ifm electronic

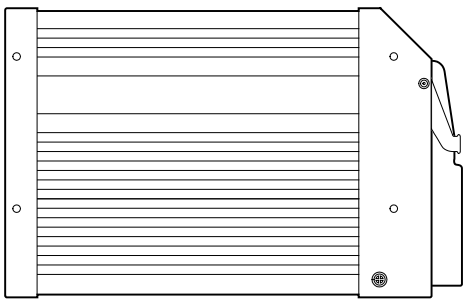
CE

ecomat mobile

Original-Programmierhandbuch
ClassicController
ecomat100
CR0020

Laufzeitsystem V06
CODESYS® V2.3

Deutsch



7391027_03_DE 2015-06-25

Inhaltsverzeichnis

1 Über diese Anleitung

Inhalt

202

1.1 Copyright

6088

1.2 Übersicht: Dokumentations-Module für ecomatmobile-Geräte

17405

Die Dokumentation für **ecomat*mobile***-Geräte besteht aus folgenden Modulen:

1.	Datenblatt
Inhalt:	Technische Daten in Tabellenform
Quelle:	→ www.ifm.com > Land wählen > [Datenblattsuche] > CR0020 > [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] > CR0020 > [Betriebsanleitungen]

3.	Programmierhandbuch + Online-Hilfe
Inhalt:	Beschreibung der Konfiguration und der Funktionen der Geräte-Software
Quelle:	→ www.ifm.com > Land wählen > [Datenblattsuche] > CR0020 > [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] > CR0020 > [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 **ecomat*mobile*** finden Sie auch hier:
→ **ecomat*mobile***-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 0x123 0b010	Dezimalzahl Hexadezimalzahl Binärzahl
[...]	Bezeichnung von Tasten, Schaltflächen oder Anzeigen

1.5 Wie ist diese Dokumentation aufgebaut?

204
1508

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

Und so finden Sie sich zurecht:

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

Bei Fehlfunktionen oder Unklarheiten setzen Sie sich bitte mit dem Hersteller in Verbindung:
→ 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 (CR0020,CR0505)

9185

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
2012-01-09	Speicherbausteine FRAMREAD, FRAMWRITE	vertauschte Parameter SRC, DST in der Tabelle "Zulässige Werte"
2012-07-16	Laufzeitsystem	Upgrade auf V06
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"
2014-07-31	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 *Systemmerker* • CAN / CANopen Fehler und Fehlerbehandlung: nur noch im Systemhandbuch "Know-How" • Fehlercodes, EMCY-Codes: nun im Anhang, Kapitel *Fehler-Tabellen*
2015-03-10	Verfügbarer Speicher	Darstellung verbessert
2015-05-22	FBs INPUT_ANALOG, INPUT_CURRENT, INPUT_VOLTAGE	zulässige Eingangskanäle
2015-05-26	FB J1939_x_GLOBAL_REQUEST	Beschreibung präzisiert
2015-06-10	diverse FBs	Beschreibung FB-Eingang CHANNEL korrigiert

2 Sicherheitshinweise

Inhalt

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!
  ⇨ Wiederanlaufsperre 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).

6827

- ► 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.

3 Systembeschreibung

Inhalt

975

3.1 Angaben zum Gerät

11568

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

- ClassicController: CR0020 ab Gerätestand AI
  Laufzeitsystem V06b

3.2 Hardware-Beschreibung

Inhalt

14081

3.2.1 Hardware-Aufbau

Inhalt

15332

Startvoraussetzung

19658

Das Gerät startet erst, wenn am Versorgungsanschluss VBBS (unter anderem Versorgung der Relais auf der Standardseite) und an Klemme 15 eine ausreichende Spannung anliegt.
Klemme 15 ist in Fahrzeugen die vom Zündschloss geschaltete Plusleitung.

Relais

19659

Der Controller verfügt über 2 interne Ausgangsrelais, die jeweils 12 Ausgänge von der Klemmenspannung VBBx trennen können (x = O / R).

Das Ausgangsrelais (Monitoring Relay) wird zweikanalig vom Mikrocontroller angesteuert. Dazu wird der eine Kanal durch eine UND-Verknüpfung des Watchdog-Signals (interne Mikrocontroller-Überwachung) und des System-Merkerbits RELAIS über einen Halbleiterschalter angesteuert. Der andere Kanal wird nur mittels des System-Merkerbits ERROR über einen Halbleiterschalter angesteuert. Im betätigten Zustand werden die zu überwachenden Ausgänge über den Relaiskontakt (nicht zwangsgeführt) an die Klemmenspannung VBBx gelegt.

Das Clamp-Relais wird einkanalig über den Systemmerker RELAIS_CLAMP_15 angesteuert (→ Grafik).
RELAIS_CLAMP_15 ist nach dem Starten der Steuerung aktiv.
Das Clamp-Relais schaltet die Spannung VBBO an die zweite Ausgangsgruppe.

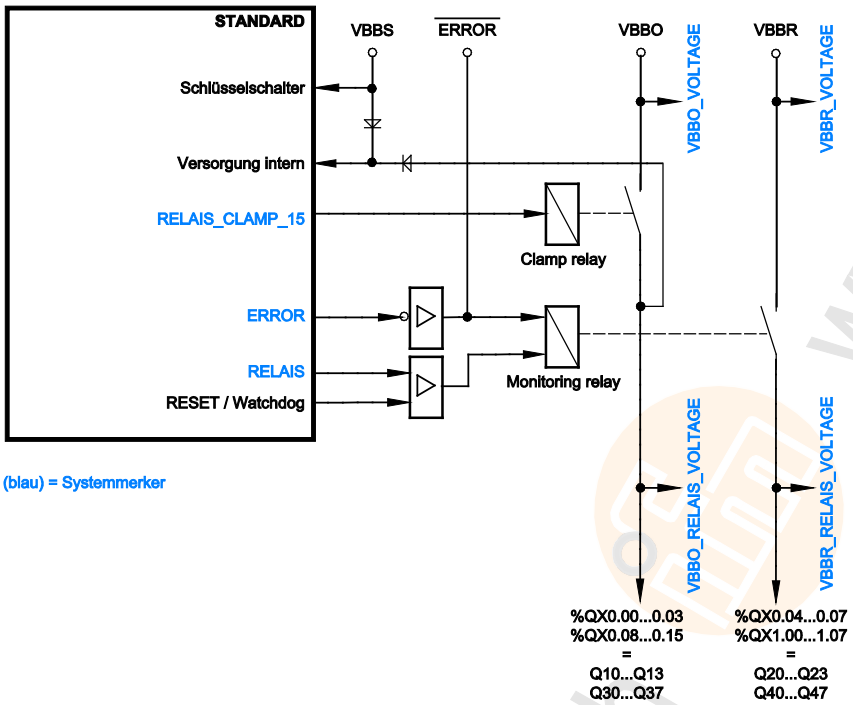
Das Clamp-Relais gewährleistet die interne Versorgung des Geräts, solange VBBO weiterhin anliegt, auch wenn VBBS gewollt oder ungewollt wegfällt.

Das Clamp-Relais unterliegt der vollen Kontrolle im Anwendungsprogramm und kann über einen Setz-/Rücksetzbefehl des Systemmerkers RELAIS_CLAMP_15 geschaltet werden.

Prinzipschaltung

19660

Aus dem nachfolgenden Prinzipschaltbild kann die Abhängigkeit der Relais von den anliegenden Signalen und den logischen Zuständen der Systemmerker entnommen werden.



Grafik: Prinzipaufbau der Versorgung und der Relais

Verfügbarer Speicher

13736

FLASH-Speicher

15366

FLASH-Speicher (nichtflüchtiger, langsamer Speicher) insgesamt im Gerät vorhanden	2 MByte

Davon sind folgende Speicherbereiche reserviert für ...

maximale Größe für das Anwendungsprogramm	704 kByte
Daten außerhalb des Anwendungsprogramms Anwender kann Daten speichern, z.B. Files, Bitmaps, Fonts	1 MByte
Daten außerhalb des Anwendungsprogramms Daten mit *FLASHREAD* (→ Seite 217) lesen oder mit *FLASHWRITE* (→ Seite 218) schreiben (bei Files: abzüglich 128 Byte für Header)	64 kByte

Der verbleibende Speicher ist reserviert für system-interne Zwecke.

SRAM

15906

SRAM (flüchtiger, schneller Speicher) insgesamt im Gerät vorhanden SRAM steht hier allgemein für alle Arten von flüchtigen, schnellen Speichern.	512 kByte

Davon sind folgende Speicherbereiche reserviert für ...

vom Anwendungsprogramm reservierte Daten	160 kByte

Der verbleibende Speicher ist reserviert für system-interne Zwecke.

FRAM

8002

FRAM (nichtflüchtiger, schneller Speicher) insgesamt im Gerät vorhanden FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.	32 kByte

Davon sind folgende Speicherbereiche reserviert für ...

im Anwendungsprogramm als VAR_RETAIN deklarierte Variablen	1 kByte
als remanent definierte Merker (ab %MB0...) ▶ Ende des Speicherbereichs im FB *MEMORY_RETAIN_PARAM* (→ Seite 214) angeben!	256 Byte
Vom Anwender frei verfügbarer remanenter Speicher Zugriff erfolgt über *FRAMREAD* (→ Seite 220) und *FRAMWRITE* (→ Seite 221)	16 kByte

Der verbleibende Speicher ist reserviert für system-interne Zwecke.

3.2.2 Funktionsweise der verzögerten Abschaltung

993

Werden die Controller von der Versorgungsspannung getrennt, werden im Normalfall sofort alle Ausgänge abgeschaltet, keine Eingangssignale mehr eingelesen und die Abarbeitung der Steuerungssoftware (Laufzeitsystem und Anwendungsprogramm) abgebrochen. Dieses geschieht unabhängig davon, in welchem Programmschritt sich der Controller befindet.

Wenn dieses Verhalten nicht gewünscht ist, muss der Controller programmgesteuert abgeschaltet werden. Das ermöglicht nach Abschalten der Zündung zum Beispiel das Sichern von Speicherständen.

Die ClassicController können durch eine entsprechende Beschaltung der Versorgungsspannungs-Eingänge und die Auswertung der zugehörigen Systemmerker, programmgesteuert abgeschaltet werden. Das Prinzipschaltbild im Kapitel *Hardware-Aufbau* (→ Seite 13) zeigt schematisch die Zusammenhänge der einzelnen Strompfade.

Klemme VBBS (23) mit Zündschalter verbinden

994

Über die Klemme 23 wird der Controller via Zündschalter abschaltbar versorgt.

Das Potential heißt in der Kraftfahrzeugtechnik "Klemme 15".

Diese Klemme wird intern überwacht. Liegt keine Versorgungsspannung an, wird der Systemmerker CLAMP_15 auf FALSE gesetzt. Das Rücksetzen des Merkers CLAMP_15 kann vom Anwendungsprogramm überwacht werden

Klemme VBBO (5) mit Batterie verbinden (nicht geschaltet)

995

Über die Klemme 5 werden bis zu 12 Ausgänge der Ausgangsgruppe VBBO mit Spannung versorgt. Gleichzeitig wird über diese Klemme die Selbsthaltung der Steuerungselektronik versorgt.

Selbsthaltung

996

Die Selbsthaltung ist aktiv, wenn VBBO an Spannung liegt **und** der Systemmerker RELAY_CLAMP_15 (und damit das Relais [Clamp]) gesetzt ist.

Wird der Systemmerker RELAY_CLAMP_15 zurückgesetzt, fällt das Relais [Clamp] ab. Liegt in diesem Moment keine Spannung an Klemme 23 an, wird die Selbsthaltung aufgehoben und der Controller schaltet sich vollständig ab.

3.2.3 Relais: wichtige Hinweise!

19480

Zuordnung Relais – Potentiale: → Datenblatt
Max. Summenstrom je Relaiskontakt (= je Ausgangsgruppe): → Datenblatt

ACHTUNG

Gefahr der Zerstörung der Relaiskontakte!
"Klebende" Relaiskontakte können auch im Notfall nicht mehr die Ausgänge von der Versorgung trennen!

Falls VBBS (Klemme 15) und VBBO gleichzeitig von der Versorgung getrennt werden, jedoch die Potentiale VBBx an der Versorgung angeschlossen bleiben, dann können die Relais schon abfallen, bevor die Ausgänge vom System deaktiviert werden.

In diesem Fall trennen die Relais **unter Last** die Ausgänge von der Versorgung. Dies schränkt die Lebensdauer der Relais deutlich ein.

► Bei dauerhaftem Anschluss von VBBx an Versorgung:
  • auch VBBO dauerhaft anschließen und
  • die Ausgänge programmgesteuert mit Hilfe von VBBS (Klemme 15) abschalten.

3.2.4 Überwachungskonzept

Inhalt

991

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

- die Steuerung schaltet die internen Relais ab
  - > die Ausgänge werden stromlos, behalten aber ihren logischen Zustand
  - > das Programm läuft weiter

oder:

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

Funktionsweise des Überwachungskonzeptes

Inhalt

997

Während des Programmablaufes steht das Ausgangsrelais unter voller Softwarekontrolle des Anwenders. So kann z.B. ein paralleler Kontakt der Sicherheitskette als Eingangssignal ausgewertet und das Ausgangsrelais entsprechend abgeschaltet werden. Zur weiteren Sicherheit müssen die entsprechenden nationalen Vorschriften beachtet werden.

Tritt während des Programmablaufs ein Fehler auf, kann durch das Systemmerker-Bit ERROR das Relais spannungsfrei geschaltet werden, um kritische Anlagenteile abzutrennen.

Durch Rücksetzen des Systemmerker-Bits RELAIS (über das Systemmerker-Bit ERROR oder direkt), werden alle Ausgänge abgeschaltet. Die Ausgänge im Strompfad VBBR werden direkt durch das Ausgangsrelais getrennt. Die Ausgänge im Strompfad VBBO werden also nur softwaregesteuert getrennt.

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!
  ⇨ Wiederanlaufsperre realisieren!
- ► Dazu im Fehlerfall die in Frage kommenden Ausgänge im Programm logisch abschalten!

Bei Auftreten eines Watchdog-Fehlers wird die Programmabarbeitung automatisch unterbrochen und der Controller zurückgesetzt. Der Controller startet anschließend neu, wie nach einem Power-On.

Überwachung der Klemmenspannung VBBR

20109

Über das Potential VBBR werden bis zu 12 Ausgänge der Ausgangsgruppe mit Spannung versorgt. Die Klemmenspannung wird überwacht:

ERROR_VBBR = TRUE	Versorgungsspannung fehlt oder ist zu niedrig
ERROR_VBBR = FALSE	Versorgungsspannung ist in Ordnung

► Diese Information im Anwendungsprogramm verarbeiten!

Überwachungs- und Sicherungsmechanismen

Inhalt

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,
  - Querschluss (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.

3.2.5 Eingänge (Technologie)

14090

Analog-Eingänge

2426

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

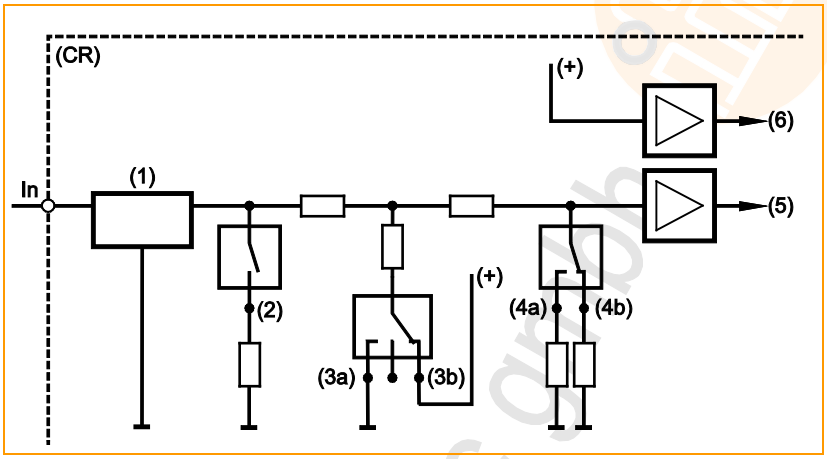
- Stromeingang 0...20 mA
- Spannungseingang 0...10 V
- Spannungseingang 0...32 V

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



In = Anschluss Multifunktions-Eingang n
(CR) = Gerät
(1) = Eingangsfilter
(2) = analoge Strommessung
(3a) = Binär-Eingang plus-schaltend
(3b) = Binär-Eingang minus-schaltend
(4a) = analoge Spannungsmessung 0...10 V
(4b) = analoge Spannungsmessung 0...32 V
(5) = Spannung
(6) = Referenz-Spannung

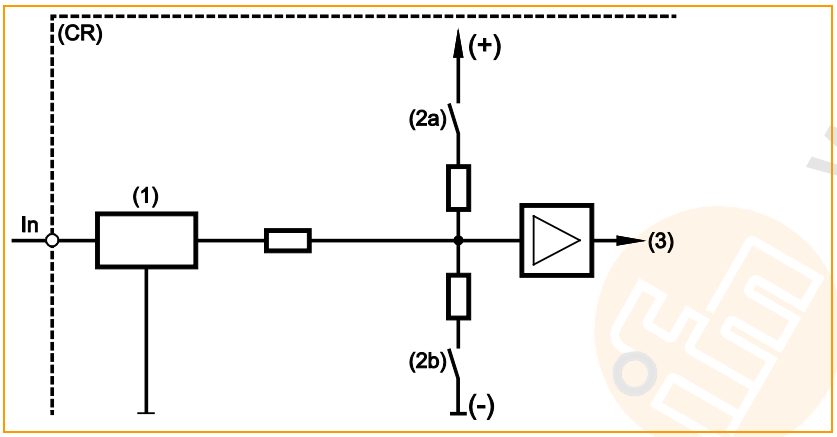
Grafik: Prinzipschaltung Multifunktions-Eingang

Binär-Eingänge

1015
7345

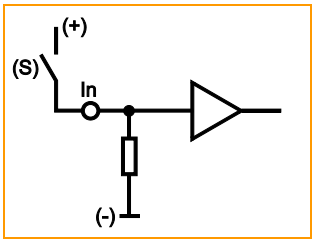
Der Binär-Eingang kann in folgenden Modi betrieben werden:
- binärer Eingang plus-schaltend (BL) für positives Gebersignal
- binärer Eingang, minus-schaltend (BH) für negatives Gebersignal

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.



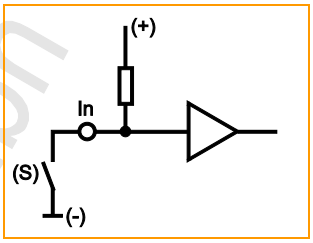
In = Anschluss Binär-Eingang n
(CR) = Gerät
(1) = Input-Filter
(2a) = Eingang minus-schaltend
(2b) = Eingang plus-schaltend
(3) = Spannung

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



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

Prinzipschaltung Binär-Eingang plus-schaltend (BL)
für positives Sensorsignal:
Eingang = offen ⇨ Signal = Low (Supply)



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

Prinzipschaltung Binär-Eingang minus-schaltend (BH)
für negatives Sensorsignal:
Eingang = offen ⇨ Signal = High (GND)

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

Eingangsgruppe I0 (I00...07 / ANALOG0...7)

20389

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

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

Diagnosefähige Sensoren nach NAMUR können ausgewertet werden.

Alle Eingänge zeigen das gleiche Verhalten bei Funktion und Diagnose.

Detaillierte Beschreibung → Kapitel *Adressbelegung Ein-/Ausgänge* (→ Seite 239)

Im Anwendungsprogramm können die Systemvariablen ANALOG00...ANALOGxx zur kundenspezifischen Diagnose der Eingänge dienen.

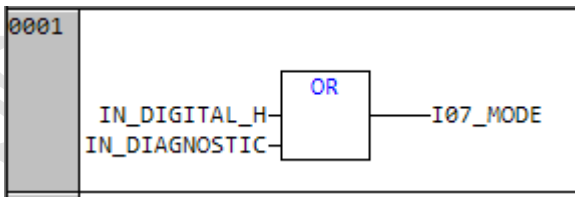
Werden die Analogeingänge auf Strommessung konfiguriert, wird bei Überschreiten des Endwertes (> 21 mA) in den sicheren Spannungsmessbereich (0...30V DC) geschaltet und das jeweilige Fehlerbit im Merkerbyte ERROR_I0 gesetzt. Wird der Grenzwert wieder unterschritten, schaltet der Eingang selbsttätig auf den Strommessbereich zurück.

► Die Konfiguration jedes einzelnen Eingangs erfolgt über das Anwendungsprogramm:
  - FB *INPUT_ANALOG* (→ Seite 150) > Eingang MODE
  - Konfigurationsbyte Ixx_MODE

15380

Beispiel mit Konfigurationsbyte Ixx_MODE

Die Zuweisung setzt den gewählten Eingang auf die Betriebsart IN_DIGITAL_H mit Diagnose:



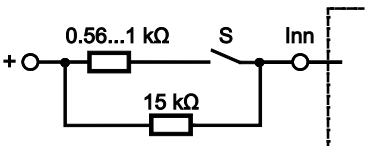
13956

> Das Diagnose-Ergebnis zeigen z.B. folgende Systemmerker:

Systemmerker (Symbolname)	Typ	Beschreibung
ERROR_BREAK_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Eingangsgruppe x: Leiterbruch-Fehler oder (Widerstandseingang): Schluss nach Versorgung [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SHORT_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Eingangsgruppe x: Kurzschluss-Fehler [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler

Diagnose bei nichtelektronischen Schaltern:

► Schalter mit einer zusätzlichen Widerstandsbeschaltung versehen!



Grafik: Nichtelektronischer Schalter S am Eingang Inn

Eingangsgruppe I1 (I10...17 / FRQ0...3)

19487

Eingänge I10...13

19490

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
- Ausgang (→ Kapitel *Ausgänge (Technologie)* (→ Seite 28))

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

Diese Eingänge sind nicht konfigurierbar.

Eingänge I14...17 / FRQ0...3

19497

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
- schneller Eingang für z.B. Inkrementalgeber und Frequenz- oder Periodendauermessung

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

Diagnosefähige Sensoren nach NAMUR können ausgewertet werden.

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

FAST_COUNT (→ Seite 157)	Zählerbaustein für schnelle Eingangsimpulse
FREQUENCY (→ Seite 158)	misst die Frequenz des am gewählten Kanal ankommenden Signals
INC_ENCODER (→ Seite 159)	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
PERIOD (→ Seite 162)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_RATIO (→ Seite 164)	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 166)	liest ein Kanalpaar mit schnellen Eingängen ein und vergleicht die Phasenlage der Signale

Eingangsgruppe I2 (I20...27)

19489

Eingänge I20...23

19499

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
- Ausgang (→ Kapitel *Ausgänge (Technologie)* (→ Seite 28))

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

Diese Eingänge sind nicht konfigurierbar.

Eingänge I24...27 / CYL0...3

19500

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
- schneller Eingang für z.B. Inkrementalgeber und Frequenz- oder Periodendauermessung

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

Diagnosefähige Sensoren nach NAMUR können ausgewertet werden.

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

FAST_COUNT (→ Seite 157)	Zählerbaustein für schnelle Eingangsimpulse
FREQUENCY (→ Seite 158)	misst die Frequenz des am gewählten Kanal ankommenden Signals
INC_ENCODER (→ Seite 159)	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
PERIOD (→ Seite 162)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_RATIO (→ Seite 164)	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 166)	liest ein Kanalpaar mit schnellen Eingängen ein und vergleicht die Phasenlage der Signale

Eingangsgruppe I3 (I30...37)

19501

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)
- binärer Eingang, minus-schaltend (BH) für negatives Gebersignal
- Ausgang (→ Kapitel *Ausgänge (Technologie)* (→ Seite 28))

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

Diagnosefähige Sensoren nach NAMUR können ausgewertet werden.

Alle Eingänge zeigen das gleiche Verhalten bei Funktion und Diagnose.

Detaillierte Beschreibung → Kapitel *Adressbelegung Ein-/Ausgänge* (→ Seite 239)

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

Eingangsgruppe I4 (I40...47)

19502

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)
- Ausgang (→ Kapitel *Ausgänge (Technologie)* (→ Seite 28))

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

Diagnosefähige Sensoren nach NAMUR können ausgewertet werden.

Alle Eingänge zeigen das gleiche Verhalten bei Funktion und Diagnose.

Detaillierte Beschreibung → Kapitel *Adressbelegung Ein-/Ausgänge* (→ Seite 239)

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

3.2.6 Ausgänge (Technologie)

Inhalt

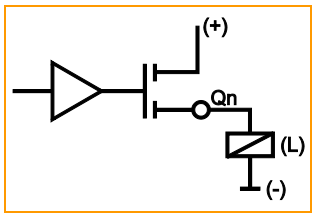
14093

Binär-Ausgänge

14094

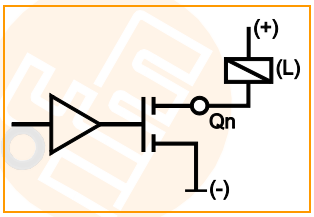
Bei den Geräte-Ausgängen sind folgende Betriebsarten möglich (→ Datenblatt):
- binärer Ausgang, plus-schaltend (BH) mit/ohne Diagnosefunktion
- binärer Ausgang, minus-schaltend (BL) ohne Diagnosefunktion

15450



Qn = Anschluss Ausgang n
(L) = Last

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



Qn = Anschluss Ausgang n
(L) = Last

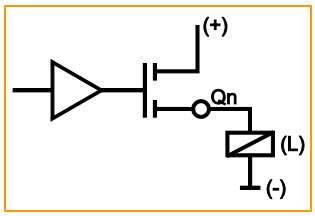
Prinzipschaltung Binär-Ausgang minus-schaltend (BL)
für negatives Ausgangssignal

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 Q1Q2 (Q10...13 / Q20...23)

19507

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) mit Diagnosefunktion und Protection
- analoger Ausgang, stromgeregelt (PWMi)
- analoger Ausgang mit Pulsweitenmodulation (PWM)
- binärer Eingang (→ Kapitel *Eingänge (Technologie)* (→ Seite 22))

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

Werden die Ausgänge nicht als PWM-Ausgänge genutzt, wird die Diagnose über die integrierten Strommesskanäle realisiert, die auch für die stromgeregelten Ausgangsfunktionen genutzt werden.

► Die Konfiguration jedes einzelnen Ausgangs erfolgt über das Anwendungsprogramm:
Lastströme anzeigen → FB *OUTPUT_CURRENT* (→ Seite 170)
PWM-Ausgang: → FB *PWM1000* (→ Seite 179)
Konfigurationsbyte Qxx_MODE

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.

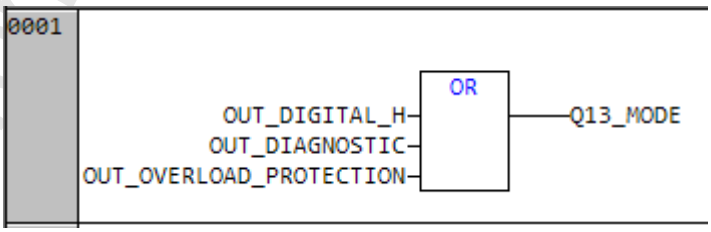
Die Ausgänge im PWM-Modus unterstützen keine Diagnosefunktionen.

Bei der Nutzung als Binärausgang erfolgt die Konfiguration mit den Systemvariablen Q1x_MODE...Q2x_MODE. Soll die Diagnose genutzt werden, muss sie zusätzlich aktiviert werden.

Leiterbruch und Kurzschluss des Ausgangssignals werden getrennt über die Systemvariablen ERROR_BREAK_Q1Q2 oder ERROR_SHORT_Q1Q2 angezeigt. Die einzelnen Ausgangs-Fehlerbits können im Anwendungsprogramm bei Bedarf ausmaskiert werden.

Beispiel:

Die Zuweisung setzt den gewählten Ausgang auf die Betriebsart OUT_DIGITAL_H mit Diagnose. Der Überlastschutz wird aktiviert (voreingestellt).



HINWEIS

Um die internen Messwiderstände zu schützen, sollte OUT_OVERLOAD_PROTECTION immer aktiv sein (max. Messstrom 4,1 A).

Zu den Grenzwerten unbedingt das Datenblatt beachten!

Die Funktion OUT_OVERLOAD_PROTECTION wird im reinen PWM-Modus nicht unterstützt.

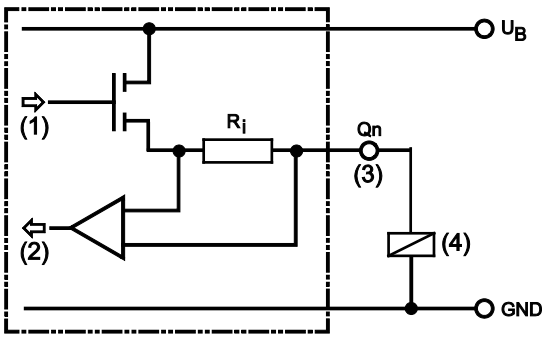
13976

Abhängig von der Umgebungstemperatur kann ab einem bestimmten Kurzschlussstrom ein Kurzschluss eventuell nicht mehr zuverlässig erkannt werden, da die Ausgangstreiber sich zum Schutz vor Zerstörung selbsttätig zeitweise deaktivieren.

