beschreibung
ERROR_REGISTER	BYTE	Zeigt den Inhalt des OBV Index 0x1001 (Error-Register)
ERROR_FIELD	ARRAY [0..5] OF WORD	Zeigt den Inhalt des OBV Index 0x1003 (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

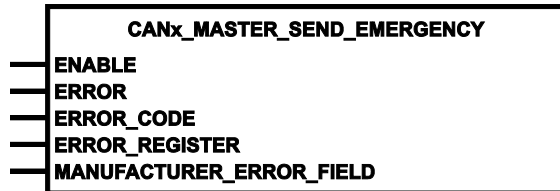
13195

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_CANopenMaster_Vxxyzz.LIB`

Symbol in CODESYS:



Beschreibung

2015

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

! Sollen anwendungsspezifische Fehlernachrichten im Objektverzeichnis gespeichert werden, muss **CANx_MASTER_EMCY_HANDLER** (→ Seite [81](#)) **nach** dem (mehrfachen) Bearbeiten von CANx_MASTER_SEND_EMERGENCY aufgerufen werden.

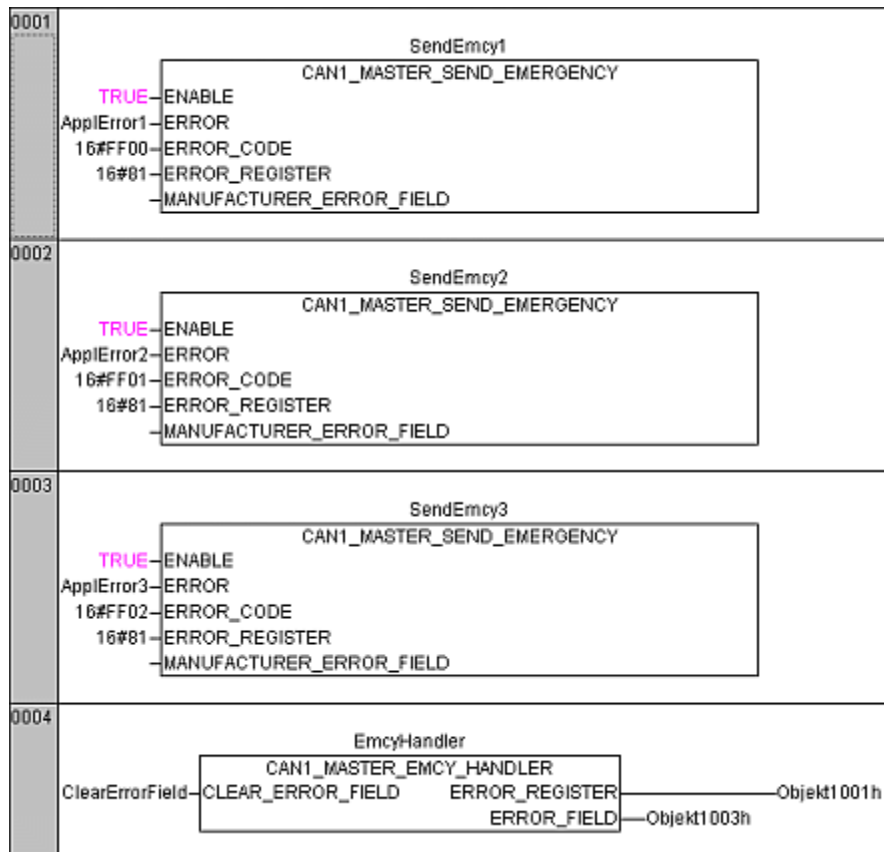
Parameter der Eingänge

2016

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
ERROR	BOOL	Über diesen Eingang wird dem FB die Information übergeben, ob der zum konfigurierten Fehlercode gehörende Fehler aktuell anliegt. FALSE ⇒ TRUE (Flanke): sendet den anstehenden Fehler-Code falls Eingang in der letzten Sekunde nicht TRUE war TRUE ⇒ FALSE (Flanke) UND Fehler steht nicht mehr an: Nach Verzögerung von ca. 1 s: > Null-Fehlerrmeldung wird 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.
ERROR_REGISTER	BYTE	ERROR_REGISTER gibt die Art des Fehlers an. Der hier angegebene Wert wird mit allen anderen aktuell aktiven Fehlernachrichten bitweise ODER-verknüpft. Der sich hierbei ergebende Wert wird ins Error-Register (Index 1001 ₁₆ /00) geschrieben und mit der EMCY-Nachricht versendet. Die Werte sollten gemäß der CANopen-Spezifikation eingetragen werden.
MANUFACTURER_ERROR_FIELD	ARRAY [0..4] OF BYTE	Hier können bis zu 5 Bytes anwendungsspezifische 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 = 0xFF00 im Fehlerregister 0x81
2. ApplError2, Code = 0xFF01 im Fehlerregister 0x81
3. ApplError3, Code = 0xFF02 im Fehlerregister 0x81

Der FB CAN1_MASTER_EMCY_HANDLER sendet die Fehlermeldungen an das Fehler-Register "Objekt 0x1001" im Fehler-Array "Objekt 0x1003".

CANx_MASTER_STATUS

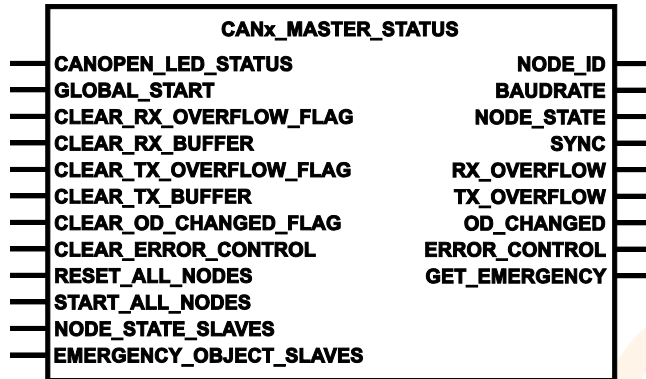
2021

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_CANopenMaster_Vxxyyzz.LIB

Symbol in CODESYS:



Beschreibung

2024

Status-Anzeige des als CANopen-Master eingesetzten Gerätes

Der FB zeigt den Status des als CANopen-Master eingesetzten Gerätes an. Weitere Möglichkeiten:

- den Status des Netzwerks überwachen
- den Status der angeschlossenen Slaves überwachen
- die Slaves im Netzwerk zurücksetzen oder starten.

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

2025

Parameter	Datentyp	Beschreibung
CANOPEN_LED_STATUS	BOOL	(Eingang ist nicht für PDM-Geräte verfügbar) TRUE: Die Status-LED der Steuerung wird in den Modus "CANopen" geschaltet: Blinkfrequenz 0,5 Hz = PRE-OPERATIONAL Blinkfrequenz 2,0 Hz = OPERATIONAL Die sonstigen LED-Diagnoseanzeigen werden durch diese Betriebsart nicht verändert.
GLOBAL_START	BOOL	TRUE: Alle angeschlossenen Netzwerkteilnehmer (Slaves) werden gleichzeitig bei der Netzwerkinitialisierung gestartet (⇒ Zustand OPERATIONAL). FALSE: Die angeschlossenen Netzwerkteilnehmer werden einzeln nacheinander gestartet.
CLEAR_RX_OVERFLOW_FLAG	BOOL	FALSE ⇒ TRUE (Flanke): Fehlerflag RX_OVERFLOW löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_RX_BUFFER	BOOL	FALSE ⇒ TRUE (Flanke): Daten im Empfangspuffer löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_TX_OVERFLOW_FLAG	BOOL	FALSE ⇒ TRUE (Flanke): Fehlerflag TX_OVERFLOW löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_TX_BUFFER	BOOL	FALSE ⇒ TRUE (Flanke): Daten im Sendepuffer löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_OD_CHANGED_FLAG	BOOL	FALSE ⇒ TRUE (Flanke): Flag OD_CHANGED löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_ERROR_CONTROL	BOOL	FALSE ⇒ TRUE (Flanke): Die Guard-Fehlerliste (ERROR_CONTROL) löschen sonst: diese Funktion wird nicht ausgeführt
RESET_ALL_NODES	BOOL	FALSE ⇒ TRUE (Flanke): Alle angeschlossenen Netzwerkteilnehmer (Slaves) werden per NMT-Kommando zurückgesetzt sonst: diese Funktion wird nicht ausgeführt
START_ALL_NODES	BOOL	FALSE ⇒ TRUE (Flanke): Alle angeschlossenen Netzwerkteilnehmer (Slaves) werden per NMT-Kommando gestartet sonst: diese Funktion wird nicht ausgeführt
NODE_STATE_SLAVES	DWORD	Zeigt den Status aller Netzknoten. ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben! Beispiel-Code → Kapitel <i>Beispiel: CANx_MASTER_STATUS</i> (→ Seite 87)
EMERGENCY_OBJECT_SLAVES	DWORD	Zeigt die zuletzt aufgetretenen Fehlermeldungen aller Netzknoten. ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben! i → Kapitel <i>Zugriff auf die Strukturen zur Laufzeit der Anwendung</i> (→ Seite 88)

Parameter der Ausgänge

2029

Parameter	Datentyp	Beschreibung
NODE_ID	BYTE	aktuelle Knoten-ID des CANopen-Masters
BAUDRATE	WORD	aktuelle Baudrate des CANopen-Masters in [kBaud]
NODE_STATE	INT	aktueller Status des CANopen-Masters
SYNC	BOOL	SYNC-Signal des CANopen-Masters TRUE: Im letzten Zyklus wurde ein SYNC-Signal gesendet FALSE: Im letzten Zyklus wurde kein SYNC-Signal gesendet
RX_OVERFLOW	BOOL	TRUE: Fehler: Empfangspuffer-Überlauf FALSE: kein Überlauf
TX_OVERFLOW	BOOL	TRUE: Fehler: Sendepuffer-Überlauf FALSE: kein Überlauf
OD_CHANGED	BOOL	TRUE: Daten im Objektverzeichnis des CANopen-Masters wurden geändert FALSE: keine Datenänderung
ERROR_CONTROL	ARRAY [0..7] OF BYTE	Das Array enthält die Liste (max. 8) der fehlenden Netzwerknoten (Guard- oder Heartbeat-Fehler)  → Kapitel <i>Zugriff auf die Strukturen zur Laufzeit der Anwendung</i> (→ Seite 88)
GET_EMERGENCY	STRUCT CANx_EMERGENCY_MESSAGE	Am Ausgang stehen die Daten für die Struktur CANx_EMERGENCY_MESSAGE zur Verfügung. Es wird immer die zuletzt empfangene EMCY-Nachricht im CANopen-Netzwerk angezeigt. Um eine Liste aller aufgetretenen Fehler zu erhalten, muss das Array "EMERGENCY_OBJECT_SLAVES" ausgewertet werden.

Parameter der internen Strukturen

2030

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

Parameter	Datentyp	Beschreibung
CANx_EMERGENCY_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_CR0303_CANopenMaster_Vxxyzz.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_CR0303_CANopenMaster_Vxxyzz.LIB angelegt.

Die folgenden Code-Fragmente zeigen Ihnen am Beispiel des Controllers CR0020 die Anwendung des FB CANx_MASTER_STATUS.

Beispiel: CANx_MASTER_STATUS

2031

Slave-Informationen

2033

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.

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

```
0001 PROGRAM MasterStatus
0002 VAR
0003   Status: CR0020_MASTER_STATUS;
0004   LedStatus: BOOL := TRUE;
0005   GlobalStartNodes: BOOL := TRUE;
0006   ClearRxOverflowFlag: BOOL;
0007   ClearRxBuffer: BOOL;
0008   ClearTxOverflowFlag: BOOL;
0009   ClearTxBuffer: BOOL;
0010   ClearOdChanged: BOOL;
0011   ClearErrorControl: BOOL;
0012   ResetAllNodes: BOOL;
0013   StartAllNodes: BOOL;
0014   NodeId: BYTE;
0015   Baudrate: WORD;
0016   NodeState: INT;
0017   Sync: BOOL;
0018   RxOverflow: BOOL;
0019   TxOverflow: BOOL;
0020   OdChanged: BOOL;
0021   GuardHeartbeatErrorArray: ARRAY[0..7] OF BYTE;
0022   GetEmergency: EMERGENCY_MESSAGE;
0023 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

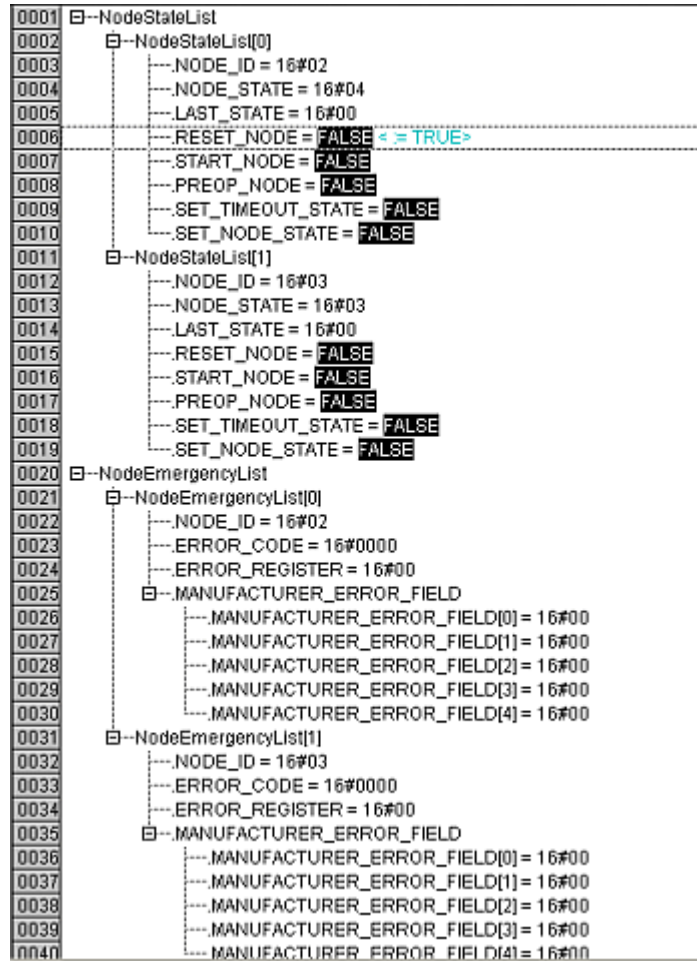
2035

```
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 Anwendung

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.



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

 zu den möglichen Fehler-Codes: → Systemhandbuch "Know-How ecomatmobile"
→ Kapitel *CAN / CANopen: Fehler und Fehlerbehandlung*.

5.2.3 Bausteine: CANopen-Slave

Inhalt

CANx_SLAVE_EMCY_HANDLER	90
CANx_SLAVE_NODEID	91
CANx_SLAVE_SEND_EMERGENCY	92
CANx_SLAVE_STATUS	94

1874

Für den CANopen-Slave stellt **ifm electronic** eine Reihe von Bausteinen zur Verfügung, die im Folgenden erklärt werden.

CANx_SLAVE_EMCY_HANDLER

13199

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_CANOpenSlave_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

2053

CANx_SLAVE_EMCY_HANDLER verwaltet den geräteeigenen Fehlerstatus des CANopen-Slaves:

- Error Register (Index 0x1001) und
 - Error Field (Index 0x1003) des CANopen Objektverzeichnis.
- Den FB in folgenden Fällen aufrufen:
- der Fehlerstatus soll ins CAN-Netzwerk übertragen werden und
 - die Fehlernachrichten des Anwendungsprogramms sollen im Objektverzeichnis gespeichert werden.

! Sollen die Fehlernachrichten im Objektverzeichnis gespeichert werden?

► **Nach** dem (mehrfachen) Bearbeiten von **CANx_SLAVE_SEND_EMERGENCY** (→ Seite [92](#)) einmalig CANx_SLAVE_EMCY_HANDLER aufrufen!

Parameter der Eingänge

2054

Parameter	Datentyp	Beschreibung
CLEAR_ERROR_FIELD	BOOL	FALSE ⇒ TRUE (Flanke): • Inhalt des ERROR_FIELD an FB-Ausgang ausgeben • Inhalt des ERROR_FIELD im Objektverzeichnis löschen sonst: diese Funktion wird nicht ausgeführt

Parameter der Ausgänge

2055

Parameter	Datentyp	Beschreibung
ERROR_REGISTER	BYTE	Zeigt den Inhalt des OBV Index 0x1001 (Error-Register)
ERROR_FIELD	ARRAY [0..5] OF WORD	Zeigt den Inhalt des OBV Index 0x1003 (Error-Field) ERROR_FIELD[0]: Anzahl der gespeicherten Fehler ERROR_FIELD[1...5]: gespeicherte Fehler, der jüngste Fehler steht im Index [1]

CANx_SLAVE_NODEID

13202

= CANx Slave Node-ID

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_CANopenSlave_Vxxyyzz.LIB

Symbol in CODESYS:



Beschreibung

2049

CANx_SLAVE_NODEID ermöglicht das Einstellen der Node-ID eines CANopen-Slaves zur Laufzeit des Anwendungsprogramms.

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): Parameter übernehmen und aktivieren sonst: diese Funktion wird nicht ausgeführt
NODEID	BYTE	Node-ID = ID des Knotens zulässige Werte = 0...127

CANx_SLAVE_SEND_EMERGENCY

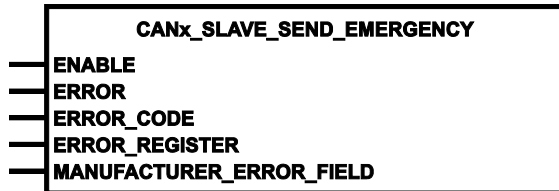
13205

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_CANOpenSlave_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

2059

CANx_SLAVE_SEND_EMERGENCY versendet anwendungsspezifische Fehlerstatus. Das sind Fehlermeldungen, die zusätzlich zu den geräteinternen Fehlermeldungen (z.B. Kurzschluss am Ausgang) gesendet werden sollen.

- Den FB aufrufen, wenn der Fehlerstatus an andere Geräte im Netzwerkverbund übertragen werden soll.

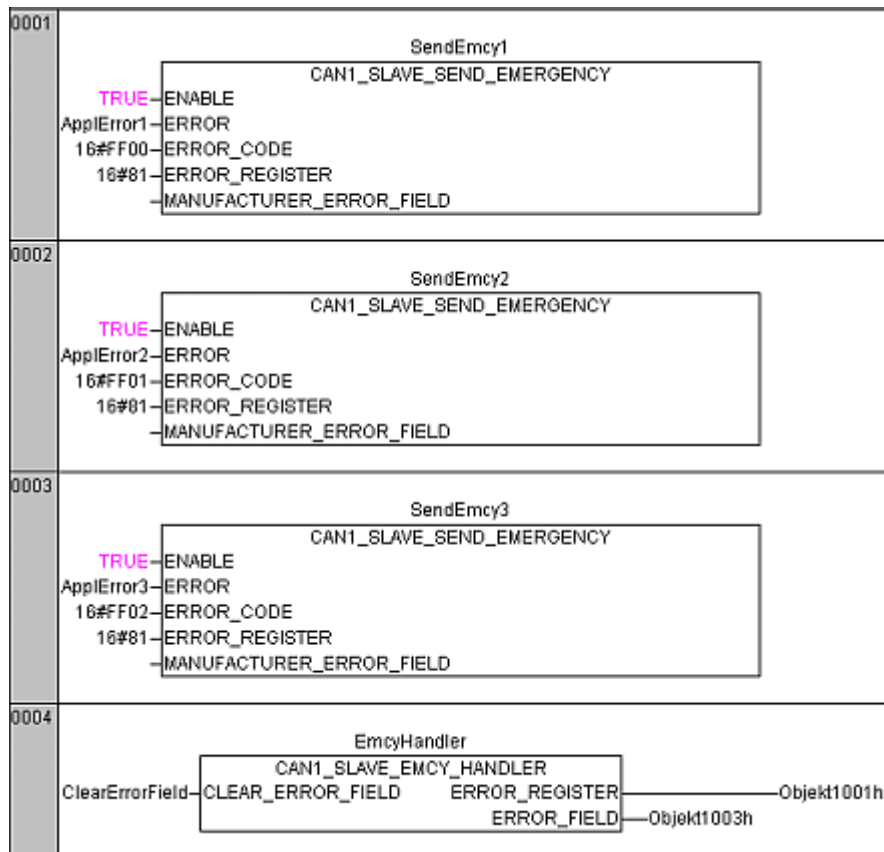
Parameter der Eingänge

2060

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
ERROR	BOOL	Über diesen Eingang wird dem FB die Information übergeben, ob der zum konfigurierten Fehlercode gehörende Fehler aktuell anliegt. FALSE ⇒ TRUE (Flanke): sendet den anstehenden Fehler-Code falls Eingang in der letzten Sekunde nicht TRUE war TRUE ⇒ FALSE (Flanke) UND Fehler steht nicht mehr an: Nach Verzögerung von ca. 1 s: > Null-Fehlermeldung wird 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.
ERROR_REGISTER	BYTE	ERROR_REGISTER gibt die Art des Fehlers an. Der hier angegebene Wert wird mit allen anderen aktuell aktiven Fehlermeldungen bitweise ODER-verknüpft. Der sich hierbei ergebende Wert wird ins Error-Register (Index 1001 _h /00) geschrieben und mit der EMCY-Nachricht versendet. Die Werte sollten gemäß der CANOpen-Spezifikation eingetragen werden.
MANUFACTURER_ERROR_FIELD	ARRAY [0..4] OF BYTE	Hier können bis zu 5 Bytes anwendungsspezifische 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 = 0xFF00 im Fehlerregister 0x81
2. ApplError2, Code = 0xFF01 im Fehlerregister 0x81
3. ApplError3, Code = 0xFF02 im Fehlerregister 0x81

Der FB CAN1_SLAVE_EM CY_HANDLER sendet die Fehlermeldungen an das Fehler-Register "Objekt 0x1001" im Fehler-Array "Objekt 0x1003".

CANx_SLAVE_STATUS

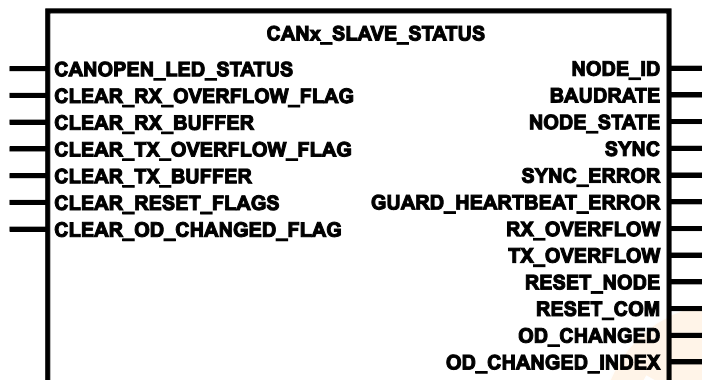
2063

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_CANOpenSlave_Vxxyyzz.LIB

Symbol in CODESYS:



Beschreibung

2066

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.

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

Beispiel:

```

0001 PROGRAM SlaveStatus
0002 VAR
0003   SlaveStatus: CR0505_SLAVE_STATUS;
0004   LedStatus: BOOL := TRUE;
0005   ClearRxOverflowFlag: BOOL;
0006   ClearRxBuffer: BOOL;
0007   ClearTxOverflowFlag: BOOL;
0008   ClearTxBuffer: BOOL;
0009   ClearResetFlags: BOOL;
0010   ClearOdChanged: BOOL;
0011   NodeId: BYTE;
0012   Baudrate: WORD;
0013   NodeState: BYTE;
0014   Sync: BOOL;
0015   SyncError: BOOL;
0016   GuardHeartbeatError: BOOL;
0017   RxOverflow: BOOL;
0018   TxOverflow: BOOL;
0019   ResetNode: BOOL;
0020   ResetCom: BOOL;
0021   OdChanged: BOOL;
0022   OdChangedIndex: INT;
0023 END_VAR
    
```

Parameter der Eingänge

2067

Parameter	Datentyp	Beschreibung
CANOPEN_LED_STATUS	BOOL	(Eingang ist nicht für PDM-Geräte verfügbar) TRUE: Die Status-LED der Steuerung wird in den Modus "CANopen" geschaltet: Blinkfrequenz 0,5 Hz = PRE-OPERATIONAL Blinkfrequenz 2,0 Hz = OPERATIONAL Die sonstigen LED-Diagnoseanzeigen werden durch diese Betriebsart nicht verändert.
GLOBAL_START	BOOL	TRUE: Alle angeschlossenen Netzwerkteilnehmer (Slaves) werden gleichzeitig bei der Netzwerkinitialisierung gestartet (⇒ Zustand OPERATIONAL). FALSE: Die angeschlossenen Netzwerkteilnehmer werden einzeln nacheinander gestartet.
CLEAR_RX_OVERFLOW_FLAG	BOOL	FALSE ⇒ TRUE (Flanke): Fehlerflag RX_OVERFLOW löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_RX_BUFFER	BOOL	FALSE ⇒ TRUE (Flanke): Daten im Empfangspuffer löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_TX_OVERFLOW_FLAG	BOOL	FALSE ⇒ TRUE (Flanke): Fehlerflag TX_OVERFLOW löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_TX_BUFFER	BOOL	FALSE ⇒ TRUE (Flanke): Daten im Sendepuffer löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_RESET_FLAGS	BOOL	FALSE ⇒ TRUE (Flanke): Flag RESET_NODE löschen Flag RESET_COM löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_OD_CHANGED_FLAGS	BOOL	FALSE ⇒ TRUE (Flanke): Flag OD_CHANGED löschen Flag OD_CHANGED_INDEX löschen sonst: diese Funktion wird nicht ausgeführt

Parameter der Ausgänge

2068

Parameter	Datentyp	Beschreibung
NODE_ID	BYTE	aktuelle Knoten-ID des CANopen-Slaves
BAUDRATE	WORD	aktuelle Baudrate des CANopen-Knotens in [kBaud]
NODE_STATE	BYTE	aktueller Status des CANopen-Slaves 0 = Bootup-Nachricht versendet 4 = CANopen-Slave im Status PRE-OPERATIONAL und wird per SDO-Zugriff konfiguriert 5 = CANopen-Slave im Status OPERATIONAL 127 = CANopen-Slave im Status PRE-OPERATIONAL
SYNC	BOOL	SYNC-Signal des CANopen-Masters TRUE: Im letzten Zyklus wurde ein SYNC-Signal empfangen FALSE: Im letzten Zyklus wurde kein SYNC-Signal empfangen
SYNC_ERROR	BOOL	TRUE: Fehler: das SYNC-Signal des Masters wurde nicht oder zu spät (nach Ablauf von ComCyclePeriod) empfangen FALSE: kein SYNC-Fehler
GUARD_HEARTBEAT_ERROR	BOOL	TRUE: Fehler: das Guard- oder Heartbeat-Signal des Masters wurde nicht oder zu spät empfangen FALSE: kein Guard- oder Heartbeat-Fehler
RX_OVERFLOW	BOOL	TRUE: Fehler: Empfangspuffer-Überlauf FALSE: kein Überlauf
TX_OVERFLOW	BOOL	TRUE: Fehler: Sendepuffer-Überlauf FALSE: kein Überlauf
RESET_NODE	BOOL	TRUE: CANopen-Stack des Slaves vom Master zurückgesetzt FALSE: CANopen-Stack des Slaves nicht zurückgesetzt
RESET_COM	BOOL	TRUE: Kommunikations-Interface des CAN-Stack wurde vom Master zurückgesetzt FALSE: Kommunikations-Interface nicht zurückgesetzt
OD_CHANGED	BOOL	TRUE: Daten im Objektverzeichnis des CANopen-Masters wurden geändert FALSE: keine Datenänderung
OD_CHANGED_INDEX	INT	Index des zuletzt geänderten Objektverzeichnis-Eintrags

5.2.4 Bausteine: CANopen SDOs

Inhalt

CANx_SDO_READ	98
CANx_SDO_WRITE.....	100

2071

Hier finden Sie **ifm**-Bausteine für den Umgang von CANopen mit Service Data Objects (SDOs).



