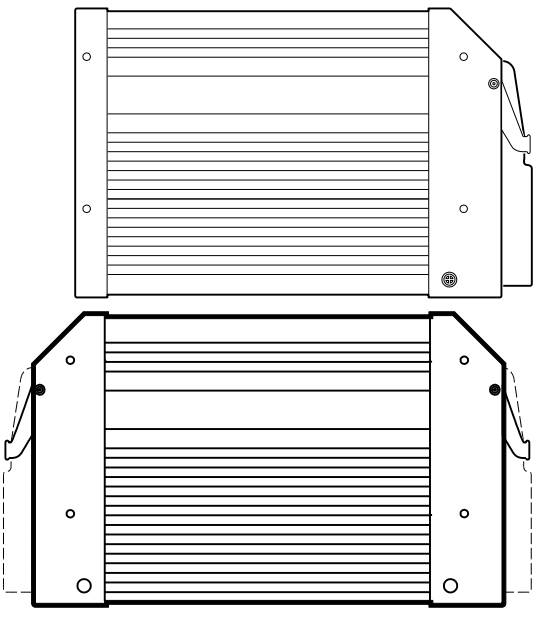
Original Programming Manual
ClassicController

CR0033
with integrated I/O modul: **CR0133**

Runtime system v02.00.03
CODESYS® V2.3

English



7391022 / 04 03 / 2018

© ifm electronic gmbh

Contents

© ifm electronic gmbh www.ifm.com

© ifm electronic gmbh www.ifm.com

© ifm electronic gmbh www.ifm.com

1 About this manual

Contents

202

1.1 Copyright

6088

© All rights reserved by ifm electronic gmbh. No part of this manual may be reproduced and used without the consent of ifm electronic gmbh.

All product names, pictures, companies or other brands used on our pages are the property of the respective rights owners:

- AS-i is the property of the AS-International Association, (→ www.as-interface.net)
- CAN is the property of the CiA (CAN in Automation e.V.), Germany (→ www.can-cia.org)
- CODESYS™ is the property of the 3S – Smart Software Solutions GmbH, Germany (→ www.codesys.com)
- DeviceNet™ is the property of the ODVA™ (Open DeviceNet Vendor Association), USA (→ www.odva.org)
- EtherNet/IP® is the property of the →ODVA™
- EtherCAT® is a registered trade mark and patented technology, licensed by Beckhoff Automation GmbH, Germany
- IO-Link® (→ www.io-link.com) is the property of the →PROFIBUS Nutzerorganisation e.V., Germany
- ISOBUS is the property of the AEF – Agricultural Industry Electronics Foundation e.V., Deutschland (→ www.aef-online.org)
- Microsoft® is the property of the Microsoft Corporation, USA (→ www.microsoft.com)
- PROFIBUS® is the property of the PROFIBUS Nutzerorganisation e.V., Germany (→ www.profibus.com)
- PROFINET® is the property of the →PROFIBUS Nutzerorganisation e.V., Germany
- Windows® is the property of the →Microsoft Corporation, USA

© ifm electronic gmbh

1.2 Overview: documentation modules for CR0033

23791
22853

The documentation for this devices consists of the following modules:
(Downloads from ifm's website → **ifm weltweit • ifm worldwide • ifm à l'échelle internationale** (→ p. 343))

Document	Contents / Description
Data sheet	Technical data in a table
Installation instructions (are supplied with the device)	• Instructions for installation, electrical installation, and commissioning • Technical data
Programming manual	• Functions of the setup menu of the device • Creation of a CODESYS project with this device • Target settings with CODESYS • Programming of the device-internal PLC with CODESYS • Description of the device-specific CODESYS function libraries
System manual "Know-How ecomatmobile"	Know-how about the following topics (examples): • Overview Templates and demo programs • CAN, CANopen • Control outputs • Visualisations • Overview of the files and libraries
System manual "The ISOBUS in the ifm controller"	Description of the configuration and the functions of the ISOBUS software in the device

© ifm electronic gmbh www.ifm.com

1.3 What do the symbols and formats mean?

203

The following symbols or pictograms illustrate the notes in our instructions:

⚠ WARNING

Death or serious irreversible injuries may result.

CAUTION

Slight reversible injuries may result.

NOTICE

Property damage is to be expected or may result.

	Important note Non-compliance can result in malfunction or interference
	Information Supplementary note
► ...	Request for action
> ...	Reaction, result
→ ...	"see"
abc	Cross-reference
123 0x123 0b010	Decimal number Hexadecimal number Binary number
[...]	Designation of pushbuttons, buttons or indications

© ifm electronic gmbh

1.4 How is this documentation structured?

16416
1508

This documentation is a combination of different types of manuals. It is for beginners and also a reference for advanced users. This document is addressed to the programmers of the applications.

How to use this manual:

- Refer to the table of contents to select a specific subject.
- Using the index you can also quickly find a term you are looking for.
- At the beginning of a chapter we will give you a brief overview of its contents.
- Abbreviations and technical terms → Appendix.

In case of malfunctions or uncertainties please contact the manufacturer at:
Contact → **ifm weltweit • ifm worldwide • ifm à l'échelle internationale** (→ p. 343)

We want to become even better! Each separate section has an identification number in the top right corner. If you want to inform us about any inconsistencies, indicate this number with the title and the language of this documentation. Thank you very much for your support!