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 (Qn=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 (Qn=TRUE) wird dann ein Kurzschluss gegen GND erkannt, wenn über den Längswiderstand R_i die Versorgungsspannung abfällt.

Ausgangsgruppe Q3 (Q30...37)

19511

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) mit Diagnosefunktion und Protection
- binärer Eingang (→ Kapitel *Eingänge (Technologie)* (→ Seite 22))

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

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.

Bei der Nutzung als Binärausgang erfolgt die Konfiguration jedes Ausgangs mit den Systemvariablen Qxx_MODE. Soll die Diagnose genutzt werden, muss sie zusätzlich aktiviert werden.

HINWEIS

Um die internen Messwiderstände zu schützen, sollte OUT_OVERLOAD_PROTECTION immer aktiv sein (voreingestellt). Je nach gewähltem Strommessbereich besteht Schutz ab 2,25 A oder ab 4,5 A. Die Funktion wird **nicht** im reinen PWM-Modus unterstützt und kann bei Bedarf abgeschaltet werden.

Zu den Grenzwerten unbedingt das Datenblatt beachten!

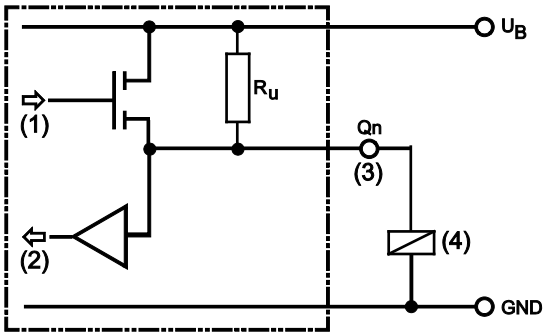
Abhängig von der Umgebungstemperatur kann ab einem bestimmten Kurzschlussstrom ein Kurzschluss eventuell nicht mehr zuverlässig erkannt werden, da die Ausgangstreiber sich zum Schutz vor Zerstörung selbsttätig zeitweise deaktivieren.

Die Leiterbruch- und die Kurzschlusserkennung sind aktiv, wenn der Ausgang **ein**geschaltet ist.

Diagnose: binäre Ausgänge (via Spannungsmessung)

19403
19397

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



Grafik: Prinzipschaltung
(1) Ausgangskanal
(2) Rücklesekanal für Diagnose
(3) Anschluss Ausgang n
(4) Last

Diagnose: Überlast

19448

Die Ausgänge haben keine Strommessung, keine Überlasterkennung.

Diagnose: Leiterbruch (via Spannungsmessung)

19404

Eine Leiterbruch-Erkennung erfolgt über den Rücklesekanal. Bei gesperrtem Ausgang (Qn=FALSE) wird dann ein Leiterbruch erkannt, wenn der Widerstand R_u den Rücklesekanal auf HIGH-Potential (VBB) zieht. Ohne den Leiterbruch würde die niederohmige Last (R_L < 10 kOhm) LOW (logisch 0) erzwingen.

Diagnose: Kurzschluss (via Spannungsmessung)

19405

Eine Kurzschluss-Erkennung erfolgt über den Rücklesekanal. Bei geschaltetem Ausgang (Qn=TRUE) wird dann ein Kurzschluss gegen GND erkannt, wenn der Rücklesekanal auf LOW-Potential (GND) gezogen wird.

Ausgangsgruppe Q4 (Q40...47)

19513

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), teilweise auch minus-schaltend (BL)
- analoger Ausgang mit Pulsweitenmodulation (PWM), teilweise als H-Brücke
- binärer Eingang (→ Kapitel *Eingänge (Technologie)* (→ Seite 22))

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

► Die Konfiguration jedes einzelnen Ausgangs erfolgt über das Anwendungsprogramm:
Lastströme anzeigen → FB *OUTPUT_CURRENT* (→ Seite 170)
PWM-Ausgang: → FB *PWM1000* (→ Seite 179)
Konfigurationsbyte Qxx_MODE

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.

Die Ausgänge im PWM-Modus unterstützen keine Diagnosefunktionen.

Bei der Nutzung als Binärausgang erfolgt die Konfiguration jedes Ausgangs mit den Systemvariablen Qxx_MODE. Soll die Diagnose genutzt werden, muss sie zusätzlich aktiviert werden.

Leiterbruch und Kurzschluss des Ausgangssignals werden (gebündelt je Ausgangsgruppe) getrennt über die Systemvariablen ERROR_BREAK_Qx oder ERROR_SHORT_Qx angezeigt. Die einzelnen Ausgangs-Fehlerbits können im Anwendungsprogramm bei Bedarf ausmaskiert werden.

HINWEIS

Um die internen Messwiderstände zu schützen, sollte OUT_OVERLOAD_PROTECTION immer aktiv sein (voreingestellt). Je nach gewähltem Strommessbereich besteht Schutz ab 2,25 A oder ab 4,5 A. Die Funktion wird **nicht** im reinen PWM-Modus unterstützt und kann bei Bedarf abgeschaltet werden.

Zu den Grenzwerten unbedingt das Datenblatt beachten!

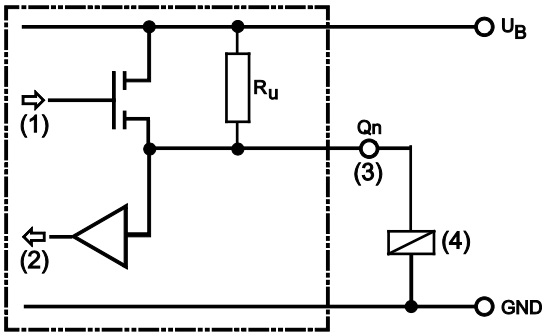
Abhängig von der Umgebungstemperatur kann ab einem bestimmten Kurzschlussstrom ein Kurzschluss eventuell nicht mehr zuverlässig erkannt werden, da die Ausgangstreiber sich zum Schutz vor Zerstörung selbsttätig zeitweise deaktivieren.

Die Leiterbruch- und die Kurzschlusserkennung sind aktiv, wenn der Ausgang **ein**geschaltet ist.

Diagnose: binäre Ausgänge (via Spannungsmessung)

19403
19397

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



Grafik: Prinzipschaltung
(1) Ausgangskanal
(2) Rücklesekanal für Diagnose
(3) Anschluss Ausgang n
(4) Last

Diagnose: Überlast

19448

Die Ausgänge haben keine Strommessung, keine Überlasterkennung.

Diagnose: Leiterbruch (via Spannungsmessung)

19404

Eine Leiterbruch-Erkennung erfolgt über den Rücklesekanal. Bei gesperrtem Ausgang (Qn=FALSE) wird dann ein Leiterbruch erkannt, wenn der Widerstand R_u den Rücklesekanal auf HIGH-Potential (VBB) zieht. Ohne den Leiterbruch würde die niederohmige Last (R_L < 10 kOhm) LOW (logisch 0) erzwingen.

Diagnose: Kurzschluss (via Spannungsmessung)

19405

Eine Kurzschluss-Erkennung erfolgt über den Rücklesekanal. Bei geschaltetem Ausgang (Qn=TRUE) wird dann ein Kurzschluss gegen GND erkannt, wenn der Rücklesekanal auf LOW-Potential (GND) gezogen wird.

3.2.7 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	**P**uls**w**eiten-**m**oduliertes 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

3.2.8 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

3.2.9 Betrieb von bidirektionalen Ein-/Ausgängen

999

Einige Anschlüsse der Controller können wahlweise als Eingang oder als Ausgang konfiguriert werden (→ Datenblatt).

ACHTUNG

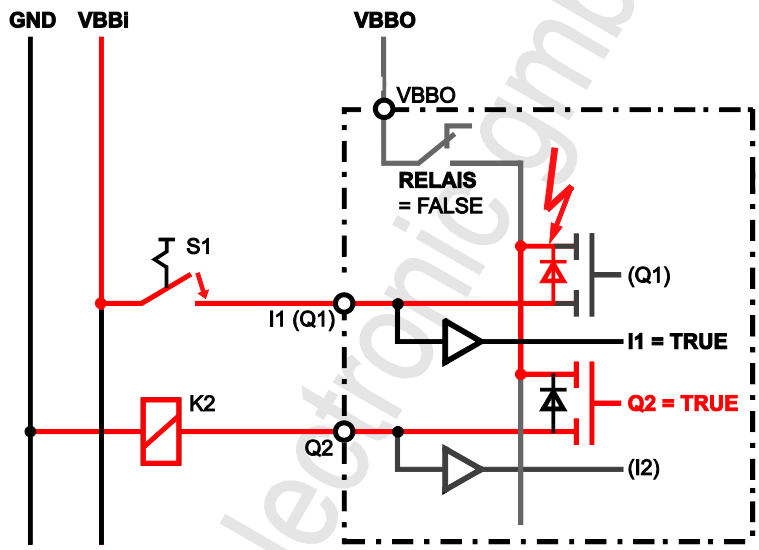
Zerstörung von Ausgängen bei unzulässiger Rückspeisung!

Wird eine Gruppe bidirektionaler Ein-/Ausgänge gemischt mit Ein- und Ausgängen betrieben, darf die Versorgung VBB dieser Ausgangsgruppe **nicht** potentialfrei werden.
Die Ausgangsgruppe wäre potentialfrei, wenn z.B. ...
- RELAIS = FALSE oder
- RELAIS_CLAMP_15 = FALSE.

Dieser potentialfreie Zustand führt über die Schutzdiode des Ausgangstransistors zu einer Spannungsrückspeisung, wenn innerhalb einer Ein-/Ausgangsgruppe...
- ein Eingang (z.B. I1) = TRUE und
- ein Ausgang derselben Gruppe (z.B. Q2) = TRUE.

> **Folge:**
Die Last am Ausgang (Q2) erhält über die Schutzdiode von Eingang (I1) Spannung. Die Schutzdiode und damit der Ausgang (Q1), über den der Rückspeisestrom in diesem Moment fließt, kann zerstört werden.

► **Abhilfe:**
Eine Ein-Ausgangsgruppe nur als Eingänge ODER nur als Ausgänge betreiben.
oder:
Hinweis unten befolgen.



Beispiel:

Merker RELAIS schaltet die Versorgung VBBO der Ausgangsgruppe aus.

Der externe Schalter S1 speist das Potential VBBi auf Eingang I1.

Wird Ausgang Q2 = TRUE (→ Grafik), dann bekommt K2 trotz RELAIS = FALSE Spannung über die Schutzdiode von Q1 (rote Linien). Wegen Überlastung brennt diese Schutzdiode durch und der Ausgang Q1 wird zerstört!

Grafik: Beispiel gefährlicher Beschaltung: Gefahr der Rückspeisung!

HINWEIS

Abhilfe bei gemischt betriebenen bidirektionalen Ein- / Ausgängen

► Merker RELAIS und/oder RELAIS_CLAMP_15 im Anwendungsprogramm dauerhaft mit TRUE beschalten:
- TRUE ----- RELAIS
- TRUE ----- RELAIS_CLAMP_15

3.2.10 Rückspeisung bei extern beschalteten Ausgängen

2422

In manchen Anwendungen werden Aktuatoren nicht nur von Ausgängen der SPS gesteuert, sondern zusätzlich von externen Schaltern. In solchen Fällen müssen die extern beschalteten Ausgänge mit Sperrdioden geschützt werden (→ Grafik unten).

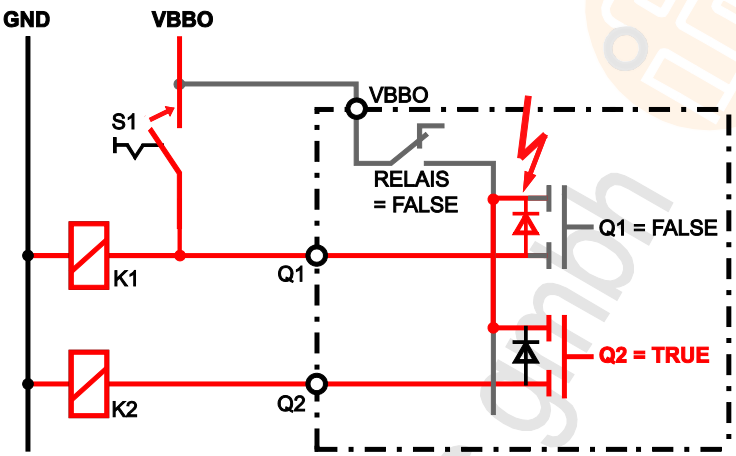
ACHTUNG

Zerstörung von Ausgängen bei unzulässiger Rückspeisung!

Werden Aktoren von extern angesteuert, darf die Potentialschiene derselben Ausgangsgruppe nicht potentialfrei werden (z.B. bei RELAIS = FALSE).

Andernfalls findet über die integrierte Schutzdiode im Ausgangstreiber des extern beschalteten Ausgangs eine Rückspeisung der Klemmenspannung VBB auf die Potentialschiene der Ausgangsgruppe statt. Dadurch steuert ein gesetzter anderer Ausgang derselben Gruppe seine an ihm angeschlossene Last an. Durch den Laststrom wird der rückspeisende Ausgang zerstört.

► Extern beschaltete Ausgänge mit Sperrdioden schützen!

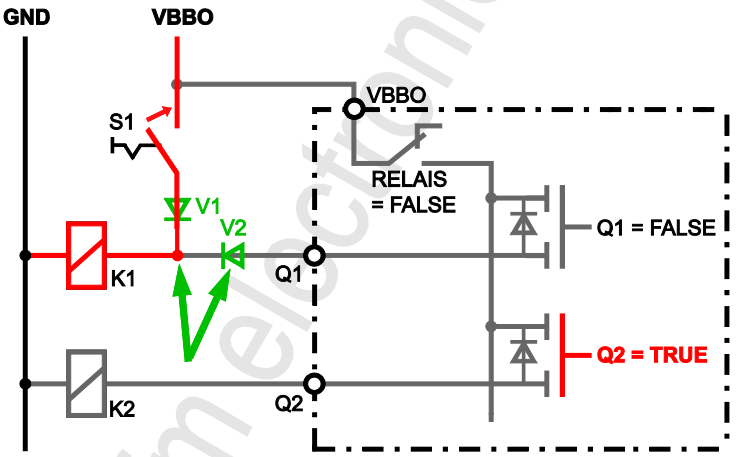


Beispiel:

Merker RELAIS schaltet die Versorgung VBBO der Ausgangsgruppe aus.

Ohne Sperrdioden speist der externe Schalter S1 die Versorgung VBBO über die interne Schutzdiode (rot) von Ausgang Q1 auf die interne Potentialschiene der Ausgänge.

Wird Ausgang Q2 = TRUE (→ Grafik), dann bekommt K2 trotz RELAIS = FALSE Spannung über die Schutzdiode von Q1 (rote Linien). Wegen Überlastung brennt diese Schutzdiode durch und der Ausgang Q1 wird zerstört!



Grafik: Beispiel Beschaltung mit Sperrdioden wegen Gefahr der Rückspeisung

Abhilfe:

Sperrdioden V1 und V2 einsetzen (→ grüne Pfeile)!

Erfolg:

Wenn RELAIS = FALSE, dann bleibt K2 ausgeschaltet, auch wenn Q2 = TRUE.

HINWEIS

Abhilfe bei extern beschalteten Ausgängen

► Die extern beschalteten Ausgänge so über Dioden entkoppeln, dass keine externe Spannung an die Ausgangsklemme der Steuerung geschaltet werden kann!

3.2.11 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 (→ *Systemmerker* (→ Seite 233)):

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

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

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 84))
- CANopen-Master (→ Kapitel *Bausteine: CANopen-Master* (→ Seite 102))
- CANopen-Slave (→ Kapitel *Bausteine: CANopen-Slave* (→ Seite 111))
- CANopen-Netzwerkvariablen (via CODESYS)
- SAE J1939 (für Antriebsmanagement, → Kapitel *Bausteine: SAE J1939* (→ Seite 124))
- Buslast-Erkennung
- Errorframe-Zähler
- Download-Schnittstelle
- 100 % Buslast ohne Paketverlust

11796

In diesem **ecomat*mobile***-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

14107

3.4.1 Software-Module für das Gerät

Inhalt

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 61)

Nachfolgend beschreiben wir diese Software-Module:

Bootloader

14111

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

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

Laufzeitsystem

14112

Grundprogramm im Gerät, stellt die Verbindung her zwischen der Hardware des Gerätes und dem Anwendungsprogramm.
→ Kapitel *Software-Module für das Gerät* (→ Seite 43)

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_CR0020_Vxxyyzz.LIB`	gerätespezifische Bibliothek Muss immer im Anwendungsprogramm enthalten sein!
`ifm_CAN1_EXT_Vxxyyzz.LIB`	(optional) wenn die CAN-Schnittstelle 1 des Geräts auf 29 Bit arbeiten soll
`ifm_CR0020_CANopenMaster_Vxxyyzz.LIB`	(optional) wenn die CAN-Schnittstelle 1 des Geräts als CANopen-Master betrieben werden soll
`ifm_CR0020_CANopenSlave_Vxxyyzz.LIB`	(optional) wenn die CAN-Schnittstelle 1 des Geräts als CANopen-Slave betrieben werden soll
`ifm_CR0020_J1939_x_Vxxyyzz.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 CR0020* (→ Seite 78)

3.4.2 Programmierhinweise für CODESYS-Projekte

Inhalt

7426

Hier erhalten Sie Tipps zum Programmieren des Geräts.

► Beachten Sie die Hinweise im CODESYS-Programmierhandbuch
→ www.ifm.com > Land wählen > [Datenblattsuche] > CR0020 > [Betriebsanleitungen],
→ **ecomat*mobile***-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 **ecomat*mobile*** 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 trägem Verhalten des Geräts!
Zykluszeit darf nicht zu lang werden!

► Beim Erstellen des Anwendungsprogramms die oben aufgeführten Empfehlungen beachten und durch Austesten überprüfen.

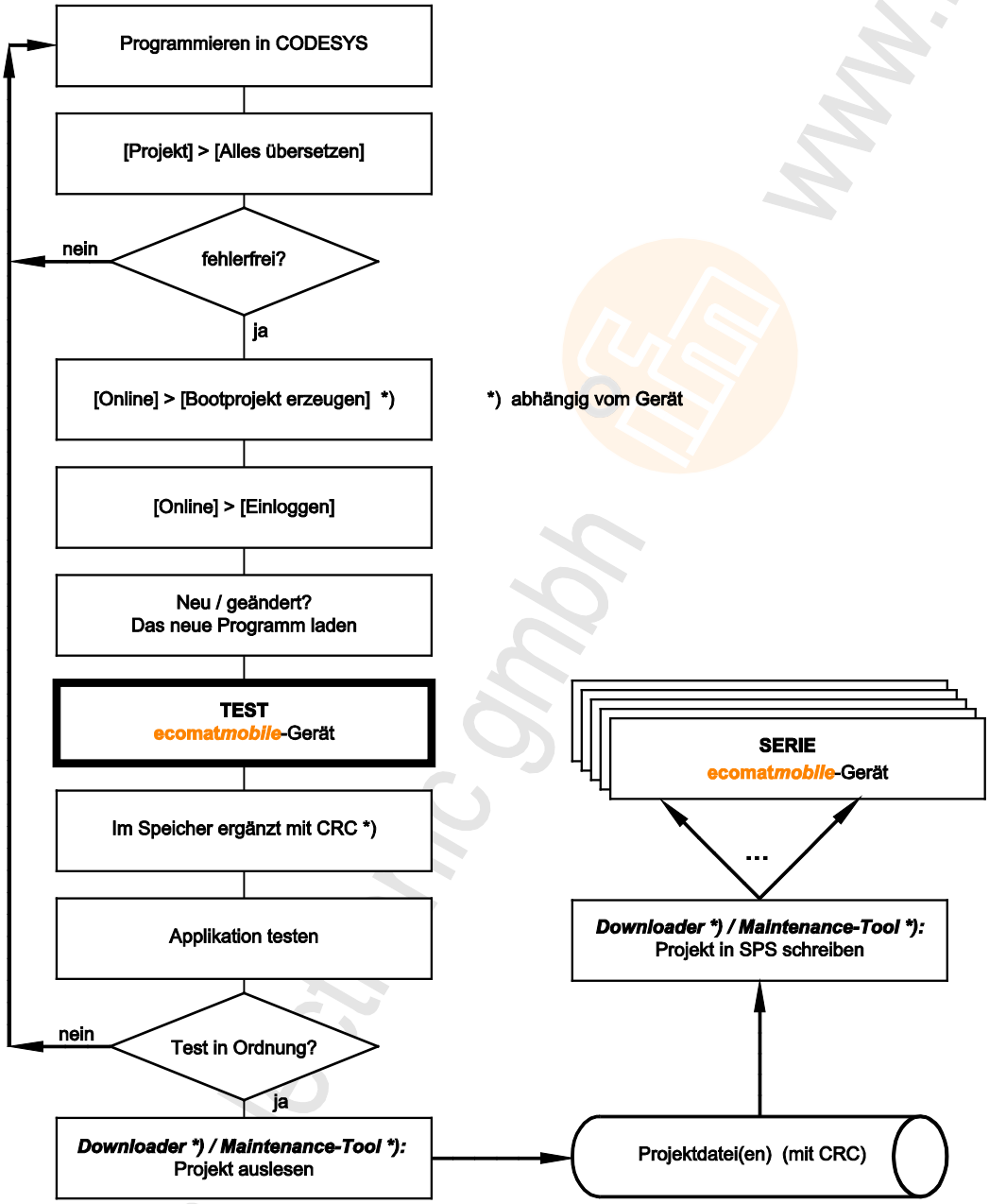
► Bei Bedarf durch Neustrukturieren der Software und des Systemaufbaus die Zykluszeit vermindern.

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]
ecomat*mobile*-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]
→ **ecomat*mobile***-DVD "Software, tools and documentation" im Register "R360 tools [D/E]"

3.4.3 Betriebszustände

Inhalt

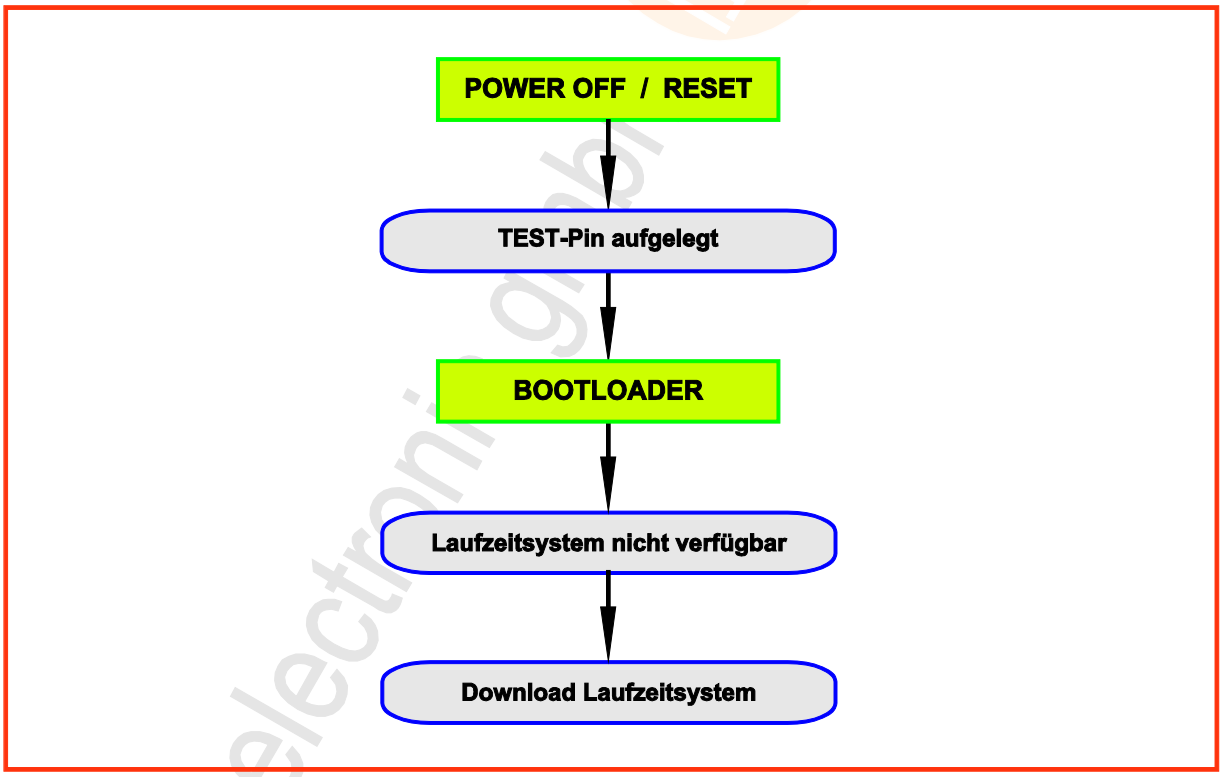
14120

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

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

Betriebszustände

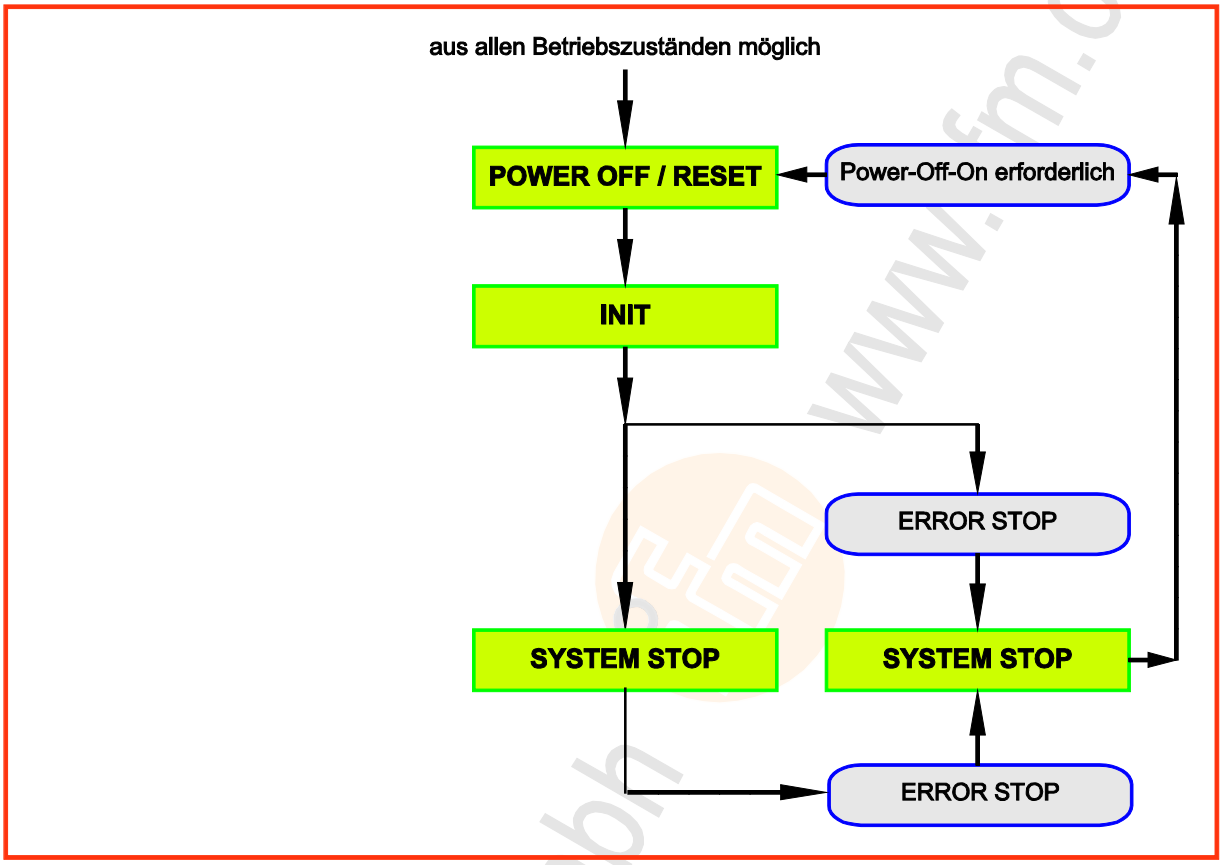
19217



Grafik: Betriebszustände (hier: Laufzeitsystem ist nicht verfügbar)

Betriebszustände: Anwendungsprogramm nicht verfügbar

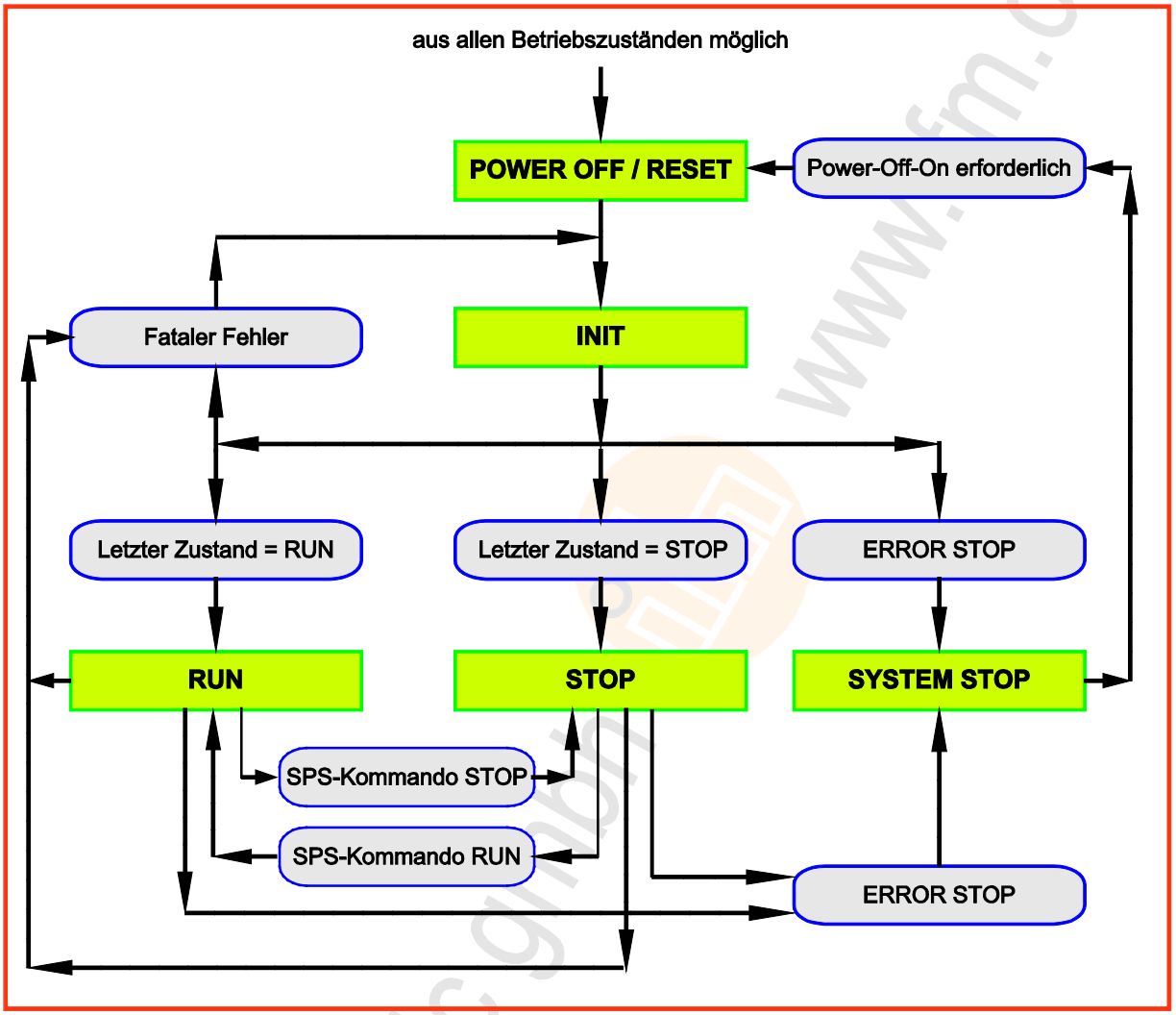
19218



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 **ecomat*mobile***-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 54))

> 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 54))

> Die LED blinkt grün (2 Hz).