CANx_SDO_READ

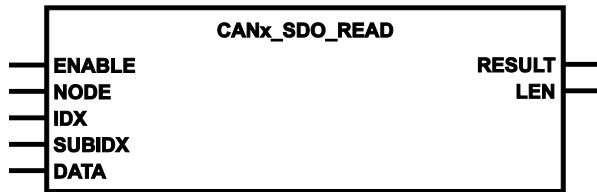
621

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

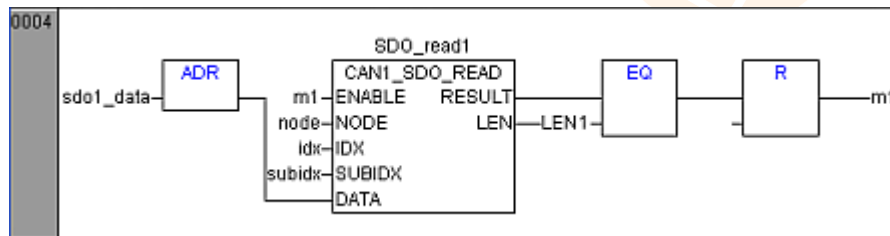
624

CANx_SDO_READ liest das → **SDO** (→ Seite [220](#)) mit den angegebenen Indizes aus dem Knoten aus.

Voraussetzung: Knoten muss sich im Zustand PRE-OPERATIONAL oder OPERATIONAL befinden.

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

Beispiel:



Parameter der Eingänge

625

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
NODE	BYTE	CANopen-ID des Knotens zulässig = 1...127 = 0x01...0x7F
IDX	WORD	Index im Objektverzeichnis
SUBIDX	BYTE	Subindex bezogen auf den Index im Objektverzeichnis
DATA	DWORD	Adresse des Empfangsdaten-Arrays zulässige Länge = 0...255 ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Parameter der Ausgänge

626

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)
LEN	WORD	Länge des Eintrags in "Anzahl der Bytes" Der Wert für LEN darf nicht größer sein als die Größe des Empfangs-Arrays. Andernfalls werden beliebige Daten in der Anwendung überschrieben.

Mögliche Ergebnisse für RESULT:

Wert dez	 hex	Beschreibung
0	00	FB ist inaktiv
1	01	FB-Ausführung wurde ohne Fehler beendet – Daten sind gültig
2	02	Funktionsbaustein ist aktiv (Aktion noch nicht beendet)
3	03	Fehler, keine Daten während der Überwachungszeit empfangen

CANx_SDO_WRITE

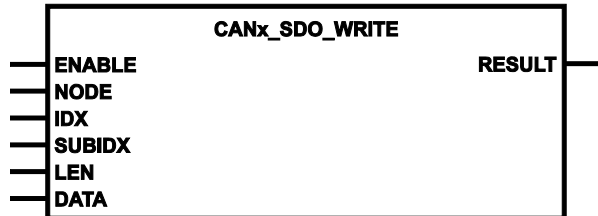
615

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

618

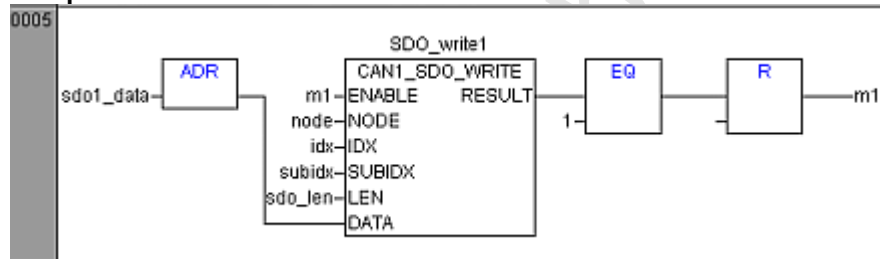
CANx_SDO_WRITE schreibt das →SDO (→ Seite [220](#)) mit den angegebenen Indizes in den Knoten.

Voraussetzung: Knoten muss sich im Zustand PRE-OPERATIONAL oder OPERATIONAL befinden.

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

! Der Wert für LEN muss kleiner sein als die Größe des Sende-Arrays. Andernfalls werden beliebige Daten versendet.

Beispiel:



Parameter der Eingänge

619

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
NODE	BYTE	CANopen-ID des Knotens zulässig = 1...127 = 0x01...0x7F
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 darf nicht größer sein als die Größe des Sendearrays. Andernfalls werden beliebige Daten versendet.
DATA	DWORD	Adresse des Sendedaten-Arrays zulässige Länge = 0...255 ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Parameter der Ausgänge

620

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für RESULT:

Wert dez hex		Beschreibung
0	00	FB ist inaktiv
1	01	FB-Ausführung wurde ohne Fehler beendet – Daten sind gültig
2	02	Funktionsbaustein ist aktiv (Aktion noch nicht beendet)
3	03	Fehler, Daten können nicht übertragen werden

5.2.5 Bausteine: SAE J1939

Inhalt	
J1939_x	103
J1939_x_GLOBAL_REQUEST	104
J1939_x_RECEIVE	106
J1939_x_RESPONSE	108
J1939_x_SPECIFIC_REQUEST	110
J1939_x_TRANSMIT	112

2273

Für SAE J1939 stellt **ifm electronic** eine Reihe von Bausteinen zur Verfügung, die im Folgenden erklärt werden.

J1939_x

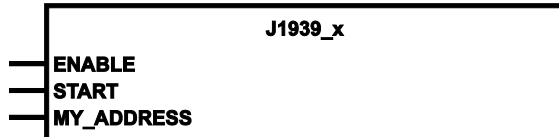
9375

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_J1939_x_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

4325

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

4313

! HINWEIS

(Nur für LZS bis V05)

J1939-Kommunikation über 1. CAN-Schnittstelle:

- ▶ Schnittstelle zuvor mit **CAN1_EXT** (→ Seite [66](#)) initialisieren!

J1939-Kommunikation über 2. CAN-Schnittstelle:

- ▶ Schnittstelle zuvor mit **CAN2** (→ Seite [71](#)) 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 ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
START	BOOL	TRUE (nur 1 Zyklus lang): J1939-Protokoll an CAN-Schnittstelle x starten FALSE: im weiteren Programmablauf
MY_ADDRESS	BYTE	J1939-Adresse des Geräts

J1939_x_GLOBAL_REQUEST

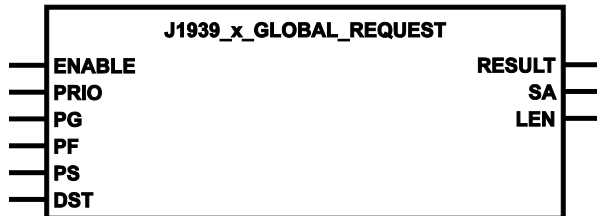
4315

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_J1939_x_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

2301

J1939_x_GLOBAL_REQUEST ist für das automatische Anfordern einzelner Nachrichten von allen (global) aktiven J1939-Netzwerkteilnehmern verantwortlich. Dazu werden dem FB die Parameter PG, PF, PS und die Adresse des Arrays DST übergeben, in dem die empfangenen Daten abgelegt werden.

Info

PGN = [Page] + [PF] + [PS]

PDU = [PRIO] + [PGN] + [J1939-Adresse] + [Daten]

13790

ACHTUNG

Daten können unzulässig überschrieben werden!

- ▶ Ein Empfangs-Array mit einer Größe von 1 785 Bytes anlegen!
Dies ist die maximale Größe einer J1939-Nachricht.
- ▶ Die Anzahl empfangener Daten prüfen:
der Wert darf nicht größer sein als das bereitgestellte Empfangs-Array!
- ▶ Für jede angefragte Nachricht eine eigene Instanz des FBs verwenden!
- ▶ Für die Zieladresse DST gilt:
 - ❗ 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.
 - RESULT = 2: der Baustein wartet auf Daten der Teilnehmer.
 - RESULT = 1: von einem Teilnehmer wurden Daten empfangen.
Der Ausgang LEN zeigt an, wie viele Datenbytes empfangen wurden.
Diese neuen Daten in DST sofort speichern / auswerten!
Der Empfang einer weiteren Nachricht überschreibt die Daten auf der Speicheradresse DST.
 - RESULT = 0: innerhalb von 1,25 Sekunden hat kein Teilnehmer am Bus eine Antwort gesendet.
Der Baustein wird wieder inaktiv.
Erst jetzt darf ENABLE wieder auf FALSE gesetzt werden!
- ▶ Für das Empfangen von Daten von mehreren Teilnehmern in schneller Folge:
den Baustein im selben SPS-Zyklus mehrmals aufrufen und direkt auswerten!

Parameter der Eingänge

463

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
PRI0	BYTE	Nachrichten-Priorität der PDU (Parameter Data Unit) zulässig = 0...7
PG	BYTE	Data Page Wert der definierten PGN (Parameter Group Number) zulässig = 0...1 (normalerweise = 0)
PF	BYTE	PDU format byte Wert der definierten PGN (Parameter Group Number) PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Wert der definierten PGN (Parameter Group Number) GE (Group Extension) = 0...255
DST	DWORD	Startadresse im Zielspeicher  Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!



Info

PGN = [Page] + [PF] + [PS]

PDU = [PRI0] + [PGN] + [J1939-Adresse] + [Daten]

Parameter der Ausgänge

464

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)
SA	BYTE	J1939-Adresse des antwortenden Geräts
LEN	WORD	Anzahl der empfangenen Bytes

Mögliche Ergebnisse für RESULT:

Wert dez	hex	Beschreibung
0	00	FB ist inaktiv
1	01	FB-Ausführung wurde ohne Fehler beendet – Daten sind gültig
2	02	Funktionsbaustein ist aktiv (Aktion noch nicht beendet)
3	03	Fehler

J1939_x_RECEIVE

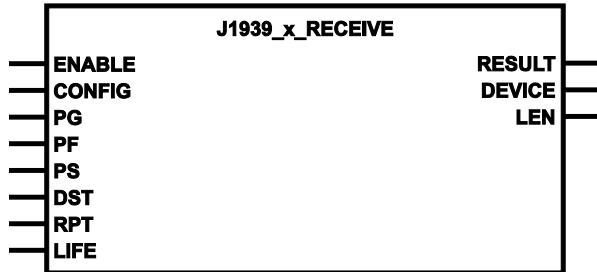
9393

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_J1939_x_Vxxyzz.LIB

Symbol in CODESYS:



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.

! Nach dem ersten Konfigurieren können diese Parameter im laufenden Anwendungsprogramm nicht mehr verändert werden: PG, PF, PS, RPT, LIFE, DST.

13790

ACHTUNG

Daten können unzulässig überschrieben werden!

- ▶ Ein Empfangs-Array mit einer Größe von 1 785 Bytes anlegen!
Dies ist die maximale Größe einer J1939-Nachricht.
- ▶ Die Anzahl empfangener Daten prüfen:
der Wert darf nicht größer sein als das bereitgestellte Empfangs-Array!

- ▶ Für die Zieladresse DST gilt:

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

- !** Nach dem ersten Setzen kann RPT nicht mehr verändert werden!

- ▶ 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.

! 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 ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
CONFIG	BOOL	TRUE (im 1. Zyklus): Datenobjekt konfigurieren FALSE: im weiteren Programmablauf
PG	BYTE	Data Page Wert der definierten PGN (Parameter Group Number) zulässig = 0...1 (normalerweise = 0)
PF	BYTE	PDU format byte Wert der definierten PGN (Parameter Group Number) PDU1 (specific) = 0...239 PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Wert der definierten PGN (Parameter Group Number) Wenn PF = PDU1 ⇒ PS = DA (Destination Address) (DA = J1939-Adresse des externen Geräts) Wenn PF = PDU2 ⇒ PS = GE (Group Extension)
DST	DWORD	Startadresse im Zielspeicher ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
RPT	TIME	Überwachungszeit Innerhalb dieses angegebenen Zeitfensters müssen die Telegramme zyklisch empfangen werden. > Andernfalls erfolgt eine Fehlermeldung. RPT = T#0s ⇒ keine Überwachung ! Nach dem ersten Setzen kann RPT nicht mehr verändert werden!
LIFE	BYTE	tolerierte Anzahl der nicht empfangenen J1939-Nachrichten

Parameter der Ausgänge

458

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)
DEVICE	BYTE	J1939-Adresse des Absenders
LEN	WORD	Anzahl der empfangenen Bytes

Mögliche Ergebnisse für RESULT:

Wert dez	hex	Beschreibung
0	00	FB ist inaktiv
1	01	FB-Ausführung wurde ohne Fehler beendet – Daten sind gültig
3	03	Fehler, keine Daten während der Überwachungszeit empfangen

J1939_x_RESPONSE

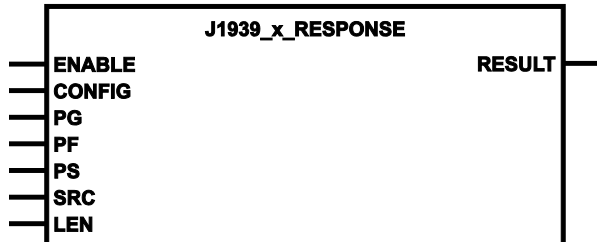
9399

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_J1939_x_Vxxyzzz.LIB

Symbol in CODESYS:



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.

- ▶ Für die Quelladresse SRC gilt:
 - ❗ Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
- ▶ Zusätzlich die Anzahl der zu übertragenden Datenbytes übergeben.

Parameter der Eingänge

451

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
CONFIG	BOOL	TRUE (im 1. Zyklus): Datenobjekt konfigurieren FALSE: im weiteren Programmablauf
PG	BYTE	Data Page Wert der definierten PGN (Parameter Group Number) zulässig = 0...1 (normalerweise = 0)
PF	BYTE	PDU format byte Wert der definierten PGN (Parameter Group Number) PDU1 (specific) = 0...239 PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Wert der definierten PGN (Parameter Group Number) Wenn PF = PDU1 ⇒ PS = DA (Destination Address) (DA = J1939-Adresse des externen Geräts) Wenn PF = PDU2 ⇒ PS = GE (Group Extension)
SRC	DWORD	Startadresse im Quellspeicher ❗ Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
LEN	WORD	Anzahl (≥ 1) der zu übertragenden Daten-Bytes

Parameter der Ausgänge

13993

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für RESULT:

Wert		Beschreibung
dez	hex	
0	00	FB ist inaktiv
1	01	Datenübertragung wurde ohne Fehler beendet
2	02	Funktionsbaustein ist aktiv (Aktion noch nicht beendet)
3	03	Fehler, Daten können nicht übertragen werden

J1939_x_SPECIFIC_REQUEST

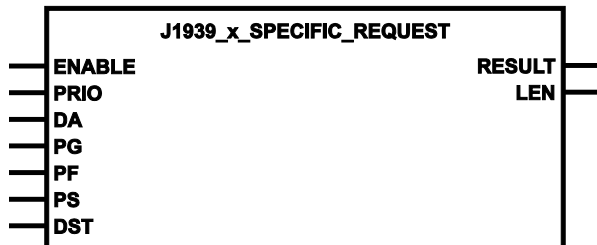
8884

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_J1939_x_Vxxyzz.LIB

Symbol in CODESYS:



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.



Info

PGN = [Page] + [PF] + [PS]

PDU = [PRIO] + [PGN] + [J1939-Adresse] + [Daten]

13790

ACHTUNG

Daten können unzulässig überschrieben werden!

- ▶ Ein Empfangs-Array mit einer Größe von 1 785 Bytes anlegen!
Dies ist die maximale Größe einer J1939-Nachricht.
- ▶ Die Anzahl empfangener Daten prüfen:
der Wert darf nicht größer sein als das bereitgestellte Empfangs-Array!

- ▶ Für die Zieladresse gilt:
 - ❗ 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.
- > Wird innerhalb von 1,25 Sekunden vom angeforderten Teilnehmer keine Antwort gesendet, meldet der FB einen Fehler (⇒ RESULT = 3).

Parameter der Eingänge

445

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
PRI0	BYTE	Nachrichten-Prioritätin der PDU (Parameter Data Unit) zulässig = 0...7
DA	BYTE	J1939-Adresse des angefragten Geräts
PG	BYTE	Data Page Wert der definierten PGN (Parameter Group Number) zulässig = 0...1 (normalerweise = 0)
PF	BYTE	PDU format byte Wert der definierten PGN (Parameter Group Number) PDU1 (specific) = 0...239 PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Wert der definierten PGN (Parameter Group Number) Wenn PF = PDU1 ⇒ PS = DA (Destination Address) (DA = J1939-Adresse des externen Geräts) Wenn PF = PDU2 ⇒ PS = GE (Group Extension)
DST	DWORD	Startadresse im Zielspeicher  Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!



Info

PGN = [Page] + [PF] + [PS]

PDU = [PRI0] + [PGN] + [J1939-Adresse] + [Daten]

Parameter der Ausgänge

446

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)
LEN	WORD	Anzahl der empfangenen Bytes

Mögliche Ergebnisse für RESULT:

Wert dez hex	Beschreibung
0 00	FB ist inaktiv
1 01	FB-Ausführung wurde ohne Fehler beendet – Daten sind gültig
2 02	Funktionsbaustein ist aktiv (Aktion noch nicht beendet)
3 03	Fehler

J1939_x_TRANSMIT

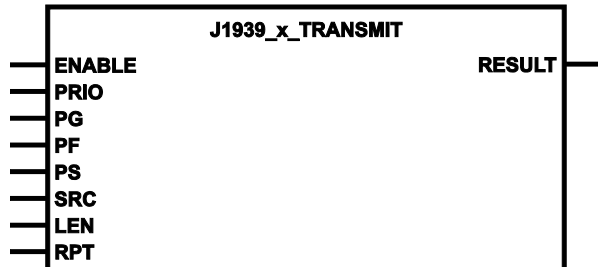
4322

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_J1939_x_Vxxyzz.LIB

Symbol in CODESYS:



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.

Info

PGN = [Page] + [PF] + [PS]

PDU = [PRIO] + [PGN] + [J1939-Adresse] + [Daten]

- ▶ Für die Quelladresse SRC gilt:
 -  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.
-  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 ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
PRI0	BYTE	Nachrichten-Priorität der PDU (Parameter Data Unit) zulässig = 0...7
PG	BYTE	Data Page Wert der definierten PGN (Parameter Group Number) zulässig = 0...1 (normalerweise = 0)
PF	BYTE	PDU format byte Wert der definierten PGN (Parameter Group Number) PDU1 (specific) = 0...239 PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Wert der definierten PGN (Parameter Group Number) Wenn PF = PDU1 ⇒ PS = DA (Destination Address) (DA = J1939-Adresse des externen Geräts) Wenn PF = PDU2 ⇒ PS = GE (Group Extension)
SRC	DWORD	Startadresse im Quellspeicher  Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
LEN	WORD	Anzahl der zu übertragenden Daten-Bytes zulässig = 1...1 785 = 0x0001...0x06F9
RPT	TIME	Wiederholzeit, innerhalb der die Daten-Telegramme zyklisch versendet werden sollen RPT = T#0s ⇒ nur einmalig versenden

Info

PGN = [Page] + [PF] + [PS]

PDU = [PRI0] + [PGN] + [J1939-Adresse] + [Daten]

Parameter der Ausgänge

440

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für RESULT:

Wert dez hex	Beschreibung
0 00	FB ist inaktiv
1 01	FB-Ausführung wurde ohne Fehler beendet – Daten sind gültig
2 02	Funktionsbaustein ist aktiv (Aktion noch nicht beendet)
3 03	Fehler, Daten können nicht übertragen werden

5.2.6 Bausteine: serielle Schnittstelle

Inhalt

SERIAL_PENDING	115
SERIAL_RX.....	116
SERIAL_SETUP	117
SERIAL_TX	118

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 Systemmerkerbit SERIAL_MODE=TRUE, dann kann die Schnittstelle frei genutzt werden. Der Programm-Download und das Debugging sind dann jedoch nur noch über die CAN-Schnittstelle möglich.

Mit den folgend aufgeführten Bausteinen kann die serielle Schnittstelle im Anwendungsprogramm genutzt werden.

SERIAL_PENDING

314

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

317

SERIAL_PENDING ermittelt die Anzahl der im seriellen Empfangspuffer gespeicherten Datenbytes. Im Gegensatz zu **SERIAL_RX** (→ Seite [116](#)) bleibt der Inhalt des Puffers nach Aufruf dieser Funktion unverändert.

Die SERIAL-Bausteine bilden die Grundlage für die Erstellung eines anwendungsspezifischen Protokolls für die serielle Schnittstelle.

Dazu das Systemmerkerbit SERIAL_MODE=TRUE setzen!

! 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 Systemmerkerbit SERIAL_MODE=TRUE, dann kann die Schnittstelle frei genutzt werden. Der Programm-Download und das Debugging sind dann jedoch nur noch über die CAN-Schnittstelle möglich.

Parameter der Ausgänge

319

Parameter	Datentyp	Beschreibung
NUMBER	WORD	Anzahl der empfangenen Datenbytes

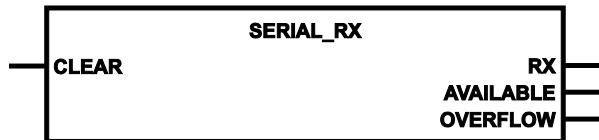
SERIAL_RX

308

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_Vxxyzz.LIB`

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 1 000 Datenbytes ein, läuft der Puffer über und es gehen Daten verloren. Dieses wird durch das Bit OVERFLOW angezeigt.

Wird eine 7-Bit-Datenübertragung genutzt, enthält das 8. Bit die Parität und muss gegebenenfalls vom Anwender ausgeblendet werden.

Die SERIAL-Bausteine bilden die Grundlage für die Erstellung eines anwendungsspezifischen Protokolls für die serielle Schnittstelle.

Dazu das Systemmerkerbit `SERIAL_MODE=TRUE` setzen!

! 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 Systemmerkerbit `SERIAL_MODE=TRUE`, dann kann die Schnittstelle frei genutzt werden. Der Programm-Download und das Debugging sind dann jedoch nur noch über die CAN-Schnittstelle möglich.

Parameter der Eingänge

312

Parameter	Datentyp	Beschreibung
CLEAR	BOOL	TRUE: Empfangspuffer löschen FALSE: Funktion wird nicht ausgeführt

Parameter der Ausgänge

313

Parameter	Datentyp	Beschreibung
RX	BYTE	empfangene Byte-Daten aus dem Empfangspuffer
AVAILABLE	WORD	Anzahl der verbleibenden Datenbytes 0 = keine gültigen Daten vorhanden
OVERFLOW	BOOL	TRUE: Überlauf des Datenpuffers ⇒ Datenverlust! FALSE: Datenpuffer ist ohne Datenverlust

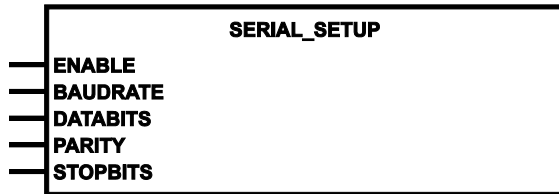
SERIAL_SETUP

302

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

305

SERIAL_SETUP initialisiert die serielle RS232-Schnittstelle.

Der FB muss nicht zwingend ausgeführt werden, um die serielle Schnittstelle verwenden zu können. Ohne FB-Aufruf gelten die folgend angegebenen Voreinstellungen.

Mit ENABLE=TRUE für einen Zyklus setzt der FB die serielle Schnittstelle auf die angegebenen Parameter. Die mit dem FB vorgenommenen Änderungen werden remanent gespeichert.

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 Systemmerkerbit SERIAL_MODE=TRUE, dann kann die Schnittstelle frei genutzt werden. Der Programm-Download und das Debugging sind dann jedoch nur noch über die CAN-Schnittstelle möglich.

5020

ACHTUNG

Der Treiberbaustein der seriellen Schnittstelle kann beschädigt werden!

Beim Trennen oder Verbinden 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 oder verbinden!

Parameter der Eingänge

306

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (nur 1 Zyklus lang): Schnittstelle initialisieren FALSE: im weiteren Programmablauf
BAUDRATE	WORD	Baudrate zulässige Werte → Datenblatt Voreinstellwert → Datenblatt
DATABITS	BYTE := 8	Anzahl der Daten-Bits zulässig = 7 oder 8
PARITY	BYTE := 0	Parität zulässig: 0=keine, 1=gerade, 2=ungerade
STOPBITS	BYTE := 1	Anzahl der Stopp-Bits zulässig = 1 oder 2

SERIAL_TX

296

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxyxyz.LIB

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 anwendungsspezifischen Protokolls für die serielle Schnittstelle.

Dazu das Systemmerkerbit SERIAL_MODE=TRUE setzen!