SYSTEM-STOP-Zustand

19222

In diesen Zustand fällt der **ecomat*mobile***-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.

Zusammenfassung Test-Eingang ist aktiv:
- Programmiermodus ist freigeben
- 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 136))

DEBUG-Modus

1086

Wird der Eingang DEBUG von *SET_DEBUG* (→ Seite 227) 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 54)) nicht mit Versorgungsspannung verbunden wird.

3.4.5 Leistungsgrenzen des Geräts

7358

Leistungsgrenzen des Geräts beachten! → Datenblatt

Überdurchschnittliche Belastungen

1488
5023

Folgende Bausteine z.B. belasten die Systemressourcen überdurchschnittlich:

Baustein	Überdurchschnittliche Belastung
CYCLE, PERIOD, PERIOD_RATIO, PHASE	Einsatz mehrerer Messkanäle mit einer hohen Eingangsfrequenz
OUTPUT_CURRENT_CONTROL, OCC_TASK	Einsatz mehrerer Stromregler gleichzeitig
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.

Als Richtwerte sollten folgende Angaben beachtet werden:

Einschränkungen für den Einsatz von FBs

1489

Stromregler	max. 8	Möglichst keine weiteren belastenden Funktionen einsetzen!
CYCLE, PERIOD, PERIOD_RATIO, PHASE	1 Kanal	Eingangsfrequenz ≤ 10 kHz
	4 Kanäle	Eingangsfrequenz ≤ 2 kHz
INC_ENCODER	max. 4	Möglichst keine weiteren belastenden Funktionen einsetzen!

1509

ACHTUNG

Gefahr von zu trägem 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

11786

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

Wird die maximale Watchdog-Zeit (ca. 100 ms) überschritten:
> das Gerät führt einen Reset durch und startet neu

Zu erkennen im Merker LAST_RESET.

4 Konfigurationen

Inhalt

1016

Die in den jeweiligen Montage- und Installationsanweisungen oder dem *Anhang* (→ Seite 233) 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

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_CR0020_Vxxyyzz.H86`),
- der Steuerungskonfiguration (`ifm_CR0020_Vxx.CFG`),
- der Gerätebibliothek (`ifm_CR0020_Vxxyyzz.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. "`CR0020`") 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**.

Das Laufzeitsystem wird mit dem eigenständigen Programm "**ifm**-Downloader" in das Gerät übertragen. (Der **ifm**-Downloader und dessen Dokumentation befindet sich auf der **ecomat*mobile***-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 Betriebsystem-Software oder des CODESYS-Laufzeitsystems kommt, gibt **ifm** davon eine neue Version heraus. Die Versionen werden fortlaufend durchnummeriert (V01, V02, V03, ...).

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

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

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 226)	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 228) gespeichert)

4.2 Programmiersystem einrichten

Inhalt

3968

4.2.1 Programmiersystem manuell einrichten

Inhalt

3963

Target einrichten

2687
11379

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

► 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_CR0020_Vxxyyzz.LIB` manuell ergänzen!

2689

HINWEIS

Es müssen immer die zum gewählten Target passenden Software-Stände zum Einsatz kommen:
- des Laufzeitsystems (`ifm_CR0020_Vxxyyzz.H86`),
- der Steuerungskonfiguration (`ifm_CR0020_Vxx.CFG`),
- der Gerätebibliothek (`ifm_CR0020_Vxxyyzz.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. "`CR0020`") 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

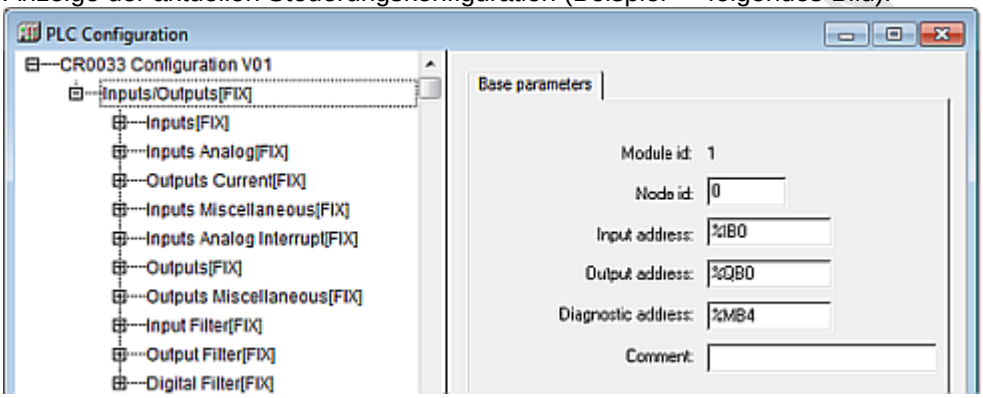
Bei der Konfiguration des Programmiersystems (→ vorheriger Abschnitt) erfolgte automatisch auch die Steuerungskonfiguration.

► 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:

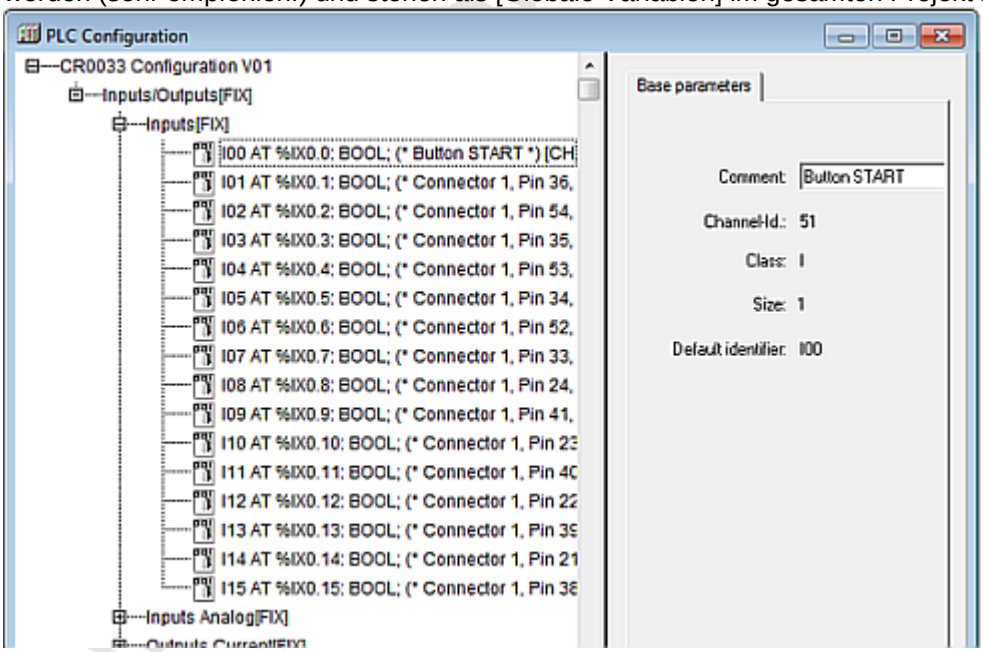
► In der linken Spalte Doppelklick auf [Steuerungskonfiguration]

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

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

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

4.3 Funktionskonfiguration, allgemein

Inhalt

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 233)) 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

1375
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_CR0020_Vxxyyzz.LIB`) Konstanten vordefiniert (z.B. IN_DIGITAL_H). Ausführliche Angaben → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243).
- In den Templates zu jeder Steuerung finden Sie Programmbausteine, die während des 1. Zyklus nach einem Neustart der Steuerung aufgerufen werden. Die dort programmierten Netzwerke dienen lediglich dazu, den Ein- und Ausgängen eine definierte Konfiguration zuzuweisen.

4.4.1 Eingänge konfigurieren

Inhalt

19567
3973

Zulässige Betriebsarten → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

► Die Konfiguration jedes einzelnen Eingangs erfolgt über das Anwendungsprogramm:
  • FB *INPUT_ANALOG* (→ Seite 150) > Eingang MODE
  • Konfigurationsbyte Ixx_MODE

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

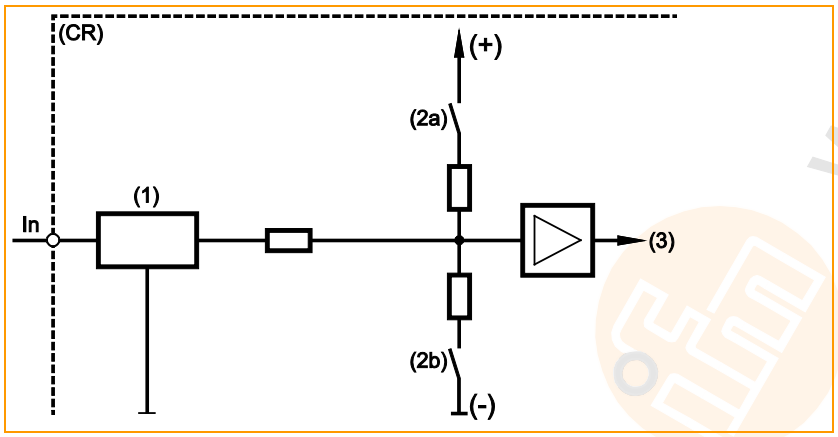
► Der Vorwiderstand darf 5 % des Eingangswiderstands RE des Geräte-Eingangs (→ Datenblatt) nicht überschreiten. Sonst wird das Signal nicht als TRUE erkannt.
Beispiel:
RE = 3 000 Ohm
⇨ max. Vorwiderstand = 150 Ohm

Binär-Eingänge

1015
7345

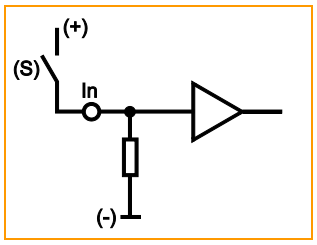
Der Binär-Eingang kann in folgenden Modi betrieben werden:
- binärer Eingang plus-schaltend (BL) für positives Gebersignal
- binärer Eingang, minus-schaltend (BH) für negatives Gebersignal

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.



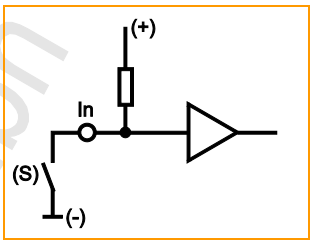
In = Anschluss Binär-Eingang n
(CR) = Gerät
(1) = Input-Filter
(2a) = Eingang minus-schaltend
(2b) = Eingang plus-schaltend
(3) = Spannung

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



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

Prinzipschaltung Binär-Eingang plus-schaltend (BL)
für positives Sensorsignal:
Eingang = offen ⇨ Signal = Low (Supply)



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

Prinzipschaltung Binär-Eingang minus-schaltend (BH)
für negatives Sensorsignal:
Eingang = offen ⇨ Signal = High (GND)

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

Schnelle Eingänge

1018

Zusätzlich verfügen die Controller über bis zu 16 schnelle Zähl-/Impulseingänge für eine Eingangsfrequenz bis 50 kHz (→ Datenblatt). Werden z.B. mechanische Schalter an diesen Eingängen angeschlossen, kann es durch Kontaktprellen zu Fehlsignalen in der Steuerung kommen. Über das Anwendungsprogramm müssen bei Bedarf diese "Fehlsignale" ausgefiltert werden.

Ferner muss beachtet werden, ob die Impulseingänge für Frequenzmessung (FRQx) und/oder Periodendauermessung (CYLx) ausgelegt sind (→ Datenblatt).

Geeignete Funktionsbausteine sind z.B.:

an FRQx-Eingängen:

FAST_COUNT (→ Seite 157)	Zählerbaustein für schnelle Eingangsimpulse
FREQUENCY (→ Seite 158)	misst die Frequenz des am gewählten Kanal ankommenden Signals

an CYLx-Eingängen:

PERIOD (→ Seite 162)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_RATIO (→ Seite 164)	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 166)	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.

Analog-Eingänge

1369

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

Wird in der Betriebsart "0...32 V" die Versorgungsspannung zurückgelesen, kann die Messung auch ratiometrisch erfolgen. Das bedeutet, ohne zusätzliche Referenzspannung können Potentiometer oder Joysticks ausgewertet werden. Ein Schwanken der Versorgungsspannung hat auf diesen Messwert dann keinen Einfluss.

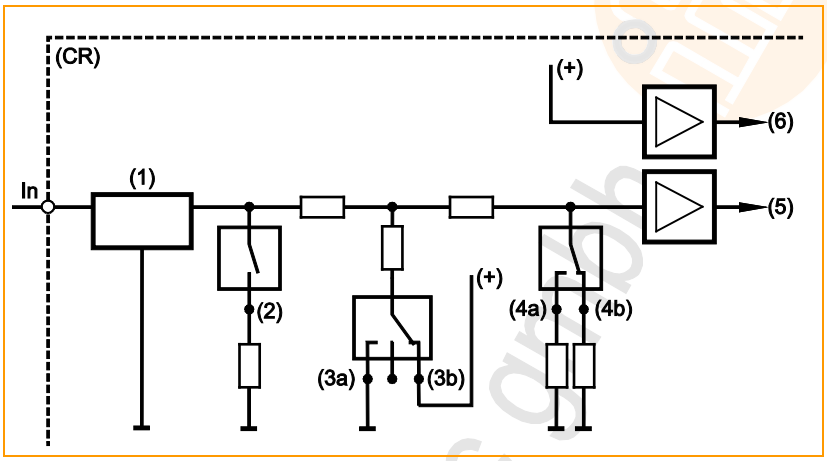
Alternativ kann ein Analog-Kanal auch binär ausgewertet werden.

HINWEIS

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

▶ Bei binärer Auswertung die höheren Eingangswiderstände berücksichtigen!

8971



In = Anschluss Multifunktions-Eingang n
(CR) = Gerät
(1) = Eingangsfilter
(2) = analoge Strommessung
(3a) = Binär-Eingang plus-schaltend
(3b) = Binär-Eingang minus-schaltend
(4a) = analoge Spannungsmessung 0...10 V
(4b) = analoge Spannungsmessung 0...32 V
(5) = Spannung
(6) = Referenz-Spannung

Grafik: Prinzipschaltung Multifunktions-Eingang

4.4.2 Ausgänge konfigurieren

Inhalt

19568
3976

Zulässige Betriebsarten → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

► Die Konfiguration jedes einzelnen Ausgangs erfolgt über das Anwendungsprogramm:
Lastströme anzeigen → FB *OUTPUT_CURRENT* (→ Seite 170)
PWM-Ausgang: → FB *PWM1000* (→ Seite 179)
Konfigurationsbyte Qxx_MODE

Binär- und PWM-Ausgänge

1346

Bei den Geräte-Ausgängen sind folgende Betriebsarten möglich (→ Datenblatt):
- binärer Ausgang, plus-schaltend (BH) mit Diagnosefunktion und Protection
- analoger Ausgang mit Pulsweitenmodulation (PWM)
- PWM-Ausgangspaar H-Brücke ohne Diagnosefunktion

PWM-Ausgänge können mit und ohne Stromregelfunktion betrieben werden.

Stromgeregelte PWM-Ausgänge werden überwiegend zur Ansteuerung von proportionalen Hydraulikfunktionen genutzt.

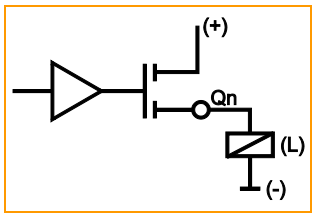
14713

⚠ WARNUNG

Sach- oder Körperschäden möglich durch Fehlfunktionen!

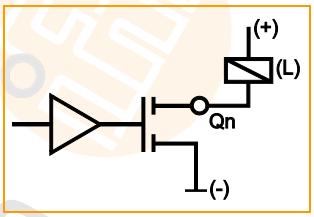
Für Ausgänge im PWM-Modus gilt:
- es gibt keine Diagnosefunktionen
- es werden keine ERROR-Merker gesetzt
- der Überlastschutz OUT_OVERLOAD_PROTECTION ist NICHT aktiv

15450



Qn = Anschluss Ausgang n
(L) = Last

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



Qn = Anschluss Ausgang n
(L) = Last

Prinzipschaltung Binär-Ausgang minus-schaltend (BL) für negatives 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.

5035

Verhalten bei Kurzschluss, andauernder Überlastung oder Leiterbruch:
(gilt ab Hardwarestand AH, jedoch **nicht** im Safety-Betrieb)

> Systemmerker ERROR_SHORT_Qx (bei Kurzschluss oder Überlast) oder ERROR_BREAK_Qx (bei Leiterbruch) wird aktiv.

> Nur bei Kurzschluss/Überlast: Laufzeitsystem schaltet den betroffenen Ausgangstreiber ab.
Betroffener Ausgang bleibt logisch auf TRUE.
Nach einer Wartezeit wird der Ausgang erneut angesteuert, was zu einem periodischen Schalten in den Kurzschluss führen kann.
Die Wartezeit steigt mit der (Über-)Belastung des Ausgangs.
Einschaltdauer bei Kurzschluss typisch 50 µs, bei Überlast deutlich länger.

► Fehlermerker im Anwendungsprogramm auswerten!
Den Ausgang logisch zurücksetzen, ggf. Maschine anhalten.
Bei Bedarf mit RELAIS=FALSE (z.B. via ERROR=TRUE) die Ausgangsgruppe VBB_R abschalten.

Nach Fehlerbeseitigung:

► Fehlermerker ERROR_..._Qx zurücksetzen.

> Ausgangsrelais gibt die Ausgangsgruppe VBB_R wieder frei.

► Ausgang neu setzen oder Maschine neu starten.

Ausgangsgruppe Q1Q2 (Q10...13 / Q20...23)

1378

Diese Ausgänge bieten eine doppelte Funktionalität. Werden sie als PWM-Ausgänge genutzt, wird die Diagnose über die integrierten Strommesskanäle, die auch für die stromgeregelten Ausgangsfunktionen genutzt werden, realisiert.

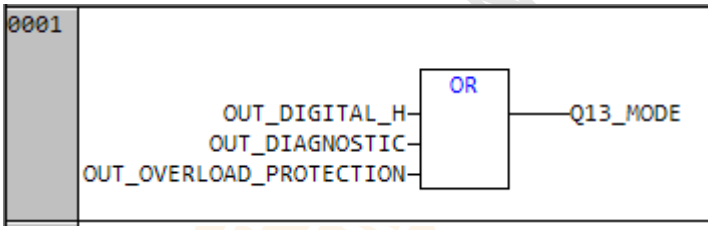
Mit *OUTPUT_CURRENT* (→ Seite 170) können Lastströme ≥ 100 mA angezeigt werden.

Bei der Nutzung als Binärausgang erfolgt die Konfiguration mit den Systemvariablen Q1x_MODE...Q2x_MODE. Soll die Diagnose genutzt werden, muss sie zusätzlich aktiviert werden.

Leiterbruch und Kurzschluss des Ausgangssignals werden getrennt über die Systemvariablen ERROR_BREAK_Q1Q2 oder ERROR_SHORT_Q1Q2 angezeigt. Die einzelnen Ausgangs-Fehlerbits können im Anwendungsprogramm bei Bedarf ausmaskiert werden.

Beispiel:

Die Zuweisung setzt den gewählten Ausgang auf die Betriebsart OUT_DIGITAL_H mit Diagnose. Der Überlastschutz wird aktiviert (voreingestellt).



HINWEIS

Um die internen Messwiderstände zu schützen, sollte OUT_OVERLOAD_PROTECTION immer aktiv sein (max. Messstrom 4,1 A).

Zu den Grenzwerten unbedingt das Datenblatt beachten!

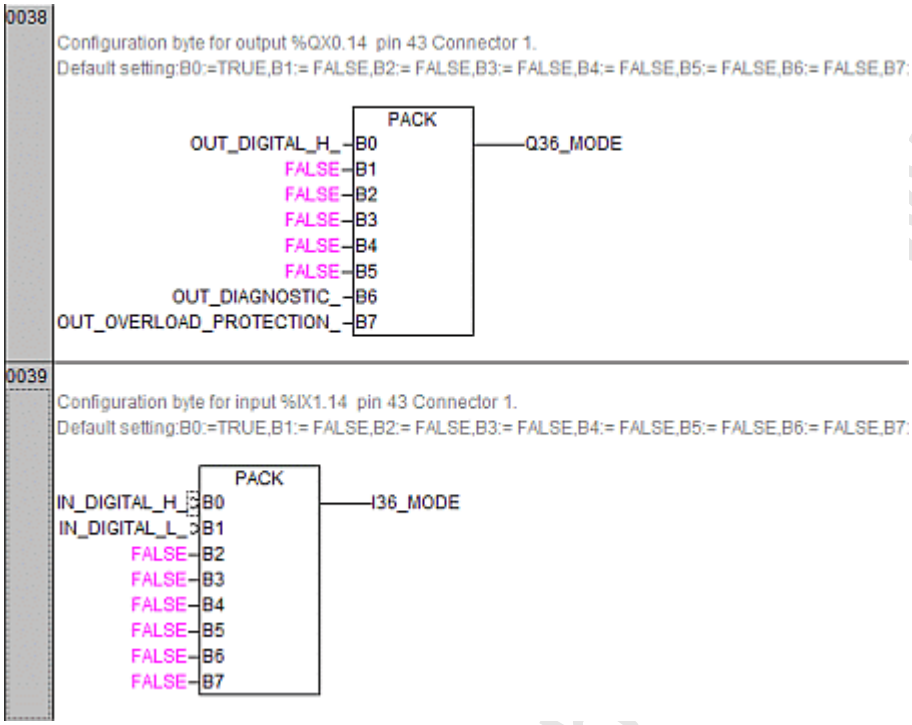
Die Funktion OUT_OVERLOAD_PROTECTION wird im reinen PWM-Modus nicht unterstützt.

Die Leiterbruch- und die Kurzschlusserkennung sind aktiv, wenn der Ausgang **ein**geschaltet ist.

Ausgangsgruppe Q3 (Q30...37)

1379

Die Konfiguration dieser Ausgänge erfolgt über die Systemvariablen Q3x_MODE. Soll die Diagnose genutzt werden, muss sie zusätzlich aktiviert werden. Gleichzeitig muss der korrespondierende Eingang durch Setzen des Systemmerkers I3x_MODE auf IN_NOMODE deaktiviert werden.



Screenshot: Diese Zuweisungen deaktivieren den Eingang und setzen den gewählten Ausgang auf die Betriebsart "OUT_DIGITAL_H mit Diagnose und Überlastschutz".

Leiterbruch und Kurzschluss des Ausgangssignals werden getrennt über die Systemvariablen ERROR_BREAK_Q3 oder ERROR_SHORT_Q3 angezeigt. Die einzelnen Ausgangs-Fehlerbits können im Anwendungsprogramm bei Bedarf ausmaskiert werden.

Zu den Grenzwerten unbedingt das Datenblatt beachten!

Die Leiterbrucherkennung ist aktiv, wenn der Ausgang **aus**geschaltet ist.
Die Kurzschlusserkennung ist aktiv, wenn der Ausgang **ein**geschaltet ist.

Ausgangsgruppe Q4 (Q40...47)

1380

Diese Ausgangsgruppe ist im Auslieferungszustand abgeschaltet, um die Diagnose über die Eingänge zu ermöglichen. Zur Nutzung der Ausgänge müssen sie aktiviert werden.

Die Konfiguration dieser Ausgänge erfolgt über die Systemvariablen Q4x_MODE. Soll die Diagnose genutzt werden, muss sie zusätzlich aktiviert werden. Gleichzeitig muss der korrespondierende Eingang durch Setzen des Systemmerkers I4x_MODE auf IN_NOMODE deaktiviert werden.

Leiterbruch und Kurzschluss des Ausgangssignals werden getrennt über die Systemvariablen ERROR_BREAK_Q4 und ERROR_SHORT_Q4 angezeigt. Die einzelnen Ausgangs-Fehlerbits können im Anwendungsprogramm bei Bedarf ausmaskiert werden.

Zur Realisierung einer H-Brückenfunktion können die Ausgänge Q41, Q42, Q45, Q46 zusätzlich in den Modus OUT_DIGITAL_L geschaltet werden.

Zu den Grenzwerten unbedingt das Datenblatt beachten!

Die Leiterbrucherkennung ist aktiv, wenn der Ausgang **aus**geschaltet ist.
Die Kurzschlusserkennung ist aktiv, wenn der Ausgang **ein**geschaltet ist.

4.5 Variablen

Inhalt

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
→ **ecomat*mobile*-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.

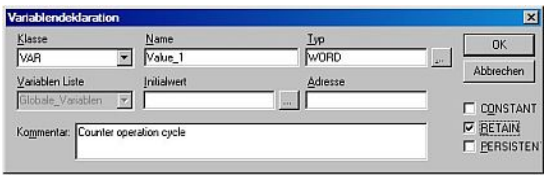
14166

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

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 CR0020

Inhalt

14235

Legende für ..._Vxxyyzz.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_CR0020_V06yyzz.LIB

19531

Dies ist die Geräte-Bibliothek. Diese **ifm**-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
ANALOG_RAW (→ Seite 149)	liefert nicht-normierte Werte des Analog-Digital-Wandlers für jeden einzelnen Eingangs-Port
CANx (siehe "*CAN2*" → Seite 92)	initialisiert die CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CAN1_BAUDRATE (→ Seite 85)	stellt die Übertragungsrate für den Busteilnehmer an der CAN-Schnittstelle 1 ein
CAN1_DOWNLOADID (→ Seite 86)	stellt den Download-Identifier für die CAN-Schnittstelle 1 ein
CANx_ERRORHANDLER (→ Seite 93)	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 94)	CAN-Schnittstelle x: konfiguriert alle Datenempfangsobjekte und liest den Empfangspuffer der Datenobjekte aus x = 2 = Nummer der CAN-Schnittstelle
CANx_RECEIVE (→ Seite 96)	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 98)	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 120)	CAN-Schnittstelle x: liest das SDO mit den angegebenen Indizes aus dem Knoten aus x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SDO_WRITE (→ Seite 122)	CAN-Schnittstelle x: schreibt das SDO mit den angegebenen Indizes in den Knoten x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_TRANSMIT (→ Seite 101)	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 224)	erzeugt über einen konfigurierbaren Speicherbereich eine Prüfsumme (CRC) und prüft die Daten des Speicherbereichs auf ungewollte Veränderung
DELAY (→ Seite 199)	verzögert die Ausgabe des Eingangswertes um die Zeit T (Totzeit-Glied)
FAST_COUNT (→ Seite 157)	Zählerbaustein für schnelle Eingangsimpulse
yFLASHREAD (→ Seite 217)	liest unterschiedliche Datentypen direkt aus dem Flash-Speicher in den RAM
FLASHWRITE (→ Seite 218)	schreibt unterschiedliche Datentypen direkt in den Flash-Speicher
FRAMREAD (→ Seite 220)	liest unterschiedliche Datentypen direkt aus dem FRAM-Speicher in den RAM FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.
FRAMWRITE (→ Seite 221)	schreibt unterschiedliche Datentypen direkt in den FRAM-Speicher FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.
FREQUENCY (→ Seite 158)	misst die Frequenz des am gewählten Kanal ankommenden Signals
GET_IDENTITY (→ Seite 226)	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 228) gespeichert)
GLR (→ Seite 200)	Der Gleichlaufregler ist ein Regler mit PID-Verhalten
INC_ENCODER (→ Seite 159)	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
INPUT_ANALOG (→ Seite 150)	Strom- und Spannungsmessung am analogen Eingangskanal
INPUT_CURRENT (→ Seite 151)	Strommessung am analogen Eingangskanal
INPUT_VOLTAGE (→ Seite 152)	Spannungsmessung am analogen Eingangskanal
MEMCPY (→ Seite 222)	schreibt und liest unterschiedliche Datentypen direkt in den Speicher
MEMORY_RETAIN_PARAM (→ Seite 214)	legt das remanente Verhalten der Daten für verschiedene Ereignisse fest

Baustein	Kurzbeschreibung
NORM (→ Seite 154)	normiert einen Wert [WORD] innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen
OCC_TASK (→ Seite 168)	OCC = Output Current Control (= stromgeregelter Ausgang) Stromregler für einen PWMi-Ausgangskanal Jede Instanz der Funktion wird im Zyklus von 5 ms aufgerufen.
OUTPUT_CURRENT (→ Seite 170)	misst den Strom (Mittelung über Dither-Periode) an einem Ausgangskanal
OUTPUT_CURRENT_CONTROL (→ Seite 171)	Stromregler für einen PWMi-Ausgangskanal
PERIOD (→ Seite 162)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_RATIO (→ Seite 164)	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 166)	liest ein Kanalpaar mit schnellen Eingängen ein und vergleicht die Phasenlage der Signale
PID1 (→ Seite 202)	PID-Regler
PID2 (→ Seite 204)	PID-Regler
PT1 (→ Seite 206)	Regelstrecke mit Verzögerung 1. Ordnung
PWM (→ Seite 173)	initialisiert und parametriert einen PWM-fähigen Ausgangskanal Festlegung der PWM-Frequenz über RELOAD
PWM100 (→ Seite 177)	initialisiert und parametriert einen PWM-fähigen Ausgangskanal PWM-Frequenz in [Hz] angeben Puls-Pausen-Verhältnis in 1 %-Schritten angeben
PWM1000 (→ Seite 179)	initialisiert und parametriert einen PWM-fähigen Ausgangskanal das Puls-Pausen-Verhältnis kann in 1 ‰-Schritten angegeben werden
SERIAL_PENDING (→ Seite 137)	ermittelt die Anzahl der im seriellen Empfangspuffer gespeicherten Datenbytes
SERIAL_RX (→ Seite 138)	liest mit jedem Aufruf ein empfangenes Datenbyte aus dem seriellen Empfangspuffer aus
SERIAL_SETUP (→ Seite 139)	initialisiert die serielle RS232-Schnittstelle
SERIAL_TX (→ Seite 141)	überträgt ein Datenbyte über die serielle RS232-Schnittstelle
SET_DEBUG (→ Seite 227)	organisiert (abhängig vom TEST-Eingang) den DEBUG-Modus oder den Monitoring-Modus
SET_IDENTITY (→ Seite 228)	setzt eine anwendungsspezifische Programmkennung
SET_INTERRUPT_I (→ Seite 143)	bedingtes Ausführen eines Programmteils nach einer Interrupt-Anforderung über einen Eingangskanal
SET_INTERRUPT_XMS (→ Seite 146)	bedingtes Ausführen eines Programmteils im Intervall von x Millisekunden
SET_PASSWORD (→ Seite 229)	setzt Benutzerkennung für Zugangskontrolle bei Programm- und Speicher-Upload
SOFTRESET (→ Seite 208)	führt einen kompletten Neustart des Geräts aus
TIMER_READ (→ Seite 210)	liest die aktuelle Systemzeit in [ms] aus Max-Wert = 49d 17h 2min 47s 295ms
TIMER_READ_US (→ Seite 211)	liest die aktuelle Systemzeit in [µs] aus Max-Wert = 1h 11min 34s 967ms 295µs

5.1.2 Bibliothek ifm_CR0020_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 103)	verwaltet den geräteeigenen Fehlerstatus des CANopen-Masters an der CAN-Schnittstelle x x = 1 = Nummer der CAN-Schnittstelle
CANx_MASTER_SEND_EMERGENCY (→ Seite 104)	versendet anwendungsspezifische Fehlerstatus des CANopen-Masters an der CAN-Schnittstelle x x = 1 = Nummer der CAN-Schnittstelle
CANx_MASTER_STATUS (→ Seite 106)	Status-Anzeige an der CAN-Schnittstelle x des als CANopen-Master eingesetzten Gerätes x = 1 = Nummer der CAN-Schnittstelle

5.1.3 Bibliothek ifm_CR0020_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 112)	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 113)	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 114)	versendet anwendungsspezifische Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x x = 1 = Nummer der CAN-Schnittstelle
CANx_SLAVE_STATUS (→ Seite 116)	zeigt den Status des an der CAN-Schnittstelle x als CANopen-Slave eingesetzten Gerätes x = 1 = Nummer der CAN-Schnittstelle

5.1.4 Bibliothek ifm_CAN1_EXT_Vxxyyzz.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 87)	initialisiert die CAN-Schnittstelle 1 auch für den Extended-Mode Modus und Baudrate einstellen
CAN1_EXT_ERRORHANDLER (→ Seite 88)	führt ein "manuelles" Bus-Recover auf der CAN-Schnittstelle 1 durch
CAN1_EXT_RECEIVE (→ Seite 89)	CAN-Schnittstelle 1: konfiguriert ein Datenempfangsobjekt und liest den Empfangspuffer des Datenobjektes aus
CANx_EXT_RECEIVE_ALL (→ Seite 94)	CAN-Schnittstelle x: konfiguriert alle Datenempfangsobjekte und liest den Empfangspuffer der Datenobjekte aus x = 1 = Nummer der CAN-Schnittstelle
CAN1_EXT_TRANSMIT (→ Seite 91)	übergibt in jedem Aufruf ein CAN-Datenobjekt (Message) an die CAN-Schnittstelle 1 zur Übertragung

5.1.5 Bibliothek ifm_J1939_x_Vxxyyzz.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 125)	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 126)	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 128)	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 130)	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 132)	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 134)	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_16bitOS05_Vxxyyzz.LIB

19535

Diese Bibliothek enthält Bausteine für Hydraulik-Steuerungen.

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

Baustein	Kurzbeschreibung
CONTROL_OCC (→ Seite 182)	OCC = Output Current Control (= stromgeregelter Ausgang) skaliert den Eingangswert [WORD] auf einen angegebenen Strombereich
JOYSTICK_0 (→ Seite 185)	skaliert Signale [INT] aus einem Joystick auf fest definierte Kennlinien, normiert auf 0…1000
JOYSTICK_1 (→ Seite 188)	skaliert Signale [INT] aus einem Joystick auf parametrierbare Kennlinien, normiert auf 0…1000
JOYSTICK_2 (→ Seite 192)	skaliert Signale [INT] aus einem Joystick auf einen parametrierbaren Kennlinien-Verlauf; die Normierung ist frei bestimmbar
NORM_HYDRAULIC (→ Seite 195)	normiert einen Wert [DINT] innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen

5.2 ifm-Bausteine für das Gerät CR0020

Inhalt

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

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_CR0020_Vxxyyzz.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-Speichermodul kann bei Dauerbetrieb dieser Funktion zerstört werden!

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

HINWEIS

Die neue 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_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:

Beschreibung

648

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

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

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

ACHTUNG

Für CR250n, CR0301, CR0302, CS0015 beachten:

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

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

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)

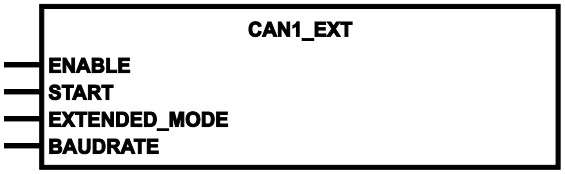
CAN1_EXT

4192

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CAN1_EXT_Vxxyyzz.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_Vxxyyzz.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-activ (= Normalzustand).

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

Das Fehlerbit CANx_BUSOFF muss anschließend im Anwendungsprogramm zurückgesetzt werden.

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

Parameter der Eingänge

2177

Parameter	Datentyp	Beschreibung
BUSOFF_RECOVER	BOOL	TRUE (nur 1 Zyklus lang): > 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

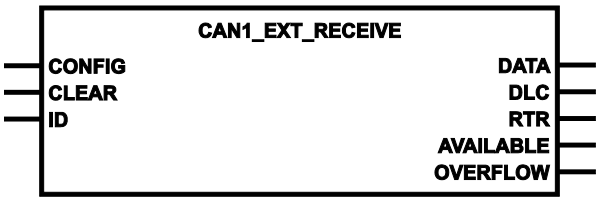
CAN1_EXT_RECEIVE

4302

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CAN1_EXT_Vxxyyzz.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 87) 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 87) 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

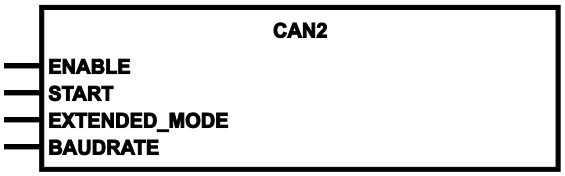
CAN2

639

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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

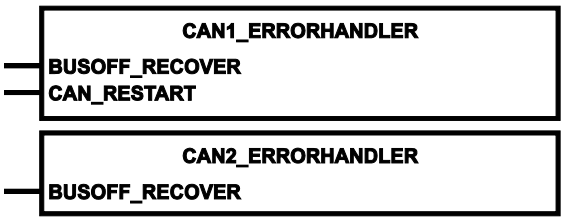
633

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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 gearteter 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 92) den Eingang START = TRUE (nur 1 Zyklus lang) setzen

> **HINWEIS**
>
> CAN2 muss grundsätzlich zum Initialisieren der zweiten CAN-Schnittstelle ausgeführt werden, bevor FBs für diese genutzt werden können.
>
> Wenn die automatische Bus-Recover-Funktion genutzt werden soll (Default-Einstellung), darf CANx_ERRORHANDLER **nicht** in das Programm eingebunden und instanziert werden!

Parameter der Eingänge

637

Parameter	Datentyp	Beschreibung
BUSOFF_RECOVER	BOOL	TRUE (nur 1 Zyklus lang): Bus-off-Zustand beheben FALSE: 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

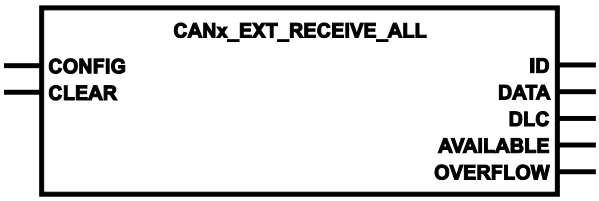
4183

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_Vxxyyzz.LIB`
CAN2_EXT_RECEIVE_ALL: Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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

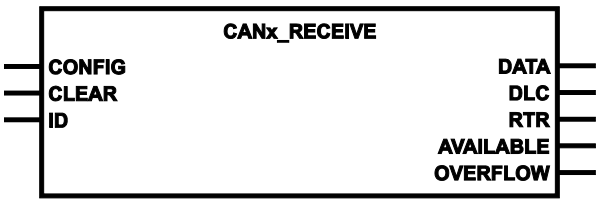
627

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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 92) 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

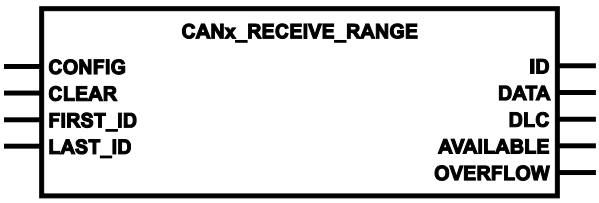
4179

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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 100)).