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 Systemmerkerbit SERIAL_MODE=TRUE, dann kann die Schnittstelle frei genutzt werden. Der Programm-Download und das Debugging sind dann jedoch nur noch über die CAN-Schnittstelle möglich.

Parameter der Eingänge

300

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
DATA	BYTE	zu übertragender Wert

5.2.7 Bausteine: SPS-Zyklus optimieren

Inhalt

Bausteine: Interrupts verarbeiten	119
---	-----

8609

Hier zeigen wir Ihnen Funktionen zum Optimieren des SPS-Zyklus.

Bausteine: Interrupts verarbeiten

Inhalt

SET_INTERRUPT_I	120
SET_INTERRUPT_XMS	123

1599

Die SPS arbeitet das gespeicherte Anwendungsprogramm 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.

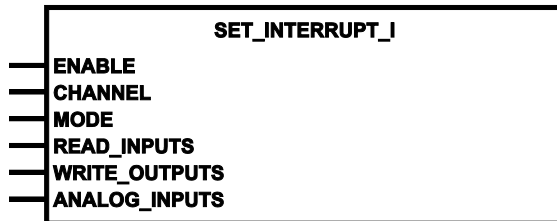
SET_INTERRUPT_I

2381

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

281
11573

SET_INTERRUPT_I organisiert das Ausführen eines Programnteils 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 Anwendungsprogramm in einem einzigen unveränderlichen Block abläuft.

Ein möglicher Lösungsansatz wäre, die Zykluszeit kurz zu halten. Dieser Weg führt oft dazu, die Anwendung auf mehrere Steuerungszyklen zu verteilen. Die Programmierung wird dadurch jedoch unübersichtlich und schwierig.

Eine andere Möglichkeit besteht darin, einen bestimmten Programnteil nur auf Anforderung durch einen Eingangsimpuls unabhängig vom Steuerungszyklus aufzurufen:

Der zeitkritische Teil des Anwendungsprogramms 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 Programnteil immer dann ausgeführt wird, wenn eine Flanke am Eingang CHANNEL erkannt wird. Werden Ein- und Ausgänge in diesem Programnteil 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 Programnteils 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 Programnteil nicht zusätzlich zyklisch aufgerufen wird, sollte er (mit Ausnahme des Initialisierungsaufwurfes) 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.

Die Laufzeit des Hauptzyklus plus die Summe der Laufzeiten aller per Interrupt aufgerufenen Programnteile muss stets innerhalb der max. zulässigen Zykluszeit bleiben!

Für die Datenkonsistenz zwischen Hauptprogramm und den im Interrupt laufenden Programnteilen ist der Anwender zuständig!

Interrupt-Prioritäten:

- Alle per Interrupt aufgerufenen Programmteile haben die gleiche Priorität der Ausführung. Mehrere gleichzeitige Interrupts werden sequenziell in Reihenfolge ihres Auftretens abgearbeitet.
- Wird eine weitere Flanke am gleichen Eingang während der Ausführung des per Interrupt aufgerufenen Programmteils erkannt, wird dieser zur Bearbeitung eingetragen und das Programm nach Beendigung direkt wieder aufgerufen. Optional können durch Setzen des Glitch-Filters störende Mehrfachimpulse ausgefiltert werden.
- Das im Interrupt laufende Programm kann durch höherpriorisierte Interrupts (z.B. CAN) unterbrochen werden.
- Belegen mehrere Interrupts den gleichen Kanal, erhält der zuletzt initialisierte FB (oder das PRG) den Kanal. Der zuvor definierte FB (oder das PRG) wird dann nicht mehr aufgerufen und liefert keine Daten mehr.

! 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 (Controller: CR0n3n, CR7n3n)

%IX0.12...%IX0.15, %IX1.4...%IX1.8 (übrige ClassicController, ExtendedController, SafetyController)

%IX0.0, %IX0.8 (SmartController: CR250n)

IN08...IN11 (CabinetController: CR030n)

IN0...IN3 (Platinensteuerung: CS0015)

Eingänge, analog:

%IX0.0...%IX0.7 (Controller: CR0n3n, CR7n3n)

alle Kanäle (Auswahl bitcodiert) (alle übrigen Controller)

Ausgänge, digital:

%QX0.0...%QX0.7 (ClassicController, ExtendedController, SafetyController)

%QX0.0, %QX0.8 (SmartController: CR250n)

OUT00...OUT03 CabinetController: CR030n()

OUT0...OUT7 (Platinensteuerung: CS0015)

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

20089

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (nur 1 Zyklus lang): Initialisierung des Bausteins FALSE: Baustein wird nicht ausgeführt
CHANNEL	BYTE	Nummer des Interrupt-Eingangs (0...3) 0...3 für die Eingänge IN08...IN11
MODE	BYTE	Art der Flanke am Eingang CHANNEL, die den Interrupt auslöst 1 = steigende Flanke (Standard-Wert) 2 = fallende Flanke 3 = steigende und fallende Flanke > 3 = Standard-Wert
READ_INPUTS	BOOL	TRUE: die Eingänge 8...11 vor Aufruf des Programms lesen und in die Eingangsmerker IN08...IN11 schreiben FALSE: nur den unter CHANNEL angegebenen Kanal lesen und in den dazugehörigen Eingangsmerker INnn schreiben
WRITE_OUTPUTS	BOOL	TRUE: die aktuellen Werte der Ausgangsmerker OUT00...03 nach Programmablauf auf die Ausgänge schreiben FALSE: keine Ausgänge schreiben
ANALOG_INPUTS	BOOL	TRUE: die Eingänge 16...23 lesen und die ungefilterten, unkalibrierten Analogwerte in die Merker A_IN16...23 schreiben FALSE: die Merker A_IN16...23 nicht schreiben

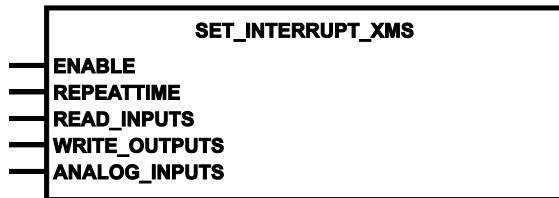
SET_INTERRUPT_XMS

272

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



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 Anwendungsprogramm in einem einzigen unveränderlichen Block abläuft.

Ein möglicher Lösungsansatz wäre, die Zykluszeit kurz zu halten. Dieser Weg führt oft dazu, die Anwendung 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 des Anwendungsprogramms 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 Initialisierungsaufwurfes) im Zyklus übersprungen werden.

Es können mehrere Timer-Interrupt-Bausteine 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.

Für die Datenkonsistenz zwischen Hauptprogramm und den im Interrupt laufenden Programmteilen ist der Anwender zuständig!

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 (Controller: CR0n3n, CR7n3n)

%IX0.12...%IX0.15, %IX1.4...%IX1.8 (übrige ClassicController, ExtendedController, SafetyController)

%IX0.0, %IX0.8 (SmartController: CR250n)

IN08...IN11 (CabinetController: CR030n)

IN0...IN3 (Platinensteuerung: CS0015)

Eingänge, analog:

%IX0.0...%IX0.7 (Controller: CR0n3n, CR7n3n)

alle Kanäle (Auswahl bitcodiert) (alle übrigen Controller)

Ausgänge, digital:

%QX0.0...%QX0.7 (ClassicController, ExtendedController, SafetyController)

%QX0.0, %QX0.8 (SmartController: CR250n)

OUT00...OUT03 CabinetController: CR030n()

OUT0...OUT7 (Platinensteuerung: CS0015)

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

20095

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (nur 1 Zyklus lang): Initialisierung des Bausteins FALSE: Baustein wird nicht ausgeführt
REPEATTIME	TIME	Zeitdauer in [ms] zwischen Ende des Programms und Neustart Die Zeitdauer zwischen zwei Aufrufen ermittelt sich damit als Summe aus REPEATTIME und Laufzeit des per Interrupt aufgerufenen Programms.
READ_INPUTS	BOOL	TRUE: die Eingänge 8...11 vor Aufruf des Programms lesen und in die Eingangsmerker IN08...IN11 schreiben FALSE: nur den unter CHANNEL angegebenen Kanal lesen und in den dazugehörigen Eingangsmerker INnn schreiben
WRITE_OUTPUTS	BOOL	TRUE: die aktuellen Werte der Ausgangsmerker OUT00...03 nach Programmablauf auf die Ausgänge schreiben FALSE: keine Ausgänge schreiben
ANALOG_INPUTS	BOOL	TRUE: die Eingänge 16...23 lesen und die ungefilterten, unkalibrierten Analogwerte in die Merker A_IN16...23 schreiben FALSE: die Merker A_IN16...23 nicht schreiben

5.2.8 Bausteine: Eingangswerte verarbeiten

Inhalt

INPUT_ANALOG.....	127
INPUT_CURRENT	128
INPUT_VOLTAGE.....	129

1602
1302

Hier zeigen wir Ihnen **ifm**-Funktionsbausteine zum Lesen und Verarbeiten der analogen oder binären Signale am Geräte-Eingang.

! HINWEIS

Die in der Steuerungskonfiguration von CODESYS erscheinenden analogen Rohwerte kommen direkt aus dem ADC. Sie sind noch nicht korrigiert!

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

Erst durch die **ifm**-FBs findet eine Fehlerkorrektur und Normierung statt. Die FBs liefern den korrigierten Wert.

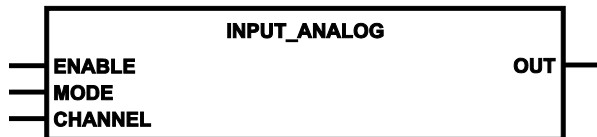
INPUT_ANALOG

519

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxyxyz.LIB

Symbol in CODESYS:



Beschreibung

522

INPUT_ANALOG ermöglicht Strom- und Spannungsmessung an den Analogkanälen.

Der FB liefert den aktuellen Analogwert am gewählten Analogkanal. Die Messung und der Ausgangswert resultiert aus der über MODE angegebenen Betriebsart:

MODE	Eingang Betriebsart	Ausgang OUT	Einheit
IN_DIGITAL_H	Digitaleingang	0 / 1	---
IN_CURRENT	Stromeingang	0...20 000	µA
IN_VOLTAGE10	Spannungseingang	0...10 000	mV
IN_VOLTAGE30	Spannungseingang	0...30 000	mV
IN_RATIO	Spannungseingang ratiometrisch	0...1 000	‰

Zur Parametrierung der Betriebsart sollten die angegebenen globalen Systemvariablen genutzt werden. Die Analogwerte werden normiert ausgegeben.

! Wird dieser FB genutzt, muss unbedingt die Systemvariable RELAIS *) gesetzt werden, sonst fehlen die internen Referenzspannungen für die Strommessung.

*) Relais nur in folgenden Geräten vorhanden: CR0020, CRnn32, CRnn33, CR0200, CR0505, CR7nnn

Parameter der Eingänge

20323

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
MODE	BYTE	IN_DIGITAL_H Digitaleingang IN_CURRENT Stromeingang 0...20 000 µA IN_VOLTAGE10 Spannungseingang 0...10 000 mV IN_VOLTAGE30 Spannungseingang 0...30 000 mV IN_RATIO ratiometrischer Analogeingang
INPUT_CHANNEL	BYTE	Nummer des Eingangskanals zulässig = 16...23

Parameter der Ausgänge

524

Parameter	Datentyp	Beschreibung
OUT	WORD	Ausgangswert entsprechend MODE bei ungültiger Einstellung: OUT = "0"

INPUT_CURRENT

513

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

516

INPUT_CURRENT liefert den aktuellen Eingangsstrom in [µA] an den analogen Stromeingängen.

 INPUT_CURRENT ist eine Kompatibilitätsfunktion für ältere Programme. In neuen Programmen sollte der leistungsfähigere FB **INPUT_ANALOG** (→ Seite [127](#)) eingesetzt werden.

Parameter der Eingänge

20325

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
INPUT_CHANNEL	BYTE	Nummer des Eingangskanals zulässig = 16...23

Parameter der Ausgänge

518

Parameter	Datentyp	Beschreibung
ACTUAL_CURRENT	WORD	Eingangsstrom in [µA]

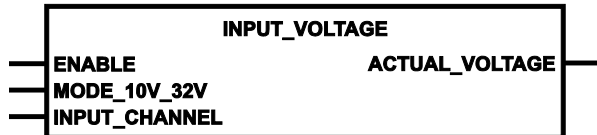
INPUT_VOLTAGE

507

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxyxyz.LIB

Symbol in CODESYS:



Beschreibung

510

INPUT_VOLTAGE liefert die aktuelle Eingangsspannung in mV an dem gewählten Analogkanal.

Die Messung bezieht sich auf den über MODE_10V_32V angegebenen Spannungsbereich (10.000 mV oder 32.000 mV).

 INPUT_VOLTAGE ist eine Kompatibilitätsfunktion für ältere Programme. In neuen Programmen sollte der leistungsfähigere FB **INPUT_ANALOG** (→ Seite [127](#)) eingesetzt werden.

Parameter der Eingänge

20327

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
MODE_10V_32V	BOOL	TRUE: Spannungsbereich 0...32 V FALSE: Spannungsbereich 0...10 V
INPUT_CHANNEL	BYTE	Nummer des Eingangskanals zulässig = 16...23

Parameter der Ausgänge

512

Parameter	Datentyp	Beschreibung
ACTUAL_VOLTAGE	WORD	Eingangsspannung in [mV]

5.2.9 Bausteine: analoge Werte anpassen

Inhalt

NORM.....	131
-----------	-----

1603

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

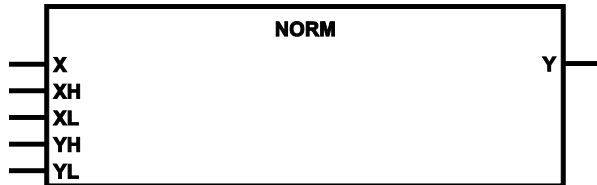
NORM

401

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_Vxxyzz.LIB`

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 Eingangswert für X muss sich im definierten Bereich zwischen XL und XH befinden! Der FB prüft NICHT den Wert X auf Plausibilität.
- > 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	Eingangswert
XH	WORD	obere Grenze des Eingangswertebereichs [Inkrement]
XL	WORD	untere Grenze des Eingangswertebereichs [Inkrement]
YH	WORD	obere Grenze des Ausgangswertebereichs
YL	WORD	untere Grenze des Ausgangswertebereichs

Parameter der Ausgänge

406

Parameter	Datentyp	Beschreibung
Y	WORD	Ausgangswert

Beispiel: NORM (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 Funktionsbaustein das Eingangssignal z.B. wie folgt um:

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

Beispiel: NORM (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 Funktionsbaustein das Eingangssignal z.B. wie folgt um:

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

5.2.10 Bausteine: Zählerfunktionen zur Frequenz- und Periodendauermessung

Inhalt	
FAST_COUNT	134
FREQUENCY	135
INC_ENCODER	136
PERIOD	138
PERIOD_RATIO	140
PHASE	142

18818

Die Controller unterstützen bis zu 4 schnelle Eingänge, die Eingangsfrequenzen bis zu 30 kHz verarbeiten können. Neben der reinen Frequenzmessung können die Eingänge FRQ auch zur Auswertung von inkrementellen Drehgebern (Zählerfunktion) eingesetzt werden.

Bedingt durch die unterschiedlichen Messmethoden können Fehler bei der Frequenzermittlung auftreten.

Zur einfachen Auswertung stehen folgende Bausteine zur Verfügung:

Baustein	zulässige Werte	Erklärung
FAST_COUNT	0...50 000 Hz	Schnelle Impulse zählen
FREQUENCY	0,1...30 000 Hz	Frequenz am angegebenen Kanal messen. Messfehler verringert sich bei hohen Frequenzen
INC_ENCODER	0...30 000 Hz	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
PERIOD	0...5 000 Hz	Frequenz und Periodendauer (Zykluszeit) am angegebenen Kanal messen
PERIOD_RATIO	0...5 000 Hz	Frequenz und Periodendauer (Zykluszeit) sowie Puls-Pause-Verhältnis [%] am angegebenen Kanal messen
PHASE	0...5 000 Hz	Liest ein Kanalpaar ein und vergleicht die Phasenlage der Signale

! Wichtig bei Einsatz der schnellen Eingänge als "normale" Digitaleingänge:

- Die erhöhte Empfindlichkeit gegen Störimpulse beachten (z.B. Kontaktprellen bei mechanischen Kontakten).
- Der Standard-Digitaleingang kann Signale bis 50 Hz auswerten.

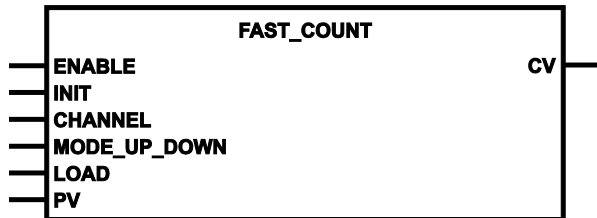
FAST_COUNT

20430

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxyxyz.LIB

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.

! Bei den *ecomatmobile*-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

17812

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Zähler angehalten
INIT	BOOL	FALSE ⇒ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...3) 0...3 für die Eingänge IN08...IN11
MODE_UP_DOWN	BOOL	TRUE: Zähler zählt abwärts FALSE: Zähler zählt aufwärts
LOAD	BOOL	TRUE: Startwert PV wird in CV geladen FALSE: Funktion wird nicht ausgeführt
PV	DWORD	Startwert (Preset value) für den Zähler

Parameter der Ausgänge

572

Parameter	Datentyp	Beschreibung
CV	DWORD	aktueller Zählerwert Verhalten beim Überlauf: • zählt der Zähler abwärts, bleibt er bei 0 stehen • zählt der Zähler aufwärts, gibt es einen Überlauf.

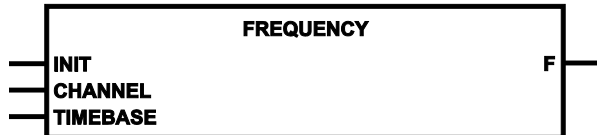
FREQUENCY

20604

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxyxyz.LIB

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.

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

Parameter der Eingänge

17814

Parameter	Datentyp	Beschreibung
INIT	BOOL	FALSE ⇒ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...3) 0...3 für die Eingänge IN08...IN11
TIMEBASE	TIME	Zeitbasis zur Frequenzmessung (max. 57 s)

8406

! 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

542

Parameter	Datentyp	Beschreibung
F	REAL	Frequenz des Eingangssignals in [Hz]

INC_ENCODER

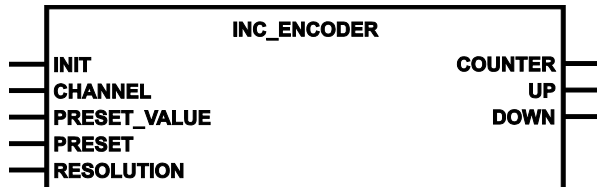
20432

= Incremental Encoder

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

4330

INC_ENCODER bietet eine Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern. Immer zwei Frequenzeingänge bilden das Eingangspaar, das über den FB ausgewertet wird.

Grenzfrequenz = 30 kHz

max. anschließbar: 4 Drehgeber (ExtendedController: max. 8 Drehgeber)

Voreinstellwert setzen:

1. Wert in PRESET_VALUE eintragen
2. PRESET für einen Zyklus auf TRUE setzen
3. PRESET wieder auf FALSE setzen

Der FB zählt die Impulse an den Eingängen, solange INIT=FALSE und PRESET=FALSE sind.

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.

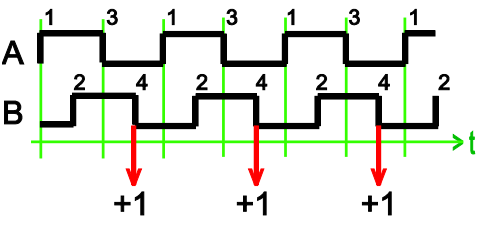
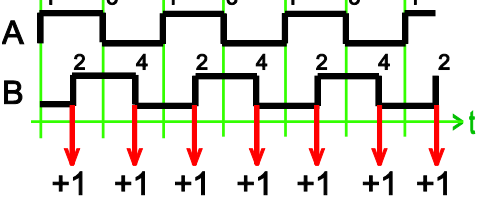
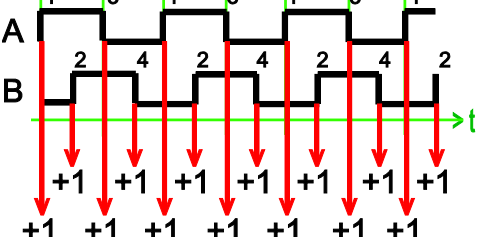
! Am selben Eingang diesen FB **nicht** gemeinsam mit einem der folgenden FBs nutzen!

- **FAST_COUNT** (→ Seite [134](#))
- **FREQUENCY** (→ Seite [135](#))
- **PERIOD** (→ Seite [138](#))
- **PERIOD_RATIO** (→ Seite [140](#))
- **PHASE** (→ Seite [142](#))

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

17822

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (nur 1 Zyklus lang): Baustein wird initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des Eingangskanal-Paares (0/1): 0 = Kanalpaar 0 = Eingänge IN08 + IN09 1 = Kanalpaar 1 = Eingänge IN10 + IN11
PRESET_VALUE	DINT	Zähler-Startwert
PRESET	BOOL	FALSE ⇒ TRUE (Flanke): PRESET_VALUE wird nach COUNTER geladen TRUE: Zähler ignoriert die Eingangsimpulse FALSE: Zähler zählt die Eingangsimpulse
RESOLUTION	BYTE	Auswertung der Drehgeber-Auflösung: 01 = zählt bei jeder vierten Flanke (= Auflösung des Drehgebers) 02 = zählt bei jeder zweiten Flanke 04 = zählt bei jeder steigenden und fallenden Flanke Alle anderen Werte zählen wie "01".

Parameter der Ausgänge

530

Parameter	Datentyp	Beschreibung
COUNTER	DINT	aktueller Zählerstand
UP	BOOL	TRUE: Zähler zählte im letzten Zyklus aufwärts FALSE: Zähler zählte im letzten Zyklus nicht aufwärts
DOWN	BOOL	TRUE: Zähler zählte im letzten Zyklus abwärts FALSE: Zähler zählte im letzten Zyklus nicht abwärts

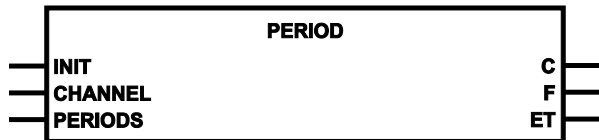
PERIOD

20606

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_Vxxyzz.LIB`

Symbol in CODESYS:



Beschreibung

373

PERIOD misst die Frequenz und die Periodendauer (Zykluszeit) in $[\mu\text{s}]$ 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 $[\mu\text{s}]$ 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 $< 0,5 \text{ Hz}$ werden nicht mehr eindeutig angezeigt!

Parameter der Eingänge

17818

Parameter	Datentyp	Beschreibung
INIT	BOOL	FALSE ⇒ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...3) 0...3 für die Eingänge IN08...IN11
PERIODS	BYTE	Anzahl der zu vergleichenden Perioden

8406

! 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] zulässig = 200...10 000 000 = 0xC8...0x989680 (ca. 46,3 Minuten)
F	REAL	Frequenz des Eingangssignals in [Hz]
ET	TIME	Verstrichene Zeit seit der letzten positiven Flanke am Eingang (nutzbar bei sehr langsamen Signalen)

PERIOD_RATIO

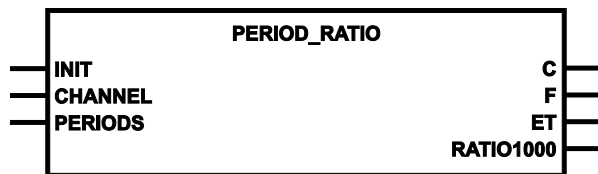
364

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_Vxxyzz.LIB`

 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-/Periodenverhältnis 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-/Periodenverhältnis 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 einem Puls/Periodenverhältnis von 100 % (Eingangssignal dauerhaft auf Versorgungsspannung) den Wert 0.

Frequenzen < 0,05 Hz werden nicht mehr eindeutig angezeigt!

Parameter der Eingänge

17820

Parameter	Datentyp	Beschreibung
INIT	BOOL	FALSE ⇒ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...3) 0...3 für die Eingänge IN08...IN11
PERIODS	BYTE	Anzahl der zu vergleichenden Perioden

8406

! 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] zulässig = 200...10 000 000 = 0xC8...0x989680 (ca. 46,3 Minuten)
F	REAL	Frequenz des Eingangssignals in [Hz]
ET	TIME	Verstrichene Zeit seit dem letzten Zustandswechsel am Eingang (nutzbar bei sehr langsamen Signalen)
RATIO1000	WORD	bei Messung der Periodendauer: Puls-/Periode-Verhältnis in [%] Voraussetzungen: • Impulsdauer ≥ 100 µs • Frequenz < 5 kHz bei anderen Messungen: RATIO1000 = 0

PHASE

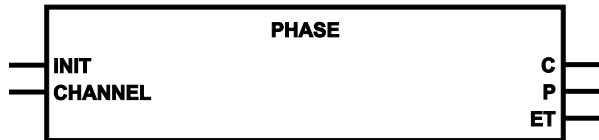
358

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_Vxxyzz.LIB`

 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.

 Bei Frequenzen kleiner 15 Hz wird eine Periodendauer bzw. Phasenverschiebung von 0 angezeigt.

Parameter der Eingänge

528

Parameter	Datentyp	Beschreibung
INIT	BOOL	FALSE ⇒ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des Eingangskanal-Paares (0/1): 0 = Kanalpaar 0 = Eingänge IN08 + IN09 1 = Kanalpaar 1 = Eingänge IN10 + IN11

8406

 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 des Signals am ersten Eingang des Kanalpaares in [µs]
P	INT	Winkel der Phasenverschiebung gültige Messung = 1...358 °
ET	TIME	Verstrichene Zeit seit der letzten positiven Flanke am zweiten Impulseingang des Kanalpaares

5.2.11 Bausteine: PWM-Funktionen

Inhalt	
PWM.....	144
PWM1000.....	148

13758

Hier finden Sie **ifm**-Bausteine, um die Ausgänge mit Pulsweitenmodulation (PWM) betreiben zu können.