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 96) 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
```

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

CANx_TRANSMIT

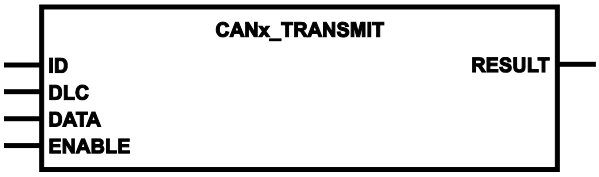
609

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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 92) die zweite CAN-Schnittstelle initialisiert werden.

Parameter der Eingänge

613

Parameter	Datentyp	Beschreibung
ID	WORD	Nummer des Datenobjekt-Identifier Zulässige Werte: 0...2 047
DLC	BYTE	Anzahl der 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

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_CR0020_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 104) 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

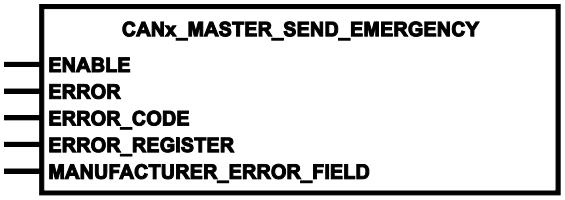
13195

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_CANopenMaster_Vxxyyzz.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 103) **nach** dem (mehrfachen) Bearbeiten von CANx_MASTER_SEND_EMERGENCY aufgerufen werden.

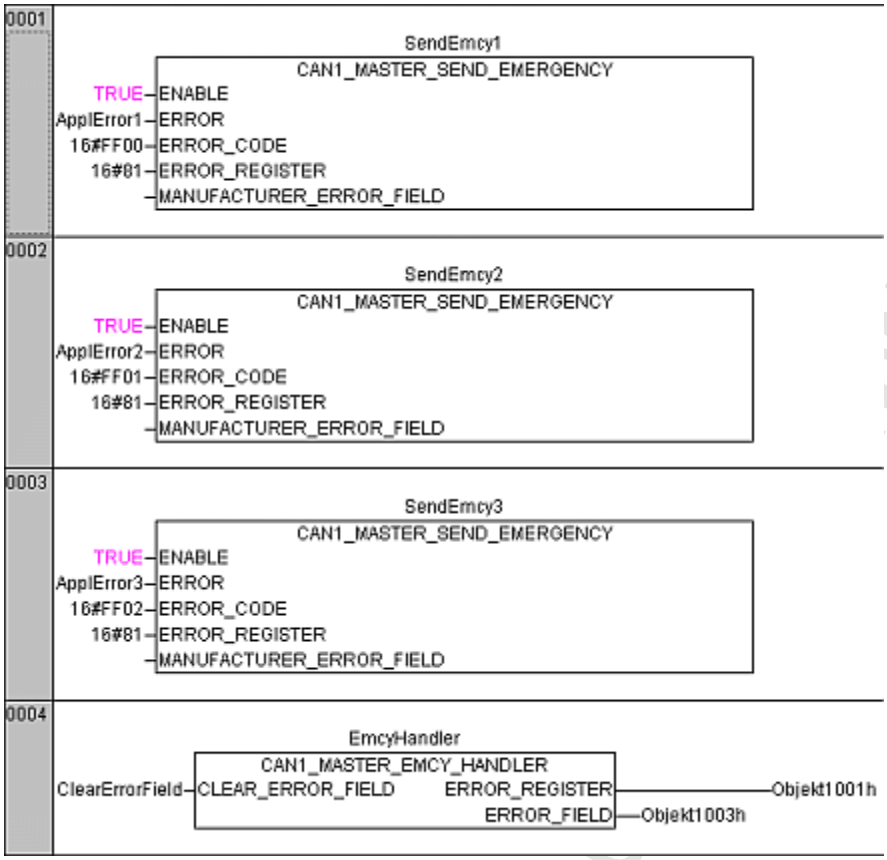
Parameter der Eingänge

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-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 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

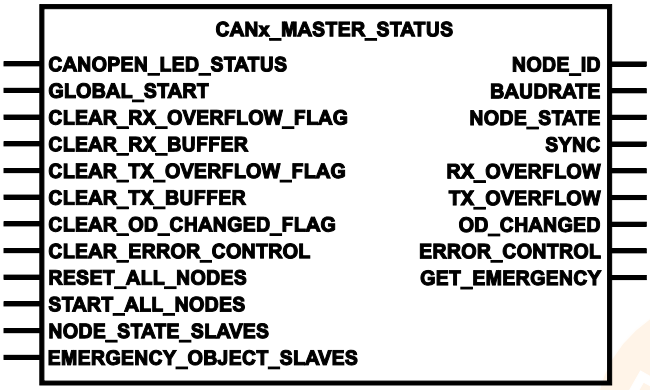
2021

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_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 Netzwerkknoten. Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben! Beispiel-Code → Kapitel *Beispiel: CANx_MASTER_STATUS* (→ Seite 109)
EMERGENCY_OBJECT_SLAVES	DWORD	Zeigt die zuletzt aufgetretenen Fehlermeldungen aller Netzwerkknoten. Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben! → Kapitel *Zugriff auf die Strukturen zur Laufzeit der Anwendung* (→ Seite 110)

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 Netzwerkknoten (Guard- oder Heartbeat-Fehler) → Kapitel *Zugriff auf die Strukturen zur Laufzeit der Anwendung* (→ Seite 110)
GET_EMERGENCY	STRUCT CANx_EMERGENY_MESSAGE	Am Ausgang stehen die Daten für die Struktur CANx_EMERGENY_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_EMERGENY_MESSAGE	STRUCT	NODE_ID: BYTE ERROR_CODE: WORD ERROR_REGISTER: BYTE MANUFACTURER_ERROR_FIELD: ARRAY [0..4] OF BYTE Die Struktur ist in den globalen Variablen der Bibliothek `ifm_CR0020_CANopenMaster_Vxxyyzz.LIB` angelegt.
CANx_NODE_STATE	STRUCT	NODE_ID: BYTE NODE_STATE: BYTE LAST_STATE: BYTE RESET_NODE: BOOL START_NODE: BOOL PREOP_NODE: BOOL SET_TIMEOUT_STATE: BOOL SET_NODE_STATE: BOOL Die Struktur ist in den globalen Variablen der Bibliothek `ifm_CR0020_CANopenMaster_Vxxyyzz.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     GuardHeartbealErrorArray: 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.

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

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

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

5.2.3 Bausteine: CANopen-Slave

Inhalt

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_CR0020_CANopenSlave_Vxxyyzz.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 114) 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_CR0020_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

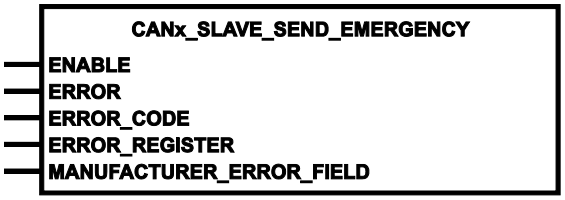
13205

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_CANopenSlave_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2059

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

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

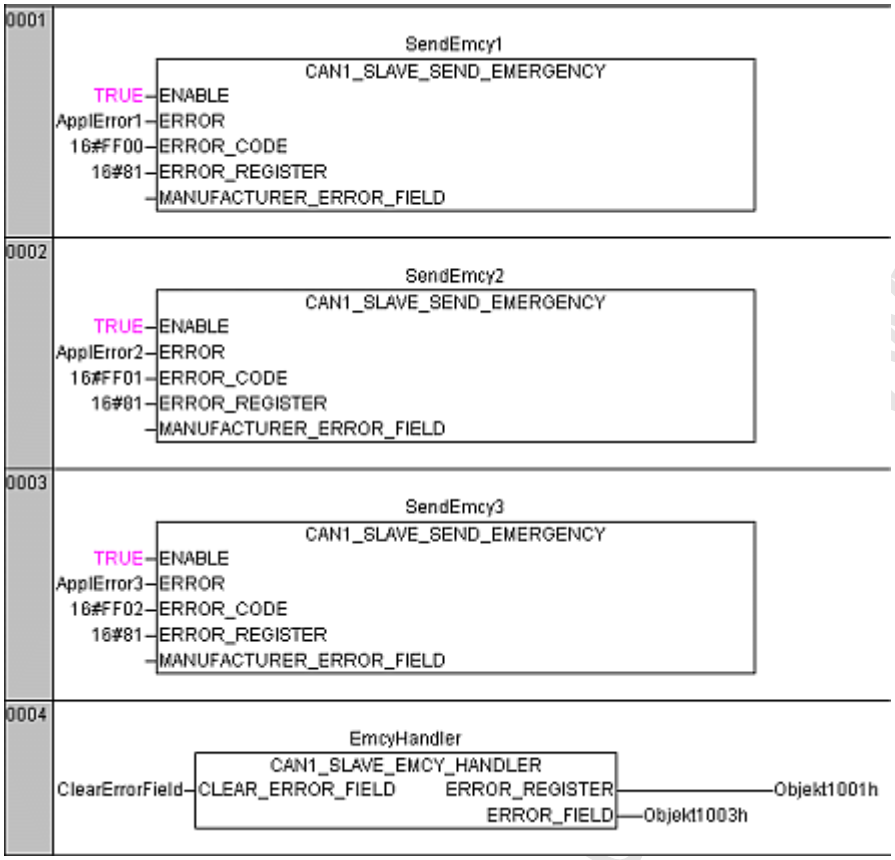
Parameter der Eingänge

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 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_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_EMCY_HANDLER sendet die Fehlermeldungen an das Fehler-Register "Objekt 0x1001" im Fehler-Array "Objekt 0x1003".

CANx_SLAVE_STATUS

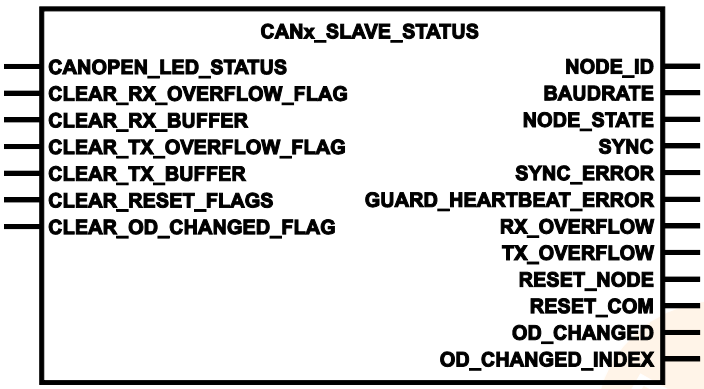
2063

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_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     GuardHeartbealError: 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

2071

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

CANx_SDO_READ

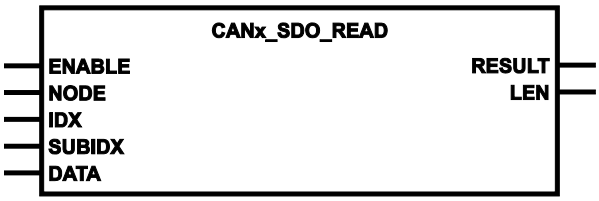
621

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:

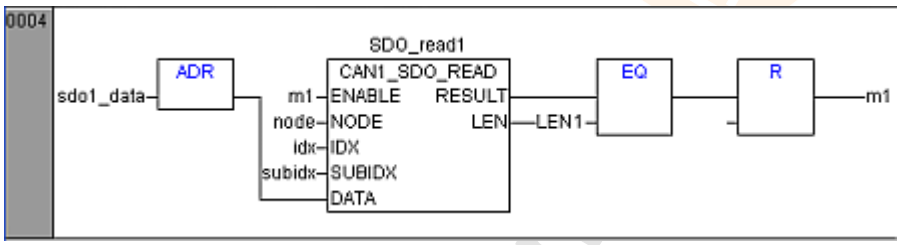


Beschreibung

624

CANx_SDO_READ liest das →*SDO* (→ Seite 262) 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	Wert 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

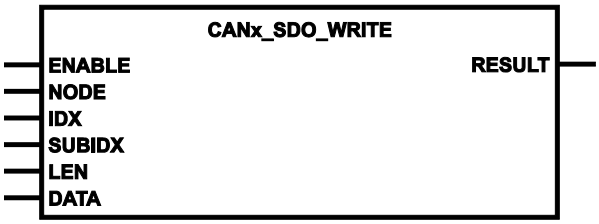
615

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

618

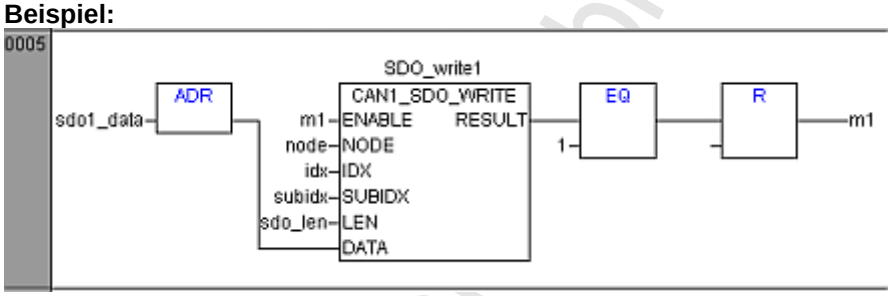
CANx_SDO_WRITE schreibt das →*SDO* (→ Seite 262) mit den angegebenen Indizes in den Knoten.

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 Sende-Arrays. 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	Wert 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

2273

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

J1939_x

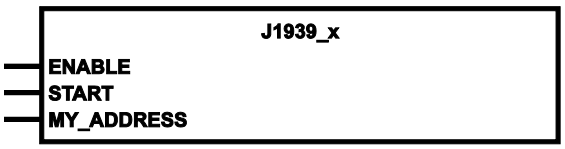
9375

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_J1939_x_Vxxyyzz.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 87) initialisieren!

J1939-Kommunikation über 2. CAN-Schnittstelle:
► Schnittstelle zuvor mit *CAN2* (→ Seite 92) 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

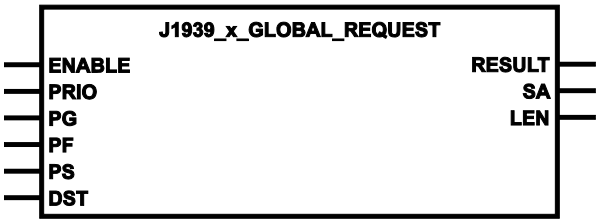
4315

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_J1939_x_Vxxyyzz.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
PRIO	BYTE	Nachrichten-Prioritätin 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 = [PRIO] + [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	Wert 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

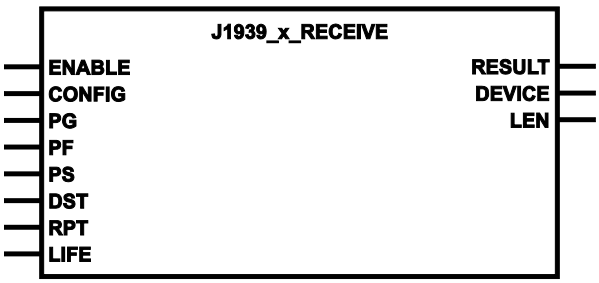
9393

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_J1939_x_Vxxyyzz.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	Wert 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

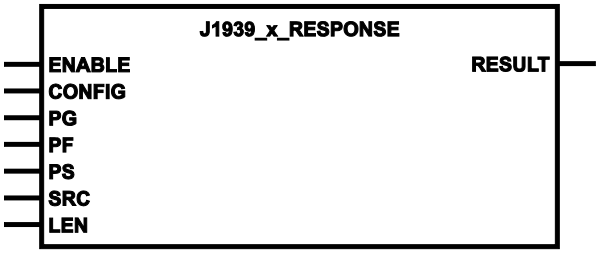
9399

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_J1939_x_Vxxyyzz.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 übertragenen 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 dez	Wert hex	Beschreibung
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

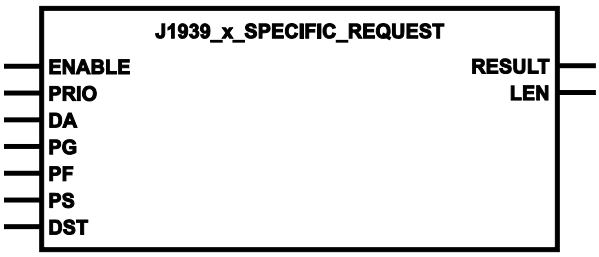
8884

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_J1939_x_Vxxyyzz.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
PRIO	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 = [PRIO] + [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	Wert 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

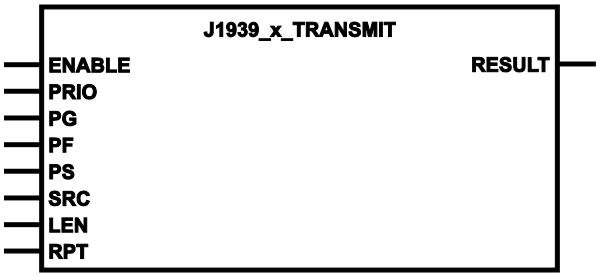
4322

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

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_J1939_x_Vxxyyzz.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
PRIO	BYTE	Nachrichten-Prioritätin 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 = [PRIO] + [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	Wert 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

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_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:

Beschreibung

317

SERIAL_PENDING ermittelt die Anzahl der im seriellen Empfangspuffer gespeicherten Datenbytes.

Im Gegensatz zu *SERIAL_RX* (→ Seite 138) 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_CR0020_Vxxyyzz.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

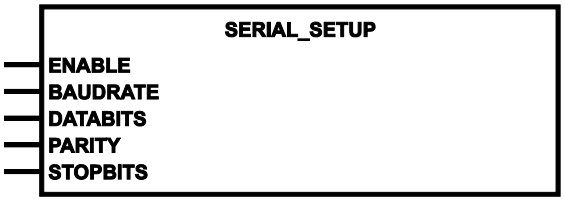
SERIAL_SETUP

302

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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_CR0020_Vxxyyzz.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

8609

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

Bausteine: Interrupts verarbeiten

Inhalt

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.

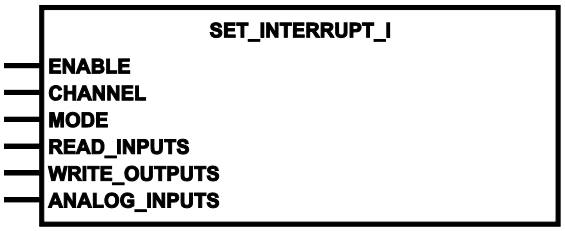
SET_INTERRUPT_I

2381

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

281
11573

SET_INTERRUPT_I organisiert das Ausführen eines Programmteils durch eine Interrupt-Anforderung über einen Eingangskanal.

In der klassischen SPS ist die Zykluszeit das Maß der Dinge für Echtzeitbetrachtungen. Gegenüber kundenspezifischen Steuerungen ist die SPS damit im Nachteil. Auch ein "Echtzeit-Betriebssystem" ändert nichts an dieser Tatsache, wenn das gesamte 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 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 Programmteil immer dann ausgeführt wird, wenn eine Flanke am Eingang CHANNEL erkannt wird. Werden Ein- und Ausgänge in diesem Programmteil genutzt, werden diese ebenfalls in der Interrupt-Routine, ausgelöst durch die Eingangs-Flanke, gelesen oder beschrieben. Über die Eingänge READ_INPUTS, WRITE_OUTPUTS oder ANALOG_INPUTS kann das Lesen oder Schreiben unterbunden werden.

Innerhalb des Programmteils können also alle zeitkritischen Ereignisse bearbeitet werden, indem Eingänge oder globale Variablen verknüpft und Ausgänge beschrieben werden. So können auch Bausteine nur genau dann ausgeführt werden, wenn sie durch ein Eingangssignal angefordert werden.

HINWEIS

Damit der per Interrupt aufgerufene Programmteil nicht zusätzlich zyklisch aufgerufen wird, sollte er (mit Ausnahme des Initialisierungsaufrufes) im Zyklus übersprungen werden.

Der Eingang (CHANNEL), der zum Auslösen des Interrupt überwacht wird, kann in der Interrupt-Routine nicht initialisiert und weiter verarbeitet werden.

Die Eingänge müssen in der Betriebsart IN_FAST sein, sonst können die Interrupts nicht gelesen werden.

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

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

19866

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 Interupt 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.

971

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

2383

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...7) 0...7 für die Eingänge IN0...IN7
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 0..7 vor Aufruf des Programms lesen und in die Eingangsmerker I00...I07 schreiben FALSE: nur den unter CHANNEL angegebenen Kanal lesen und in den dazugehörigen Eingangsmerker Ixx schreiben
WRITE_OUTPUTS	BOOL	TRUE: die aktuellen Werte der Ausgangsmerker Q00...Q07 nach Programmablauf auf die Ausgänge schreiben FALSE: keine Ausgänge schreiben
ANALOG_INPUTS	BOOL	TRUE: die Eingänge 0..7 lesen und die ungefilterten, unkalibrierten Analogwerte in die Merker ANALOG_IRQ00...07 schreiben FALSE: die Merker ANALOG_IRQ00...07 nicht schreiben

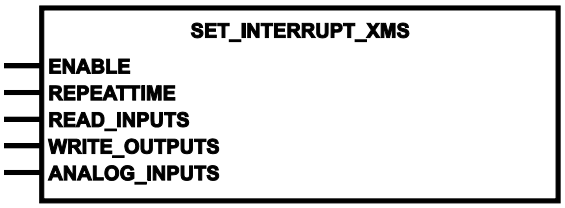
SET_INTERRUPT_XMS

272

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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 Initialisierungsaufrufes) 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.

971

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

2382

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 0..7 vor Aufruf des Programms lesen und in die Eingangsmerker I00...I07 schreiben FALSE: keine Aktualisierung der Eingänge
WRITE_OUTPUTS	BOOL	TRUE: die aktuellen Werte der Ausgangsmerker Q00...Q07 nach Programmablauf auf die Ausgänge schreiben FALSE: keine Ausgänge schreiben
ANALOG_INPUTS	BOOL	TRUE: die Eingänge 0..7 lesen und die ungefilterten, unkalibrierten Analogwerte in die Merker ANALOG_IRQ00...07 schreiben FALSE: die Merker ANALOG_IRQ00...07 nicht schreiben

5.2.8 Bausteine: Eingangswerte verarbeiten

Inhalt

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.

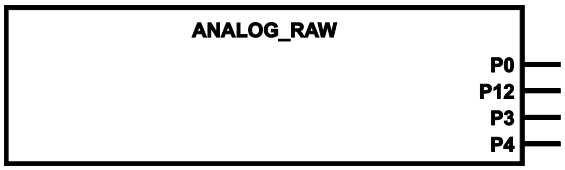
ANALOG_RAW

19589

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

9918

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

Parameter der Ausgänge

13148

Parameter	Datentyp	Beschreibung
P0	ARRAY [0..7] OF WORD	Roh-Eingangswerte der analogen Eingänge, Port 0: P0.0 für I00 ... P0.7 für I07
P12	ARRAY [0..7] OF WORD	Roh-Eingangswerte der analogen Eingänge, Ports 1+2: P12.0 für I14 ... P12.3 für I17 P12.4 für I24 ... P12.7 für I27
P3	ARRAY [0..7] OF WORD	Roh-Eingangswerte der analogen Eingänge, Port 3: P3.0 für I30 ... P3.7 für I37
P4	ARRAY [0..7] OF WORD	Roh-Eingangswerte der analogen Eingänge, Port 4: P4.0 für I40 ... P4.7 für I47

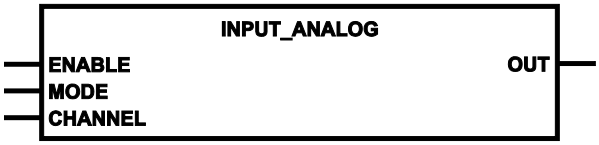
INPUT_ANALOG

519

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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

523

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 = 0...7

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_CR0020_Vxxyyzz.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 150) eingesetzt werden.

Parameter der Eingänge

517

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 = 0...7

Parameter der Ausgänge

518

Parameter	Datentyp	Beschreibung
ACTUAL_CURRENT	WORD	Eingangsstrom in [µA]

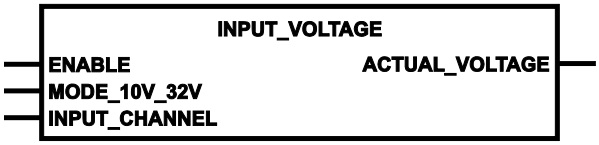
INPUT_VOLTAGE

507

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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 150) eingesetzt werden.

Parameter der Eingänge

511

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 = 0...7

Parameter der Ausgänge

512

Parameter	Datentyp	Beschreibung
ACTUAL_VOLTAGE	WORD	Eingangsspannung in [mV]

5.2.9 Bausteine: analoge Werte anpassen

Inhalt

1603

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

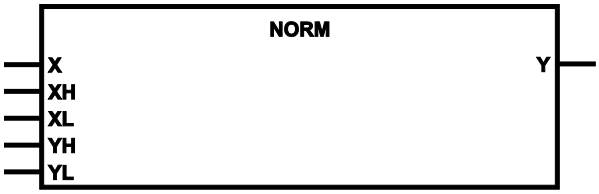
NORM

401

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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 [Inkremente]
XL	WORD	untere Grenze des Eingangswertebereichs [Inkremente]
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

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.

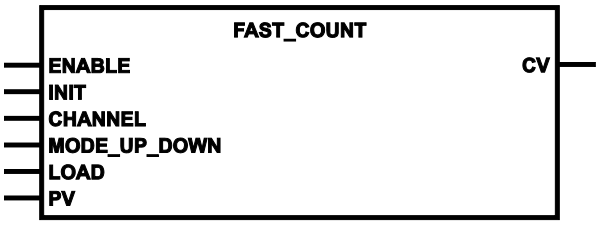
FAST_COUNT

20430

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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 **ecomat*mobile***-Controllern kann der Kanal 0 technisch bedingt nur als Aufwärtszähler eingesetzt werden. Die Kanäle 1...3 können als Auf- und Abwärtszähler genutzt werden.

Parameter der Eingänge

20433

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 I14...I17
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.

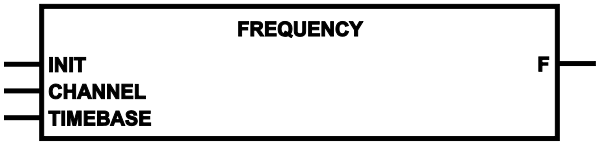
FREQUENCY

20604

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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

20610

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 I14...I17
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

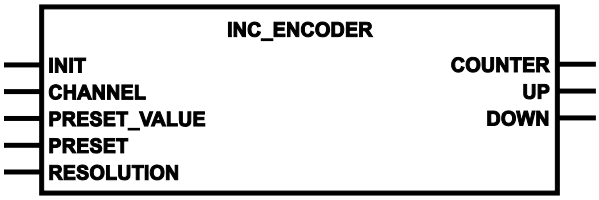
20432

= Incremental Encoder

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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)

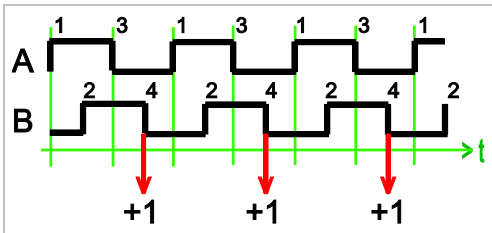
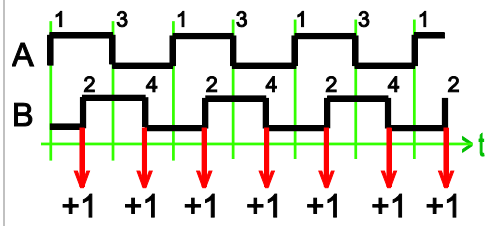
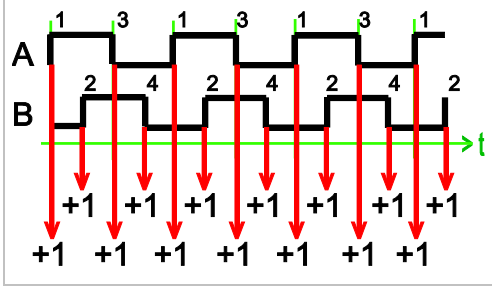
Voreinstellwert setzen:
1. Wert in PRESET_VALUE eintragen
2. PRESET für einen Zyklus auf TRUE setzen
3. PRESET wieder auf FALSE setzen

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 157)
- *FREQUENCY* (→ Seite 158)
- *PERIOD* (→ Seite 162)
- *PERIOD_RATIO* (→ Seite 164)
- *PHASE* (→ Seite 166)

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

20438

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (nur 1 Zyklus lang): Baustein wird initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des Eingangskanal-Paares (0...3) 0 = Kanalpaar 0 = Eingänge I14 + I15 1 = Kanalpaar 1 = Eingänge I16 + I17 2 = Kanalpaar 2 = Eingänge I24 + I25 3 = Kanalpaar 3 = Eingänge I26 + I27
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

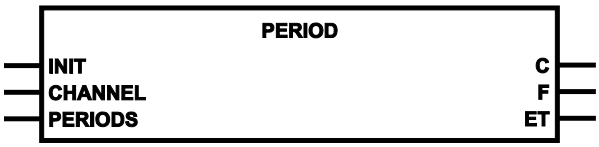
PERIOD

20606

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

373

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

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

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

Der maximale Messbereich beträgt ca. 71 min.

HINWEIS

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

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

Parameter der Eingänge

20608

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 I24...I27
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)

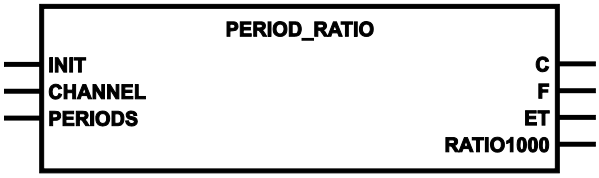
PERIOD_RATIO

20441

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

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

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

Parameter der Eingänge

20446

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 I24...I27
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

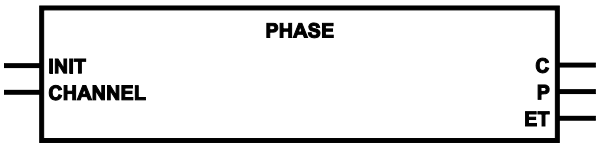
PHASE

20443

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

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

20444

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 I14 + I15 1 = Kanalpaar 1 = Eingänge I16 + I17

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

13758

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

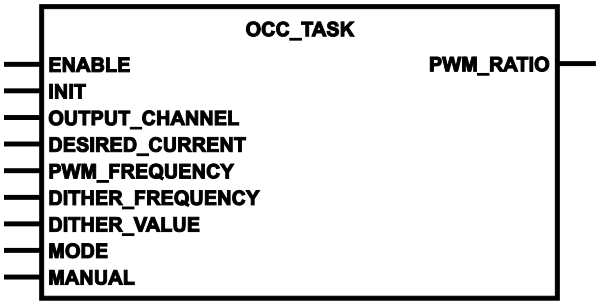
OCC_TASK

20619

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

391

OCC_TASK arbeitet als Stromregler für die PWM-Ausgänge.

Der Regler ist als adaptiver Regler konzipiert, so dass dieser selbstoptimierend arbeitet. Ist das selbstoptimierende Verhalten nicht gewünscht, kann über den Eingang MANUAL ein Wert > 0 (selbstoptimierende Verhalten wird deaktiviert) übergeben werden. Der Zahlenwert repräsentiert einen Korrekturwert, der u.a eine Auswirkung auf den I- und D-Anteil des Reglers hat. Zur Ermittlung der besten Einstellung des Reglers im MANUAL-Modus, bietet sich der Wert 50 an. Je nach gewünschtem Reglerverhalten kann der Wert dann schrittweise vergrößert (Regler wird schärfer / schneller) oder verkleinert (Regler wird schwächer / langsamer) werden.

Ist der Eingang MANUAL auf "0" gesetzt, arbeitet der Regler immer selbstoptimierend. Das Verhalten der Regelstrecke wird ständig überwacht und die aktualisierten Korrekturwerte werden automatisch in jedem Zyklus dauerhaft gespeichert. Veränderungen in der Regelstrecke werden somit sofort erkannt und korrigiert.

HINWEIS

OCC_TASK arbeitet mit einer festen Zykluszeit von 5 ms. Es müssen auch keine Istwerte zugeführt werden, da diese schon funktionsintern erfasst werden.

OCC_TASK basiert auf *PWM* (→ Seite 173).

Wird OUTPUT_CURRENT_CONTROL für die Ausgänge 4...7 genutzt, darf bei gleichzeitiger Verwendung der PWM-Ausgänge 8...11 auch dort nur der PWM-Funktionsbaustein eingesetzt werden.

► Bei der Definition des Parameters DITHER_VALUE darauf achten, dass das resultierende PWM-Ratio im Arbeitsbereich der Regelung zwischen 0...100 % bleibt:
  • PWM-Ratio + DITHER_VALUE < 100 % und
  • PWM-Ratio - DITHER_VALUE > 0 %.
  Außerhalb dieses zulässigen Bereichs kann der im Parameter DESIRED_CURRENT angegebene Strom nicht erreicht werden.

Parameter der Eingänge

20620

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
INIT	BOOL	FALSE ⇨ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
OUTPUT_CHANNEL	BYTE	Nummer des stromgeregelten Ausgangskanals (0...7) 0…3 für die Ausgänge Q10...Q13 4...7 für die Ausgänge Q20...Q23
DESIRED_CURRENT	WORD	Stromsollwert des Ausgangs in [mA]
PWM_FREQUENCY	WORD	PWM-Frequenz [Hz] für die Last am Ausgang
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.
DITHER_VALUE	BYTE	Spitze-Spitze-Wert des Dithers in [%] zulässige Werte = 0...100 = 0x00...0x64
MODE	BYTE	Reglercharakteristik: 0 = sehr langsamer Anstieg, kein Überschwingen 1 = langsamer Anstieg, kein Überschwingen 2 = minimales Überschwingen 3 = mäßiges Überschwingen zulässig
MANUAL	BYTE	Wert = 0: Regler arbeitet selbstoptimierend Wert > 0: Das selbstoptimierende Verhalten des Reglers wird überschrieben (typisch: 50)

Parameter der Ausgänge

393

Parameter	Datentyp	Beschreibung
PWM_RATIO	BYTE	Zu Kontrollzwecken: Anzeige PWM-Tastverhältnis 0...99 %

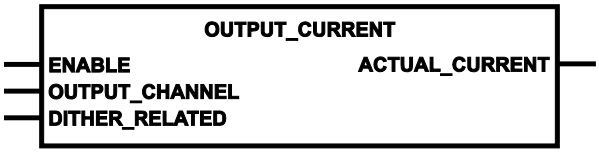
OUTPUT_CURRENT

20449

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

385

OUTPUT_CURRENT dient dem Messen des Stroms (optional: Mittelung über Dither-Periode) an einem Ausgangskanal.

Der FB liefert den aktuellen Ausgangsstrom, wenn die Ausgänge als PWM-Ausgänge oder als plus-schaltend benutzt werden. Die Strommessung erfolgt innerhalb des Gerätes, es werden also keine externen Messwiderstände benötigt.

Parameter der Eingänge

20451

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
OUTPUT_CHANNEL	BYTE	Nummer des stromgeregelten Ausgangskanals (0...7) 0…3 für die Ausgänge Q10...Q13 4...7 für die Ausgänge Q20...Q23
DITHER_RELATED	BOOL	Strom wird ermittelt als Mittelwert über... TRUE: eine Dither-Periode FALSE: eine PWM-Periode

Parameter der Ausgänge

387

Parameter	Datentyp	Beschreibung
ACTUAL_CURRENT	WORD	Ausgangsstrom in [mA]

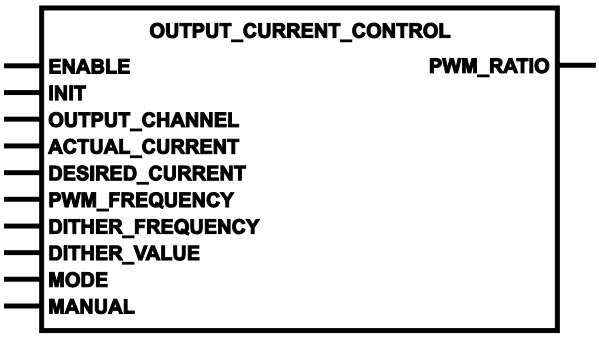
OUTPUT_CURRENT_CONTROL

20453

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

20454

OUTPUT_CURRENT_CONTROL arbeitet als Stromregler für die PWM-Ausgänge.