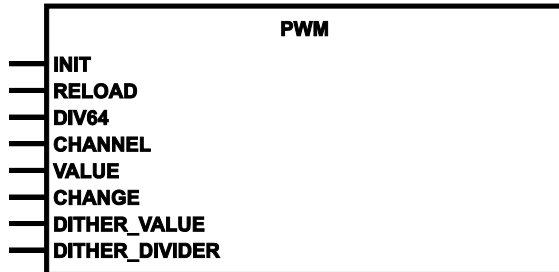
PWM

20457

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_Vxxyzz.LIB`

Symbol in CODESYS:



Beschreibung

13768

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 Anwendungsprogramms 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 0...7 gleich sein.

Bei diesen Kanälen dürfen PWM und *PWM1000* (→ Seite [148](#)) nicht gemischt werden.

Die PWM-Frequenz (und damit der RELOAD-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 z.B. mit ifm-Gerät EC2049 (Vorschaltgerät zur Strommessung).

PWM_DITHER wird einmalig für jeden Kanal in der Initialisierung des Anwendungsprogramms 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.

! Die Parameter DITHER_FREQUENCY und DITHER_VALUE können für jeden Kanal individuell eingestellt werden.

Parameter der Eingänge

20612

Parameter	Datentyp	Beschreibung
INIT	BOOL	FALSE ⇒ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
RELOAD	WORD	Wert zur Festlegung der PWM-Frequenz (→ Kapitel <i>Berechnung des RELOAD-Wertes</i> (→ Seite 146))
DIV64	BOOL	CPU-Takt / 64
CHANNEL	BYTE	Nummer des PWM-Ausgangskanals (0...7) 0...7 für die Ausgänge OUT00...OUT07
VALUE	WORD	aktueller PWM-Wert zulässig = 0...RELOAD 0 = Einschaltdauer 100 % RELOAD = Einschaltdauer 0 %
CHANGE	BOOL	TRUE: Übernahme neuer Wert von ... • VALUE: nach der aktuellen PWM-Periode • DITHER_VALUE: nach der aktuellen Dither-Periode FALSE: geänderter PWM-Wert hat keinen Einfluss auf den Ausgang
DITHER_VALUE	WORD	Spitze-Spitze-Wert des Dithers in [%] zulässig = 0...1 000 = 0x0000...0x03E8
DITHER_DIVIDER	WORD	Dither-Frequenz = PWM-Frequenz / DIVIDER * 2

PWM-Frequenz

1529

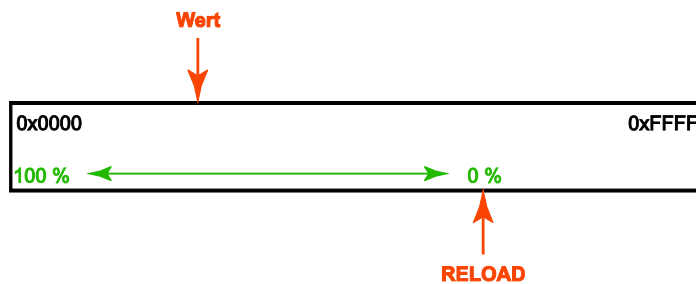
Abhängig vom Ventiltyp wird eine entsprechende PWM-Frequenz benötigt. Die PWM-Frequenz wird bei der PWM-Funktion über den Reload-Wert (Funktion PWM) oder direkt als Zahlenwert in Hz (Funktion PWM1000) übergeben. Je nach R360-Controller unterscheiden sich die PWM-Ausgänge in ihrer Arbeits-, aber nicht 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 0xFFFF rückwärts bzw. von 0x0000 aufwärts. Bei Erreichen eines übergebenen Vergleichswertes (VALUE) wird der Ausgang gesetzt. Mit Überlauf des Zählers (Zählerstandwechsel von 0x0000 nach 0xFFFF oder von 0xFFFF nach 0x0000) wird der Ausgang wieder zurückgesetzt und der Vorgang neu gestartet.

Soll dieser interne Zähler nicht zwischen 0x0000 und 0xFFFF 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.

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: CR250n - Platinensteuerung: CS0015 - PDM360smart: CR1071
DIV64 = 0	$\text{RELOAD} = 20 \text{ MHz} / f_{\text{PWM}}$	$\text{RELOAD} = 10 \text{ MHz} / f_{\text{PWM}}$
DIV64 = 1	$\text{RELOAD} = 312,5 \text{ kHz} / f_{\text{PWM}}$	$\text{RELOAD} = 156,25 \text{ kHz} / f_{\text{PWM}}$

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 0xFFFF wird.

Berechnungsbeispiele RELOAD-Wert

1532

- CabinetController: CR0303 - ClassicController: CR0020, CR0505 - ExtendedController: CR0200 - SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506	- CabinetController: CR0301, CR0302 - SmartController: CR250n - Platinensteuerung: CS0015 - PDM360smart: CR1071
Die PWM-Frequenz soll 400 Hz betragen. $\frac{20 \text{ MHz}}{400 \text{ Hz}} = 50\,000 = 0xC350 = \text{RELOAD}$	Die PWM-Frequenz soll 200 Hz betragen. $\frac{10 \text{ MHz}}{200 \text{ Hz}} = 50\,000 = 0xC350 = \text{RELOAD}$
Der zulässige Bereich des PWM-Wertes ist damit der Bereich von 0x0000...0xC350. Der Vergleichswert, bei dem der Ausgang durchschaltet, muss dann zwischen 0x0000 und 0xC350 liegen.	

Daraus ergeben sich folgende Puls-Pausen-Verhältnisse:

Puls-Pausen-Verhältnis	Einschaltdauer	Wert für Puls-Pausen-Verhältnis
Minimal	0 %	50 000 = 0xC350
Maximal	100 %	0 = 0x0000

Zwischen minimaler und maximaler Ansteuerung sind 50 000 Zwischenwerte (PWM-Werte) möglich.

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, kann z.B. durch Nutzung von **PT1** (→ Seite [171](#)) 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.

964

HINWEIS

Beim Installieren der *ecomatmobile*-DVD "Software, tools and documentation" wurden auch Projekte mit Beispielen auf Ihrem Computer im Programmverzeichnis abgelegt:

...\ifm electronic\CoDeSys V...\Projects\DEMO_PLC_DVD_V... (für Controller) oder

...\ifm electronic\CoDeSys V...\Projects\DEMO_PDM_DVD_V... (für PDMs)

Dort finden Sie auch Projekte mit Beispielen zu diesem Thema. Es wird dringend empfohlen, dem gezeigten Schema zu folgen.

 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.

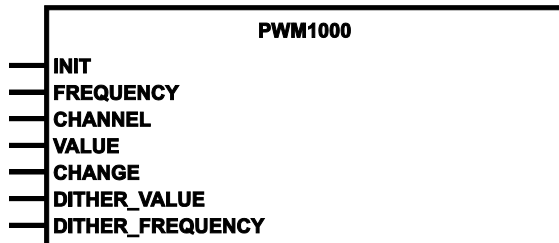
PWM1000

20465

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_Vxxyzz.LIB`

Symbol in CODESYS:



Beschreibung

7304

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 Anwendungsprogramms 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 0...7 gleich sein.

Bei diesen Kanälen dürfen **PWM** (→ Seite [144](#)) 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 z.B. mit ifm-Gerät EC2049 (Vorschaltgerät zur Strommessung).

DITHER wird einmalig für jeden Kanal in der Initialisierung des Anwendungsprogramms 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.

! Die Parameter DITHER_FREQUENCY und DITHER_VALUE können für jeden Kanal individuell eingestellt werden.

Parameter der Eingänge

17879

Parameter	Datentyp	Beschreibung
INIT	BOOL	FALSE ⇒ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
FREQUENCY	WORD	PWM-Frequenz in [Hz] zulässig = 20...250 = 0x0014...0x00FA
CHANNEL	BYTE	Nummer des PWM-Ausgangskanals (0...7) 0...7 für die Ausgänge OUT00...OUT07
VALUE	WORD	PWM-Wert (Puls-Periode-Verhältnis) in [%] zulässig = 0...1 000 = 0x0000...0x03E8 Werte > 1 000 gelten als = 1 000
CHANGE	BOOL	TRUE: Übernahme neuer Wert von ... • FREQUENCY: nach der aktuellen PWM-Periode • VALUE: nach der aktuellen PWM-Periode • DITHER_VALUE: nach der aktuellen Dither-Periode • DITHER_FREQUENCY: nach der aktuellen Dither-Periode FALSE: geänderter PWM-Wert hat keinen Einfluss auf den Ausgang
DITHER_VALUE	WORD	Spitze-Spitze-Wert des Dithers in [%] zulässig = 0...1 000 = 0x0000...0x03E8
DITHER_FREQUENCY	WORD	Dither-Frequenz in [Hz] Wertebereich = 0...FREQUENCY / 2 FREQUENCY / DITHER_FREQUENCY muss geradzahlig sein! Alle anderen Werte erhöht der FB auf den nächst passenden Wert.

5.2.12 Bausteine: Hydraulikregelung

Inhalt	
JOYSTICK_0.....	151
JOYSTICK_1.....	154
JOYSTICK_2.....	158
NORM_HYDRAULIC.....	161

9986

Die Bibliothek `ifm_HYDRAULIC_CR0303_Vxxyzz.Lib` enthält folgende Bausteine:

JOYSTICK_0 (→ Seite 151)	skaliert Signale [INT] aus einem Joystick auf fest definierte Kennlinien, normiert auf 0...1000
JOYSTICK_1 (→ Seite 154)	skaliert Signale [INT] aus einem Joystick auf parametrierbare Kennlinien, normiert auf 0...1000
JOYSTICK_2 (→ Seite 158)	skaliert Signale [INT] aus einem Joystick auf einen parametrierbaren Kennlinien-Verlauf, die Normierung ist frei bestimmbar
NORM_HYDRAULIC (→ Seite 161)	normiert einen Wert [DINT] innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen

Aus der Bibliothek `UTIL.Lib` (im CODESYS-Paket) werden folgende Bausteine benötigt:

- RAMP_INT
- CHARCURVE

Diese Bausteine werden von den FBs der Hydraulik-Bibliothek automatisch aufgerufen und parametriert.

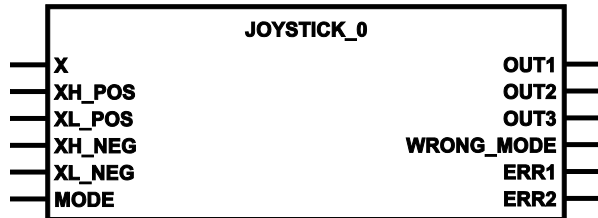
JOYSTICK_0

13225

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_hydraulic_CR0303_Vxxyzz.Lib

Symbol in CODESYS:



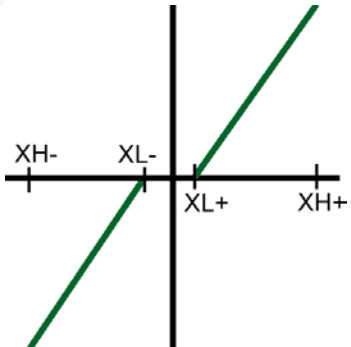
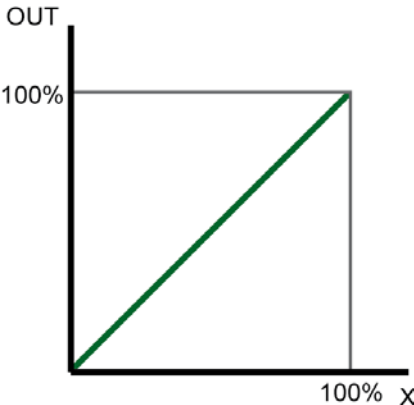
Beschreibung

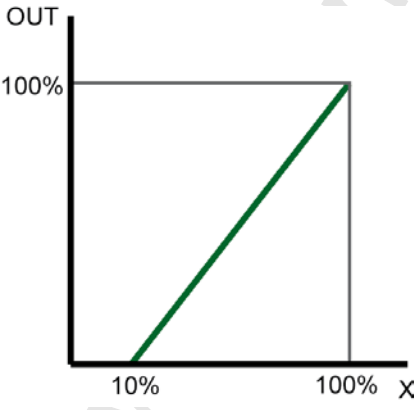
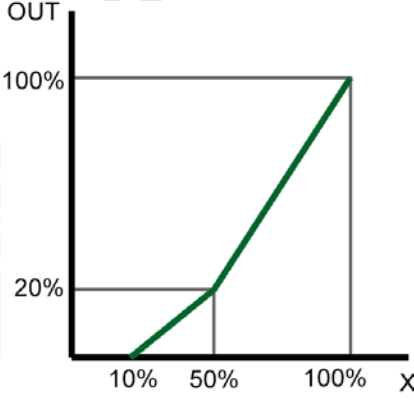
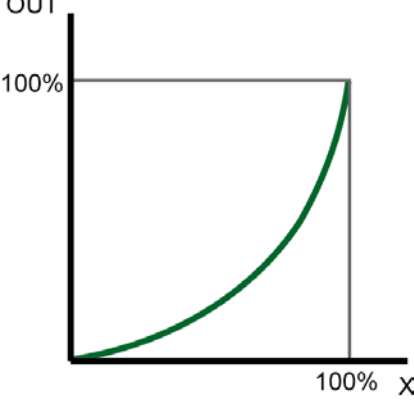
432

JOYSTICK_0 skaliert Signale aus einem Joystick auf fest definierte Kennlinien, normiert auf 0...1000.

Bei diesem FB sind die Kennlinien-Werte fest vorgegeben (→ Grafiken):

- Steigende Flanke der Rampe = 5 Inkremente/SPS-Zyklus
 Schnelle Controller haben sehr kurze Zykluszeiten!
- Fallende Flanke der Rampe = keine Rampe

Die Parameter XL_POS (XL+), XH_POS (XH+), XL_NEG (XL-) und XH_NEG (XH-) dienen dazu, die Joystickbewegung nur im erwünschten Bewegungsbereich auszuwerten. Die Werte für den positiven und den negativen Bereich dürfen sich unterscheiden. Die Werte für XL_NEG und XH_NEG sind hier negativ.	
Modus 0: Kennlinie linear für den Bereich XL bis XH	

Modus 1: Kennlinie linear mit Totbereich Werte fest eingestellt auf: Totbereich: 0...10% von 1000 Inkrementen	
Modus 2: Kennlinie 2-stufig linear mit Totbereich Werte fest eingestellt auf: Totbereich: 0...10% von 1000 Inkrementen Stufe: X = 50 % von 1000 Inkrementen Y = 20 % von 1000 Inkrementen	
Kennlinie Modus 3: Kurve ansteigend (Verlauf ist fest eingestellt)	

Parameter der Eingänge

433

Parameter	Datentyp	Beschreibung
X	INT	Eingangswert [Inkrement]
XH_POS	INT	Max. Sollwert positive Richtung [Inkrement] (auch negative Werte zulässig)
XL_POS	INT	Min. Sollwert positive Richtung [Inkrement] (auch negative Werte zulässig)
XH_NEG	INT	Max. Sollwert negative Richtung [Inkrement] (auch negative Werte zulässig)
XL_NEG	INT	Min. Sollwert negative Richtung [Inkrement] (auch negative Werte zulässig)
MODE	BYTE	Modus Auswahl Kennlinie: 0 = linear (X OUT = 0 0 ... 1000 1000) 1 = linear mit Totbereich (X OUT = 0 0 ... 100 0 ... 1000 1000) 2 = 2-stufig linear mit Totbereich (X OUT = 0 0 ... 100 0 ... 500 200 ... 1000 1000) 3 = Kurve ansteigend (Verlauf ist fest eingestellt)

Parameter der Ausgänge

6252

Parameter	Datentyp	Beschreibung
OUT1	WORD	normierter Ausgangswert: 0...1000 Inkremente z.B. für Ventil links
OUT2	WORD	normierter Ausgangswert: 0...1000 Inkremente z.B. für Ventil rechts
OUT3	INT	normierter Ausgangswert: -1000...0...1000 Inkremente z.B. für Ventil an Ausgangsmodul (z.B. CR2011 oder CR2031)
WRONG_MODE	BOOL	Fehler: Ungültiger Modus
ERR1	BYTE	Fehler-Code für steigende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der <code>util.lib</code>) (mögliche Meldungen → folgende Tabelle)
ERR2	BYTE	Fehler-Code für fallende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der <code>util.lib</code>) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERR1 und ERR2:

Wert dez hex	Beschreibung
0 00	kein Fehler
1 01	Fehler in Zahlenreihe: Falsche Reihenfolge
2 02	Fehler: Eingangswert IN ist nicht im Wertebereich der Zahlenreihe
4 04	Fehler: Ungültige Anzahl N für Zahlenreihe

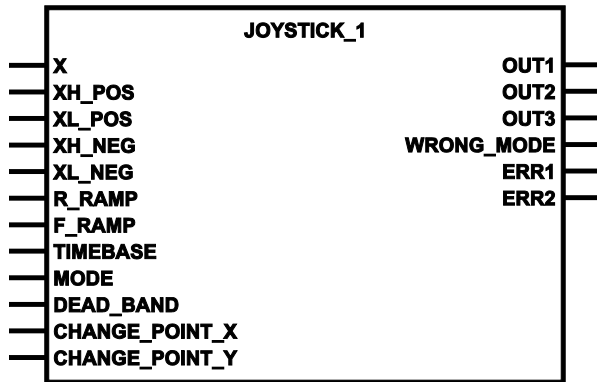
JOYSTICK_1

13226

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_hydraulic_CR0303_Vxxyzz.Lib

Symbol in CODESYS:

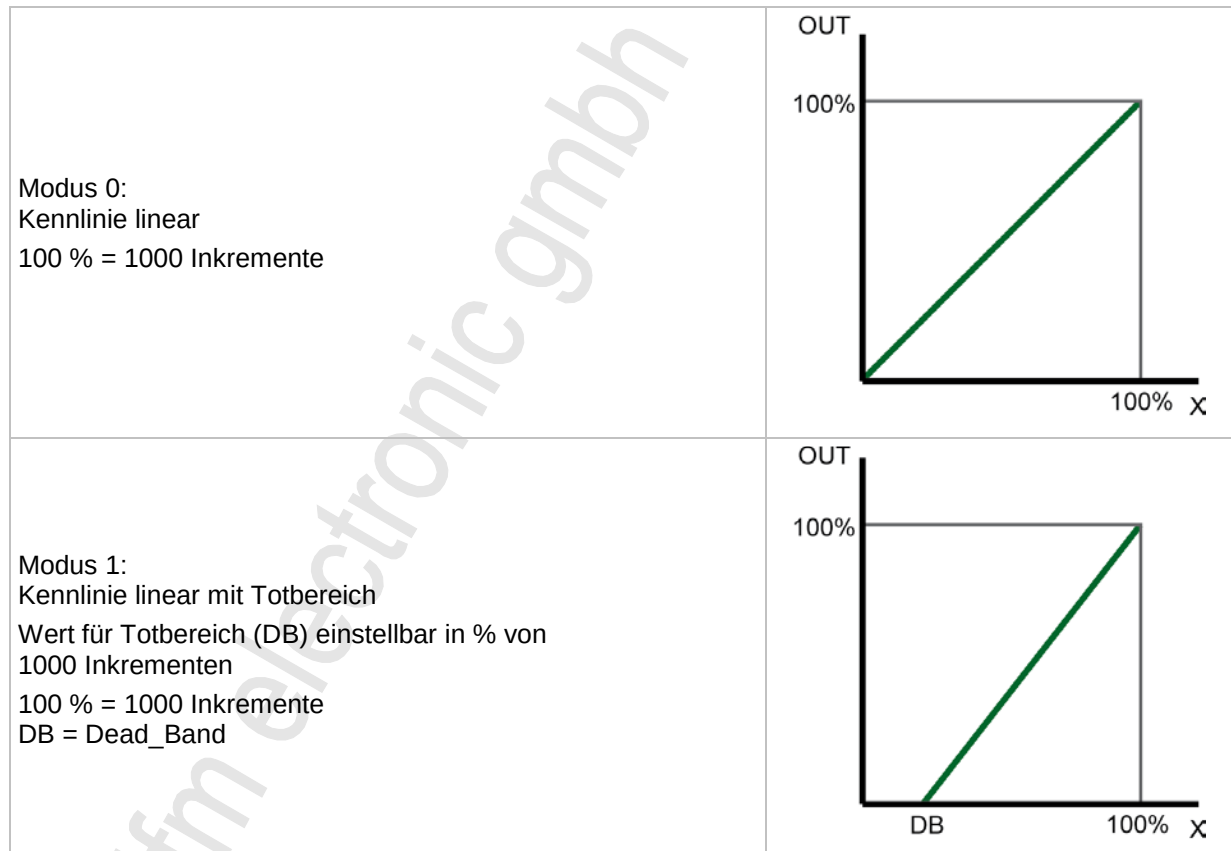


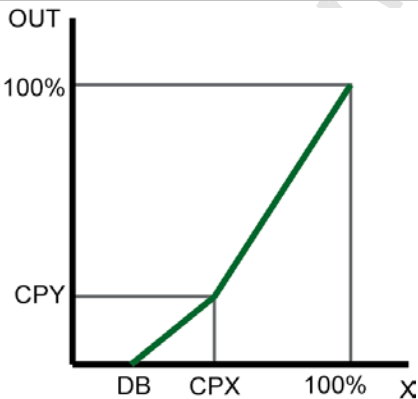
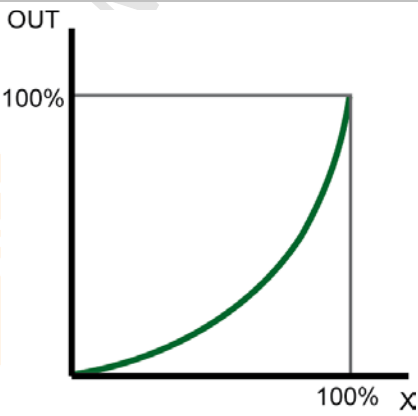
Beschreibung

425

JOYSTICK_1 skaliert Signale aus einem Joystick auf parametrierbare Kennlinien, normiert auf 0...1000.

Bei diesem FB sind die Kennlinien-Werte parametrierbar (→ Grafiken):



Modus 2: Kennlinie 2-stufig linear mit Totbereich Werte parametrierbar auf: Totbereich: 0...DB in % von 1000 Inkrementen Stufe: X = CPX in % von 1000 Inkrementen Y = CPY in % von 1000 Inkrementen 100 % = 1000 Inkremente DB = Dead_Band CPX = Change_Point_X CPY = Change_Point_Y	
Kennlinie Modus 3: Kurve ansteigend (Verlauf ist fest eingestellt)	

Parameter der Eingänge

6256

Parameter	Datentyp	Beschreibung
X	INT	Eingangswert [Inkrement]
XH_POS	INT	Max. Sollwert positive Richtung [Inkrement] (auch negative Werte zulässig)
XL_POS	INT	Min. Sollwert positive Richtung [Inkrement] (auch negative Werte zulässig)
XH_NEG	INT	Max. Sollwert negative Richtung [Inkrement] (auch negative Werte zulässig)
XL_NEG	INT	Min. Sollwert negative Richtung [Inkrement] (auch negative Werte zulässig)
R_RAMP	INT	Steigende Flanke der Rampe in [Inkrement/SPS-Zyklus] 0 = keine Rampe
F_RAMP	INT	Fallende Flanke der Rampe in [Inkrement/SPS-Zyklus] 0 = keine Rampe
TIMEBASE	TIME	Referenz für steigende und fallende Flanke der Rampe: t#0s = steigende / fallende Flanke in [Inkrement/SPS-Zyklus]  Schnelle Controller haben sehr kurze Zykluszeiten! sonst = steigende / fallende Flanke in [Inkrement/TIMEBASE]
MODE	BYTE	Modus Auswahl Kennlinie: 0 = linear (X OUT = 0 0 ... 1000 1000) 1 = linear mit Totbereich (X OUT = 0 0 ... DB 0 ... 1000 1000) 2 = 2-stufig linear mit Totbereich (X OUT = 0 0 ... DB 0 ... CPX CPY ... 1000 1000) 3 = Kurve ansteigend (Verlauf ist fest eingestellt)
DEAD_BAND	BYTE	Einstellbarer Totbereich in [% von 1000 Inkrementen]
CHANGE_POINT_X	BYTE	Für Modus 2: Rampenstufe, Wert für X in [% von 1000 Inkrementen]
CHANGE_POINT_Y	BYTE	Für Modus 2: Rampenstufe, Wert für Y in [% von 1000 Inkrementen]

Parameter der Ausgänge

6252

Parameter	Datentyp	Beschreibung
OUT1	WORD	normierter Ausgangswert: 0...1000 Inkremente z.B. für Ventil links
OUT2	WORD	normierter Ausgangswert: 0...1000 Inkremente z.B. für Ventil rechts
OUT3	INT	normierter Ausgangswert: -1000...0...1000 Inkremente z.B. für Ventil an Ausgangsmodul (z.B. CR2011 oder CR2031)
WRONG_MODE	BOOL	Fehler: Ungültiger Modus
ERR1	BYTE	Fehler-Code für steigende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der <code>util.lib</code>) (mögliche Meldungen → folgende Tabelle)
ERR2	BYTE	Fehler-Code für fallende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der <code>util.lib</code>) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERR1 und ERR2:

Wert dez hex		Beschreibung
0	00	kein Fehler
1	01	Fehler in Zahlenreihe: Falsche Reihenfolge
2	02	Fehler: Eingangswert IN ist nicht im Wertebereich der Zahlenreihe
4	04	Fehler: Ungültige Anzahl N für Zahlenreihe

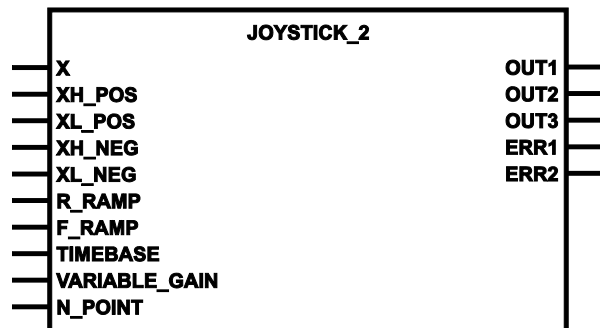
JOYSTICK_2

13229

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_hydraulic_CR0303_Vxxyzz.Lib

Symbol in CODESYS:

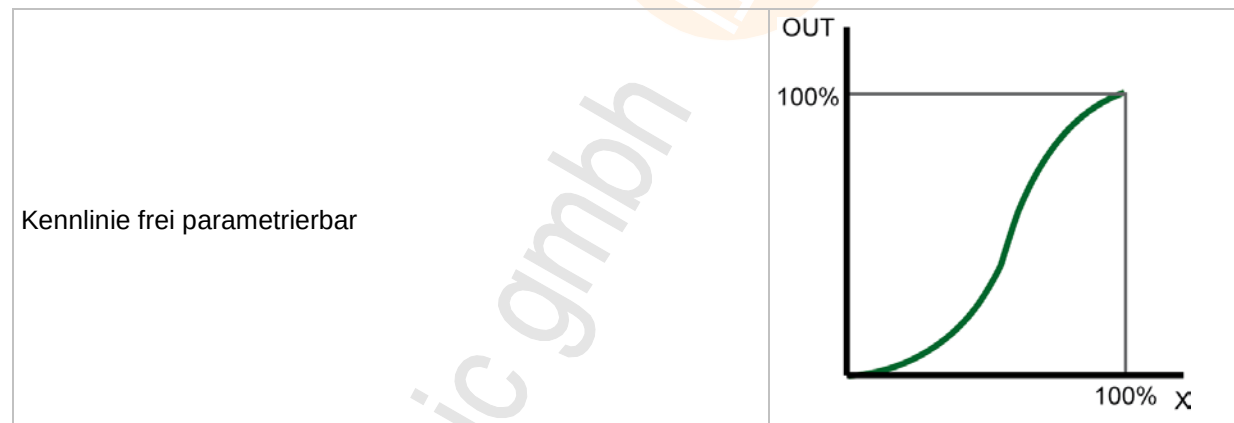


Beschreibung

418

JOYSTICK_2 skaliert Signale aus einem Joystick auf einen parametrierbaren Kennlinien-Verlauf. Die Normierung ist frei bestimmbar.

Bei diesem FB ist der Kennlinien-Verlauf frei parametrierbar (→ Grafik):



Parameter der Eingänge

6261

Parameter	Datentyp	Beschreibung
X	INT	Eingangswert [Inkrement]
XH_POS	INT	Max. Sollwert positive Richtung [Inkrement] (auch negative Werte zulässig)
XL_POS	INT	Min. Sollwert positive Richtung [Inkrement] (auch negative Werte zulässig)
XH_NEG	INT	Max. Sollwert negative Richtung [Inkrement] (auch negative Werte zulässig)
XL_NEG	INT	Min. Sollwert negative Richtung [Inkrement] (auch negative Werte zulässig)
R_RAMP	INT	Steigende Flanke der Rampe in [Inkrement/SPS-Zyklus] 0 = keine Rampe
F_RAMP	INT	Fallende Flanke der Rampe in [Inkrement/SPS-Zyklus] 0 = keine Rampe
TIMEBASE	TIME	Referenz für steigende und fallende Flanke der Rampe: t#0s = steigende / fallende Flanke in [Inkrement/SPS-Zyklus] ! Schnelle Controller haben sehr kurze Zykluszeiten! sonst = steigende / fallende Flanke in [Inkrement/TIMEBASE]
VARIABLE_GAIN	ARRAY [0..10] OF POINT	Wertepaare, die den Kurven-Verlauf beschreiben Es werden die ersten in N_POINT angegebenen Wertepaare verwertet. n = 2...11 Beispiel: 9 Wertepaare als Variable VALUES deklariert: VALUES : ARRAY [0..10] OF POINT := (X:=0,Y:=0), (X:=200,Y:=0), (X:=300,Y:=50), (X:=400,Y:=100), (X:=700,Y:=500), (X:=1000,Y:=900), (X:=1100,Y:=950), (X:=1200,Y:=1000), (X:=1400,Y:=1050); Zwischen den Werten dürfen auch Leerzeichen stehen.
N_POINT	BYTE	Anzahl der Punkte (Wertepaare in VARIABLE_GAIN), womit die Kurven-Charakteristik definiert ist: n = 2...11

Parameter der Ausgänge

420

Parameter	Datentyp	Beschreibung
OUT1	WORD	normierter Ausgangswert: 0...1000 Inkremente z.B. für Ventil links
OUT2	WORD	normierter Ausgangswert: 0...1000 Inkremente z.B. für Ventil rechts
OUT3	INT	normierter Ausgangswert: -1000...0...1000 Inkremente z.B. für Ventil an Ausgangsmodul (z.B. CR2011 oder CR2031)
ERR1	BYTE	Fehler-Code für steigende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der <code>util.lib</code>) (mögliche Meldungen → folgende Tabelle)
ERR2	BYTE	Fehler-Code für fallende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der <code>util.lib</code>) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERR1 und ERR2:

Wert dez hex		Beschreibung
0	00	kein Fehler
1	01	Fehler in Zahlenreihe: Falsche Reihenfolge
2	02	Fehler: Eingangswert IN ist nicht im Wertebereich der Zahlenreihe
4	04	Fehler: Ungültige Anzahl N für Zahlenreihe

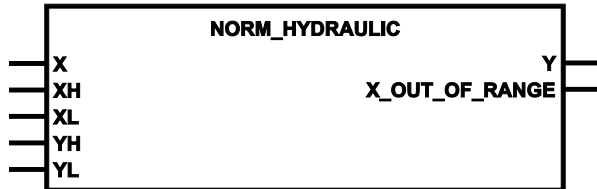
NORM_HYDRAULIC

13233

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_hydraulic_CR0303_Vxxyzz.Lib

Symbol in CODESYS:



Beschreibung

397

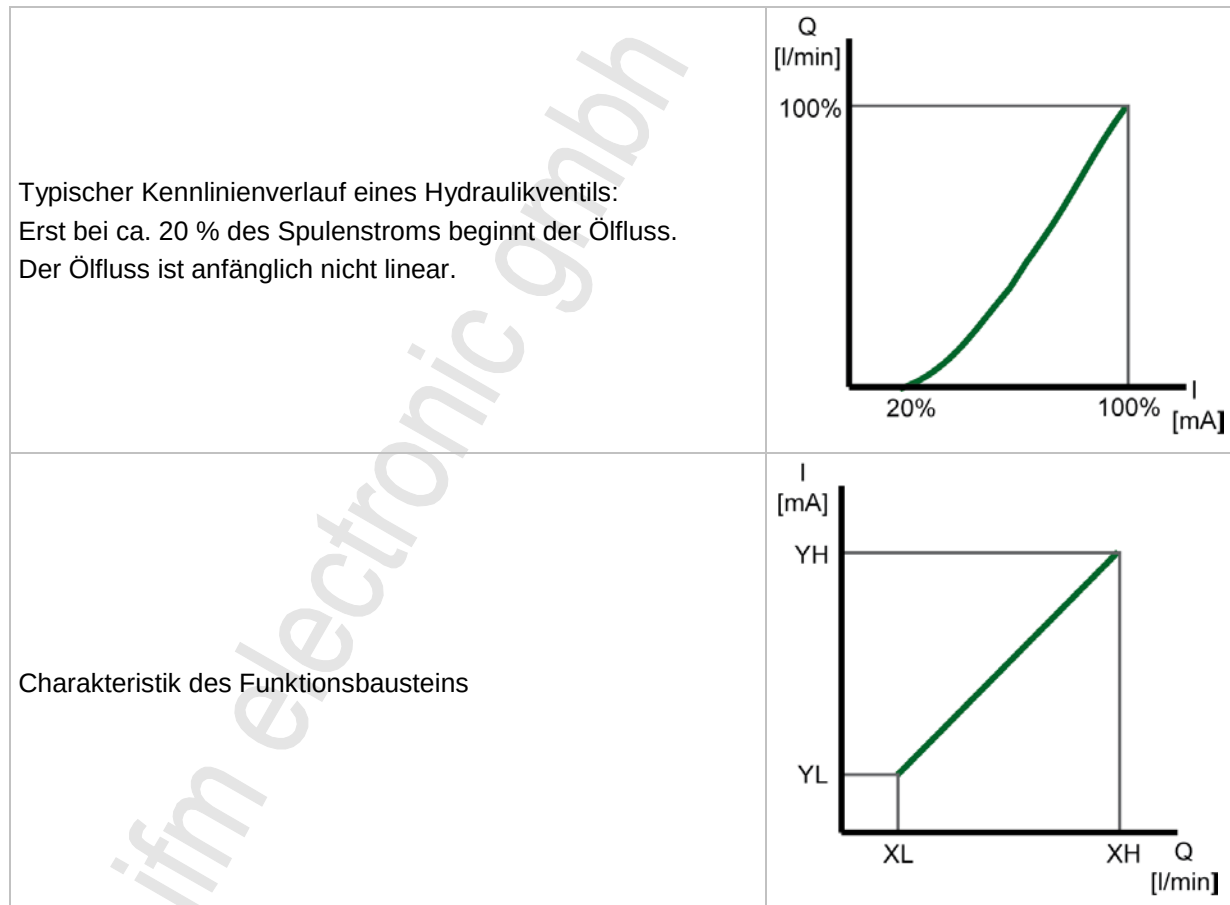
NORM_HYDRAULIC normiert Eingangswerte innerhalb festgesetzter Grenzen auf Werte mit neuen Grenzen.

 Dieser FB entspricht NORM_DINT aus der CODESYS-Bibliothek UTIL.Lib.

Der FB normiert einen Wert vom Typ DINT, der innerhalb der Grenzen zwischen XH und XL liegt, auf einen Ausgangswert innerhalb der Grenzen zwischen YH und YL.

Bedingt durch Rundungsfehler können Abweichungen beim normierten Wert um 1 auftreten. Werden die Grenzen (XH/XL oder YH/YL) invertiert angegeben, erfolgt auch die Normierung invertiert.

Wenn X außerhalb der Grenzen XL...XH liegt, wird die Fehlermeldung X_OUT_OF_RANGE = TRUE.



Parameter der Eingänge

398

Parameter	Datentyp	Beschreibung
X	DINT	Eingangswert
XH	DINT	Max. Eingangswert [Inkrement]
XL	DINT	Min. Eingangswert [Inkrement]
YH	DINT	Max. Ausgangswert [Inkrement], z.B.: Ventilstrom [mA], Durchfluss [l/min]
YL	DINT	Min. Ausgangswert [Inkrement], z.B.: Ventilstrom [mA], Durchfluss [l/min]

Parameter der Ausgänge

399

Parameter	Datentyp	Beschreibung
Y	DINT	Ausgangswert
X_OUT_OF_RANGE	BOOL	Fehler: X liegt außerhalb der Grenzen von XH und XL

Beispiel: NORM_HYDRAULIC

400

Parameter	Fall 1	Fall 2	Fall 3
oberer Grenzwert Eingang XH	100	100	2000
unterer Grenzwert Eingang XL	0	0	0
oberer Grenzwert Ausgang YH	2000	0	100
unterer Grenzwert Ausgang YL	0	2000	0
nicht normierter Wert X	20	20	20
normierter Wert Y	400	1600	1

- Fall 1:
Eingang mit relativ grober Auflösung.
Ausgang mit hoher Auflösung.
1 X-Inkrement ergibt 20 Y-Inkrement.
- Fall 2:
Eingang mit relativ grober Auflösung.
Ausgang mit hoher Auflösung.
1 X-Inkrement ergibt 20 Y-Inkrement.
Ausgangssignal ist gegenüber dem Eingangssignal invertiert.
- Fall 3:
Eingang mit hoher Auflösung.
Ausgang mit relativ grober Auflösung.
20 X-Inkrement ergeben 1 Y-Inkrement.

5.2.13 Bausteine: Regler

Inhalt

Einstellregel für einen Regler	163
DELAY	164
GLR	165
PID1	167
PID2	169
PT1	171

1634

Der nachfolgende Abschnitt beschreibt im Detail die Bausteine, die zum Aufbau von Software-Reglern im **ecomatmobile**-Gerät bereitgestellt werden. Die Bausteine können auch als Basis für die Entwicklung von eigenen Regelungsfunktionen genutzt werden.

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 K_P wird anschließend solange vergrößert, bis die Regel- und die Stellabweichung bei $K_P = K_{P_{kritisch}}$ 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 .

K_P sollte gleich groß wie K_D gewählt werden.

Idealisiert ist die Regelstrecke wie folgt einzustellen:

Regeleinrichtung	$K_P = K_D$	T_N	T_V
P	$2,0 \cdot K_{P_{kritisch}}$	—	—
PI	$2,2 \cdot K_{P_{kritisch}}$	$0,83 \cdot T_{kritisch}$	—
PID	$1,7 \cdot K_{P_{kritisch}}$	$0,50 \cdot T_{kritisch}$	$0,125 \cdot T_{kritisch}$

! Bei diesem Einstellverfahren darauf achten, dass die Regelstrecke durch die auftretenden Schwingungen keinen Schaden nimmt. Bei empfindlichen Regelstrecken darf K_P nur bis zu einem Wert erhöht werden, bei dem sicher noch keine Schwingungen auftreten.

Dämpfung von Überschwüngen

1629

Um Überschwüngen zu dämpfen, kann **PT1** (→ Seite [171](#)) (Tiefpass) eingesetzt werden. Dazu wird der Sollwert X_S durch das PT1-Glied gedämpft, bevor er der Reglerfunktion zugeführt wird.

Die Einstellgröße T_1 sollte ca. 4...5-mal größer sein als T_N des Reglers.

DELAY

585

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxyxyz.LIB

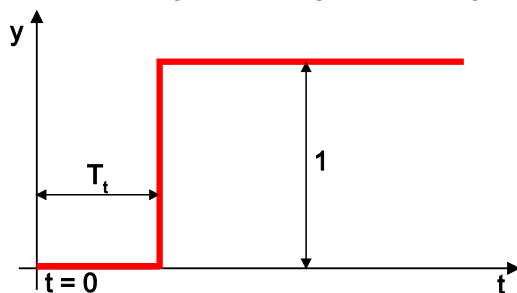
Symbol in CODESYS:



Beschreibung

588

DELAY verzögert die Ausgabe des Eingangswertes um die Zeit T (Totzeit-Glied).



Grafik: Zeitlicher Verlauf von DELAY

Die Totzeit wird durch die Dauer des SPS-Zyklus beeinflusst.

Die Totzeit darf nicht länger sein als 100 • SPS-Zykluszeit (Speichergrenze!).

Wird eine größere Verzögerung eingestellt, wird die Auflösung der Werte am Ausgang des FB schlechter, wodurch kurze Werteänderungen verloren gehen können.

⚠ Damit der FB einwandfrei arbeitet: FB in jedem SPS-Zyklus aufrufen!

Parameter der Eingänge

589

Parameter	Datentyp	Beschreibung
X	WORD	Eingangswert
T	TIME	Verzögerungszeit (Totzeit) zulässig: 0...100 • Zykluszeit

Parameter der Ausgänge

590

Parameter	Datentyp	Beschreibung
Y	WORD	Eingangswert, verzögert um die Zeit T

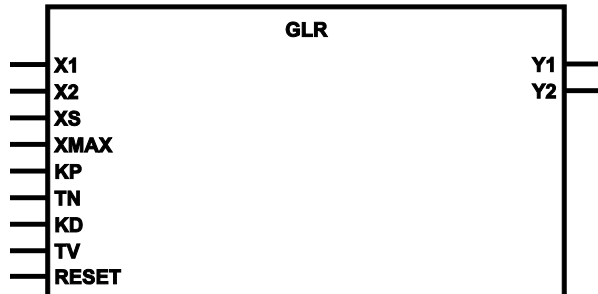
GLR

531

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxyxyz.LIB

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
XMAX	WORD	Maximaler Istwert zur Festlegung des Istwert-Wertebereichs
KP	Byte	Proportional-Anteil des Ausgangsignals (/ 10) (nur positive Werte zulässig)
TN	TIME	Nachstellzeit (Integral-Anteil)
KD	BYTE	Differential-Anteil des Ausgangsignals (/ 10) (nur positive Werte zulässig)
TV	TIME	Vorhaltezeit (Differential-Anteil)
RESET	BOOL	TRUE: Regler zurücksetzen FALSE: Funktion wird nicht ausgeführt

Parameter der Ausgänge

536

Parameter	Datentyp	Beschreibung
Y1	WORD	Stellgröße Kanal 1
Y2	WORD	Stellgröße Kanal 2

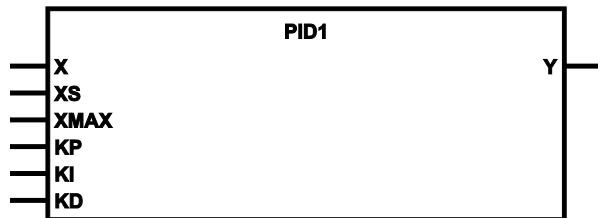
PID1

351

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxyxyz.LIB

Symbol in CODESYS:



Beschreibung

354

PID1 organisiert einen PID-Regler.

Die Änderung der Stellgröße eines PID-Reglers setzt sich aus einem proportionalen, integralen und differentiellen 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

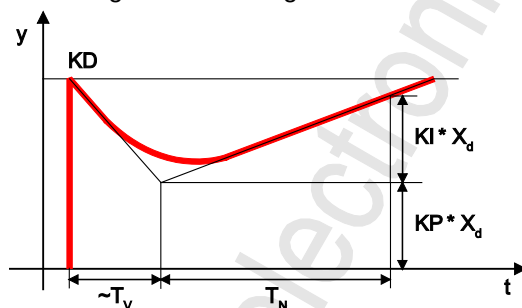
0 = maximaler Wert.

Beachten Sie, dass die Eingangsgrößen KI und KD zykluszeitabhängig sind. Um ein stabiles, reproduzierbares Regelverhalten zu bekommen, sollte der FB zeitgesteuert aufgerufen werden.

Wenn $X > X_S$, dann wird die Stellgröße erhöht.

Wenn $X < X_S$, 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	Eingangswert
XS	WORD	Sollwert
XMAX	WORD	Maximaler Istwert zur Festlegung des Istwert-Wertebereichs
KP	BYTE	Proportional-Anteil des Ausgangsignals
KI	BYTE	Integral-Anteil des Ausgangsignals
KD	BYTE	Differential-Anteil des Ausgangsignals

Parameter der Ausgänge

356

Parameter	Datentyp	Beschreibung
Y	WORD	Stellgröße (0...1000 ‰)

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.

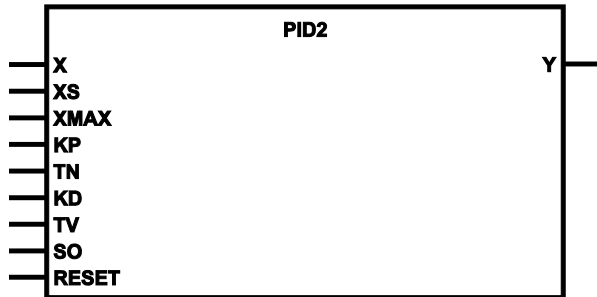
PID2

9167

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

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 (**Differential-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.

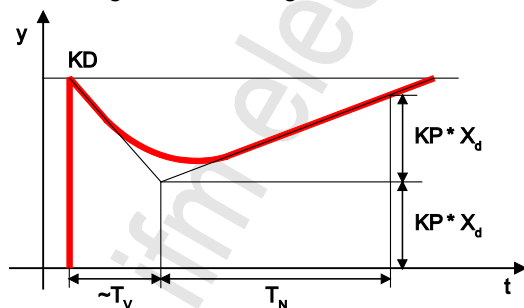
Wenn $X > X_S$, dann wird die Stellgröße erhöht.

Wenn $X < X_S$, dann wird die Stellgröße reduziert.

Eine Führungsgröße wird intern zur Stellgröße hinzuaddiert:

$$Y = Y + 65\,536 - (X_S / X_{MAX} * 65\,536).$$

Die Stellgröße Y hat folgenden zeitlichen Verlauf.



Grafik: Typische Sprungantwort eines PID-Reglers

Parameter der Eingänge

348

Parameter	Datentyp	Beschreibung
X	WORD	Eingangswert
XS	WORD	Sollwert
XMAX	WORD	Maximaler Istwert zur Festlegung des Istwert-Wertebereichs
KP	Byte	Proportional-Anteil des Ausgangssignals (/ 10) (nur positive Werte zulässig)
TN	TIME	Nachstellzeit (Integral-Anteil)
KD	BYTE	Differential-Anteil des Ausgangssignals (/ 10) (nur positive Werte zulässig)
TV	TIME	Vorhaltezeit (Differential-Anteil)
SO	BOOL	TRUE: Selbstoptimierung aktiv FALSE: Selbstoptimierung nicht aktiv
RESET	BOOL	TRUE: Regler zurücksetzen FALSE: Funktion wird nicht ausgeführt

Parameter der Ausgänge

349

Parameter	Datentyp	Beschreibung
Y	WORD	Stellgröße (0...1000 ‰)

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 ($TN \geq 50 \text{ ms}$)
- 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.

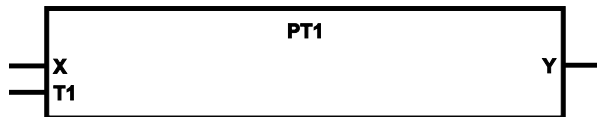
PT1

338

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxyxyz.LIB

Symbol in CODESYS:



Beschreibung

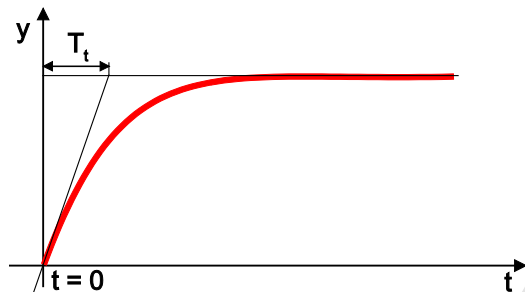
341

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.

! Der Ausgang des FB kann instabil werden, wenn T1 kleiner ist als die SPS-Zykluszeit.

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 [Inkrement]
T1	TIME	Verzögerungszeit (Zeitkonstante)

Parameter der Ausgänge

343

Parameter	Datentyp	Beschreibung
Y	INT	Ausgangswert

5.2.14 Bausteine: Software-Reset

Inhalt

SOFTRESET	173
-----------------	-----

1594

Hiermit kann die Steuerung per Kommando im Anwendungsprogramm neu gestartet werden.

SOFTRESET

260

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxyxyz.LIB

Symbol in CODESYS:



Beschreibung

263

SOFTRESET führt einen kompletten Neustart des Geräts aus.

Die Funktion kann z.B. in Verbindung mit CANopen genutzt werden, wenn ein Node-Reset ausgeführt werden soll. Der FB SOFTRESET führt einen sofortigen Neustart der Steuerung durch. Der aktuelle Zyklus wird nicht beendet.

Vor dem Neustart erfolgt das Speichern der Retain- Variablen.
Der Neustart wird im Fehlerspeicher protokolliert.

! Bei einer laufenden Kommunikation: die lange Reset-Phase beachten, da andernfalls Guarding-Fehler gemeldet werden.

Parameter der Eingänge

264

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert

5.2.15 Bausteine: Zeit messen / setzen

Inhalt

TIMER_READ	175
TIMER_READ_US	176

1601

Mit folgenden Bausteinen der **ifm electronic** können Sie...

- Zeiten messen und im Anwendungsprogramm auswerten,
- bei Bedarf Zeitwerte ändern.

TIMER_READ

236

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

239

TIMER_READ liest die aktuelle Systemzeit aus.

Mit Anlegen der Versorgungsspannung bildet das Gerät einen Zeittakt, der in einem Register aufwärts gezählt wird. Dieses Register kann mittels des Funktionsaufrufes ausgelesen und z.B. zur Zeitmessung genutzt werden.

! Der System-Timer läuft maximal bis 0xFFFF FFFF (entspricht 49d 17h 2min 47s 295ms) und startet anschließend wieder mit 0.

Parameter der Ausgänge

241

Parameter	Datentyp	Beschreibung
T	TIME	Aktuelle Systemzeit [ms]

TIMER_READ_US

657

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

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 1h 11min 34s 967ms 295µs und startet anschließend wieder mit 0.

Parameter der Ausgänge

662

Parameter	Datentyp	Beschreibung
TIME_US	DWORD	Aktuelle Systemzeit [µs]

5.2.16 Bausteine: Daten im Speicher sichern, lesen und wandeln

Inhalt

Speicherarten zur Datensicherung.....	177
Manuelle Datensicherung.....	178

13795

Speicherarten zur Datensicherung

13805

Das Gerät bietet folgende Speicher:

Flash-Speicher

13803

Eigenschaften:

- 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

FRAM-Speicher

13802

FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.

Eigenschaften:

- schnelles Schreiben und Lesen
- unbegrenzte Schreib-/Lesehäufigkeit
- beliebige Speicherbereiche wählbar
- Daten sichern mit FRAMWRITE
- Daten lesen mit FRAMREAD

Manuelle Datensicherung

Inhalt	
FLASHREAD	179
FLASHWRITE	180
FRAMREAD	182
FRAMWRITE	183
MEMCPY	184

13801

Neben der Möglichkeit, die Daten automatisch zu sichern, können über FB-Aufrufe Anwenderdaten manuell in integrierte Speicher gesichert und von dort wieder gelesen werden.

 Der Programmierer kann sich anhand der Speicheraufteilung (→ Kapitel *Verfügbarer Speicher* (→ Seite [14](#))) darüber informieren, welcher Speicherbereich frei zur Verfügung steht.

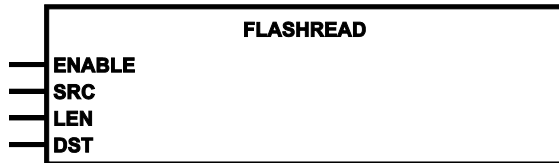
FLASHREAD

561

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_Vxxyzz.LIB`

Symbol in CODESYS:



Beschreibung

564

FLASHREAD ermöglicht das Lesen unterschiedlicher Datentypen direkt aus dem Flash-Speicher in den RAM.

- > 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.
- > Das Lesen erfolgt komplett in dem Zyklus, in dem der FB aufgerufen wird.
- ▶ Darauf achten, dass der Zielspeicherbereich im RAM groß genug ist.
- ▶ Für die Zieladresse DST gilt:
 - ❗ Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Parameter der Eingänge

20045

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
SRC	WORD	relative Quell-Anfangsadresse im Speicher zulässig = 0...16 383 = 0x0000...0x3FFF
LEN	WORD	Anzahl der Datenbytes zulässig = 0...16 383 = 0x0000...0x3FFF
DST	DWORD	Anfangsadresse der Zielvariablen ❗ Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

FLASHWRITE

555

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

558

**WARNUNG**

Gefahr durch unkontrollierten Prozessablauf!