Der Regler ist als adaptiver Regler konzipiert, so dass dieser selbstoptimierend arbeitet. Ist das selbstoptimierende Verhalten nicht gewünscht, kann über den Eingang MANUAL ein Wert > 0 übergeben werden; damit wird das selbstoptimierende Verhalten deaktiviert. Der Zahlenwert repräsentiert einen Korrekturwert, der u.a eine Auswirkung auf den I- und D-Anteil des Reglers hat. Zur Ermittlung der besten Einstellung des Reglers im MANUAL-Modus, bietet sich der Wert 50 an. Je nach gewünschtem Reglerverhalten kann der Wert dann schrittweise vergrößert (Regler wird schärfer / schneller) oder verkleinert (Regler wird schwächer / langsamer) werden.

Ist der Baustein-Eingang MANUAL auf "0" gesetzt, arbeitet der Regler immer selbstoptimierend. Das Verhalten der Regelstrecke wird ständig überwacht und die aktualisierten Korrekturwerte werden automatisch in jedem Zyklus dauerhaft gespeichert. Veränderungen in der Regelstrecke werden somit sofort erkannt und korrigiert.

HINWEIS

Um einen stabilen Ausgangswert zu bekommen, sollte OUTPUT_CURRENT_CONTROL zyklisch in gleichmäßigen Zeitabständen aufgerufen werden.

OUTPUT_CURRENT_CONTROL basiert auf *PWM* (→ Seite 173).

Wird OUTPUT_CURRENT_CONTROL für die Ausgänge 4...7 genutzt, darf bei gleichzeitiger Verwendung der PWM-Ausgänge 8...11 auch dort nur der PWM-Funktionsbaustein eingesetzt werden.

► Bei der Definition des Parameters DITHER_VALUE darauf achten, dass das resultierende PWM-Ratio im Arbeitsbereich der Regelung zwischen 0...100 % bleibt:
  • PWM-Ratio + DITHER_VALUE < 100 % und
  • PWM-Ratio - DITHER_VALUE > 0 %.
  Außerhalb dieses zulässigen Bereichs kann der im Parameter DESIRED_CURRENT angegebene Strom nicht erreicht werden.

Parameter der Eingänge

20455

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
INIT	BOOL	FALSE ⇨ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
OUTPUT_CHANNEL	BYTE	Nummer des stromgeregelten Ausgangskanals (0...7) 0…3 für die Ausgänge Q10...Q13 4...7 für die Ausgänge Q20...Q23
ACTUAL_CURRENT	WORD	Aktueller Strom des PWM-Ausgangs in [mA] ► Den Ausgangswert von *OUTPUT_CURRENT* (→ Seite 170) dem Eingang ACTUAL_CURRENT zuführen!
DESIRED_CURRENT	WORD	Stromsollwert des Ausgangs in [mA]
PWM_FREQUENCY	WORD	PWM-Frequenz [Hz] für die Last am Ausgang
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.
DITHER_VALUE	BYTE	Spitze-Spitze-Wert des Dithers in [%] zulässige Werte = 0...100 = 0x00...0x64
MODE	BYTE	Reglercharakteristik: 0 = sehr langsamer Anstieg, kein Überschwingen 1 = langsamer Anstieg, kein Überschwingen 2 = minimales Überschwingen 3 = mäßiges Überschwingen zulässig
MANUAL	BYTE	Wert = 0: Regler arbeitet selbstoptimierend Wert > 0: Das selbstoptimierende Verhalten des Reglers wird überschrieben (typisch: 50)

Parameter der Ausgänge

381

Parameter	Datentyp	Beschreibung
PWM_RATIO	BYTE	Zu Kontrollzwecken: Anzeige PWM-Tastverhältnis 0...99 %

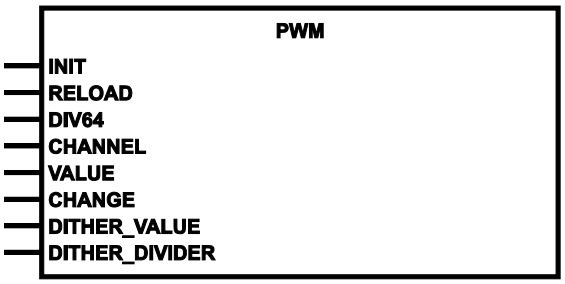
PWM

20457

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

20467

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 4...11 gleich sein.
>
> Bei diesen Kanälen dürfen PWM und *PWM1000* (→ Seite 179) nicht gemischt werden.
>
> Die PWM-Frequenz (und damit der RELAOD-Wert) ist intern auf 5 kHz begrenzt.

Je nachdem, ob eine hohe oder niedrige PWM-Frequenz benötigt wird, muss der Eingang DIV64 auf FALSE (0) oder TRUE (1) gesetzt werden.

Während des zyklischen Programmablaufes ist INIT auf FALSE gesetzt. Der FB wird aufgerufen und dabei der neue PWM-Wert übergeben. Der Wert wird übernommen, wenn der Eingang CHANGE = TRUE ist.

Eine Strommessung für den initialisierten PWM-Kanal kann realisiert werden:
- mit *OUTPUT_CURRENT* (→ Seite 170)
- oder 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

20458

Parameter	Datentyp	Beschreibung
INIT	BOOL	FALSE ⇨ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
RELOAD	WORD	Wert zur Festlegung der PWM-Frequenz (→ Kapitel *Berechnung des RELOAD-Wertes* (→ Seite 175))
DIV64	BOOL	CPU-Takt / 64
CHANNEL	BYTE	Nummer des PWM-Ausgangskanals (0...15) 0…3 für die Ausgänge Q10...Q13 4...7 für die Ausgänge Q20...Q23 8...15 für die Ausgänge Q40...Q47
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

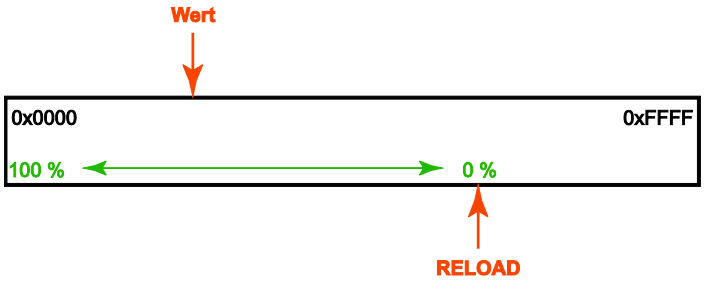
Abhängig vom Ventiltyp wird eine entsprechende PWM-Frequenz benötigt. Die PWM-Frequenz wird bei der PWM-Funktion über den Reload-Wert (Funktion PWM) oder direkt als Zahlenwert in Hz (Funktion PWM1000) übergeben. Je nach R360-Controller unterscheiden sich die PWM-Ausgänge in ihrer Arbeits-, aber nicht 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	RELOAD = 20 MHz / f_{PWM}	RELOAD = 10 MHz / f_{PWM}
DIV64 = 1	RELOAD = 312,5 kHz / f_{PWM}	RELOAD = 156,25 kHz / f_{PWM}

Je nachdem, ob eine hohe oder niedrige PWM-Frequenz benötigt wird, muss der Eingang DIV64 auf FALSE (0) oder TRUE (1) gesetzt werden. Bei PWM-Frequenzen unter 305 Hz oder 152 Hz (je nach Controller) muss DIV64 auf "1" gesetzt werden, damit der Reload-Wert nicht größer als 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. $\frac{20\ \text{MHz}}{400\ \text{Hz}} = 50\,000 = 0\text{xC350} = \text{RELOAD}$		Die PWM-Frequenz soll 200 Hz betragen. $\frac{10\ \text{MHz}}{200\ \text{Hz}} = 50\,000 = 0\text{xC350} = \text{RELOAD}$
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 206) 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 **ecomat*mobile***-DVD "Software, tools and documentation" wurden auch Projekte mit Beispielen auf Ihrem Computer im Programmverzeichnis abgelegt:
`…\ifm electronic\CoDeSys V…\Projects\DEMO_PLC_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.

PWM100

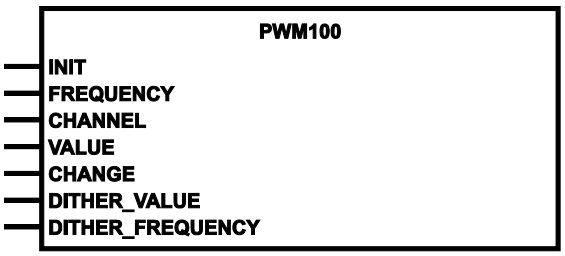
20461

Baustein-Typ = Funktionsbaustein (FB)

Neue **ecomat*mobile***-Controller unterstützen nur noch *PWM1000* (→ Seite 179).

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

20462

PWM100 organisiert die Initialisierung und Parametrierung der PWM-Ausgänge.

Der FB ermöglicht eine einfache Anwendung der PWM-Funktion im Gerät. Die PWM-Frequenz kann direkt in [Hz] und das Puls-Pausen-Verhältnis in 1 %-Schritten angegeben werden. Zum Aufbau von Reglern ist dieser Baustein durch die relativ grobe Abstufung **nicht** geeignet.

Der FB wird einmalig für jeden Kanal in der Initialisierung des 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 4...11 gleich sein.

Bei diesen Kanälen dürfen *PWM* (→ Seite 173) und PWM100 nicht gemischt werden.

Die PWM-Frequenz ist intern auf 5 kHz begrenzt.

Während des zyklischen Programmablaufes ist INIT auf FALSE gesetzt. Der FB wird aufgerufen und dabei der neue PWM-Wert übergeben. Der Wert wird übernommen, wenn der Eingang CHANGE = TRUE ist.

Eine Strommessung für den initialisierten PWM-Kanal kann realisiert werden:
- mit *OUTPUT_CURRENT* (→ Seite 170)
- oder 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

20463

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...15) 0…3 für die Ausgänge Q10...Q13 4...7 für die Ausgänge Q20...Q23 8...15 für die Ausgänge Q40...Q47
VALUE	BYTE	aktueller PWM-Wert
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	BYTE	Spitze-Spitze-Wert des Dithers in [%] zulässige Werte = 0...100 = 0x00...0x64
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.

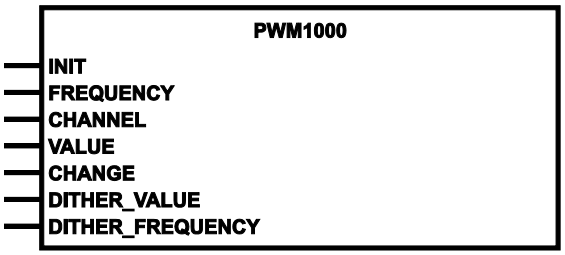
PWM1000

20465

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

20466

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 4...11 gleich sein.

Bei diesen Kanälen dürfen *PWM* (→ Seite 173) und PWM1000 nicht gemischt werden.

Die PWM-Frequenz ist intern auf 5 kHz begrenzt.

Während des zyklischen Programmablaufes ist INIT auf FALSE gesetzt. Der FB wird aufgerufen und dabei der neue PWM-Wert übergeben. Der Wert wird übernommen, wenn der Eingang CHANGE = TRUE ist.

Eine Strommessung für den initialisierten PWM-Kanal kann realisiert werden:
- mit *OUTPUT_CURRENT* (→ Seite 170)
- oder 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

20471

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...15) 0…3 für die Ausgänge Q10...Q13 4...7 für die Ausgänge Q20...Q23 8...15 für die Ausgänge Q40...Q47
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

19540

Die Bibliothek `ifm_hydraulic_16bitOS05_Vxxyyzz.LIB` enthält folgende Bausteine:

CONTROL_OCC (→ Seite 182)	OCC = Output Current Control (= stromgeregelter Ausgang) skaliert den Eingangswert [WORD] auf einen angegebenen Strombereich
JOYSTICK_0 (→ Seite 185)	skaliert Signale [INT] aus einem Joystick auf fest definierte Kennlinien, normiert auf 0...1000
JOYSTICK_1 (→ Seite 188)	skaliert Signale [INT] aus einem Joystick auf parametrierbare Kennlinien, normiert auf 0...1000
JOYSTICK_2 (→ Seite 192)	skaliert Signale [INT] aus einem Joystick auf einen parametrierbaren Kennlinien-Verlauf; die Normierung ist frei bestimmbar
NORM_HYDRAULIC (→ Seite 195)	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.

Aus der Bibliothek `ifm_CR0020_Vxxyyzz.LIB` werden folgende Bausteine benötigt:

OUTPUT_CURRENT (→ Seite 170)	misst den Strom (Mittelung über Dither-Periode) an einem Ausgangskanal
OUTPUT_CURRENT_CONTROL (→ Seite 171)	Stromregler für einen PWMi-Ausgangskanal

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

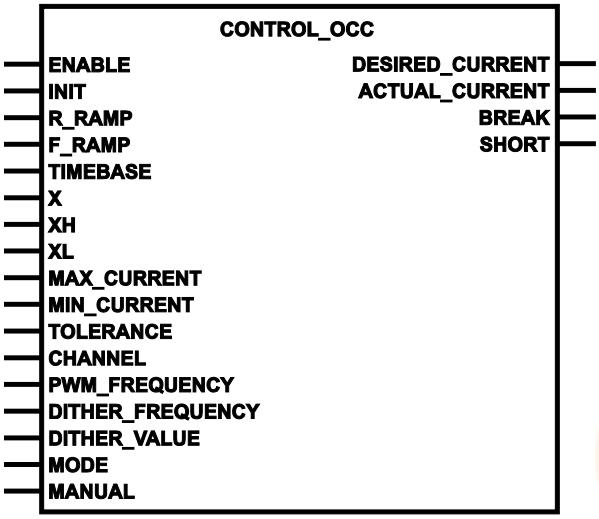
CONTROL_OCC

6245

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_HYDRAULIC_16bitOS05_Vxxyyzz.Lib`

Symbol in CODESYS:



Beschreibung

600

CONTROL_OCC skaliert den Eingangswert X auf einen angegebenen Strombereich.

Jede Instanz des FB wird in jedem SPS-Zyklus einmalig aufgerufen. Der FB nutzt *OUTPUT_CURRENT_CONTROL* (→ Seite 171) und *OUTPUT_CURRENT* (→ Seite 170) aus der Bibliothek `ifm_CR0020_Vxxyyzz.LIB` . Der Regler ist als adaptiver Regler konzipiert, so dass dieser selbstoptimierend arbeitet.

Ist das selbstoptimierende Verhalten nicht gewünscht, kann über den Eingang MANUAL ein Wert > 0 übergeben werden ⇨ das selbstoptimierende Verhalten wird deaktiviert.

Der Zahlenwert in MANUAL repräsentiert einen Korrekturwert, der u. a. eine Auswirkung auf den I- und den D-Anteil des Reglers hat. Zur Ermittlung der besten Einstellung des Reglers im MANUAL-Modus bietet sich der Wert 50 an.

Wert MANUAL vergrößern: ⇨ Regler wird schärfer / schneller
Wert MANUAL verkleinern: ⇨ Regler wird schwächer / langsamer

Ist der Eingang MANUAL auf "0" gesetzt, arbeitet der Regler immer selbstoptimierend. Das Verhalten der Regelstrecke wird ständig überwacht und die aktualisierten Korrekturwerte werden automatisch in jedem Zyklus dauerhaft gespeichert. Veränderungen in der Regelstrecke werden somit sofort erkannt und korrigiert.

Der Eingang X von CONTROL_OCC sollte von einem Ausgang der JOYSTICK-FBs gespeist werden.

Parameter der Eingänge

6247

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
INIT	BOOL	FALSE ⇨ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
R_RAMP	INT	Steigende Flanke der Rampe in [Inkremente/SPS-Zyklus] oder [Inkremente/TIMEBASE] 0 = ohne Rampe
F_RAMP	INT	Fallende Flanke der Rampe in [Inkremente/SPS-Zyklus] oder [Inkremente/TIMEBASE] 0 = ohne Rampe
TIMEBASE	TIME	Referenz für steigende und fallende Flanke der Rampe: t#0s = steigende / fallende Flanke in [Inkremente/SPS-Zyklus] Schnelle Controller haben sehr kurze Zykluszeiten! sonst = steigende / fallende Flanke in [Inkremente/TIMEBASE]
X	WORD	Eingangswert
XH	WORD	obere Grenze des Eingangswertebereichs [Inkremente]
XL	WORD	untere Grenze des Eingangswertebereichs [Inkremente]
MAX_CURRENT	WORD	Max. Ventilstrom in [mA]
MIN_CURRENT	WORD	Min. Ventilstrom in [mA]
TOLERANCE	BYTE	Toleranz für min. Ventilstrom in [Inkremente] Bei Überschreiten der Toleranz erfolgt Sprung auf MIN_CURRENT
CHANNEL	BYTE	Nummer des stromgeregelten Ausgangskanals (0...7) 0…3 für die Ausgänge Q10...Q13 4...7 für die Ausgänge Q20...Q23 Für den FB xxx_E (falls vorhanden) gilt: 0…3 für die Ausgänge Q10_E...Q13_E 4...7 für die Ausgänge Q20_E...Q23_E
PWM_FREQUENCY	WORD	PWM-Frequenz [Hz] für die Last am Ausgang
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.
DITHER_VALUE	BYTE	Spitze-Spitze-Wert des Dithers in [%] zulässige Werte = 0...100 = 0x00...0x64
MODE	BYTE	Reglercharakteristik: 0 = sehr langsamer Anstieg, kein Überschwingen 1 = langsamer Anstieg, kein Überschwingen 2 = minimales Überschwingen 3 = mäßiges Überschwingen zulässig
MANUAL	BYTE	Wert = 0: Regler arbeitet selbstoptimierend Wert > 0: Das selbstoptimierende Verhalten des Reglers wird überschrieben (typisch: 50)

Parameter der Ausgänge

602

Parameter	Datentyp	Beschreibung
DESIRED_CURRENT	WORD	Stromsollwert in [mA] für OCC (zu Kontrollzwecken)
ACTUAL_CURRENT	WORD	Ausgangsstrom in [mA]
BREAK	BOOL	Fehler: Leitung am Ausgang unterbrochen
SHORT	BOOL	Fehler: Kurzschluss in Leitung am Ausgang

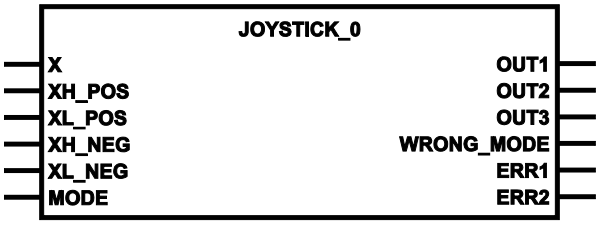
JOYSTICK_0

13224

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_hydraulic_16bitOS05_Vxxyyzz.Lib`

Symbol in CODESYS:



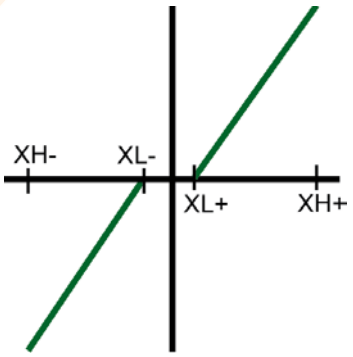
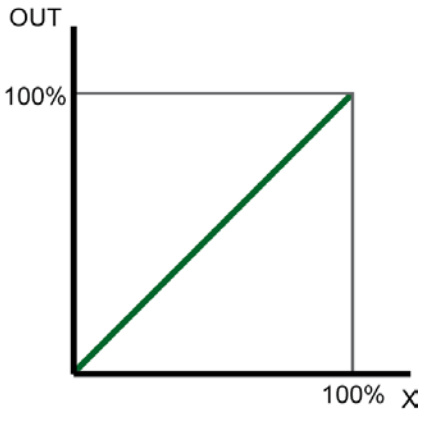
Beschreibung

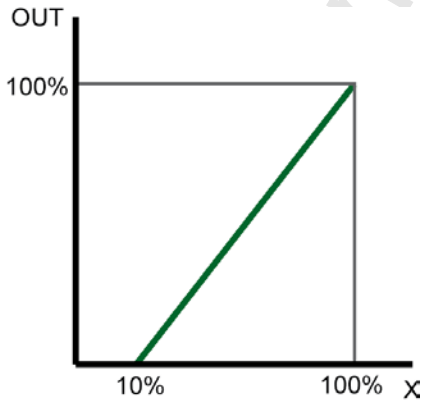
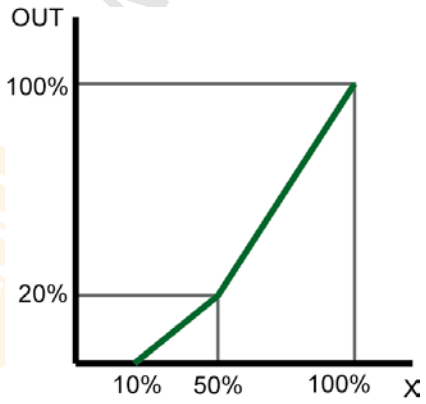
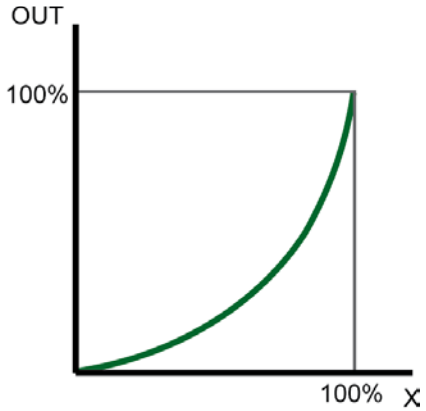
432

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 [Inkremente]
XH_POS	INT	Max. Sollwert positive Richtung [Inkremente] (auch negative Werte zulässig)
XL_POS	INT	Min. Sollwert positive Richtung [Inkremente] (auch negative Werte zulässig)
XH_NEG	INT	Max. Sollwert negative Richtung [Inkremente] (auch negative Werte zulässig)
XL_NEG	INT	Min. Sollwert negative Richtung [Inkremente] (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 `util.lib`) (mögliche Meldungen → folgende Tabelle)
ERR2	BYTE	Fehler-Code für fallende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der `util.lib`) (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

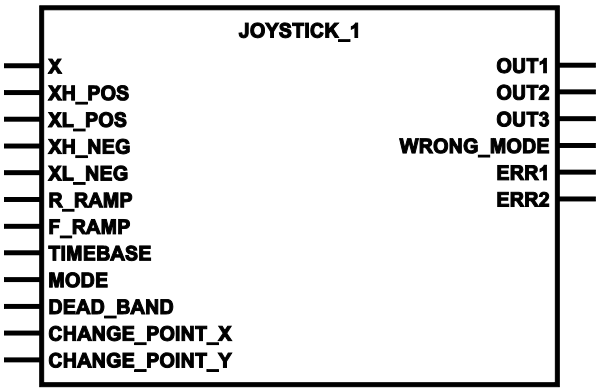
JOYSTICK_1

13227

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_hydraulic_16bitOS05_Vxxyyzz.Lib`

Symbol in CODESYS:

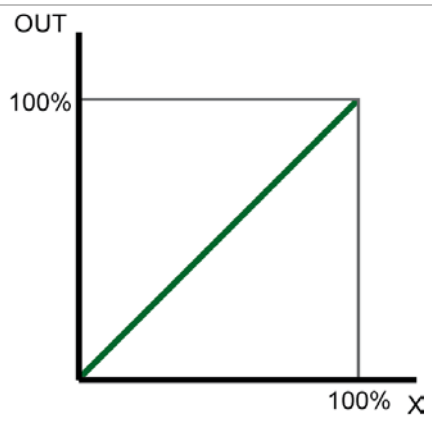
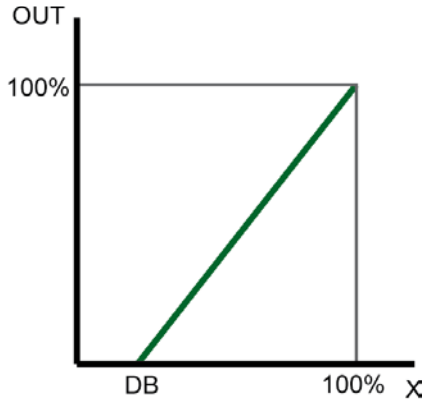


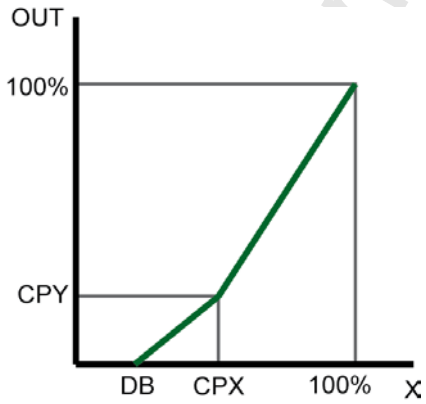
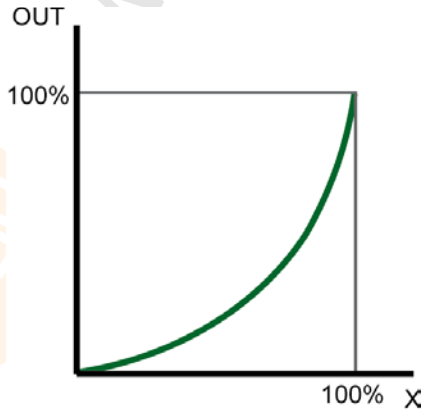
Beschreibung

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 0: Kennlinie linear 100 % = 1000 Inkremente	
Modus 1: Kennlinie linear mit Totbereich Wert für Totbereich (DB) einstellbar in % von 1000 Inkrementen 100 % = 1000 Inkremente DB = Dead_Band	

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 [Inkremente]
XH_POS	INT	Max. Sollwert positive Richtung [Inkremente] (auch negative Werte zulässig)
XL_POS	INT	Min. Sollwert positive Richtung [Inkremente] (auch negative Werte zulässig)
XH_NEG	INT	Max. Sollwert negative Richtung [Inkremente] (auch negative Werte zulässig)
XL_NEG	INT	Min. Sollwert negative Richtung [Inkremente] (auch negative Werte zulässig)
R_RAMP	INT	Steigende Flanke der Rampe in [Inkremente/SPS-Zyklus] 0 = keine Rampe
F_RAMP	INT	Fallende Flanke der Rampe in [Inkremente/SPS-Zyklus] 0 = keine Rampe
TIMEBASE	TIME	Referenz für steigende und fallende Flanke der Rampe: t#0s = steigende / fallende Flanke in [Inkremente/SPS-Zyklus] Schnelle Controller haben sehr kurze Zykluszeiten! sonst = steigende / fallende Flanke in [Inkremente/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 `util.lib`) (mögliche Meldungen → folgende Tabelle)
ERR2	BYTE	Fehler-Code für fallende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der `util.lib`) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERR1 und ERR2:

Wert dez	Wert 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

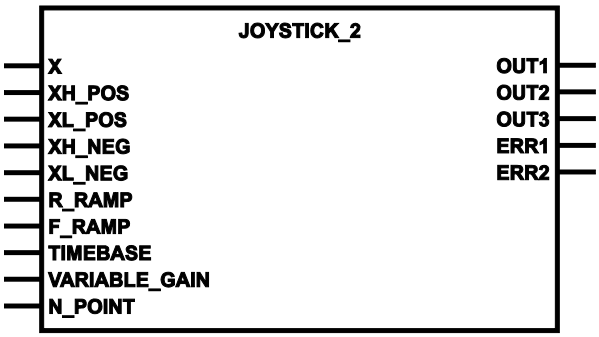
JOYSTICK_2

13228

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_hydraulic_16bitOS05_Vxxyyzz.Lib`

Symbol in CODESYS:

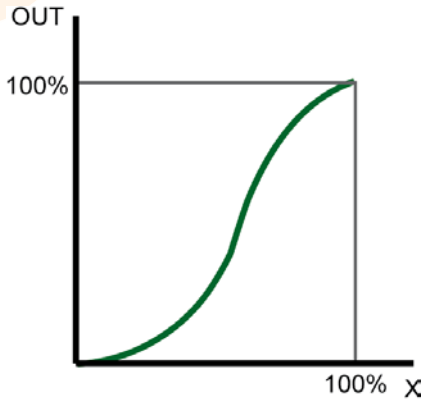


Beschreibung

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):

Kennlinie frei parametrierbar	

Parameter der Eingänge

6261

Parameter	Datentyp	Beschreibung
X	INT	Eingangswert [Inkremente]
XH_POS	INT	Max. Sollwert positive Richtung [Inkremente] (auch negative Werte zulässig)
XL_POS	INT	Min. Sollwert positive Richtung [Inkremente] (auch negative Werte zulässig)
XH_NEG	INT	Max. Sollwert negative Richtung [Inkremente] (auch negative Werte zulässig)
XL_NEG	INT	Min. Sollwert negative Richtung [Inkremente] (auch negative Werte zulässig)
R_RAMP	INT	Steigende Flanke der Rampe in [Inkremente/SPS-Zyklus] 0 = keine Rampe
F_RAMP	INT	Fallende Flanke der Rampe in [Inkremente/SPS-Zyklus] 0 = keine Rampe
TIMEBASE	TIME	Referenz für steigende und fallende Flanke der Rampe: t#0s = steigende / fallende Flanke in [Inkremente/SPS-Zyklus] Schnelle Controller haben sehr kurze Zykluszeiten! sonst = steigende / fallende Flanke in [Inkremente/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 `util.lib`) (mögliche Meldungen → folgende Tabelle)
ERR2	BYTE	Fehler-Code für fallende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der `util.lib`) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERR1 und ERR2:

Wert dez	Wert 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

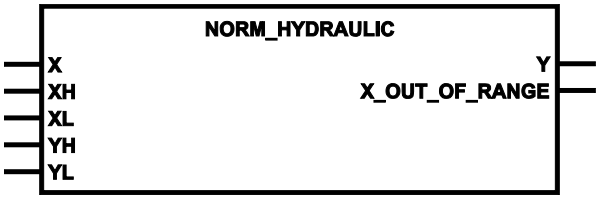
NORM_HYDRAULIC

13232

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_hydraulic_16bitOS05_Vxxyyzz.Lib`

Symbol in CODESYS:



Beschreibung

397

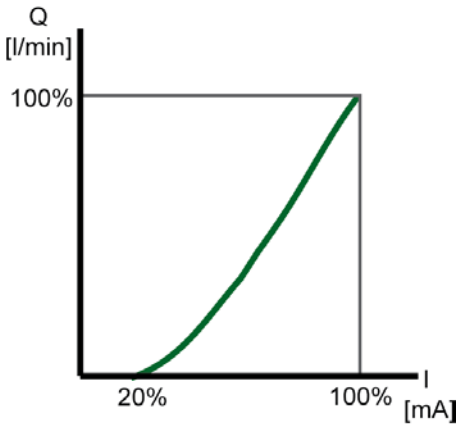
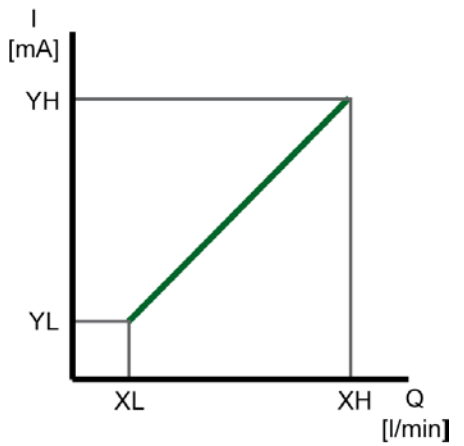
NORM_HYDRAULIC normiert Eingangswerte innerhalb festgesetzter Grenzen auf Werte mit neuen Grenzen.

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.

Typischer Kennlinienverlauf eines Hydraulikventils: Erst bei ca. 20 % des Spulenstroms beginnt der Ölfluss. Der Ölfluss ist anfänglich nicht linear.	
Charakteristik des Funktionsbausteins	

Parameter der Eingänge

398

Parameter	Datentyp	Beschreibung
X	DINT	Eingangswert
XH	DINT	Max. Eingangswert [Inkremente]
XL	DINT	Min. Eingangswert [Inkremente]
YH	DINT	Max. Ausgangswert [Inkremente], z.B.: Ventilstrom [mA], Durchfluss [l/min]
YL	DINT	Min. Ausgangswert [Inkremente], 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-Inkremente.
- Fall 2:
  Eingang mit relativ grober Auflösung.
  Ausgang mit hoher Auflösung.
  1 X-Inkrement ergibt 20 Y-Inkremente.
  Ausgangssignal ist gegenüber dem Eingangssignal invertiert.
- Fall 3:
  Eingang mit hoher Auflösung.
  Ausgang mit relativ grober Auflösung.
  20 X-Inkremente ergeben 1 Y-Inkrement.

5.2.13 Bausteine: Regler

Inhalt

1634

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

Einstellregel für einen Regler

1627

Für Regelstrecken, deren Zeitkonstanten nicht bekannt sind, ist das Einstellverfahren nach Ziegler und Nickols im geschlossenen Regelkreis vorteilhaft:

Einstellregel

1628

Die Regeleinrichtung wird zunächst als eine reine P-Regeleinrichtung betrieben. Dazu wird die Vorhaltezeit T_V auf 0 und die Nachstellzeit T_N auf einen sehr großen Wert (ideal auf unendlich) für eine träge Strecke eingestellt. Bei einer schnellen Regelstrecke sollte ein kleines T_N gewählt werden.

Der Proportionalbeiwert KP wird anschließend solange vergrößert, bis die Regel- und die Stellabweichung bei KP = $KP_{kritisch}$ Dauerschwingungen mit konstanter Amplitude ausführen. Es ist damit die Stabilitätsgrenze erreicht.

Anschließend muss die Periodendauer $T_{kritisch}$ der Dauerschwingung ermittelt werden.

Nur bei Bedarf einen D-Anteil hinzufügen.

T_V sollte ca. 2...10-mal kleiner sein als T_N.

KP sollte gleich groß wie KD gewählt werden.

Idealisiert ist die Regelstrecke wie folgt einzustellen:

Regeleinrichtung	KP = KD	TN	TV
P	2,0 • $KP_{kritisch}$	—	—
PI	2,2 • $KP_{kritisch}$	0,83 • $T_{kritisch}$	—
PID	1,7 • $KP_{kritisch}$	0,50 • $T_{kritisch}$	0,125 • $T_{kritisch}$

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

Dämpfung von Überschwingungen

1629

Um Überschwingungen zu dämpfen, kann *PT1* (→ Seite 206) (Tiefpass) eingesetzt werden. Dazu wird der Sollwert XS durch das PT1-Glied gedämpft, bevor er der Reglerfunktion zugeführt wird.

Die Einstellgröße T1 sollte ca. 4...5-mal größer sein als TN des Reglers.

DELAY

585

Baustein-Typ = Funktionsbaustein (FB)

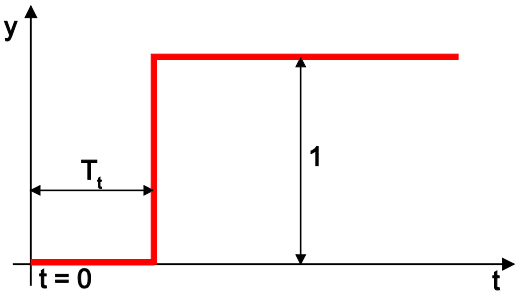
Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

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

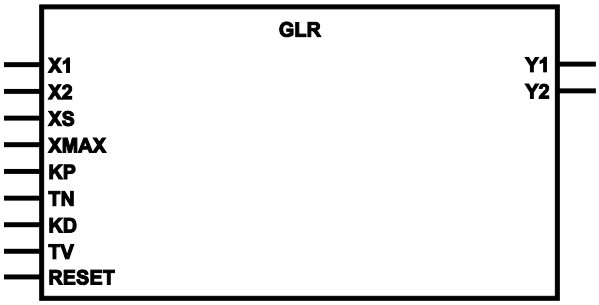
GLR

531

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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

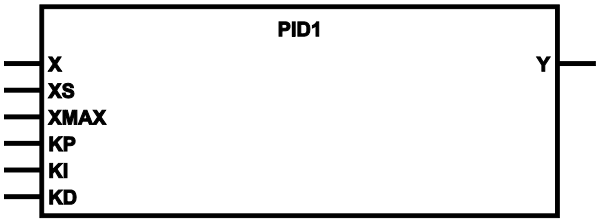
PID1

351

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

354

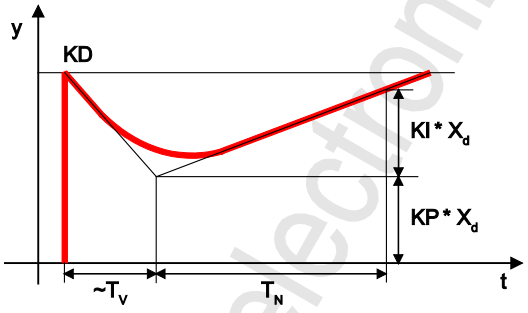
PID1 organisiert einen PID-Regler.

Die Änderung der Stellgröße eines PID-Reglers setzt sich aus einem proportionalen, integralen und differentialen Anteil zusammen. Die Stellgröße ändert sich zunächst um einen von der Änderungsgeschwindigkeit der Eingangsgröße abhängigen Betrag (D-Anteil). Nach Ablauf der Vorhaltezeit geht die Stellgröße auf den dem Proportionalbereich entsprechenden Wert zurück und ändert sich dann entsprechend der Nachstellzeit.

> **HINWEIS**
>
> Die Stellgröße Y ist bereits auf die PWM-Funktion normiert (RELOAD-Wert = 65 535). Beachten Sie dabei die umgekehrte Logik:
> 65 535 = minimaler Wert
> 0 = maximaler Wert.
>
> Beachten Sie, dass die Eingangsgrößen KI und KD zykluszeitabhängig sind. Um ein stabiles, reproduzierbares Regelverhalten zu bekommen, sollte der FB zeitgesteuert aufgerufen werden.

Wenn X > XS, dann wird die Stellgröße erhöht.