Der Zustand der Ein-/Ausgänge wird während der Ausführung von FLASHWRITE "eingefroren".

- ▶ Diesen Funktionsbaustein nicht bei laufender Maschine ausführen!

FLASHWRITE ermöglicht das Schreiben unterschiedlicher Datentypen direkt in den Flash-Speicher.

Mit diesem FB sollen während der Inbetriebnahme große Datenmengen gesichert werden, auf die im Prozess nur lesend zugegriffen wird.

Der Flash-Speicher ist in 256 Byte große Pages organisiert.

- ▶ Wurde eine Page schon einmal (auch nur teilweise) beschrieben, muss der komplette Flash-Speicherbereich vor einem erneuten Schreibzugriff auf diese Page gelöscht werden. Dies geschieht durch einen Schreibzugriff auf die Adresse 0.
- ▶ Niemals mehrfach in eine Page schreiben! Erst immer alles löschen! Sonst entstehen Traps oder Watchdog-Fehler.
- ▶ **!** Den Flash-Speicherbereich nicht öfter als 100mal löschen, da ansonsten die Datenkonsistenz in anderen Flash-Speicherbereichen nicht mehr gewährleistet werden kann.
- ▶ In jedem SPS-Zyklus darf FLASHWRITE nur einmalig gestartet werden!
- ▶ Für die Zieladresse DST gilt:
 - !** Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
- > 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.
- !** Falls Startadresse SRC außerhalb des zulässigen Bereichs: kein Datentransfer!

Parameter der Eingänge

20049

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
DST	WORD	Relative Zieladresse im Speicher zulässig = 0...16 383 = 0x0000...0x3FFF
LEN	WORD	Anzahl der Datenbytes zulässig = 0...16 383 = 0x0000...0x3FFF
SRC	DWORD	Anfangsadresse der Quellvariablen ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

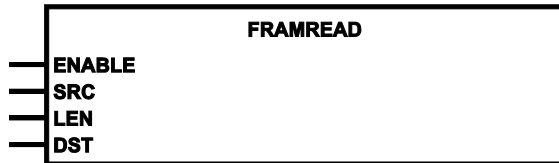
FRAMREAD

549

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_Vxyxyz.LIB`

Symbol in CODESYS:



Beschreibung

552

FRAMREAD ermöglicht das schnelle Lesen unterschiedlicher Datentypen direkt aus dem Anwender-Retain-Speicher (FRAM¹⁾).

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.

Würde durch die angegebene Anzahl an Bytes der FRAM-Speicherbereich überschritten werden, werden nur die Daten bis zum Ende des FRAM-Speicherbereichs gelesen.

► Für die Zieladresse DST gilt:

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

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

Parameter der Eingänge

20065

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
SRC	INT	relative Quell-Anfangsadresse im Speicher zulässig = 0...2 047 = 0x0000...0x07FF
LEN	INT	Anzahl der Datenbytes zulässig = 0...128 = 0x0000...0x0080
DST	DINT	Anfangsadresse der Zielvariablen ❗ Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

FRAMWRITE

543

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_Vxxyzz.LIB`

Symbol in CODESYS:



Beschreibung

546

FRAMWRITE ermöglicht das schnelle Schreiben unterschiedlicher Datentypen direkt in den Anwender-Retain-Speicher (FRAM¹⁾).

Der FB schreibt den Inhalt ab der Adresse SRC in den spannungsausfallsicheren FRAM-Speicher. Dabei werden genau so viele Bytes übertragen, wie diese über LEN angegeben sind. Würde durch die angegebene Anzahl an Bytes der FRAM-Speicherbereich überschritten werden, werden nur die Daten bis zum Ende des FRAM-Speicherbereichs geschrieben.

► Für die Quelladresse SRC gilt:

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

❗ Falls Zieladresse DST außerhalb des zulässigen Bereichs: kein Datentransfer!

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

Parameter der Eingänge

20073

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
DST	INT	Relative Zieladresse im Speicher zulässig = 512...2 047 = 0x0200...0x07FF
LEN	INT	Anzahl der Datenbytes zulässig = 0...128 = 0x0000...0x0080
SRC	DINT	Anfangsadresse der Quellvariablen ❗ Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

MEMCPY

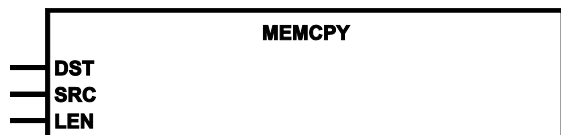
409

= Memory Copy

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

412

MEMCPY ermöglicht das Schreiben und Lesen unterschiedlicher Datentypen direkt in den Speicher. Der FB schreibt den Inhalt ab der Adresse von SRC an die Adresse DST.

- Für die Adressen SRC und DST gilt:
 - ❗ 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 Word-Variablen zu übertragen.

Parameter der Eingänge

413

Parameter	Datentyp	Beschreibung
DST	DWORD	Startadresse im Zielspeicher ❗ Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
SRC	DWORD	Startadresse im Quellspeicher ❗ Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
LEN	WORD	Anzahl (≥ 1) der zu übertragenden Daten-Bytes

5.2.17 Bausteine: Datenzugriff und Datenprüfung

Inhalt

CHECK_DATA	186
GET_IDENTITY	188
SET_DEBUG	189
SET_IDENTITY	190
SET_PASSWORD	191

1598

Die Bausteine in diesem Kapitel steuern den Datenzugriff und ermöglichen ein Prüfen der Daten.

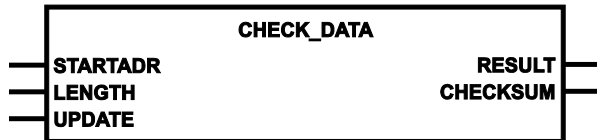
CHECK_DATA

603

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_Vxyxyz.LIB`

Symbol in CODESYS:



Beschreibung

606

CHECK_DATA erzeugt über einen konfigurierbaren Speicherbereich eine Prüfsumme (CRC) und prüft die Daten des Speicherbereichs auf ungewollte Veränderung.

- ▶ Für jeden zu überwachenden Speicherbereich eine eigene Instanz des FB erzeugen.
- ▶ **!** Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
- ▶ Zusätzlich die Anzahl der Datenbytes LENGTH (Länge ab der STARTADR) angeben.

Ungewollte Änderung: Fehler!

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.

Gewollte Änderung:

Nur wenn der Eingang UPDATE auf TRUE gesetzt ist, sind Datenänderungen im Speicher (z.B. vom Anwendungsprogramm oder *ecomatmobile*-Gerät) zulässig. Der Wert der Prüfsumme wird dann neu berechnet. Der Ausgang RESULT ist wieder permanent TRUE.

Parameter der Eingänge

607

Parameter	Datentyp	Beschreibung
STARTADR	DINT	Startadresse des überwachten Datenspeichers (WORD-Adresse ab %MW0) ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
LENGTH	WORD	Länge des überwachten Datenspeichers in [Byte]
UPDATE	BOOL	TRUE: Datenänderungen zulässig FALSE: Datenänderungen nicht zulässig

Parameter der Ausgänge

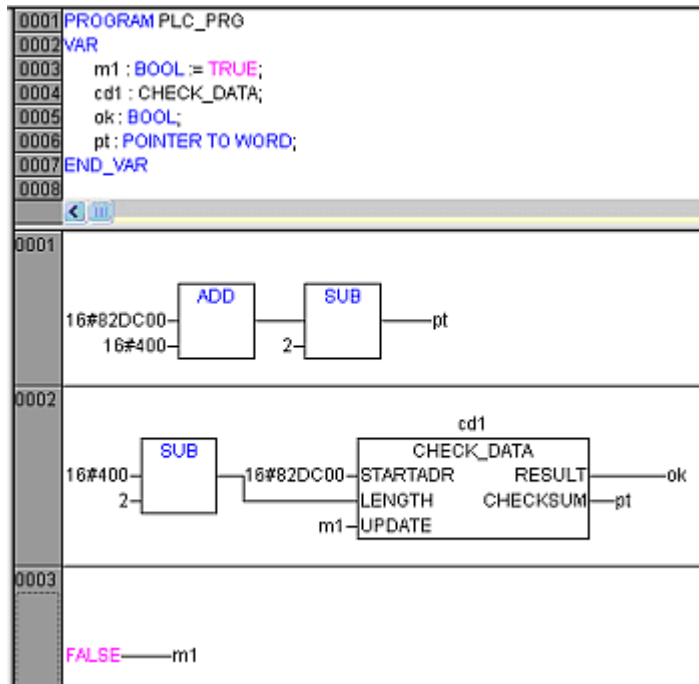
608

Parameter	Datentyp	Beschreibung
RESULT	BOOL	TRUE: CRC-Checksumme in Ordnung FALSE: CRC-Checksumme fehlerhaft (Daten wurden geändert)
CHECKSUM	DWORD	aktuelle CRC-Prüfsumme

Beispiel: CHECK_DATA

4168

Im folgenden Beispiel ermittelt das Programm die Prüfsumme und legt sie über den Pointer pt im RAM ab:



! Das hier gezeigte Verfahren ist für den Flash-Speicher nicht geeignet.

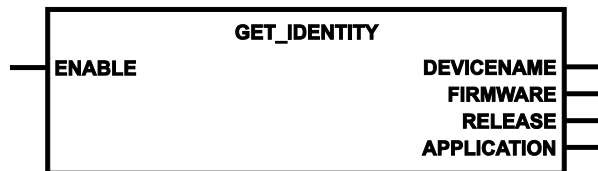
GET_IDENTITY

2212

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0303_Vxxyzz.LIB`

Symbol in CODESYS:



Beschreibung

2344

GET_IDENTITY liest die im Gerät gespeicherten spezifischen Kennungen:

- Hardware-Name und Hardware-Version des Geräts
- Name des Laufzeitsystems im Gerät
- Version und Ausgabe des Laufzeitsystems im Gerät
- Name der Anwendung (wurde zuvor mit *SET_IDENTITY* (→ Seite [190](#)) gespeichert)

Parameter der Eingänge

2609

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert

Parameter der Ausgänge

2610

Parameter	Datentyp	Beschreibung
DEVICENAME	STRING(31)	Hardware-Name und Hardware-Version des Geräts als Zeichenkette von max. 31 Zeichen z.B.: "CR0403 01.00.00"
FIRMWARE	STRING(31)	Name des Laufzeitsystems im Gerät als Zeichenkette von max. 31 Zeichen z.B.: "CR0403"
RELEASE	STRING(31)	Version und Ausgabe des Laufzeitsystems im Gerät als Zeichenkette von max. 31 Zeichen z.B.: "V01.00.00 120215"
APPLICATION	STRING(79)	Name der Anwendung als String von max. 79 Zeichen z.B.: "Crane1704"

SET_DEBUG

290

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



Beschreibung

293

SET_DEBUG organisiert den DEBUG-Modus ohne aktiven Test-Eingang (→ Kapitel *TEST-Betrieb* (→ Seite 39)).

Wird der Eingang DEBUG auf TRUE gesetzt, kann z.B. das Programmiersystem oder der Downloader mit dem Gerät kommunizieren und Systemkommandos ausführen (z.B. für Servicefunktionen über das GSM-Modem CANremote).

! Ein Software-Download ist in dieser Betriebsart nicht möglich, da der Test-Eingang nicht mit Versorgungsspannung verbunden wird. Nur lesender Zugriff ist möglich.

Parameter der Eingänge

294

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
DEBUG	BOOL	TRUE: Debugging über die Schnittstellen möglich FALSE: Debugging über die Schnittstellen nicht möglich

SET_IDENTITY

284

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

Symbol in CODESYS:



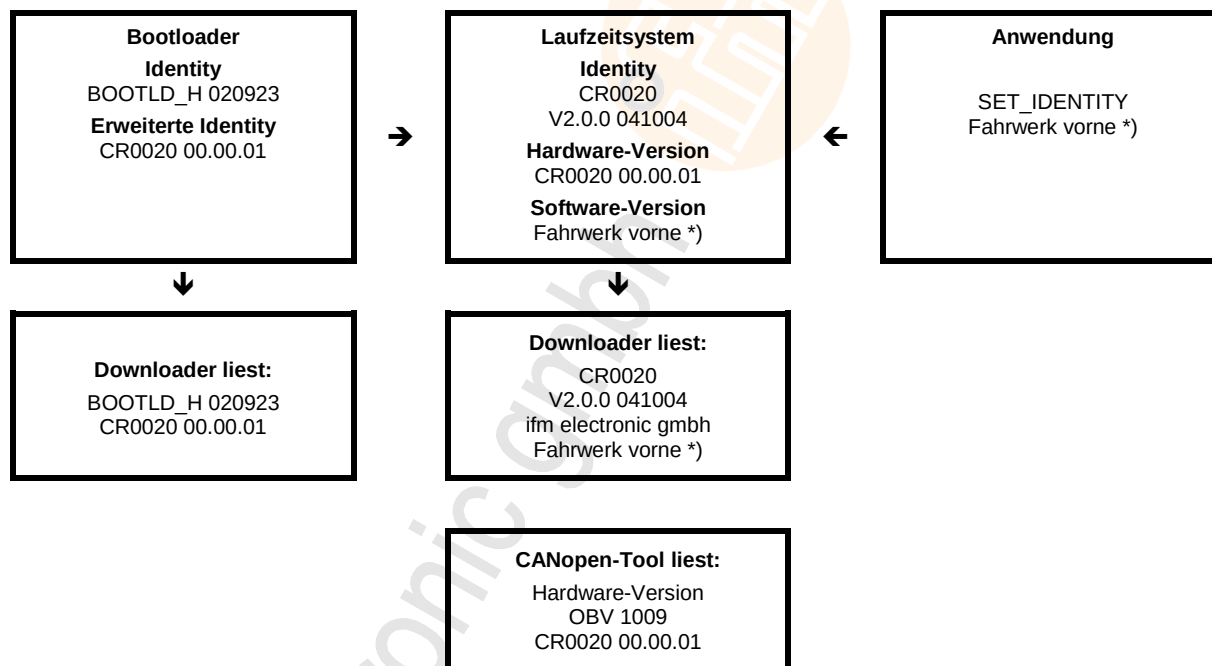
Beschreibung

287

SET_IDENTITY setzt eine anwendungsspezifische Programmkennung.

Mit dem FB kann durch das Anwendungsprogramm 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):



*)  'Fahrwerk vorne' steht hier stellvertretend für einen kundenspezifischen Text.

Parameter der Eingänge

288

Parameter	Datentyp	Beschreibung
ID	STRING(80)	beliebiger Text mit einer maximalen Länge von 80 Zeichen

SET_PASSWORD

266

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek ifm_CR0303_Vxxyzz.LIB

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 Anwendungsprogramm oder der Datenspeicher nur ausgelesen werden, wenn das richtige Passwort eingegeben wurde.

Wird an den Eingang PASSWORD ein Leer-String (Default-Zustand) übergeben, ist ein Upload des Anwendungsprogramms oder des Datenspeichers jederzeit möglich.

Ein neues Passwort wird nur nach dem Löschen des bisherigen Passwortes übernommen.

! Beim Laden eines neuen Anwendungsprogramms wird die Kennung wieder zurückgesetzt.

Parameter der Eingänge

270

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (nur 1 Zyklus lang): Parameter übernehmen FALSE: Baustein wird nicht ausgeführt
PASSWORD	STRING(16)	Benutzerkennung Wenn PASSWORD = "", dann ist Zugriff ohne Passworteingabe möglich.

6 Diagnose und Fehlerbehandlung

Inhalt

Diagnose	192
Fehler	192
Reaktion im Fehlerfall.....	192
Reaktion auf System-Fehler.....	193
CAN / CANopen: Fehler und Fehlerbehandlung.....	193

19598

Das Laufzeitsystem (LZS) überprüft das Gerät durch interne Fehler-Checks:

- in der Startphase (Reset-Phase)
- während der Ausführung des Anwendungsprogramms

→ Kapitel **Betriebszustände** (→ Seite [35](#))

So wird eine möglichst hohe Betriebssicherheit gewährleistet.

6.1 Diagnose

19601

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.

6.2 Fehler

19602

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.

6.3 Reaktion im Fehlerfall

12653

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

> die Betriebs-LED leuchtet rot,

 Vollständige Aufstellung der gerätespezifischen Fehler-Codes und Diagnosemeldungen

→ Kapitel **Systemmerker** (→ Seite [194](#))

6.4 Reaktion auf System-Fehler

19654

! Für die sichere Verarbeitung der Daten im Anwendungsprogramm ist allein dessen Programmierer verantwortlich.

- Die spezifischen Fehlermerker im Anwendungsprogramm verarbeiten!
Über den Fehlermerker erhält man eine Fehlerbeschreibung.
Diese Fehlermerker können bei Bedarf weiter verarbeitet werden.

Bei schweren Fehlern kann zusätzlich das Systemmerker-Bit ERROR gesetzt werden.

ERROR = TRUE bewirkt gleichzeitig Folgendes:

- via Anwendungsprogramm alle relevanten Ausgänge auf FALSE setzen,
- die Betriebs-LED leuchtet rot,
- der ERROR-Ausgang wird auf FALSE gesetzt.

Nach der Analyse und Beseitigung der Fehler-Ursache:

- Grundsätzlich alle Fehlermerker durch das Anwendungsprogramm zurücksetzen.
Ohne ausdrückliches Rücksetzen der Fehlermerker bleiben die Merker gesetzt mit entsprechender Auswirkung im Anwendungsprogramm.

6.5 CAN / CANopen: Fehler und Fehlerbehandlung

19604

→ Systemhandbuch "Know-How *ecomatmobile*"

→ Kapitel *CAN / CANopen: Fehler und Fehlerbehandlung*

7 Anhang

Inhalt

Systemmerker	194
Adressbelegung und E/A-Betriebsarten	200
Fehler-Tabellen	208

1664

Hier stellen wir Ihnen – ergänzend zu den Angaben in den Datenblättern – zusammenfassende Tabellen zur Verfügung.

7.1 Systemmerker

Inhalt

Systemmerker: CAN	195
Systemmerker: Fehlermerker	196
Systemmerker: LED (Standard-Seite)	197
Systemmerker: Spannungen	198
Systemmerker: 24 Eingänge und 18 Ausgänge	199

12167



Die zu den Systemmerkern gehörenden Merkeradressen können sich bei einer Erweiterung der Steuerungskonfiguration ändern.

- Für die Programmierung nur die Symbolnamen der Systemmerker nutzen!

→ Systemhandbuch "Know-How *ecomatmobile*"

→ Kapitel *Fehler-Codes und Diagnoseinformationen*

7.1.1 Systemmerker: CAN

18830

Systemmerker (Symbolname)	Typ	Beschreibung														
CANx_BAUDRATE	WORD	CAN-Schnittstelle x: eingestellte Baudrate in [kBaud]														
CANx_BUSOFF	BOOL	CAN-Schnittstelle x: Fehler "CAN-Bus off"  Zurücksetzen des Fehler-Codes setzt auch den Merker zurück														
CANx_LASTERROR	BYTE	CAN-Schnittstelle x: Fehlernummer der letzten CAN-Übertragung: <table><tr><td>0 = kein Fehler</td><td>Initial-Wert</td></tr><tr><td>1 = Stuff Error</td><td>mehr als 5 gleiche Bits in Reihe auf dem Bus</td></tr><tr><td>2 = Form Error</td><td>empfangenes Telegramm hatte falsches Format</td></tr><tr><td>3 = Ack Error</td><td>gesendetes Telegramm wurde nicht bestätigt</td></tr><tr><td>4 = Bit1 Error</td><td>außerhalb des Arbitrierungsbereichs wurde ein rezessives Bit gesendet, aber ein dominantes Bit auf dem Bus gelesen</td></tr><tr><td>5 = Bit0 Error</td><td>es wurde versucht, ein dominantes Bit zu senden, aber es wurde ein rezessiver Pegel gelesen ODER: während Bus-off Recovery wurde eine Sequenz von 11 rezessiven Bits gelesen</td></tr><tr><td>6 = CRC Error</td><td>die Prüfsumme der empfangenen Nachricht war falsch</td></tr></table>	0 = kein Fehler	Initial-Wert	1 = Stuff Error	mehr als 5 gleiche Bits in Reihe auf dem Bus	2 = Form Error	empfangenes Telegramm hatte falsches Format	3 = Ack Error	gesendetes Telegramm wurde nicht bestätigt	4 = Bit1 Error	außerhalb des Arbitrierungsbereichs wurde ein rezessives Bit gesendet, aber ein dominantes Bit auf dem Bus gelesen	5 = Bit0 Error	es wurde versucht, ein dominantes Bit zu senden, aber es wurde ein rezessiver Pegel gelesen ODER: während Bus-off Recovery wurde eine Sequenz von 11 rezessiven Bits gelesen	6 = CRC Error	die Prüfsumme der empfangenen Nachricht war falsch
0 = kein Fehler	Initial-Wert															
1 = Stuff Error	mehr als 5 gleiche Bits in Reihe auf dem Bus															
2 = Form Error	empfangenes Telegramm hatte falsches Format															
3 = Ack Error	gesendetes Telegramm wurde nicht bestätigt															
4 = Bit1 Error	außerhalb des Arbitrierungsbereichs wurde ein rezessives Bit gesendet, aber ein dominantes Bit auf dem Bus gelesen															
5 = Bit0 Error	es wurde versucht, ein dominantes Bit zu senden, aber es wurde ein rezessiver Pegel gelesen ODER: während Bus-off Recovery wurde eine Sequenz von 11 rezessiven Bits gelesen															
6 = CRC Error	die Prüfsumme der empfangenen Nachricht war falsch															
CANx_WARNING	BOOL	CAN-Schnittstelle x: Warnschwelle erreicht (≥ 96)  Reset des Merkers ist via Schreibzugriff möglich														
DOWNLOADID	WORD	CAN-Schnittstelle x: eingestellter Download-Identifizier														

CANx steht für x = 1...2 = Nummer der CAN-Schnittstelle

7.1.2 Systemmerker: Fehlermerker

18831

Systemmerker (Symbolname)	Typ	Beschreibung
ERROR	BOOL	TRUE: sicherer Zustand eingenommen alle Ausgänge = AUS (z.B. fataler Fehler / Error-Stop) FALSE: kein schwerer Fehler aufgetreten
ERROR_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	BYTE	Eingangsgruppe x: Peripheriefehler [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_MEMORY	BOOL	Speicherfehler
ERROR_OVERCURRENT_OUTxx xx = 12...17	BOOL	Fehler: Überstrom am Ausgang OUTxx
ERROR_POWER	BOOL	Spannungs-Fehler für VBBS / Klemme 15: TRUE: Wert außerhalb des zulässigen Bereichs oder: Differenz (VBB15 - VBBS) zu groß > allgemeiner Fehler FALSE: Wert in Ordnung
ERROR_TEMPERATURE	BOOL	Temperatur-Fehler TRUE: Wert außerhalb des zulässigen Bereichs > allgemeiner Fehler FALSE: Wert in Ordnung
WARNING_OVERCURRENT_OUTxx xx = 12...17	BOOL	Warnung: Überstrom am Ausgang OUTxx

7.1.3 Systemmerker: LED (Standard-Seite)

12817

Systemmerker (Symbolname)	Typ	Beschreibung
LED	WORD	LED-Farbe für "LED eingeschaltet": 0x0000 = LED_GREEN (voreingestellt) 0x0001 = LED_BLUE 0x0002 = LED_RED 0x0003 = LED_WHITE 0x0004 = LED_BLACK 0x0005 = LED_MAGENTA 0x0006 = LED_CYAN 0x0007 = LED_YELLOW
LED_X	WORD	LED-Farbe für "LED ausgeschaltet": 0x0000 = LED_GREEN 0x0001 = LED_BLUE 0x0002 = LED_RED 0x0003 = LED_WHITE 0x0004 = LED_BLACK (voreingestellt) 0x0005 = LED_MAGENTA 0x0006 = LED_CYAN 0x0007 = LED_YELLOW
LED_MODE	WORD	LED-Blinkfrequenz: 0x0000 = LED_2HZ (blinkt mit 2 Hz; voreingestellt) 0x0001 = LED_1HZ (blinkt mit 1 Hz) 0x0002 = LED_05HZ (blinkt mit 0,5 Hz) 0x0003 = LED_0HZ (leuchtet dauernd mit Wert in LED)

7.1.4 Systemmerker: Spannungen

18833

Systemmerker (Symbolname)	Typ	Beschreibung
REF_VOLTAGE	WORD	Wert • 0,1 = Spannung am Referenzspannungsausgang in [V]
REFERENCE_VOLTAGE_5	BOOL	Referenzspannungsausgang mit 5 V aktiviert
REFERENCE_VOLTAGE_10	BOOL	Referenzspannungsausgang mit 10 V aktiviert
SERIAL_MODE	BOOL	serielle Schnittstelle (RS232) für die Verwendung in der Anwendung aktivieren TRUE: RS232-Schnittstelle kann in der Anwendung verwendet werden, jedoch nicht mehr zum Programmieren, Debuggen oder Monitoren des Geräts. FALSE: RS232-Schnittstelle kann in der Anwendung nicht verwendet werden. Programmieren, Debuggen oder Monitoren des Geräts ist möglich.
SERIAL_BAUDRATE	WORD	Baudrate der RS232-Schnittstelle
SUPPLY_VOLTAGE	WORD	Wert • 0,1 = Versorgungsspannung an VBBS in [V]
TEST	BOOL	TRUE: Test-Eingang ist aktiv: • Programmiermodus ist freigeben • Software-Download ist möglich • Zustand des Anwendungsprogramms ist abfragbar • kein Schutz der gespeicherten Software möglich FALSE: laufender Betrieb der Anwendung

7.1.5 Systemmerker: 24 Eingänge und 18 Ausgänge

18834

Systemmerker (Symbolname)	Typ	Beschreibung
A_INxx xx = 00...15	WORD	Diagnosewert am Eingang INxx: gefilterter A/D-Wandler-Rohwert (10 Bit) ohne Kalibrierung und Normierung
A_INxx xx = 16...23	WORD	Analog-Eingang xx: gefilterter A/D-Wandler-Rohwert (10 Bit) ohne Kalibrierung und Normierung
A_INxx_MODE xx = 16...23	BYTE	Betriebsart des Analog-Eingangs xx → Kapitel <i>Mögliche Betriebsarten Ein-/Ausgänge</i> (→ Seite 205)
CURRxx xx = 12...17	WORD	PWM-Ausgang xx: gefilterte A/D-Wandler-Rohwerte (12 Bit) der Strommessung ohne Kalibrierung und Normierung
INxx xx = 00...23	BOOL	Status am Binäreingang xx Voraussetzung: Eingang ist als Binäreingang konfiguriert (MODE = IN_DIGITAL_H oder IN_DIGITAL_L) TRUE: Spannung am Binäreingang > 70 % von VBBS FALSE: Spannung am Binäreingang < 30 % von VBBS oder: nicht als Binäreingang konfiguriert oder: falsch konfiguriert
INxx_MODE xx = 00...15	BYTE	Betriebsart des Eingangs INxx → Kapitel <i>Mögliche Betriebsarten Ein-/Ausgänge</i> (→ Seite 205)
OUTxx xx = 00...17	BOOL	Status am Binärausgang xx: TRUE: Ausgang aktiviert FALSE: Ausgang deaktiviert

7.2 Adressbelegung und E/A-Betriebsarten

Inhalt

Adressbelegung Ein-/Ausgänge	200
Mögliche Betriebsarten Ein-/Ausgänge	205

1656

→ auch Datenblatt

7.2.1 Adressbelegung Ein-/Ausgänge

Inhalt

Eingänge: Adressbelegung	201
Ausgänge: Adressbelegung	203

2371

Eingänge: Adressbelegung

20099

Abkürzungen → Kapitel *Hinweise zur Anschlussbelegung* (→ Seite [24](#))

Betriebsarten der Ein- und Ausgänge → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))

IEC-Adresse	Symbolische Adresse	Konfiguration mit ...	Hinweis
%IX0.0	IN00	IN00_MODE	Binär-Eingang
%IX0.1	IN01	IN01_MODE	Binär-Eingang
%IX0.2	IN02	IN02_MODE	Binär-Eingang
%IX0.3	IN03	IN03_MODE	Binär-Eingang
%IX0.4	IN04	IN04_MODE	Binär-Eingang
%IX0.5	IN05	IN05_MODE	Binär-Eingang
%IX0.6	IN06	IN06_MODE	Binär-Eingang
%IX0.7	IN07	IN07_MODE	Binär-Eingang
%IX0.8	IN08	IN08_MODE	Binär-Eingang
%IX0.9	IN09	IN09_MODE	Binär-Eingang
%IX0.10	IN10	IN10_MODE	Binär-Eingang
%IX0.11	IN11	IN11_MODE	Binär-Eingang
%IX0.12	IN12	IN12_MODE	Binär-Eingang
%IX0.13	IN13	IN13_MODE	Binär-Eingang
%IX0.14	IN14	IN14_MODE	Binär-Eingang
%IX0.15	IN15	IN15_MODE	Binär-Eingang
%IX0.16	IN16	A_IN16_MODE	Binär-Eingang
%IX0.17	IN17	A_IN17_MODE	Binär-Eingang
%IX0.18	IN18	A_IN18_MODE	Binär-Eingang
%IX0.19	IN19	A_IN19_MODE	Binär-Eingang
%IX0.20	IN20	A_IN20_MODE	Binär-Eingang
%IX0.21	IN21	A_IN21_MODE	Binär-Eingang
%IX0.22	IN22	A_IN22_MODE	Binär-Eingang
%IX0.23	IN23	A_IN23_MODE	Binär-Eingang
%IW2	A_IN16	FB	Analog-Eingang
%IW3	A_IN17	FB	Analog-Eingang
%IW4	A_IN18	FB	Analog-Eingang
%IW5	A_IN19	FB	Analog-Eingang
%IW6	A_IN20	FB	Analog-Eingang
%IW7	A_IN21	FB	Analog-Eingang
%IW8	A_IN22	FB	Analog-Eingang
%IW9	A_IN23	FB	Analog-Eingang
%IW10	A_IN00	IN00_MODE	Diagnose zu IN00
%IW11	A_IN01	IN01_MODE	Diagnose zu IN01
%IW12	A_IN02	IN02_MODE	Diagnose zu IN02
%IW13	A_IN03	IN03_MODE	Diagnose zu IN03
%IW14	A_IN04	IN04_MODE	Diagnose zu IN04

IEC-Adresse	Symbolische Adresse	Konfiguration mit ...	Hinweis
%IW15	A_IN05	IN05_MODE	Diagnose zu IN05
%IW16	A_IN06	IN06_MODE	Diagnose zu IN06
%IW17	A_IN07	IN07_MODE	Diagnose zu IN07
%IW18	A_IN08	IN08_MODE	Diagnose zu IN08
%IW19	A_IN09	IN09_MODE	Diagnose zu IN09
%IW20	A_IN10	IN10_MODE	Diagnose zu IN10
%IW21	A_IN11	IN11_MODE	Diagnose zu IN11
%IW22	A_IN12	IN12_MODE	Diagnose zu IN12
%IW23	A_IN13	IN13_MODE	Diagnose zu IN13
%IW24	A_IN14	IN14_MODE	Diagnose zu IN14
%IW25	A_IN15	IN15_MODE	Diagnose zu IN15
%IW26	CURR12		Strommessung an OUT12
%IW27	CURR13		Strommessung an OUT13
%IW28	CURR14		Strommessung an OUT14
%IW29	CURR15		Strommessung an OUT15
%IW30	CURR16		Strommessung an OUT16
%IW31	CURR17		Strommessung an OUT17
%IW37	SUPPLY_VOLTAGE		
%IW38	REFERENCE_VOLTAGE		

Ausgänge: Adressbelegung

20100

Abkürzungen → Kapitel *Hinweise zur Anschlussbelegung* (→ Seite [24](#))Betriebsarten der Ein- und Ausgänge → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite [205](#))

IEC-Adresse	Symbolische Adresse	Konfiguration mit ...	Hinweis
%QX0.0	OUT00		Binär-Ausgang PWM-Ausgang
%QX0.1	OUT01		Binär-Ausgang PWM-Ausgang
%QX0.2	OUT02		Binär-Ausgang PWM-Ausgang
%QX0.3	OUT03		Binär-Ausgang PWM-Ausgang
%QX0.4	OUT04		Binär-Ausgang PWM-Ausgang
%QX0.5	OUT05		Binär-Ausgang PWM-Ausgang
%QX0.6	OUT06		Binär-Ausgang PWM-Ausgang
%QX0.7	OUT07		Binär-Ausgang PWM-Ausgang
%QX0.8	OUT08		Binär-Ausgang
%QX0.9	OUT09		Binär-Ausgang
%QX0.10	OUT10		Binär-Ausgang
%QX0.11	OUT11		Binär-Ausgang
%QX0.12	OUT12		Binär-Ausgang
%QX0.13	OUT13		Binär-Ausgang
%QX0.14	OUT14		Binär-Ausgang
%QX0.15	OUT15		Binär-Ausgang
%QX0.16	OUT16		Binär-Ausgang
%QX0.17	OUT17		Binär-Ausgang
%QB3	IN00_MODE		Konfiguration für IN00
%QB4	IN01_MODE		Konfiguration für IN01
%QB5	IN02_MODE		Konfiguration für IN02
%QB6	IN03_MODE		Konfiguration für IN03
%QB7	IN04_MODE		Konfiguration für IN04
%QB8	IN05_MODE		Konfiguration für IN05
%QB9	IN06_MODE		Konfiguration für IN06
%QB10	IN07_MODE		Konfiguration für IN07
%QB11	IN08_MODE		Konfiguration für IN08
%QB12	IN09_MODE		Konfiguration für IN09
%QB13	IN10_MODE		Konfiguration für IN10
%QB14	IN11_MODE		Konfiguration für IN11
%QB15	IN12_MODE		Konfiguration für IN12
%QB16	IN13_MODE		Konfiguration für IN13

IEC-Adresse	Symbolische Adresse	Konfiguration mit ...	Hinweis
%QB17	IN14_MODE		Konfiguration für IN14
%QB18	IN15_MODE		Konfiguration für IN15
%QB19	A_IN16_MODE		Konfiguration für A_IN16
%QB20	A_IN17_MODE		Konfiguration für A_IN17
%QB21	A_IN18_MODE		Konfiguration für A_IN18
%QB22	A_IN19_MODE		Konfiguration für A_IN19
%QB23	A_IN20_MODE		Konfiguration für A_IN20
%QB24	A_IN21_MODE		Konfiguration für A_IN21
%QB25	A_IN22_MODE		Konfiguration für A_IN22
%QB26	A_IN23_MODE		Konfiguration für A_IN23
%QB27	REFERENCE_VOLTAGE_5		
%QB28	REFERENCE_VOLTAGE_10		

7.2.2 Mögliche Betriebsarten Ein-/Ausgänge

Inhalt	
Eingänge: Betriebsarten.....	206
Ausgänge: Betriebsarten.....	207

2386

Eingänge: Betriebsarten

20103

Mögliche Konfigurations-Kombinationen (wo zulässig) entstehen durch Addition der Konfigurations-Werte.

 = diese Konfiguration ist voreingestellt

Eingänge	mögliche Betriebsart	einstellen mit ...	FB-Eingang	Wert	
				dez	hex
IN00...07	IN_DIGITAL_H plus	INxx_MODE	--	1	01
	IN_DIAGNOSTIC bei IN_DIGITAL_H	INxx_MODE	--	64	40
IN08...11	IN_DIGITAL_H plus	INxx_MODE	--	1	01
	IN_DIAGNOSTIC bei IN_DIGITAL_H	INxx_MODE	--	64	40
	IN_FAST für Interrupt-FBs	INxx_MODE	--	128	80
	Frequenzmessung 0...30 000 Hz	FB FREQUENCY			
	Periodendauermessung 0,1...5 000 Hz	FB PERIOD			
	Periodendauer- und Ratiomessung 0,1...5 000 Hz	FB PERIOD_RATIO			
	Zähler 0...50 Hz	FB FAST_COUNT			
	Drehgeber erfassen 0...30 000 Hz	FB INC_ENCODER			
IN12...15	IN_DIGITAL_H plus	INxx_MODE	--	1	01
	IN_DIGITAL_L minus	INxx_MODE	--	2	02
	IN_DIAGNOSTIC bei IN_DIGITAL_H	INxx_MODE	--	64	40
IN16...23	IN_DIGITAL_H plus	A_INxx_MODE	--	1	01
A_IN16...23	IN_CURRENT 0...20 000 µA	A_INxx_MODE	--	4	04
A_IN16...23	IN_VOLTAGE10 0...10 000 mV	A_INxx_MODE	--	8	08
A_IN16...23	IN_VOLTAGE32 0...32 000 mV	A_INxx_MODE	--	16	10
A_IN16...23	IN_RATIO32 0...1 000 %	A_INxx_MODE	--	32	20
IN16...23	IN_DIAGNOSTIC bei IN_DIGITAL_H	A_INxx_MODE	--	64	40

Betriebsarten mit folgendem Funktionsbaustein einstellen:

FAST_COUNT (→ Seite 134)	Zählerbaustein für schnelle Eingangsimpulse
FREQUENCY (→ Seite 135)	misst die Frequenz des am gewählten Kanal ankommenden Signals
INC_ENCODER (→ Seite 136)	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
PERIOD (→ Seite 138)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_RATIO (→ Seite 140)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] über die angegebenen Perioden am angegebenen Kanal. Zusätzlich wird das Puls-/Periodenverhältnis in [%] angegeben.

Ausgänge: Betriebsarten

20104

 = diese Konfiguration ist voreingestellt

Ausgänge	mögliche Betriebsart		einstellen mit ...	FB-Eingang	Wert	
					dez	hex
OUT00...07	Binärer Ausgang	plus-schaltend	--	--	--	--
	analoger Ausgang mit Pulsweitenmodulation		PWM1000			
OUT08...11	Binärer Ausgang	plus-schaltend	--	--	--	--
OUT12...17	Binärer Ausgang mit Diagnose	plus-schaltend	--	--	--	--

Betriebsarten mit folgendem Funktionsbaustein einstellen:

PWM1000 (→ Seite 148)	initialisiert und parametrisiert einen PWM-fähigen Ausgangskanal das Puls-Pausen-Verhältnis kann in 1 %-Schritten angegeben werden
---	---

Ausgänge: zulässige Betriebsarten

20105

Betriebsart		OUT00	OUT01	OUT02	OUT03	OUT04	OUT05	OUT06	OUT07
OUT_DIGITAL_H	plus	X	X	X	X	X	X	X	X
OUT_CURRENT_RANGE	4 A	X	X	X	X	X	X	X	X
PWM		X	X	X	X	X	X	X	X
OUT_OVERLOAD_PROTECTION		X	X	X	X	X	X	X	X

Betriebsart		OUT08	OUT09	OUT10	OUT11	--	--	--	--
OUT_DIGITAL_H	plus	X	X	X	X	--	--	--	--
OUT_CURRENT_RANGE	4 A	X	X	X	X	--	--	--	--
OUT_OVERLOAD_PROTECTION		X	X	X	X	--	--	--	--

Betriebsart		OUT12	OUT13	OUT14	OUT15	OUT16	OUT17	--	--
OUT_DIGITAL_H	plus	X	X	X	X	X	X	--	--
OUT_CURRENT_RANGE	10 A	X	X	X	X	X	X	--	--
OUT_DIAGNOSTIC		X	X	X	X	X	X	--	--
OUT_OVERLOAD_PROTECTION		X	X	X	X	X	X	--	--

7.3 Fehler-Tabellen

Inhalt

Fehlermerker	208
Fehler: CAN / CANopen	208

19606

7.3.1 Fehlermerker

19608

→ Kapitel *Systemmerker* (→ Seite [194](#))

7.3.2 Fehler: CAN / CANopen

19610
19604

→ Systemhandbuch "Know-How ecomatmobile"

→ Kapitel *CAN / CANopen: Fehler und Fehlerbehandlung*

EMCY-Codes: CANx

13094

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

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

EMCY-Codes: E/As, System

2671

Die folgenden EMCY-Meldungen werden automatisch versendet, wenn der FB *CANx_MASTER_EMCY_HANDLER* (→ Seite [81](#)) zyklisch aufgerufen wird.

EMCY-Code Objekt 0x1003		Objekt 0x1001	herstellerspezifische Informationen					Beschreibung
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	
00	21	03	10					Diagnose Analog-Stromeingänge
00	31	05						Klemmenspannung VBBO/VBBS
00	61	11						Speicherfehler

8 Begriffe und Abkürzungen

A

Adresse

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

Anleitung

Übergeordnetes Wort für einen der folgenden Begriffe:

Montageanleitung, Datenblatt, Benutzerinformation, Bedienungsanleitung, Gerätehandbuch, Installationsanleitung, Onlinehilfe, Systemhandbuch, Programmierhandbuch, usw.

Anwendungsprogramm

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

Architektur

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

B

Baud

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

1 MBd = 1024 x 1024 Bd = 1 048 576 Bd

Bestimmungsgemäße Verwendung

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

Bootloader

Im Auslieferungszustand enthalten **ecomatmobile**-Controller nur den Bootloader.

Der Bootloader ist ein Startprogramm, mit dem das Laufzeitsystem und das Anwendungsprogramm auf dem Gerät nachgeladen werden können.

Der Bootloader enthält Grundroutinen...

- zur Kommunikation der Hardware-Module untereinander,
- zum Nachladen des Laufzeitsystems.

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

Bus

Serielle Datenübertragung mehrerer Teilnehmer an derselben Leitung.

C

CAN

CAN = **C**ontroller **A**rea **N**etwork

CAN gilt als Feldbussystem für größere Datenmengen, das prioritätengesteuert arbeitet. Es gibt mehrere höhere Protokolle, die auf CAN aufsetzen, z. B. 'CANopen' oder 'J1939'.

CAN-Stack

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

CiA

CiA = CAN in Automation e.V.

Anwender- und Herstellerorganisation in Erlangen, Deutschland. Definitions- und Kontrollorgan für das CANopen-Protokoll.

Homepage → www.can-cia.org

CiA DS 304

DS = **D**raft **S**tandard

CANopen-Geräteprofil für sichere Kommunikation

CiA DS 401

DS = **D**raft **S**tandard

CANopen-Geräteprofil für digitale und analoge E/A-Baugruppen

CiA DS 402

DS = **D**raft **S**tandard

CANopen-Geräteprofil für Antriebe

CiA DS 403

DS = **D**raft **S**tandard

CANopen-Geräteprofil für Bediengeräte

CiA DS 404

DS = **D**raft **S**tandard

CANopen-Geräteprofil für Messtechnik und Regler

CiA DS 405

DS = **D**raft **S**tandard

CANopen-Spezifikation der Schnittstelle zu programmierbaren Steuerungen (IEC 61131-3)

CiA DS 406

DS = **D**raft **S**tandard

CANopen-Geräteprofil für Drehgeber / Encoder

CiA DS 407

DS = **D**raft **S**tandard

CANopen-Anwendungsprofil für den öffentlichen Nahverkehr

COB-ID

COB = **C**ommunication **O**bject = Kommunikationsobjekt

ID = **I**dentifier = Kennung

ID eines CANopen-Kommunikationsobjekts

Entspricht dem Identifier der CAN-Nachricht, mit der das Kommunikationsobjekt über den CAN-Bus gesendet wird.

CODESYS

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

Homepage → www.codesys.com

CSV-Datei

CSV = **C**omma **S**eparated **V**alues (auch: **C**haracter **S**eparated **V**alues)

Eine CSV-Datei ist eine Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten. Die Dateinamen-Erweiterung lautet .csv.

Beispiel: Quell-Tabelle mit Zahlenwerten:

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

Daraus entsteht folgende CSV-Datei:

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

D

Datentyp

Abhängig vom Datentyp können unterschiedlich große Werte gespeichert werden.

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

DC

Direct **C**urrent = Gleichstrom

Diagnose

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

Je nach Gerät können auch die Ein- und Ausgänge auf einwandfreie Funktion überwacht werden:

- Drahtbruch,
- Kurzschluss,
- Wert außerhalb des Sollbereichs.

Zur Diagnose können Konfigurations-Dateien herangezogen werden, die während des "normalen" Betriebs des Gerätes erzeugt wurden.

Der korrekte Start der Systemkomponenten wird während der Initialisierungs- und Startphase überwacht.

Zur weiteren Diagnose können auch Selbsttests durchgeführt werden.

Dither

to dither (engl.) = schwanken / zittern.

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

DLC

Data **L**ength **C**ode = bei CANopen die Anzahl der Daten-Bytes in einer Nachricht.

Für →SDO: DLC = 8

DRAM

DRAM = **D**ynamic **R**andom **A**ccess **M**emory.

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

DTC

DTC = **D**iagnostic **T**rouble **C**ode = Fehler-Code

Beim Protokoll J1939 werden Störungen und Fehler über zugeordnete Nummern – den DTCs – verwaltet und gemeldet.

E

ECU

(1) **E**lectronic **C**ontrol **U**nit = Steuergerät oder Mikrocontroller

(2) **E**ngine **C**ontrol **U**nit = Steuergerät eines Motors

EDS-Datei

EDS = **E**lectronic **D**ata **S**heet = elektronisch hinterlegtes Datenblatt, z.B. für:

- Datei für das Objektverzeichnis im CANopen-Master,
- CANopen-Gerätebeschreibungen.

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

Embedded Software

System-Software, Grundprogramm im Gerät, praktisch das →Laufzeitsystem.

Die Firmware stellt die Verbindung her zwischen der Hardware des Gerätes und dem Anwendungsprogramm. Die Firmware wird vom Hersteller der Steuerung als Teil des Systems geliefert und kann vom Anwender nicht verändert werden.

EMCY

Abkürzung für Emergency (engl.) = Notfall

Nachricht im CANopen-Protokoll, mit der Fehler gemeldet werden.

EMV

EMV = **E**lektro-**M**agnetische **V**erträglichkeit.

Gemäß der EG-Richtlinie (2004/108/EG) zur elektromagnetischen Verträglichkeit (kurz EMV-Richtlinie) werden Anforderungen an die Fähigkeit von elektrischen und elektronischen Apparaten, Anlagen, Systemen oder Bauteilen gestellt, in der vorhandenen elektromagnetischen Umwelt zufriedenstellend zu arbeiten. Die Geräte dürfen ihre Umgebung nicht stören und dürfen sich von äußerlichen elektromagnetischen Störungen nicht ungünstig beeinflussen lassen.

Ethernet

Ethernet ist eine weit verbreitete, herstellernerneutrale Netzwerktechnologie, mit der Daten mit einer Geschwindigkeit von 10 bis 10 000 Millionen Bit pro Sekunde (Mbps) übertragen werden können. Ethernet gehört zu der Familie der sogenannten „bestmöglichen Datenübermittlung“ auf einem nicht exklusiven Übertragungsmedium. 1972 entwickelt, wurde das Konzept 1985 als IEEE 802.3 spezifiziert.

EUC

EUC = **E**quipment **U**nder **C**ontrol (kontrollierte Einrichtung).

EUC ist eine Einrichtung, Maschine, Gerät oder Anlage, verwendet zur Fertigung, Stoffumformung, zum Transport, zu medizinischen oder anderen Tätigkeiten (→ IEC 61508-4, Abschnitt 3.2.3). Das EUC umfasst also alle Einrichtungen, Maschinen, Geräte oder Anlagen, die →Gefährdungen verursachen können und für die sicherheitsgerichtete Systeme erforderlich sind.

Falls eine vernünftigerweise vorhersehbare Aktivität oder Inaktivität zu durch das EUC verursachten Gefährdungen mit unverträglichem Risiko führt, sind Sicherheitsfunktionen erforderlich, um einen sicheren Zustand für das EUC zu erreichen oder aufrecht zu erhalten. Diese Sicherheitsfunktionen werden durch ein oder mehrere sicherheitsgerichtete Systeme ausgeführt.

F

Fehlanwendung

Das ist die Verwendung eines Produkts in einer Weise, die vom Konstrukteur nicht vorgesehen ist. Eine Fehlanwendung führt meist zu einer →Gefährdung von Personen oder Sachen.

Vor vernünftigerweise, vorhersehbaren Fehlanwendungen muss der Hersteller des Produkts in seinen Benutzerinformationen warnen.

FiFo

FIFO (**F**irst **I**n, **F**irst **O**ut) = Arbeitsweise des Stapelspeichers: Das Datenpaket, das zuerst in den Stapelspeicher geschrieben wurde, wird auch als erstes gelesen. Pro Identifier steht ein solcher Zwischenspeicher (als Warteschlange) zur Verfügung.

Flash-Speicher

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

Vorteile von Flash-Speicher

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

Nachteile von Flash-Speicher

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

FRAM

FRAM, oder auch FeRAM, bedeutet **F**erroelectric **R**andom **A**ccess **M**emory. Der Speicher- und Löschvorgang erfolgt durch eine Polarisationsänderung in einer ferroelektrischen Schicht.

Vorteile von FRAM gegenüber herkömmlichen Festwertspeichern:

- nicht flüchtig,
- kompatibel zu gängigen EEPROMs, jedoch:
- Zugriffszeit ca. 100 ns,
- fast unbegrenzt viele Zugriffszyklen möglich.

H

Heartbeat

Heartbeat (engl.) = Herzschlag.

Die Teilnehmer senden regelmäßig kurze Signale. So können die anderen Teilnehmer prüfen, ob ein Teilnehmer ausgefallen ist.

HMI

HMI = **H**uman **M**achine **I**nterface = Mensch-Maschine-Schnittstelle

I

ID – Identifier

ID = **I**dentifier = Kennung

Name zur Unterscheidung der an einem System angeschlossenen Geräte / Teilnehmer oder der zwischen den Teilnehmern ausgetauschten Nachrichtenpakete.

IEC 61131

Norm: Grundlagen Speicherprogrammierbarer Steuerungen

- Teil 1: Allgemeine Informationen
- Teil 2: Betriebsmittelanforderungen und Prüfungen
- Teil 3: Programmiersprachen
- Teil 5: Kommunikation
- Teil 7: Fuzzy-Control-Programmierung

IEC-User-Zyklus

IEC-User-Zyklus = SPS-Zyklus im CODESYS-Anwendungsprogramm.

IP-Adresse

IP = Internet Protocol = 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 Anhängfahrzeugen

- Teil 1: Bit-Übertragungsschicht und Sicherungsschicht
- Teil 2: Anwendungsschicht für die Bremsausrüstung
- Teil 3: Anwendungsschicht für andere als die Bremsausrüstung
- Teil 4: Diagnose

ISO 16845

Norm: Straßenfahrzeuge – Steuergerätenetz (CAN) – Prüfplan zu Konformität

J

J1939

→ SAE J1939

K

Klemme 15

Klemme 15 ist in Fahrzeugen die vom Zündschloss geschaltete Plusleitung.

L

Laufzeitsystem

Grundprogramm im Gerät, stellt die Verbindung her zwischen der Hardware des Gerätes und dem Anwendungsprogramm.

→ Kapitel *Software-Module für das Gerät* (→ Seite [28](#))

LED

LED = **L**ight **E**mitting **D**iode = Licht aussendende Diode.

Leuchtdiode, auch Luminiszenzdiode, ein elektronisches Element mit hoher, farbiger Leuchtkraft auf kleinem Volumen bei vernachlässigbarer Verlustleistung.

Link

Ein Link ist ein Querverweis zu einer anderen Stelle im Dokument oder auf ein externes Dokument.

LSB

Least **S**ignificant **B**it/Byte = Niederwertigstes Bit/Byte in einer Reihe von Bit/Bytes.

M

MAC-ID

MAC = **M**anufacturer's **A**ddress **C**ode

= Hersteller-Seriennummer.

→ ID = **I**dentifier = 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 [214](#))

MRAM

MRAM = **M**agnetoresistive **R**andom **A**ccess **M**emory

Die Informationen werden mit magnetischen Ladungselementen gespeichert. Dabei wird die Eigenschaft bestimmter Materialien ausgenutzt, die ihren elektrischen Widerstand unter dem Einfluss magnetischer Felder ändern.

Vorteile von MRAM gegenüber herkömmlichen Festwertspeichern:

- nicht flüchtig (wie FRAM), jedoch:
- Zugriffszeit nur ca. 35 ns,
- unbegrenzt viele Zugriffszyklen möglich.

MSB

Most **S**ignificant **B**it/Byte = Höchstwertiges Bit/Byte einer Reihe von Bits/Bytes.

N

NMT

NMT = **N**etwork **M**anagement = Netzwerk-Verwaltung (hier: im CANopen-Protokoll).
Der NMT-Master steuert die Betriebszustände der NMT-Slaves.