Wenn X < XS, dann wird die Stellgröße reduziert.

Die Stellgröße Y hat folgenden zeitlichen Verlauf:



Grafik: Typische Sprungantwort eines PID-Reglers

Parameter der Eingänge

355

Parameter	Datentyp	Beschreibung
X	WORD	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.

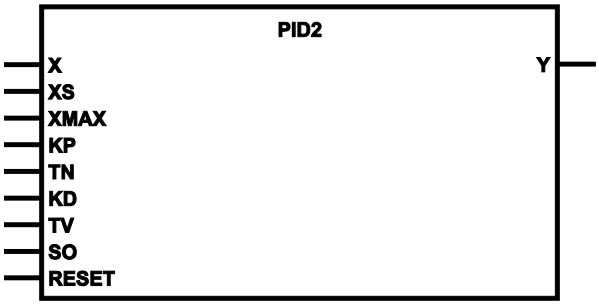
PID2

9167

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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 (**D**ifferential-Anteil). Nach Ablauf der Vorhaltezeit TV geht die Stellgröße auf den dem Proportionalbereich entsprechenden Wert zurück und ändert sich dann entsprechend der Nachstellzeit TN.

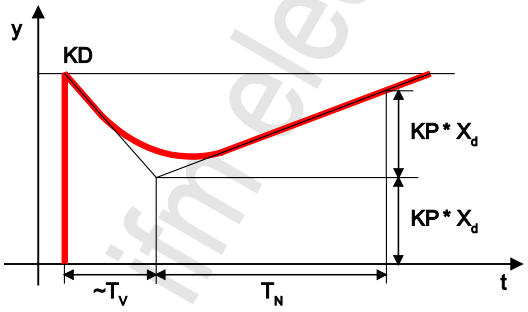
Die an den Eingängen KP und KD eingegebenen Werte werden intern durch 10 geteilt. Damit kann eine feinere Abstufung erreicht werden (z.B: KP = 17, das entspricht 1,7).

> **HINWEIS**
>
> Die Stellgröße Y ist bereits auf die PWM-Funktion normiert (RELOAD-Wert = 65 535). Beachten Sie dabei die umgekehrte Logik:
> 65 535 = minimaler Wert
> 0 = maximaler Wert.
>
> Beachten Sie, dass die Eingangsgröße KD zykluszeitabhängig ist. Um ein stabiles, reproduzierbares Regelverhalten zu bekommen, sollte der FB zeitgesteuert aufgerufen werden.

Wenn X > XS, dann wird die Stellgröße erhöht.
Wenn X < XS, dann wird die Stellgröße reduziert.

Eine Führungsgröße wird intern zur Stellgröße hinzuaddiert:
Y = Y + 65 536 - (XS / XMAX * 65 536).

Die Stellgröße Y hat folgenden zeitlichen Verlauf.



Grafik: Typische Sprungantwort eines PID-Reglers

Parameter der Eingänge

348

Parameter	Datentyp	Beschreibung
X	WORD	Eingangswert
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)
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 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.

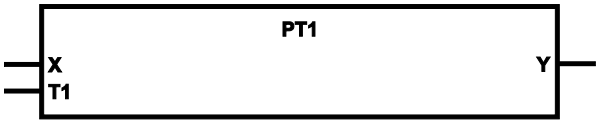
PT1

338

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

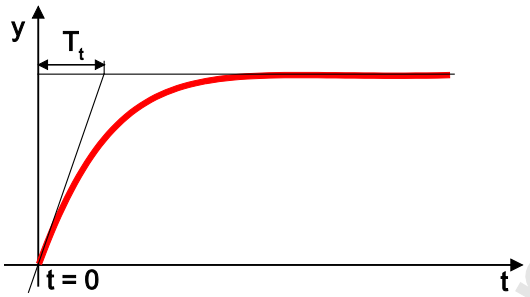
341

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 [Inkremente]
T1	TIME	Verzögerungszeit (Zeitkonstante)

Parameter der Ausgänge

343

Parameter	Datentyp	Beschreibung
Y	INT	Ausgangswert

5.2.14 Bausteine: Software-Reset

Inhalt

1594

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

SOFTRESET

260

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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

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_CR0020_Vxxyyzz.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_CR0020_Vxxyyzz.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

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

Automatische Datensicherung

Inhalt

2347

Die **ecomat*mobile***-Geräte bieten die Möglichkeit, Daten (BOOL, BYTE, WORD, DWORD) remanent (= spannungsausfallsicher) im Speicher zu sichern. Voraussetzung ist, dass die Daten als RETAIN-Variablen angelegt wurden (→ CODESYS).

Man unterscheidet zwischen Variablen, die als RETAIN deklariert wurden, und Variablen im Merkerbereich, der als Block mit *MEMORY_RETAIN_PARAM* (→ Seite 214) als remanent konfiguriert werden kann.
Details → Kapitel *Variablen* (→ Seite 76)

Der Vorteil des automatischen Speicherns ist, dass auch bei einem plötzlichen Spannungsabfall oder einer Unterbrechung der Versorgungsspannung die aktuellen Werte der Daten erhalten bleiben (z.B. Zählerstände).

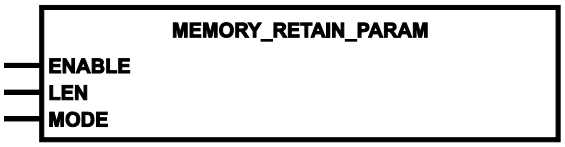
MEMORY_RETAIN_PARAM

2372

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2374

MEMORY_RETAIN_PARAM legt das remanente Verhalten der Daten für verschiedene Ereignisse fest. In CODESYS als VAR_RETAIN deklarierte Variablen haben von vornherein ein remanentes Verhalten.

Remanente Daten behalten (wie die als VAR_RETAIN deklarierte Variablen) ihren Wert nach einem unkontrolliertem Beenden wie auch nach normalem Aus- und Einschalten der Steuerung. Bei erneutem Start arbeitet das Programm mit den gespeicherten Werten weiter.

Für (mit MODE) wählbare Gruppen von Ereignissen legt dieser FB fest, wie viele (LEN) Datenbytes (ab Merkerbyte %MB0) Retain-Verhalten haben sollen, auch wenn sie nicht ausdrücklich als VAR_RETAIN deklariert wurden.

Ereignis	MODE = 0	MODE = 1	MODE = 2	MODE = 3
Power OFF ⇨ ON	Daten werden neu initialisiert	Daten sind remanent	Daten sind remanent	Daten sind remanent
Reset warm	Daten werden neu initialisiert	Daten sind remanent	Daten sind remanent	Daten sind remanent
Reset kalt	Daten werden neu initialisiert	Daten werden neu initialisiert	Daten sind remanent	Daten sind remanent
Reset Ursprung	Daten werden neu initialisiert	Daten werden neu initialisiert	Daten sind remanent	Daten sind remanent
Anwendungsprogramm laden	Daten werden neu initialisiert	Daten werden neu initialisiert	Daten sind remanent	Daten sind remanent
Laufzeitsystem laden	Daten werden neu initialisiert	Daten werden neu initialisiert	Daten werden neu initialisiert	Daten sind remanent

Bei MODE = 0 habe nur solche Daten Retain-Verhalten wie bei MODE=1, die ausdrücklich als VAR_RETAIN deklariert wurden.

Wird der FB nie aufgerufen, verhalten sich die Merkerbytes nach MODE = 0. Die Merkerbytes, die oberhalb des konfigurierten Bereichs liegen, verhalten sich ebenfalls nach MODE = 0.

Eine einmal getätigte Konfiguration bleibt auf dem Gerät erhalten, auch wenn die Anwendung oder das Laufzeitsystem neu geladen werden.

Parameter der Eingänge

2375

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
LEN	WORD	Anzahl der Datenbytes ab Merkeradresse %MB0, die remanentes Verhalten haben sollen zulässig = 0...4 096 = 0x0...0x1000 LEN > 4 096 wird automatisch zu LEN = 4 096 korrigiert
MODE	BYTE	Ereignisse, bei denen diese Variablen Retain-Verhalten haben sollen (0...3; → Tabelle oben) Bei MODE > 3 bleibt die zuletzt gültige Einstellung erhalten

Manuelle Datensicherung

Inhalt

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 15)) darüber informieren, welcher Speicherbereich frei zur Verfügung steht.

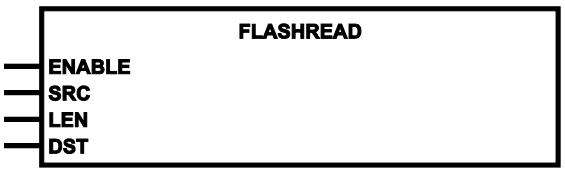
FLASHREAD

561

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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

565

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...65 535 = 0x0000...0xFFFF
LEN	WORD	Anzahl der Datenbytes zulässig = 0...65 535 = 0x0000...0xFFFF
DST	DWORD	Anfangsadresse der Zielvariablen Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

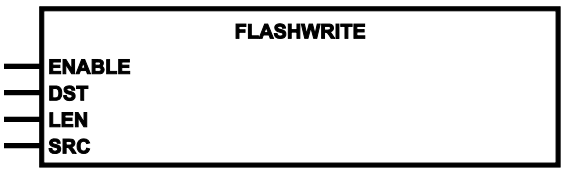
FLASHWRITE

555

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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

559

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 Ziel-Anfangsadresse im Speicher zulässig = 0...65 535 = 0x0000...0xFFFF
LEN	WORD	Anzahl der Datenbytes zulässig = 0...65 535 = 0x0000...0xFFFF
SRC	DWORD	Anfangsadresse der Quellvariablen Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

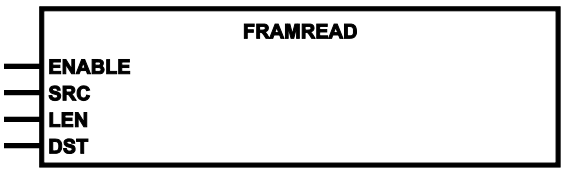
FRAMREAD

549

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

552

FRAMREAD ermöglicht das schnelle Lesen unterschiedlicher Datentypen direkt aus dem Anwender-Retain-Speicher (FRAM[1]).

Der FB liest den Inhalt ab der Adresse von SRC aus dem FRAM-Speicher. Dabei werden genau so viele Bytes übertragen, wie diese unter LEN angegeben sind.
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

553

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...1 023 = 0x0000...0x03FF
LEN	WORD	Anzahl der Datenbytes
DST	DWORD	Anfangsadresse der Zielvariablen Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

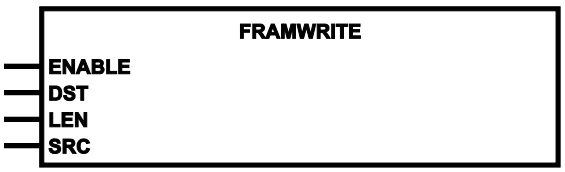
FRAMWRITE

543

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

546

FRAMWRITE ermöglicht das schnelle Schreiben unterschiedlicher Datentypen direkt in den Anwender-Retain-Speicher (FRAM[1]).

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

547

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 Ziel-Anfangsadresse im Speicher zulässig = 0...1 023 = 0x0000...0x03FF
LEN	WORD	Anzahl der Datenbytes
SRC	DWORD	Anfangsadresse der Quellvariablen Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

MEMCPY

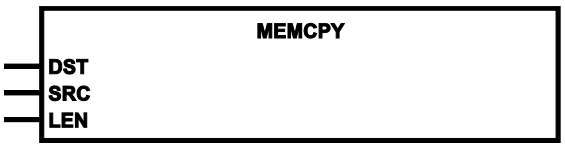
409

= Memory Copy

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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

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_CR0020_Vxxyyzz.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 **ecomat*mobile***-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

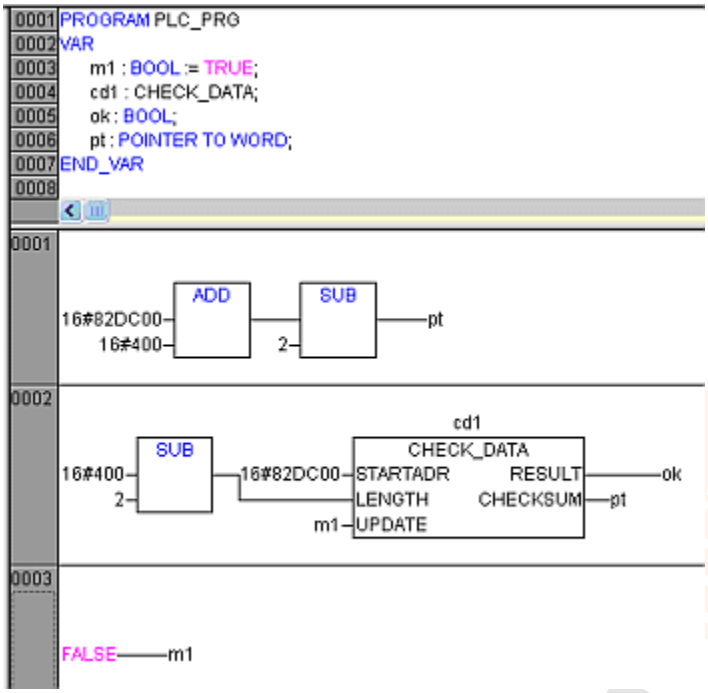
608

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.

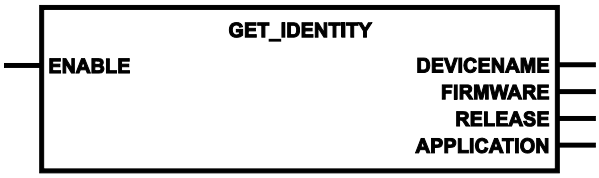
GET_IDENTITY

2212

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR0020_Vxxyyzz.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 228) 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_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:

Beschreibung

293

SET_DEBUG organisiert den DEBUG-Modus ohne aktiven Test-Eingang (→ Kapitel *TEST-Betrieb* (→ Seite 54)).

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_CR0020_Vxxyyzz.LIB`

Symbol in CODESYS:

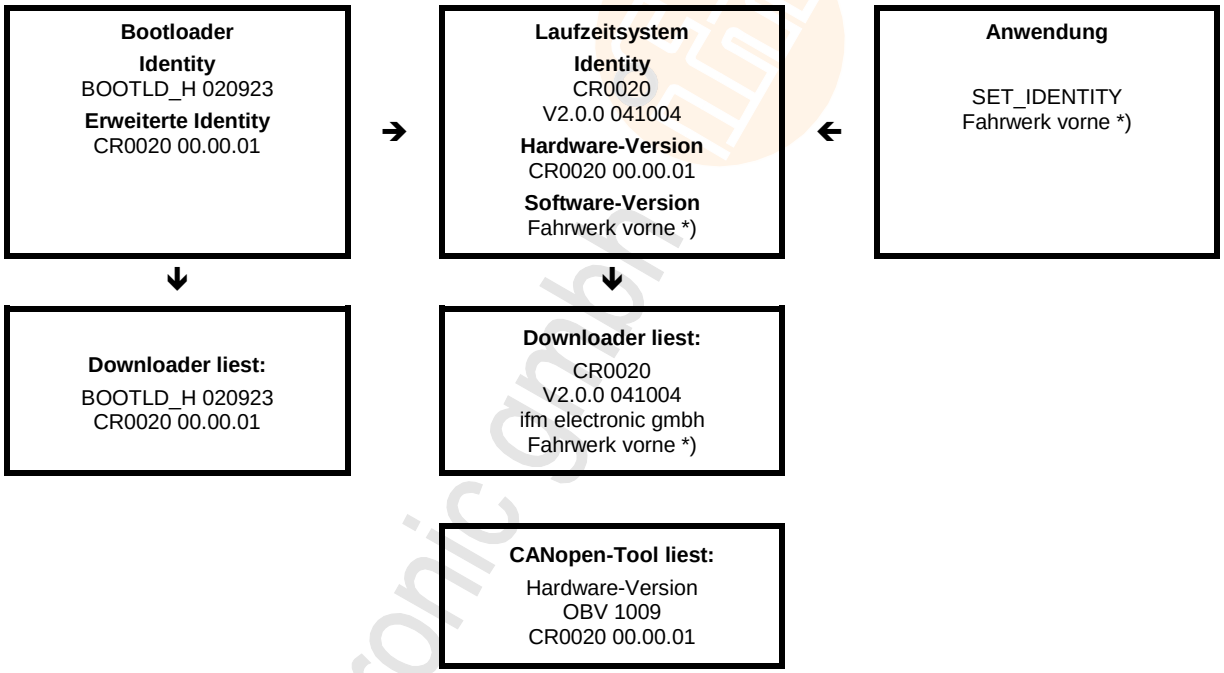
Beschreibung

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_CR0020_Vxxyyzz.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 Password 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

19598

Das Laufzeitsystem (LZS) überprüft das Gerät durch interne Fehler-Checks:
• in der Startphase (Reset-Phase)
• während der Ausführung des Anwendungsprogramms
→ Kapitel *Betriebszustände* (→ Seite 50)

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

19603
12217

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

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

HINWEIS

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

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

Vollständige Aufstellung der gerätespezifischen Fehler-Codes und Diagnosemeldungen
→ Kapitel *Systemmerker* (→ Seite 233)

6.4 Relais: wichtige Hinweise!

1446

Durch die logische Verknüpfung über das Systemmerker-Bit RELAIS oder RELAY_CLAMP_15 (→ Kapitel *Selbsthaltung* (→ Seite 16)) werden auch alle anderen Ausgänge abgeschaltet.

Je nach Anwendung muss nun entschieden werden, ob durch Rücksetzen des Systemmerker-Bits ERROR das Relais – und damit auch die Ausgänge – wieder eingeschaltet werden dürfen.

Zusätzlich besteht auch die Möglichkeit, das Systemmerker-Bit ERROR als "frei definierten Fehler" durch das Anwendungsprogramm zu setzen.

ACHTUNG

Vorzeitiger Verschleiß der Relaiskontakte möglich.

► Nur im "Notfall" diese Funktion zum generellen Abschalten der Ausgänge nutzen.

► Im Normalfall die Relais nur lastfrei schalten!
Dazu via Anwendungsprogramm alle relevanten Ausgänge auf FALSE setzen!

6.5 Reaktion auf System-Fehler

2258

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 und
- die Ausgangsrelais schalten ab.
- Somit fallen die darüber gesicherten Ausgänge ab.

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.

6.6 CAN / CANopen: Fehler und Fehlerbehandlung

19604

→ Systemhandbuch "Know-How ecomat*mobile*"
→ Kapitel *CAN / CANopen: Fehler und Fehlerbehandlung*

7 Anhang

Inhalt

1664

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

7.1 Systemmerker

Inhalt

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 ecomat*mobile*"
→ Kapitel *Fehler-Codes und Diagnoseinformationen*

7.1.1 Systemmerker: CAN

19555

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: 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 dominates 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-Identifier

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

7.1.2 Systemmerker: SAE-J1939

19556

Systemmerker (Symbolname)	Typ	Beschreibung
J1939_TASK	BOOL	Mit J1939_TASK wird die Zeitanforderung beim Versenden von J1939-Telegrammen eingehalten. Sollen J1939-Telegramme mit einer Wiederholzeit ≤ 50 ms versendet werden, setzt das Laufzeitsystem automatisch J1939_TASK=TRUE. Für Anwendungen, bei denen die Zeitanforderungen ≥ SPS-Zykluszeit sind: ▶ Systemlast reduzieren mit J1939_TASK=FALSE! TRUE: J1939-Task ist aktiv (= Initialwert) Der Task wird alle 2 ms aufgerufen Der J1939-Stack sendet seine Telegramme im benötigten Zeitraster FALSE: J1939-Task ist nicht aktiv

7.1.3 Systemmerker: Fehlermerker (Standard-Seite)

19557

Systemmerker (Symbolname)	Typ	Beschreibung
ERROR	BOOL	TRUE = Sammelfehlermeldung setzen, Relais ausschalten
ERROR_BREAK_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	WORD	Ausgangsgruppe x: Leiterbruch-Fehler [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
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_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_SHORT_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	WORD	Ausgangsgruppe x: Kurzschluss-Fehler [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_TEMPERATURE	BOOL	Temperatur-Fehler TRUE: Wert außerhalb des zulässigen Bereichs > allgemeiner Fehler FALSE: Wert in Ordnung
ERROR_VBBx	BOOL	Versorgungsspannungs-Fehler an VBBx (x = O | R): TRUE: Wert außerhalb des zulässigen Bereichs > allgemeiner Fehler FALSE: Wert in Ordnung

7.1.4 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)

7.1.5 Systemmerker: Spannungen (Standard-Seite)

19558

Systemmerker (Symbolname)	Typ	Beschreibung
CLAMP_15	BOOL	Überwachung Spannung an Klemme 15
RELAIS	BOOL	TRUE: Relais aktiviert Ausgänge werden mit Spannung versorgt FALSE: Relais ausgeschaltet Ausgänge sind spannungslos
RELAIS_CLAMP_15	BOOL	Relais Klemme 15 (Pin 5)
SERIAL_BAUDRATE	WORD	Baudrate der RS232-Schnittstelle
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.
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

7.1.6 Systemmerker: 16...40 Eingänge und 24...0 Ausgänge (Standard-Seite)

19560

Systemmerker (Symbolname)	Typ	Beschreibung
ANALOGx x = 0...7	WORD	Analog-Eingang xx: gefilterter A/D-Wandler-Rohwert (12 Bit) ohne Kalibrierung und Normierung
Ixx xx = 00...07 / 10...17 / 20...27 / 30...37 / 40...47	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
Ixx_MODE xx = 00...07 / 14...17 / 24...27 / 30...37 / 40...47	BYTE	Betriebsart des Eingangs Ixx → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)
Qxx xx = 10...13 / 20...23 / 30...37 / 40...47	BOOL	Status am Binärausgang xx: Voraussetzung: Ausgang ist als Binärausgang konfiguriert TRUE: Ausgang aktiviert FALSE: Ausgang deaktiviert (= Initialwert) oder: nicht als Binärausgang konfiguriert
Qxx_MODE xx = 10...13 / 20...23 / 30...37 / 40...47	BYTE	Betriebsart des Ausgangs Qxx → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

7.2 Adressbelegung und E/A-Betriebsarten

Inhalt

1656

→ auch Datenblatt

7.2.1 Adressbelegung Ein-/Ausgänge

Inhalt

2371

Eingänge: Adressbelegung (Standard-Seite) (16...40 Eingänge)

19572

Abkürzungen → Kapitel *Hinweise zur Anschlussbelegung* (→ Seite 35)
Betriebsarten der Ein- und Ausgänge → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

IEC-Adresse	Symbolische Adresse
%IX0.00 %IW03	I00 ANALOG0
%IX0.01 %IW3	I01 ANALOG1
%IX0.02 %IW4	I02 ANALOG2
%IX0.03 %IW5	I03 ANALOG3
%IX0.04 %IW6	I04 ANALOG4
%IX0.05 %IW7	I05 ANALOG5
%IX0.06 %IW8	I06 ANALOG6
%IX0.07 %IW9	I07 ANALOG7
%IX0.08	I10
%IX0.09	I11
%IX0.10	I12
%IX0.11	I13
%IX0.12	I14
%IX0.13	I15
%IX0.14	I16
%IX0.15	I17
%IX1.00	I20
%IX1.01	I21
%IX1.02	I22
%IX1.03	I23
%IX1.04	I24
%IX1.05	I25
%IX1.06	I26
%IX1.07	I27
%IX1.08	I30
%IX1.09	I31
%IX1.10	I32
%IX1.11	I33
%IX1.12	I34
%IX1.13	I35
%IX1.14	I36
%IX1.15	I37

IEC-Adresse	Symbolische Adresse
%IX2.00	I40
%IX2.01	I41
%IX2.02	I42
%IX2.03	I43
%IX2.04	I44
%IX2.05	I45
%IX2.06	I46
%IX2.07	I47

Ausgänge: Adressbelegung (Standard-Seite) (0...24 Ausgänge)

19573

Abkürzungen → Kapitel *Hinweise zur Anschlussbelegung* (→ Seite 35)
Betriebsarten der Ein- und Ausgänge → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 243)

IEC-Adresse	Symbolische Adresse
%QX0.00	Q10
%QX0.01	Q11
%QX0.02	Q12
%QX0.03	Q13
%QX0.04	Q20
%QX0.05	Q21
%QX0.06	Q22
%QX0.07	Q23
%QX0.08	Q30
%QX0.09	Q31
%QX0.10	Q32
%QX0.11	Q33
%QX0.12	Q34
%QX0.13	Q35
%QX0.14	Q36
%QX0.15	Q37
%QX1.00	Q40
%QX1.01	Q41
%QX1.02	Q42
%QX1.03	Q43
%QX1.04	Q44
%QX1.05	Q45
%QX1.06	Q46
%QX1.07	Q47

7.2.2 Mögliche Betriebsarten Ein-/Ausgänge

Inhalt

2386

Eingänge: Betriebsarten (Standard-Seite) (16...40 Eingänge)

19575

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

= diese Konfiguration ist voreingestellt

Eingänge	mögliche Betriebsart		einstellen mit FB	FB-Eingang	Wert	
					dez	hex
I00...I07	IN_NOMODE	Aus	INPUT_ANALOG	MODE	0	00
	IN_DIGITAL_H	plus	INPUT_ANALOG	MODE	1	01
	IN_CURRENT	0...20 000 µA	INPUT_ANALOG	MODE	4	04
	IN_VOLTAGE10	0...10 000 mV	INPUT_ANALOG	MODE	8	08
	IN_VOLTAGE30	0...30 000 mV	INPUT_ANALOG	MODE	16	10
	IN_RATIO	0...1 000 ‰	INPUT_ANALOG	MODE	32	20
	IN_DIAGNOSTIC	bei IN_DIGITAL_H	INPUT_ANALOG	MODE	64	40
I10...I13	IN_NOMODE	Aus	INPUT_ANALOG	MODE	0	00
	IN_DIGITAL_H	plus	INPUT_ANALOG	MODE	1	01
I14...I17	IN_NOMODE	Aus	INPUT_ANALOG	MODE	0	00
	IN_DIGITAL_H	plus	INPUT_ANALOG	MODE	1	01
	IN_DIAGNOSTIC	bei IN_DIGITAL_H	INPUT_ANALOG	MODE	64	40
	IN_FAST	bei IN_DIGITAL_H	INPUT_ANALOG	MODE	128	80
	Frequenzmessung	0...50 000 Hz	FREQUENCY PHASE			
	Periodendauermessung	0,1...5 000 Hz	PERIOD			
	Periodendauer- und Ratiomessung	0,1...5 000 Hz	PERIOD_RATIO			
	Zähler	0...30 000 Hz	FAST_COUNT			
	Drehgeber erfassen	0...50 000 Hz	INC_ENCODER			
I20...I23	IN_NOMODE	Aus	INPUT_ANALOG	MODE	0	00
	IN_DIGITAL_H	plus	INPUT_ANALOG	MODE	1	01
I24...I27	IN_NOMODE	Aus	INPUT_ANALOG	MODE	0	00
	IN_DIGITAL_H	plus	INPUT_ANALOG	MODE	1	01
	IN_DIGITAL_L	minus	INPUT_ANALOG	MODE	2	02
	IN_DIAGNOSTIC	bei IN_DIGITAL_H	INPUT_ANALOG	MODE	64	40
	IN_FAST	bei IN_DIGITAL_H	INPUT_ANALOG	MODE	128	80
	Frequenzmessung	0...1 000 Hz	FREQUENCY PHASE			

Eingänge	mögliche Betriebsart		einstellen mit FB	FB-Eingang	Wert	
					dez	hex
	Periodendauermessung	0,1...1 000 Hz	PERIOD			
	Periodendauer- und Ratiomessung	0,1...1 000 Hz	PERIOD_RATIO			
	Zähler	0…1 000 Hz	FAST_COUNT			
	Drehgeber erfassen	0…1 000 Hz	INC_ENCODER			
I30…I37	IN_NOMODE	Aus	INPUT_ANALOG	MODE	0	00
	IN_DIGITAL_H	plus	INPUT_ANALOG	MODE	1	01
	IN_DIGITAL_L	minus	INPUT_ANALOG	MODE	2	02
	IN_DIAGNOSTIC	bei IN_DIGITAL_H	INPUT_ANALOG	MODE	64	40
I40…I47	IN_NOMODE	Aus	INPUT_ANALOG	MODE	0	00
	IN_DIGITAL_H	plus	INPUT_ANALOG	MODE	1	01
	IN_DIAGNOSTIC	bei IN_DIGITAL_H	INPUT_ANALOG	MODE	64	40

Betriebsarten mit folgendem Funktionsbaustein einstellen:

FAST_COUNT (→ Seite 157)	Zählerbaustein für schnelle Eingangsimpulse
FREQUENCY (→ Seite 158)	misst die Frequenz des am gewählten Kanal ankommenden Signals
INC_ENCODER (→ Seite 159)	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
INPUT_ANALOG (→ Seite 150)	Strom- und Spannungsmessung am analogen Eingangskanal
PERIOD (→ Seite 162)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_RATIO (→ Seite 164)	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 166)	liest ein Kanalpaar mit schnellen Eingängen ein und vergleicht die Phasenlage der Signale

Ausgänge: Betriebsarten (Standard-Seite) (0...24 Ausgänge)

19576

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

= diese Konfiguration ist voreingestellt

Ausgänge	mögliche Betriebsart		einstellen mit	Wert	
				dez	hex
Q10…Q13	OUT_NOMODE	Aus	Qxx_MODE	0	00
	OUT_DIGITAL_H	plus	Qxx_MODE	1	01
	OUT_CURRENT		Qxx_MODE	4	04
	OUT_DIAGNOSTIC		Qxx_MODE	64	40
	OUT_OVERLOAD_PROTECTION		Qxx_MODE	128	80
Q20…Q23	OUT_NOMODE	Aus	Qxx_MODE	0	00
	OUT_DIGITAL_H	plus	Qxx_MODE	1	01
	OUT_CURRENT		Qxx_MODE	4	04
	OUT_DIAGNOSTIC		Qxx_MODE	64	40
	OUT_OVERLOAD_PROTECTION		Qxx_MODE	128	80
Q30…Q37	OUT_NOMODE	Aus	Qxx_MODE	0	00
	OUT_DIGITAL_H	plus	Qxx_MODE	1	01
	OUT_DIAGNOSTIC		Qxx_MODE	64	40
	OUT_OVERLOAD_PROTECTION		Qxx_MODE	128	80
Q40, Q43, Q44, Q47	OUT_NOMODE	Aus	Qxx_MODE	0	00
	OUT_DIGITAL_H	plus	Qxx_MODE	1	01
	OUT_DIAGNOSTIC		Qxx_MODE	64	40
Q41, Q42, Q45, Q46	OUT_NOMODE	Aus	Qxx_MODE	0	00
	OUT_DIGITAL_H	plus	Qxx_MODE	1	01
	OUT_DIGITAL_L	minus	Qxx_MODE	2	02
	OUT_DIAGNOSTIC		Qxx_MODE	64	40

Ausgänge: zulässige Betriebsarten

19577

Betriebsart		Q10	Q11	Q12	Q13	Q20	Q21	Q22	Q23
OUT_NOMODE	Aus	X	X	X	X	X	X	X	X
OUT_DIGITAL_H	plus	X	X	X	X	X	X	X	X
OUT_CURRENT		X	X	X	X	X	X	X	X
OUT_DIAGNOSTIC		X	X	X	X	X	X	X	X
OUT_OVERLOAD_PROTECTION		X	X	X	X	X	X	X	X
PWM		X	X	X	X	X	X	X	X

Betriebsart		Q30	Q31	Q32	Q33	Q34	Q35	Q36	Q37
OUT_NOMODE	Aus	X	X	X	X	X	X	X	X
OUT_DIGITAL_H	plus	X	X	X	X	X	X	X	X
OUT_DIAGNOSTIC		X	X	X	X	X	X	X	X

Betriebsart		Q40	Q41	Q42	Q43	Q44	Q45	Q46	Q47
OUT_NOMODE	Aus	X	X	X	X	X	X	X	X
OUT_DIGITAL_H	plus	X	X	X	X	X	X	X	X
OUT_DIGITAL_L	minus	--	X	X	--	--	X	X	--
OUT_DIAGNOSTIC		X	X	X	X	X	X	X	X
OUT_OVERLOAD_PROTECTION		X	X	X	X	X	X	X	X
PWM		X	--	--	X	X	--	--	X
H-Brücke		--	X	X	--	--	X	X	--

7.2.3 Adressen / Variablen der E/As

Inhalt

2376

Eingänge: Adressen und Variablen (Standard-Seite) (16...40 Eingänge)

19579

IEC-Adresse	E/A-Variable	Bemerkung
%QB8	I00_MODE	Konfigurations-Byte für %IX0.00 (I00)
%QB9	I01_MODE	Konfigurations-Byte für %IX0.01 (I01)
%QB10	I02_MODE	Konfigurations-Byte für %IX0.02 (I02)
%QB11	I03_MODE	Konfigurations-Byte für %IX0.03 (I03)
%QB12	I04_MODE	Konfigurations-Byte für %IX0.04 (I04)
%QB13	I05_MODE	Konfigurations-Byte für %IX0.05 (I05)
%QB14	I06_MODE	Konfigurations-Byte für %IX0.06 (I06)
%QB15	I07_MODE	Konfigurations-Byte für %IX0.07 (I07)
%QB16	I14_MODE	Konfigurations-Byte für %IX0.14 (I14)