Node

Node (engl.) = Knoten. Damit ist ein Teilnehmer im Netzwerk gemeint.

Node Guarding

Node (engl.) = Knoten, hier: Netzwerkteilnehmer

Guarding (engl.) = Schutz

Parametrierbare, zyklische Überwachung von jedem entsprechend konfigurierten →Slave. Der →Master prüft, ob die Slaves rechtzeitig antworten. Die Slaves prüfen, ob der Master regelmäßig anfragt. Somit können ausgefallene Netzwerkteilnehmer schnell erkannt und gemeldet werden.

O

Obj / Objekt

Oberbegriff für austauschbare Daten / Botschaften innerhalb des CANopen-Netzwerks.

Objektverzeichnis

Das **Objektverzeichnis** OBV enthält alle CANopen-Kommunikationsparameter eines Gerätes, sowie gerätespezifische Parameter und Daten.

OBV

Das **Objektverzeichnis** OBV enthält alle CANopen-Kommunikationsparameter eines Gerätes, sowie gerätespezifische Parameter und Daten.

OPC

OPC = **O**LE for **P**rocess **C**ontrol = Objektverknüpfung und -einbettung für Prozesssteuerung
Standardisierte Software-Schnittstelle zur herstellerunabhängigen Kommunikation in der Automatisierungstechnik

OPC-Client (z.B. Gerät zum Parametrieren oder Programmieren) meldet sich nach dem Anschließen am OPC-Server (z.B. Automatisierungsgerät) automatisch bei diesem an und kommuniziert mit ihm.

operational

Operational (engl.) = betriebsbereit

Betriebszustand eines CANopen-Teilnehmers. In diesem Modus können →SDOs, →NMT-Kommandos und →PDOs übertragen werden.

P

PC-Karte

→ PCMCIA-Karte

PCMCIA-Karte

PCMCIA = Personal Computer Memory Card International Association, ein Standard für Erweiterungskarten mobiler Computer.

Seit der Einführung des Cardbus-Standards 1995 werden PCMCIA-Karten auch als PC-Karte (engl.: PC Card) bezeichnet.

PDM

PDM = **P**rocess and **D**ialog **M**odule = **P**rozess- und **D**ialog-**M**onitor.

Gerät zur Kommunikation des Bedieners mit der Maschine / Anlage.

PDO

PDO = **P**rocess **D**ata **O**bject = Nachrichten-Objekt mit Prozessdaten.

Die zeitkritischen Prozessdaten werden mit Hilfe der "Process Data Objects" (PDOs) übertragen. Die PDOs können beliebig zwischen den einzelnen Knoten ausgetauscht werden (PDO-Linking).

Zusätzlich wird festgelegt, ob der Datenaustausch ereignisgesteuert (asynchron) oder synchronisiert erfolgen soll. Je nach der Art der zu übertragenden Daten kann die richtige Wahl der Übertragungsart zu einer erheblichen Entlastung des →CAN-Bus führen.

Dem Protokoll entsprechend, sind diese Dienste nicht bestätigte Dienste: es gibt keine Kontrolle, ob die Nachricht auch beim Empfänger ankommt. Netzwerkvariablen-Austausch entspricht einer "1-zu-n-Verbindung" (1 Sender zu n Empfängern).

PDU

PDU = **P**rotocol **D**ata **U**nit = Protokoll-Daten-Einheit.

Die PDU ist ein Begriff aus dem →CAN-Protokoll →SAE J1939. Sie bezeichnet einen Bestandteil der Ziel- oder Quelladresse.

PES

Programable **e**lectronic **s**ystem = Programmierbares elektronisches System ...

- zur Steuerung, zum Schutz oder zur Überwachung,
- auf der Basis einer oder mehrerer programmierbarer Geräte,
- einschließlich aller Elemente dieses Systems, wie Ein- und Ausgabegeräte.

PGN

PGN = **P**arameter **G**roup **N**umber = Parameter-Gruppennummer

PGN = 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

Der PID-Regler (proportional–integral–derivative controller) besteht aus folgenden Anteilen:

- P = Proportional-Anteil
- I = Integral-Anteil
- D = Differential-Anteil (jedoch nicht beim Controller CR04nn, CR253n).

Piktogramm

Piktogramme sind bildhafte Symbole, die eine Information durch vereinfachte grafische Darstellung vermitteln (→ Kapitel *Was bedeuten die Symbole und Formatierungen?* (→ Seite [7](#))).

Pre-Op

Pre-Op = PRE-OPERATIONAL mode (engl.) = Zustand vor 'betriebsbereit'.

Betriebszustand eines CANopen-Teilnehmers. Nach dem Einschalten der Versorgungsspannung geht jeder Teilnehmer automatisch in diesem Zustand. Im CANopen-Netz können in diesem Modus nur →SDOs und →NMT-Kommandos übertragen werden, jedoch keine Prozessdaten.

Prozessabbild

Mit Prozessabbild bezeichnet man den Zustand der Ein- und Ausgänge, mit denen die SPS innerhalb eines →Zyklus arbeitet.

- Am Zyklus-Beginn liest die SPS die Zustände aller Eingänge in das Prozessabbild ein. Während des Zyklus kann die SPS Änderungen an den Eingängen nicht erkennen.
- Im Laufe des Zyklus werden die Ausgänge nur virtuell (im Prozessabbild) geändert.
- Am Zyklus-Ende schreibt die SPS die virtuellen Ausgangszustände auf die realen Ausgänge.

PWM

PWM = Puls-Weiten-Modulation

Bei dem PWM-Ausgangssignal handelt es sich um ein getaktetes Signal zwischen GND und Versorgungsspannung.

Innerhalb einer festen Periode (PWM-Frequenz) wird das Puls-/Pausenverhältnis variiert. Durch die angeschlossene Last stellt sich je nach Puls-/Pausenverhältnis der entsprechende Effektivstrom ein.

R

ratiometrisch

Ratio (lat.) = Verhältnis

Messungen können auch ratiometrisch erfolgen = Verhältnismessung. Wenn das Ausgangssignal eines Sensors proportional zu seiner Versorgungsspannung ist, kann durch ratiometrische Messung (= Messung im Verhältnis zur Versorgung) der Einfluss von Schwankungen der Versorgung reduziert, im Idealfall sogar beseitigt werden.

→ Analogeingang

RAW-CAN

RAW-CAN bezeichnet das reine →CAN-Protokoll, das ohne ein zusätzliches Kommunikationsprotokoll auf dem CAN-Bus (auf ISO/OSI-Schicht 2) arbeitet. Das CAN-Protokoll ist international nach →ISO 11898-1 definiert und garantiert zusätzlich in →ISO 16845 die Austauschbarkeit von CAN-Chips.

remanent

Remanente Daten sind gegen Datenverlust bei Spannungsausfall geschützt.

Z.B. kopiert das →Laufzeitsystem die remanenten Daten automatisch in einen →Flash-Speicher, sobald die Spannungsversorgung unter einen kritischen Wert sinkt. Bei Wiederkehr der Spannungsversorgung lädt das Laufzeitsystem die remanenten Daten zurück in den Arbeitsspeicher. Dagegen sind die Daten im Arbeitsspeicher einer Steuerung flüchtig und bei Unterbrechung der Spannungsversorgung normalerweise verloren.

ro

ro = read only (engl.) = nur lesen

Unidirektionale Datenübertragung: Daten können nur gelesen werden, jedoch nicht verändert.

RTC

RTC = **Real Time Clock** = Echtzeituhr

Liefert (batteriegepuffert) aktuell Datum und Uhrzeit. Häufiger Einsatz beim Speichern von Fehlermeldungsprotokollen.

rw

rw = read/write (engl.) = lesen und schreiben

Bidirektionale Datenübertragung: Daten können sowohl gelesen als auch verändert werden.

S

SAE J1939

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

Norm: Recommended Practice for a Serial Control and Communications Vehicle Network

- Teil 2: Agricultural and Forestry Off-Road Machinery Control and Communication Network
- Teil 3: On Board Diagnostics Implementation Guide
- Teil 5: Marine Stern Drive and Inboard Spark-Ignition Engine On-Board Diagnostics Implementation Guide

- Teil 11: Physical Layer – 250 kBits/s, Shielded Twisted Pair
- Teil 13: Off-Board Diagnostic Connector
- Teil 15: Reduced Physical Layer, 250 kBits/s, Un-Shielded Twisted Pair (UTP)
- Teil 21: Data Link Layer
- Teil 31: Network Layer
- Teil 71: Vehicle Application Layer
- Teil 73: Application Layer – Diagnostics
- Teil 81: Network Management Protocol

SD-Card

Eine SD Memory Card (Kurzform für **Secure Digital Memory Card**; deutsch: Sichere digitale Speicherkarte) ist ein digitales Speichermedium, das nach dem Prinzip der →Flash-Speicherung arbeitet.

SDO

SDO = **S**ervice **D**ata **O**bject = Nachrichten-Objekt mit Servicedaten.

Das SDO dient dem Zugriff auf Objekte in einem CANopen-Objektverzeichnis. Dabei fordern 'Clients' die gewünschten Daten von 'Servern' an. Die SDOs bestehen immer aus 8 Bytes.

Beispiele:

- Automatische Konfiguration aller →Slaves über SDOs beim Systemstart.
- Auslesen der Fehlernachrichten aus dem →Objektverzeichnis.

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

Selbsttest

Testprogramm, das aktiv Komponenten oder Geräte testet. Das Programm wird durch den Anwender gestartet und dauert eine gewisse Zeit. Das Ergebnis davon ist ein Testprotokoll (Log-Datei), aus dem entnommen werden kann, was getestet wurde und ob das Ergebnis positiv oder negativ ist.

Slave

Passiver Teilnehmer am Bus, antwortet nur auf Anfrage des →Masters. Slaves haben im Bus eine eindeutige →Adresse.

Steuerungskonfiguration

Bestandteil der CODESYS-Bedienoberfläche.

- ▶ Programmierer teilt dem Programmiersystem mit, welche Hardware programmiert werden soll.
- > CODESYS lädt die zugehörigen Bibliotheken.
- > Lesen und schreiben der Peripherie-Zustände (Ein-/Ausgänge) ist möglich.

stopped

stopped (engl.) = angehalten

Betriebszustand eines CANopen-Teilnehmers. In diesem Modus werden nur →NMT-Kommandos übertragen.

Symbole

Piktogramme sind bildhafte Symbole, die eine Information durch vereinfachte grafische Darstellung vermitteln (→ Kapitel *Was bedeuten die Symbole und Formatierungen?* (→ Seite [7](#))).

Systemvariable

Variable, auf die via IEC-Adresse oder Symbolname aus der SPS zugegriffen werden kann.

T

Target

Das Target enthält für CODESYS die Hardware-Beschreibung des Zielgeräts, z.B.: Ein- und Ausgänge, Speicher, Dateiablageorte.

Entspricht einem elektronischen Datenblatt.

TCP

Das **T**ransmission **C**ontrol **P**rotocol ist Teil der Protokollfamilie TCP/IP. Jede TCP/IP-Datenverbindung hat einen Sender und einen Empfänger. Dieses Prinzip ist eine verbindungsorientierte Datenübertragung. In der TCP/IP-Protokollfamilie übernimmt TCP als verbindungsorientiertes Protokoll die Aufgabe der Datensicherheit, der Datenflusssteuerung und ergreift Maßnahmen bei einem Datenverlust. (vgl.: →UDP)

Template

Template (englisch = Schablone) ist eine Vorlage, die mit Inhalten gefüllt werden kann.

Hier: Eine Struktur von vorkonfigurierten Software-Elementen als Basis für ein Anwendungsprogramm.

U

UDP

UDP (**U**ser **D**atagram **P**rotocol) ist ein minimales, verbindungsloses Netzprotokoll, das zur Transportschicht der Internetprotokollfamilie gehört. Aufgabe von UDP ist es, Daten, die über das Internet übertragen werden, der richtigen Anwendung zukommen zu lassen.

Derzeit sind Netzwerkvariablen auf Basis von →CAN und UDP implementiert. Die Variablenwerte werden dabei auf der Basis von Broadcast-Nachrichten automatisch ausgetauscht. In UDP sind diese als Broadcast-Telegramme realisiert, in CAN als →PDOs.

Dem Protokoll entsprechend, sind diese Dienste nicht bestätigte Dienste: es gibt keine Kontrolle, ob die Nachricht auch beim Empfänger ankommt. Netzwerkvariablen-Austausch entspricht einer "1-zu-n-Verbindung" (1 Sender zu n Empfängern).

V

Verwendung, bestimmungsgemäß

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

W

Watchdog

Der Begriff Watchdog (englisch; Wachhund) wird verallgemeinert für eine Komponente eines Systems verwendet, die die Funktion anderer Komponenten beobachtet. Wird dabei eine mögliche Fehlfunktion 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

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.

9 Index

A

Adressbelegung	200
Adressbelegung der Ausgänge	203
Adressbelegung Ein-/Ausgänge	200
Adressbelegung und E/A-Betriebsarten	200
Adresse	209
Analogeingänge	
Konfiguration und Diagnose	51
Analog-Eingänge	17
Analogwerte anpassen	130
Angaben zum Gerät	12
Anhang	194
Anlaufverhalten der Steuerung	11
Anleitung	209
Anschlussbelegung	24
Anwendungsprogramm	29, 209
Anwendungsprogramm erstellen	33
Architektur	209
Ausgänge	
Adressbelegung	203
Betriebsarten	207
zulässige Betriebsarten	207
Ausgänge (Technologie)	21
Ausgänge konfigurieren	53
Ausgangsgruppe Q0Q1 (OUT00...07)	22
Ausgangsgruppe Q2 (OUT08...11)	22
Ausgangsgruppe Q3 (OUT12...17)	23

B

Baud	209
Bausteine	
analoge Werte anpassen	130
CAN Layer 2	63
CANopen SDOs	97
CANopen-Master	80
CANopen-Slave	89
Daten im Speicher sichern, lesen und wandeln	177
Datenzugriff und Datenprüfung	185
Eingangswerte verarbeiten	126
Hydraulikregelung	150
Interrupts verarbeiten	119
PWM-Funktionen	143
Regler	163
SAE J1939	102
serielle Schnittstelle	114
Software-Reset	172
SPS-Zyklus optimieren	119
Zählerfunktionen zur Frequenz- und Periodendauermessung	133
Zeit messen / setzen	174
Beachten!	10
Beispiel	
CANx_MASTER_SEND_EMERGENCY	83
CANx_MASTER_STATUS	87
CANx_SLAVE_SEND_EMERGENCY	93
CHECK_DATA	187
Initialisieren von CANx_RECEIVE_RANGE in 4 Zyklen	78
NORM (1)	132
NORM (2)	132
NORM_HYDRAULIC	162

Berechnung des RELOAD-Wertes	146
Berechnungsbeispiele RELOAD-Wert	146
Bestimmungsgemäße Verwendung	209
Betriebsarten der Ein-/Ausgänge	205
Betriebsmodi	39
Betriebszustände	35
Anwendungsprogramm nicht verfügbar	36
Anwendungsprogramm verfügbar	37
Bibliothek ifm_CAN1_EXT_Vxxyzz.LIB	60
Bibliothek ifm_CR0303_CANopenMaster_V04yynn.LIB	60
Bibliothek ifm_CR0303_CANopenSlave_V04yynn.LIB	61
Bibliothek ifm_CR0303_V05yzz.LIB	58
Bibliothek ifm_hydraulic_CR0303_V01yzz.LIB	62
Bibliothek ifm_J1939_x_Vxxyzz.LIB	61
Bibliotheken	30
Binärausgänge	
Konfiguration und Diagnose	53
Binär-Ausgänge	21
Binäreingänge	
Konfiguration und Diagnose	52
Binär-Eingänge	18
Bootloader	29, 209
Bootloader-Zustand	38
Boot-Projekt speichern	34
Bus	209

C

CAN	210
Schnittstellen und Protokolle	27
CAN / CANopen	
Fehler und Fehlerbehandlung	193
CAN1_BAUDRATE	64
CAN1_DOWNLOADID	65
CAN1_EXT	66
CAN1_EXT_ERRORHANDLER	67
CAN1_EXT_RECEIVE	68
CAN1_EXT_TRANSMIT	70
CAN2	71
CAN-Schnittstellen	27
CAN-Stack	210
CANx_ERRORHANDLER	72
CANx_EXT_RECEIVE_ALL	73
CANx_MASTER_EMCY_HANDLER	81
CANx_MASTER_SEND_EMERGENCY	82
CANx_MASTER_STATUS	84
CANx_RECEIVE	75
CANx_RECEIVE_RANGE	77
CANx_SDO_READ	98
CANx_SDO_WRITE	100
CANx_SLAVE_EMCY_HANDLER	90
CANx_SLAVE_NODEID	91
CANx_SLAVE_SEND_EMERGENCY	92
CANx_SLAVE_STATUS	94
CANx_TRANSMIT	79
CHECK_DATA	186
CiA	210
CiA DS 304	210
CiA DS 401	210
CiA DS 402	210
CiA DS 403	210

Index

CiA DS 404	210
CiA DS 405	210
CiA DS 406	210
CiA DS 407	210
COB-ID	211
CODESYS	211
CODESYS-Programmierhandbuch	6
Copyright	5
CSV-Datei	211

D

Dämpfung von Überschwüngen	163
Daten sichern, lesen und wandeln	177
Datentyp	211
Datenzugriff und Datenprüfung	185
DC	212
Debug	39
DEBUG-Modus	39
DELAY	164
Diagnose	192, 212
binäre Ausgänge (via Strommessung)	23
Kurzschluss (via Strommessung)	23
Leiterbruch (via Strommessung)	23
Überlast (via Strommessung)	23
Diagnose und Fehlerbehandlung	192
Dither	147, 212
DLC	212
DRAM	212
DTC	212

E

ECU	212
EDS-Datei	213
Eingänge	
Adressbelegung	201
Betriebsarten	206
Eingänge (Technologie)	17
Eingänge konfigurieren	50
Eingangsgruppe I0 (IN00...07)	19
Eingangsgruppe I1 (IN08...11 / FRQ00...03)	19
Eingangsgruppe I2 (IN12...15)	19
Eingangsgruppe I3 (A_IN16...23)	20
Eingangswerte verarbeiten	126
Einmalige Mechanismen	16
Einsatz als Binäreingänge	52
Einstellempfehlung	168, 170
Einstellregel	163
Einstellregel für einen Regler	163
Embedded Software	213
EMCY	213
EMCY-Codes	
CANx	208
E/As, System	208
EMV	213
Ethernet	213
EUC	213

F

FAST_COUNT	134
FB, FUN, PRG in CODESYS	32
FBs für PWM-Funktionen	54

Fehlanwendung	213
Fehler	192
CAN / CANopen	208
Fehlermerker	208
Fehler-Tabellen	208
FiFo	214
FLASHREAD	179
Flash-Speicher	177, 214
FLASH-Speicher	14
FLASHWRITE	180
FRAM	14, 214
FRAMREAD	182
FRAM-Speicher	177
FRAMWRITE	183
FREQUENCY	135
Funktionskonfiguration	48, 49
Funktionskonfiguration der Ein- und Ausgänge	49
Funktionskonfiguration, allgemein	48

G

Gerätekonfiguration	41
GET_IDENTITY	188
GLR	165

H

Hardware-Aufbau	13
Hardware-Beschreibung	12
Heartbeat	214
Hinweise zur Anschlussbelegung	24
Historie der Anleitung (CR030n)	9
HMI	214

I

ID – Identifier	214
IEC 61131	215
IEC-User-Zyklus	215
ifm weltweit • ifm worldwide • ifm à l'échelle internationale	231
ifm-Bausteine für das Gerät CR0303	63
ifm-Bibliotheken für das Gerät CR0303	57
ifm-Downloader nutzen	34
ifm-Funktionselemente	57
ifm-Maintenance-Tool nutzen	34
INC_ENCODER	136
INIT-Zustand (Reset)	38
INPUT_ANALOG	127
INPUT_CURRENT	128
INPUT_VOLTAGE	129
Installation verifizieren	43
Interrupt	40
Interruptverarbeitung	119
IP-Adresse	215
ISO 11898	215
ISO 11992	215
ISO 16845	215

J

J1939	215
J1939_x	103
J1939_x_GLOBAL_REQUEST	104
J1939_x_RECEIVE	106

Index

J1939_x_RESPONSE	108
J1939_x_SPECIFIC_REQUEST	110
J1939_x_TRANSMIT	112
JOYSTICK_0	151
JOYSTICK_1	154
JOYSTICK_2	158

K

Kein Laufzeitsystem	38
Klemme 15	215
Konfiguration der Ein- und Ausgänge (Voreinstellung)	48, 49
Konfigurationen	41

L

Laufzeitsystem	29, 216
Laufzeitsystem aktualisieren	43
Laufzeitsystem einrichten	41
Laufzeitsystem neu installieren	42
LED	25, 216
LED im Anwendungsprogramm steuern	25
Leistungsgrenzen des Geräts	40
Link	216
LSB	216

M

MAC-ID	216
manuell	178
Manuelle Datensicherung	178
Master	216
MEMCPY	184
MMI	216
Mögliche Betriebsarten Ein-/Ausgänge	205
MRAM	216
MSB	216

N

Nach Einschalten der Versorgungsspannung	15
Netzwerkvariablen	56
NMT	217
Node	217
Node Guarding	217
NORM	131
NORM_HYDRAULIC	161
Notizen • Notes • Notes	227

O

Obj / Objekt	217
Objektverzeichnis	217
OBV	217
OPC	217
operational	217

P

Parameter der internen Strukturen	86
PC-Karte	217
PCMCIA-Karte	218
PDM	218
PDO	218
PDU	218
PERIOD	138

PERIOD_RATIO	140
PES	218
PGN	218
PHASE	142
PID1	167
PID2	169
PID-Regler	167, 169, 218
Piktogramm	218
Piktogramme	7
Pre-Op	219
Prinzipschaltung	13
Programmierhinweise für CODESYS-Projekte	31
Programmiersystem einrichten	44
Programmiersystem manuell einrichten	44
Programmiersystem über Templates einrichten	47
Prozessabbild	219
PT1	171
PWM	144, 219
PWM1000	148
PWM-Ausgänge	21, 54
PWM-Dither	147
PWM-Frequenz	145

R

Rampenfunktion	147
rationometrisch	219
RAW-CAN	219
Reaktion auf System-Fehler	193
Reaktion im Fehlerfall	192
remanent	219
Reset	38
Retain-Variablen	56
ro	219
RTC	220
Run	38
RUN-Zustand	38
rw	220

S

SAE J1939	102, 220
Schnelle Eingänge	52
Schnittstellen-Beschreibung	26
SD-Card	220
SDO	220
Selbsttest	220
Serial Mode	39
SERIAL_MODE	39
SERIAL_PENDING	115
SERIAL_RX	116
SERIAL_SETUP	117
SERIAL_TX	118
Serielle Schnittstelle	26, 114
SET_DEBUG	189
SET_IDENTITY	190
SET_INTERRUPT_I	120
SET_INTERRUPT_XMS	123
SET_PASSWORD	191
Sicherheitshinweise	10
Sicherheitshinweise zu Reed-Relais	24, 50

Index

Slave.....	221
Slave-Informationen.....	87
SOFTRESET	173
Software.....	28
Software-Module für das Gerät	28
Software-Reset.....	172
Software-Steuerungskonfiguration.....	45
Speicher, verfügbar	14
Speicherarten zur Datensicherung.....	177
SRAM.....	14
Startvoraussetzung.....	13
Status-LED	25
Steuerungskonfiguration.....	45, 221
Steuerungskonfiguration aktivieren (z.B. CR0033).....	46
Stopp.....	38
stopped	221
STOP-Zustand.....	38
Struktur Emergency_Message	87
Struktur Knoten-Status	87
Symbole.....	221
Systembeschreibung	12
Systemmerker.....	194
24 Eingänge und 18 Ausgänge	199
CAN	195
Fehlermerker	196
LED (Standard-Seite).....	197
Spannungen	198
SYSTEM-STOP-Zustand.....	38
Systemvariable	221
Systemvariablen	48
Systemvoraussetzungen	12
Systemzeit	174

T

Target.....	221
Target einrichten.....	45
TCP	221
Template.....	221
Test.....	39
TEST-Betrieb	39
TIMER_READ.....	175
TIMER_READ_US.....	176

U

Über diese Anleitung	5
Überdurchschnittliche Belastungen.....	40
Übersicht	
Dokumentations-Module für ecomatmobile-Geräte	6
Überwachungs- und Sicherungsmechanismen.....	15
Überwachungskonzept.....	15
UDP	222

V

Variablen.....	55
Verfügbarer Speicher	14
Verfügbarkeit von PWM.....	54
Verhalten des Watchdog	40
Verwendung, bestimmungsgemäß.....	222
Vorkenntnisse	11

W

Was bedeuten die Symbole und Formatierungen?.....	7
Watchdog	40, 222
Welche Vorkenntnisse sind notwendig?.....	11
Wenn Laufzeitsystem / Anwendungsprogramm läuft.....	16
Wenn TEST-Pin nicht aktiv.....	16
Wie ist diese Dokumentation aufgebaut?.....	8
wo	222

Z

Zugriff auf die Strukturen zur Laufzeit der Anwendung.....	88
Zykluszeit.....	222
Zykluszeit beachten!	32

10 Notizen • Notes • Notes



© ifm electronic gmbh

www.ifm.com

© ifm electronic gmbh



www.ifm.com

© ifm electronic gmbh



www.ifm.com

© ifm electronic gmbh



www.ifm.com

11 ifm weltweit • ifm worldwide • ifm à l'échelle internationale

Stand: 2015-03-06

8310

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 (Shanghai) Co. Ltd. • 201203 Shanghai • Tel. +86 21 / 3813 4800
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 Győr • 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. • Chiba-shi, Chiba 261-7118 • Tel. +81 043-299-2070
MAL	ifm electronic Pte. Ltd. • 47100 Puchong Selangor • Tel. +603 8063 9522
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 Billingsstad • Tel. +47 66 / 98 33 50
NL	ifm electronic b.v. • 3843 GA Harderwijk • Tel. +31 341 / 438 438
P	ifm electronic s.a. • 4410-136 São Félix da Marinha • Tel. +351 223 / 71 71 08
PL	ifm electronic Sp. z o.o. • 40-106 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 • 41250 Göteborg • Tel. +46 31 / 750 23 00
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	SCM Alliances Co., Ltd. • Bangkok 10 400 • Tel. +66 02 615 4888
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.