%QB17	I15_MODE	Konfigurations-Byte für %IX0.15 (I15)
%QB18	I16_MODE	Konfigurations-Byte für %IX0.16 (I16)
%QB19	I17_MODE	Konfigurations-Byte für %IX0.17 (I17)
%QB20	I24_MODE	Konfigurations-Byte für %IX1.04 (I24)
%QB21	I25_MODE	Konfigurations-Byte für %IX1.05 (I25)
%QB22	I26_MODE	Konfigurations-Byte für %IX1.06 (I26)
%QB23	I27_MODE	Konfigurations-Byte für %IX1.07 (I27)
%QB24	I30_MODE	Konfigurations-Byte für %IX1.08 (I30)
%QB25	I31_MODE	Konfigurations-Byte für %IX1.09 (I31)
%QB26	I32_MODE	Konfigurations-Byte für %IX1.10 (I32)
%QB27	I33_MODE	Konfigurations-Byte für %IX1.11 (I33)
%QB28	I34_MODE	Konfigurations-Byte für %IX1.12 (I34)
%QB29	I35_MODE	Konfigurations-Byte für %IX1.13 (I35)
%QB30	I36_MODE	Konfigurations-Byte für %IX1.14 (I36)
%QB31	I37_MODE	Konfigurations-Byte für %IX1.15 (I37)
%QB32	I40_MODE	Konfigurations-Byte für %IX2.00 (I40)
%Q33	I41_MODE	Konfigurations-Byte für %IX2.01 (I41)
%QB34	I42_MODE	Konfigurations-Byte für %IX2.02 (I42)
%QB35	I43_MODE	Konfigurations-Byte für %IX2.03 (I43)
%QB36	I44_MODE	Konfigurations-Byte für %IX2.04 (I44)
%QB37	I45_MODE	Konfigurations-Byte für %IX2.05 (I45)
%QB38	I46_MODE	Konfigurations-Byte für %IX2.06 (I46)
%QB39	I47_MODE	Konfigurations-Byte für %IX2.07 (I47)

IEC-Adresse	E/A-Variable	Bemerkung
%IW3	ANALOG0	Analog-Wert an I00
%IW4	ANALOG1	Analog-Wert an I01
%IW5	ANALOG2	Analog-Wert an I02
%IW6	ANALOG3	Analog-Wert an I03
%IW7	ANALOG4	Analog-Wert an I04
%IW8	ANALOG5	Analog-Wert an I05
%IW9	ANALOG6	Analog-Wert an I06
%IW10	ANALOG7	Analog-Wert an I07

Ausgänge: Adressen und Variablen (Standard-Seite) (0...24 Ausgänge)

19580

IEC-Adresse	E/A-Variable	Bemerkung
%QB40	Q10_MODE	Konfigurations-Byte für %QX0.00 (Q10)
%QB41	Q11_MODE	Konfigurations-Byte für %QX0.01 (Q11)
%QB42	Q12_MODE	Konfigurations-Byte für %QX0.02 (Q12)
%QB43	Q13_MODE	Konfigurations-Byte für %QX0.03 (Q13)
%QB44	Q20_MODE	Konfigurations-Byte für %QX0.04 (Q20)
%QB45	Q21_MODE	Konfigurations-Byte für %QX0.05 (Q21)
%QB46	Q22_MODE	Konfigurations-Byte für %QX0.06 (Q22)
%QB47	Q23_MODE	Konfigurations-Byte für %QX0.07 (Q23)
%QB48	Q30_MODE	Konfigurations-Byte für %QX0.08 (Q30)
%QB49	Q31_MODE	Konfigurations-Byte für %QX0.09 (Q31)
%QB50	Q32_MODE	Konfigurations-Byte für %QX0.10 (Q32)
%QB51	Q33_MODE	Konfigurations-Byte für %QX0.11 (Q33)
%QB52	Q34_MODE	Konfigurations-Byte für %QX0.12 (Q34)
%QB53	Q35_MODE	Konfigurations-Byte für %QX0.13 (Q35)
%QB54	Q36_MODE	Konfigurations-Byte für %QX0.14 (Q36)
%QB55	Q37_MODE	Konfigurations-Byte für %QX0.15 (Q37)
%QB56	Q40_MODE	Konfigurations-Byte für %QX1.00 (Q40)
%QB57	Q41_MODE	Konfigurations-Byte für %QX1.01 (Q41)
%QB58	Q42_MODE	Konfigurations-Byte für %QX1.02 (Q42)
%QB59	Q43_MODE	Konfigurations-Byte für %QX1.03 (Q43)
%QB60	Q44_MODE	Konfigurations-Byte für %QX1.04 (Q44)
%QB61	Q45_MODE	Konfigurations-Byte für %QX1.05 (Q45)
%QB62	Q46_MODE	Konfigurations-Byte für %QX1.06 (Q46)
%QB63	Q47_MODE	Konfigurations-Byte für %QX1.07 (Q47)

7.3 Fehler-Tabellen

Inhalt

19606

7.3.1 Fehlermerker

19608

→ Kapitel *Systemmerker* (→ Seite 233)

7.3.2 Fehler: CAN / CANopen

19610
19604

→ Systemhandbuch "Know-How ecomat*mobile*"
→ 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					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Beschreibung
00	80	11	---	---	---	---	---	CANx Monitoring SYNC-Error (nur Slave)
00	81	11	---	---	---	---	---	CANx Warngrenze (> 96)
10	81	11	---	---	---	---	---	CANx Empfangspuffer Überlauf
11	81	11	---	---	---	---	---	CANx Sendepuffer Überlauf
30	81	11	---	---	---	---	---	CANx Guard-/Heartbeat-Error (nur Slave)

EMCY-Codes: E/As, System (Standard-Seite)

19552

Die folgenden EMCY-Meldungen werden automatisch versendet, wenn der FB *CANx_MASTER_EMCY_HANDLER* (→ Seite 103) zyklisch aufgerufen wird.

EMCY-Code Objekt 0x1003		Objekt 0x1001	herstellerspezifische Informationen					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Beschreibung
00	21	03	I0	I1	I2	I3	I4	Diagnose Eingänge
00	23	03	Q1Q2	Q3	Q4			Diagnose Ausgänge Unterbrechung
02	23	03	Q1Q2	Q3	Q4			Diagnose Ausgänge Kurzschluss
00	31	05						Klemmenspannung VBBO/VBBS
00	33	05						Ausgangsspannung VBBR
00	42	09						Übertemperatur
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 **ecomat*mobile***-Controller nur den Bootloader.
Der Bootloader ist ein Startprogramm, mit dem das Laufzeitsystem und das Anwendungsprogramm auf dem Gerät nachgeladen werden können.
Der Bootloader enthält Grundroutinen...
- zur Kommunikation der Hardware-Module untereinander,
- zum Nachladen des Laufzeitsystems.

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

Bus

Serielle Datenübertragung mehrerer Teilnehmer an derselben Leitung.

C

CAN

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

CAN-Stack

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

CiA

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

CiA DS 304

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

CiA DS 401

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

CiA DS 402

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

CiA DS 403

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

CiA DS 404

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

CiA DS 405

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

CiA DS 406

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

CiA DS 407

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

COB-ID

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

CODESYS

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

CSV-Datei

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

Beispiel: Quell-Tabelle mit Zahlenwerten:

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

Daraus entsteht folgende CSV-Datei:

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

D

Datentyp

Abhängig vom Datentyp können unterschiedlich große Werte gespeichert werden.

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

DC

Direct **C**urrent = Gleichstrom

Diagnose

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

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

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

Dither

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

DLC

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

DRAM

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

DTC

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

E

ECU

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

EDS-Datei

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

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

Embedded Software

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

EMCY

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

EMV

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

Ethernet

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

EUC

EUC = **E**quipment **U**nder **C**ontrol (kontrollierte Einrichtung).
EUC ist eine Einrichtung, Maschine, Gerät oder Anlage, verwendet zur Fertigung, Stoffumformung, zum Transport, zu medizinischen oder anderen Tätigkeiten (→ IEC 61508-4, Abschnitt 3.2.3). Das EUC umfasst also alle Einrichtungen, Maschinen, Geräte oder Anlagen, die →Gefährdungen verursachen können und für die sicherheitsgerichtete Systeme erforderlich sind.
Falls eine vernünftigerweise vorhersehbare Aktivität oder Inaktivität zu durch das EUC verursachten Gefährdungen mit unvertretbarem Risiko führt, sind Sicherheitsfunktionen erforderlich, um einen sicheren Zustand für das EUC zu erreichen oder aufrecht zu erhalten. Diese Sicherheitsfunktionen werden durch ein oder mehrere sicherheitsgerichtete Systeme ausgeführt.

F

Fehlanwendung

Das ist die Verwendung eines Produkts in einer Weise, die vom Konstrukteur nicht vorgesehen ist.
Eine Fehlanwendung führt meist zu einer →Gefährdung von Personen oder Sachen.
Vor vernünftigerweise, vorhersehbaren Fehlanwendungen muss der Hersteller des Produkts in seinen Benutzerinformationen warnen.

FiFo

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

Flash-Speicher

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

Vorteile von Flash-Speicher

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

Nachteile von Flash-Speicher

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

FRAM

FRAM, oder auch FeRAM, bedeutet **Fe**rroelectric **R**andom **A**ccess **M**emory. Der Speicher- und Löschvorgang erfolgt durch eine Polarisationsänderung in einer ferroelektrischen Schicht.
Vorteile von FRAM gegenüber herkömmlichen Festwertspeichern:
- nicht flüchtig,
- kompatibel zu gängigen EEPROMs, jedoch:
- Zugriffszeit ca. 100 ns,
- fast unbegrenzt viele Zugriffszyklen möglich.

H

Heartbeat

Heartbeat (engl.) = Herzschlag.
Die Teilnehmer senden regelmäßig kurze Signale. So können die anderen Teilnehmer prüfen, ob ein Teilnehmer ausgefallen ist.

HMI

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

I

ID – Identifier

ID = **Id**entifier = Kennung

Name zur Unterscheidung der an einem System angeschlossenen Geräte / Teilnehmer oder der zwischen den Teilnehmern ausgetauschten Nachrichtenpakete.

IEC 61131

Norm: Grundlagen Speicherprogrammierbarer Steuerungen
- Teil 1: Allgemeine Informationen
- Teil 2: Betriebsmittelanforderungen und Prüfungen
- Teil 3: Programmiersprachen
- Teil 5: Kommunikation
- Teil 7: Fuzzy-Control-Programmierung

IEC-User-Zyklus

IEC-User-Zyklus = SPS-Zyklus im CODESYS-Anwendungsprogramm.

IP-Adresse

IP = **I**nternet **P**rotocol = Internet-Protokoll.
Die IP-Adresse ist eine Nummer, die zur eindeutigen Identifizierung eines Internet-Teilnehmers notwendig ist. Zur besseren Übersicht wird die Nummer in 4 dezimalen Werten geschrieben, z. B. 127.215.205.156.

ISO 11898

Norm: Straßenfahrzeuge – CAN-Protokoll
- Teil 1: Bit-Übertragungsschicht und physikalische Zeichenabgabe
- Teil 2: High-speed medium access unit
- Teil 3: Fehlertolerante Schnittstelle für niedrige Geschwindigkeiten
- Teil 4: Zeitgesteuerte Kommunikation
- Teil 5: High-speed medium access unit with low-power mode

ISO 11992

Norm: Straßenfahrzeuge – Austausch von digitalen Informationen über elektrische Verbindungen zwischen Zugfahrzeugen und Anhängefahrzeugen
- Teil 1: Bit-Übertragungsschicht und Sicherungsschicht
- Teil 2: Anwendungsschicht für die Bremsausrüstung
- Teil 3: Anwendungsschicht für andere als die Bremsausrüstung
- Teil 4: Diagnose

ISO 16845

Norm: Straßenfahrzeuge – Steuergerätenetz (CAN) – Prüfplan zu Konformität

J

J1939

→ SAE J1939

K

Klemme 15

Klemme 15 ist in Fahrzeugen die vom Zündschloss geschaltete Plusleitung.

L

Laufzeitsystem

Grundprogramm im Gerät, stellt die Verbindung her zwischen der Hardware des Gerätes und dem Anwendungsprogramm.
→ Kapitel *Software-Module für das Gerät* (→ Seite 43)

LED

LED = **L**ight **E**mitting **D**iode = Licht aussendende Diode.
Leuchtdiode, auch Luminiszenzdiode, ein elektronisches Element mit hoher, farbiger Leuchtkraft auf kleinem Volumen bei vernachlässigbarer Verlustleistung.

Link

Ein Link ist ein Querverweis zu einer anderen Stelle im Dokument oder auf ein externes Dokument.

LSB

Least **S**ignificant **B**it/Byte = Niederwertigstes Bit/Byte in einer Reihe von Bit/Bytes.

M

MAC-ID

MAC = **M**anufacturer‘s **A**ddress **C**ode
= Hersteller-Seriennummer.
→ID = **Id**entifier = Kennung
Jede Netzwerkkarte verfügt über eine so genannte MAC-Adresse, ein unverwechselbarer, auf der ganzen Welt einzigartiger Zahlencode – quasi eine Art Seriennummer. So eine MAC-Adresse ist eine Aneinanderreihung von 6 Hexadezimalzahlen, etwa "00-0C-6E-D0-02-3F".

Master

Wickelt die komplette Organisation auf dem →Bus ab. Der Master entscheidet über den zeitlichen Buszugriff und fragt die →Slaves zyklisch ab.

MMI

MMI = **M**ensch-**M**aschine-**I**nterface
→ *HMI* (→ Seite 256)

MRAM

MRAM = **M**agnetoresistive **R**andom **A**ccess **M**emory
Die Informationen werden mit magnetischen Ladungselementen gespeichert. Dabei wird die Eigenschaft bestimmter Materialien ausgenutzt, die ihren elektrischen Widerstand unter dem Einfluss magnetischer Felder ändern.
Vorteile von MRAM gegenüber herkömmlichen Festwertspeichern:
- nicht flüchtig (wie FRAM), jedoch:
- Zugriffszeit nur ca. 35 ns,
- unbegrenzt viele Zugriffszyklen möglich.

MSB

Most **S**ignificant **B**it/Byte = Höchstwertiges Bit/Byte einer Reihe von Bits/Bytes.

N

NMT

NMT = **N**etwork **M**anagemen**t** = Netzwerk-Verwaltung (hier: im CANopen-Protokoll).
Der NMT-Master steuert die Betriebszustände der NMT-Slaves.

Node

Node (engl.) = Knoten. Damit ist ein Teilnehmer im Netzwerk gemeint.

Node Guarding

Node (engl.) = Knoten, hier: Netzwerkteilnehmer
Guarding (engl.) = Schutz
Parametrierbare, zyklische Überwachung von jedem entsprechend konfigurierten →Slave. Der →Master prüft, ob die Slaves rechtzeitig antworten. Die Slaves prüfen, ob der Master regelmäßig anfragt. Somit können ausgefallene Netzwerkteilnehmer schnell erkannt und gemeldet werden.

O

Obj / Objekt

Oberbegriff für austauschbare Daten / Botschaften innerhalb des CANopen-Netzwerks.

Objektverzeichnis

Das **Ob**jekt**v**erzeichnis OBV enthält alle CANopen-Kommunikationsparameter eines Gerätes, sowie gerätespezifische Parameter und Daten.

OBV

Das **Ob**jekt**v**erzeichnis OBV enthält alle CANopen-Kommunikationsparameter eines Gerätes, sowie gerätespezifische Parameter und Daten.

OPC

OPC = **O**LE for **P**rocess **C**ontrol = Objektverknüpfung und -einbettung für Prozesssteuerung
Standardisierte Software-Schnittstelle zur herstellerunabhängigen Kommunikation in der Automatisierungstechnik
OPC-Client (z.B. Gerät zum Parametrieren oder Programmieren) meldet sich nach dem Anschließen am OPC-Server (z.B. Automatisierungsgerät) automatisch bei diesem an und kommuniziert mit ihm.

operational

Operational (engl.) = betriebsbereit
Betriebszustand eines CANopen-Teilnehmers. In diesem Modus können →SDOs, →NMT-Kommandos und →PDOs übertragen werden.

P

PC-Karte

→ PCMCIA-Karte

PCMCIA-Karte

PCMCIA = Personal Computer Memory Card International Association, ein Standard für Erweiterungskarten mobiler Computer.
Seit der Einführung des Cardbus-Standards 1995 werden PCMCIA-Karten auch als PC-Karte (engl.: PC Card) bezeichnet.

PDM

PDM = **P**rocess and **D**ialog **M**odule = **P**rozess- und **D**ialog-**M**onitor.
Gerät zur Kommunikation des Bedieners mit der Maschine / Anlage.

PDO

PDO = **P**rocess **D**ata **O**bject = Nachrichten-Objekt mit Prozessdaten.
Die zeitkritischen Prozessdaten werden mit Hilfe der "Process Data Objects" (PDOs) übertragen. Die PDOs können beliebig zwischen den einzelnen Knoten ausgetauscht werden (PDO-Linking). Zusätzlich wird festgelegt, ob der Datenaustausch ereignisgesteuert (asynchron) oder synchronisiert erfolgen soll. Je nach der Art der zu übertragenden Daten kann die richtige Wahl der Übertragungsart zu einer erheblichen Entlastung des →CAN-Bus führen.
Dem Protokoll entsprechend, sind diese Dienste nicht bestätigte Dienste: es gibt keine Kontrolle, ob die Nachricht auch beim Empfänger ankommt. Netzwerkvariablen-Austausch entspricht einer "1-zu-n-Verbindung" (1 Sender zu n Empfängern).

PDU

PDU = **P**rotocol **D**ata **U**nit = Protokoll-Daten-Einheit.
Die PDU ist ein Begriff aus dem →CAN-Protokoll →SAE J1939. Sie bezeichnet einen Bestandteil der 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 →Zyklusses arbeitet.

- Am Zyklus-Beginn liest die SPS die Zustände aller Eingänge in das Prozessabbild ein. Während des Zyklusses kann die SPS Änderungen an den Eingängen nicht erkennen.
- Im Laufe des Zyklusses werden die Ausgänge nur virtuell (im Prozessabbild) geändert.
- Am Zyklus-Ende schreibt die SPS die virtuellen Ausgangszustände auf die realen Ausgänge.

PWM

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

R

ratiometrisch

Ratio (lat.) = Verhältnis
Messungen können auch ratiometrisch erfolgen = Verhältnismessung. Wenn das Ausgangssinal eines Sensors proportional zu seiner Versorgungsspannung ist, kann durch ratiometrische Messung (= Messung im Verhältnis zur Versorgung) der Einfluss von Schwankungen der Versorgung reduziert, im Idealfall sogar beseitigt werden.
→ Analogeingang

RAW-CAN

RAW-CAN bezeichnet das reine →CAN-Protokoll, das ohne ein zusätzliches Kommunikationsprotokoll auf dem CAN-Bus (auf ISO/OSI-Schicht 2) arbeitet. Das CAN-Protokoll ist international nach →ISO 11898-1 definiert und garantiert zusätzlich in →ISO 16845 die Austauschbarkeit von CAN-Chips.

remanent

Remanente Daten sind gegen Datenverlust bei Spannungsausfall geschützt.
Z.B. kopiert das →Laufzeitsystem die remanenten Daten automatisch in einen →Flash-Speicher, sobald die Spannungsversorgung unter einen kritischen Wert sinkt. Bei Wiederkehr der Spannungsversorgung lädt das Laufzeitsystem die remanenten Daten zurück in den Arbeitsspeicher. Dagegen sind die Daten im Arbeitsspeicher einer Steuerung flüchtig und bei Unterbrechung der Spannungsversorgung normalerweise verloren.

ro

ro = read only (engl.) = nur lesen
Unidirektionale Datenübertragung: Daten können nur gelesen werden, jedoch nicht verändert.

RTC

RTC = **R**eal **T**ime **C**lock = Echtzeituhr
Liefert (batteriegepuffert) aktuell Datum und Uhrzeit. Häufiger Einsatz beim Speichern von Fehlermeldungsprotokollen.

rw

rw = read/write (engl.) = lesen und schreiben
Bidirektionale Datenübertragung: Daten können sowohl gelesen als auch verändert werden.

S

SAE J1939

Das Netzwerkprotokoll SAE J1939 beschreibt die Kommunikation auf einem →CAN-Bus in Nutzfahrzeugen zur Übermittlung von Diagnosedaten (z.B.Motordrehzahl, Temperatur) und Steuerungsinformationen.
Norm: Recommended Practice for a Serial Control and Communications Vehicle Network
- Teil 2: Agricultural and Forestry Off-Road Machinery Control and Communication Network
- Teil 3: On Board Diagnostics Implementation Guide
- Teil 5: Marine Stern Drive and Inboard Spark-Ignition Engine On-Board Diagnostics Implementation Guide
- Teil 11: Physical Layer – 250 kBits/s, Shielded Twisted Pair
- Teil 13: Off-Board Diagnostic Connector
- Teil 15: Reduced Physical Layer, 250 kBits/s, Un-Shielded Twisted Pair (UTP)
- Teil 21: Data Link Layer
- Teil 31: Network Layer
- Teil 71: Vehicle Application Layer
- Teil 73: Application Layer – Diagnostics
- Teil 81: Network Management Protocol

SD-Card

Eine SD Memory Card (Kurzform für **S**ecure **D**igital Memory Card; deutsch: Sichere digitale Speicherkarte) ist ein digitales Speichermedium, das nach dem Prinzip der →Flash-Speicherung arbeitet.

SDO

SDO = **S**ervice **D**ata **O**bject = Nachrichten-Objekt mit Servicedaten.
Das SDO dient dem Zugriff auf Objekte in einem CANopen-Objektverzeichnis. Dabei fordern 'Clients' die gewünschten Daten von 'Servern' an. Die SDOs bestehen immer aus 8 Bytes.
Beispiele:
- Automatische Konfiguration aller →Slaves über SDOs beim Systemstart.
- Auslesen der Fehlernachrichten aus dem →Objektverzeichnis.

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

Selbsttest

Testprogramm, das aktiv Komponenten oder Geräte testet. Das Programm wird durch den Anwender gestartet und dauert eine gewisse Zeit. Das Ergebnis davon ist ein Testprotokoll (Log-Datei), aus dem entnommen werden kann, was getestet wurde und ob das Ergebnis positiv oder negativ ist.

Slave

Passiver Teilnehmer am Bus, antwortet nur auf Anfrage des →Masters. Slaves haben im Bus eine eindeutige →Adresse.

Steuerungskonfiguration

Bestandteil der CODESYS-Bedienoberfläche.

► Programmierer teilt dem Programmiersystem mit, welche Hardware programmiert werden soll.
> CODESYS lädt die zugehörigen Bibliotheken.
> Lesen und schreiben der Peripherie-Zustände (Ein-/Ausgänge) ist möglich.

stopped

stopped (engl.) = angehalten
Betriebszustand eines CANopen-Teilnehmers. In diesem Modus werden nur →NMT-Kommandos übertragen.

Symbole

Piktogramme sind bildhafte Symbole, die eine Information durch vereinfachte grafische Darstellung vermitteln (→ Kapitel *Was bedeuten die Symbole und Formatierungen?* (→ 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 Fehlfunktionen erkannt, so wird dies entweder signalisiert oder geeignete Programm-Verzweigungen eingeleitet. Das Signal oder die Verzweigungen dienen als Auslöser für andere kooperierende Systemkomponenten, die das Problem lösen sollen.

WO

wo = write only (engl.) = nur schreiben
Unidirektionale Datenübertragung: Daten können nur verändert werden, jedoch nicht gelesen.

Z

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

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z

10 Notizen • Notes • Notes

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 Györ • Tel. +36 96 / 518-397
I	ifm electronic s.a. • 20041 Agrate-Brianza (MI) • Tel. +39 039 / 68.99.982
IL	Astragal Ltd. • Azur 58001 • Tel. +972 3 -559 1660
IND	ifm electronic India Branch Office • Kolhapur, 416234 • Tel. +91 231-267 27 70
J	efector co., ltd. • 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 Billingstad • Tel. +47 66 / 98 33 50
NL	ifm electronic b.v. • 3843 GA Harderwijk • Tel. +31 341 / 438 438
P	ifm electronic s.a. • 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 Allianze 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.