We reserve the right to make alterations which can result in a change of contents of the documentation. You can find the current version on **ifm's** website:
→ **ifm weltweit • ifm worldwide • ifm à l'échelle internationale** (→ p. 343)

16420

NOTE

These instructions are valid for the device without and with integrated I/O module.

► In both cases, make sure to set up the PLC configuration for the device CR0033!

You can find more information about the integrated I/O module in:
→ chapter **Integrated I/O module: Description** (→ p. 248) in the appendix of this documentation.

1.5 History of the instructions (CR0033 + CR0133)

15794

What has been changed in this manual? An overview:

Date	Theme	Change
2013-06-24	various	new document structure
2014-02-03	integrated I/O module	CR0133 description added
2014-04-28	Various function blocks	More precise description of the function block input CHANNEL
2014-06-24	FB PID2	Graphic corrected
2014-06-30	Name of the documentation	"System manual" renamed as "Programming manual"
2014-07-04	Device output ERROR (clamp 13)	Output is not available. Reference note removed.
2014-07-24	FB INPUT_ANALOG FB SET_INPUT_MODE FB SET_OUTPUT_MODE	Possible error codes for ERROR output added
2014-07-31	Chapter "Error codes, diagnostics"	Error code tables added
2014-07-31	FB PHASE	Description of parameters of outputs C, ET corrected
2014-07-31	FB OUTPUT_CURRENT_CONTROL	If preset value = 0 mA >> control to 0 "within 100 ms" instead of "at once"
2014-08-08	Chapter "Inputs of the integrated I/Omodule"	completed by sections "Analogue inputs" and "binary inputs"

© ifm electronic gmbh www.ifm.com

Date	Theme	Change
2014-08-08	Chapter "Object directory of the integrated I/O module"	in the headlines "SDOs" replaced by "Object directory"
2014-08-26	Description of inputs, outputs	highside / lowside replaced by positive / negative switching
2014-08-29	Fast inputs	Note on automatically set input resistance
2014-09-29	Flag OUT_OVERLOAD_PROTECTION	generally replaced by "overload protection"
2014-11-12	Chapter "Outputs (technology)"	Section "Diagnostics of the binary outputs" supplemented or corrected
2015-01-13	Structure of documentation for error codes, system flags	• error flags: now only in the appendix, chapter **System flags** • CAN / CANopen errors and error handling: now only in the system manual "Know-How" • error codes, EMCY codes: now in the appendix, chapter **Error tables**
2015-03-10	Available memory	Description improved
2015-05-26	FB J1939_x_GLOBAL_REQUEST	More precise description
2015-06-10	Various function blocks	Description of the FB input CHANNEL corrected
2015-07-20	Inputs IN12...IN15	now without operating mode 19
2015-07-27	FB GET_IDENTITY	added with output SERIALNUMBER
2015-08-04	Outputs OUT00...01	Diagnosis via current and voltage measurement
2015-08-24	I/O module ExB01	the not realized I/O modes are deleted now
2015-10-22	System flag bit SERIAL_MODE	Debugging of the application program via USB is not possible
2016-04-27	FBs for fast inputs	Note in case of higher frequencies added
2017-01-04	FB INC_ENCODER	Can be used at the same input on the standard side of the device in combination with **one** function block FAST_COUNT, FREQUENCY, FREQUENCY_PERIOD, PERIOD, PERIOD_RATIO, PHASE
2017-01-13	Software manual for CODESYS 2.3	hint to download from the ifm homepage removed
2017-01-13	System manual for ISOBUS	hint to download from ifm homepage added
2017-02-22	FB INC_ENCODER	Can be used at the same input on the standard side of the device in combination with **one** function block PERIOD, PERIOD_RATIO, PHASE
2017-04-28	System flag LAST_RESET	from RTS v02.00.01: value ever is = "0"
2017-06-02	FRAM, MEMCPY, MEMSET: Declaration for "remanent memory freely available to the user"	removed from manual, because value and start address depend from hardware and software of the device
2017-06-02	Fast inputs	Internal resistance of the signal source must be substantially lower than the input resistance of the used input
2017-12-08	Addresses and variables of the I/Os	Declaration of the input bytes and output bytes removed because invalid
2018--03-05	FB INPUT	Value corrected for analogue input resistance measurement (16...30 000 Ω)

© ifm electronic gmbh

2 Safety instructions

Contents

213

2.1 Please note!

214
11212

No characteristics are warranted on the basis of the information, notes and examples provided in this manual. The drawings, representations and examples imply no responsibility for the system and no application-specific particularities.

- ► The manufacturer of the machine/equipment is responsible for ensuring the safety of the machine/equipment.
- ► Follow the national and international regulations of the country in which the machine/installation is to be placed on the market!

⚠ WARNING

Non-observance of these instructions can lead to property damage or bodily injury!
ifm electronic gmbh does not assume any liability in this regard.

- ► The acting person must have read and understood the safety instructions and the corresponding chapters in this manual before working on and with this device.
- ► The acting person must be authorised to work on the machine/equipment.
- ► The acting person must have the qualifications and training required to perform this work.
- ► Adhere to the technical data of the devices!
  You can find the current data sheet on **ifm's** homepage.
- ► Observe the installation and wiring information as well as the functions and features of the devices!
  → supplied installation instructions or on **ifm's** homepage
- ► Please note the corrections and notes in the release notes for the existing hardware, software and documentation, available on the **ifm** website

Website → **ifm weltweit • ifm worldwide • ifm à l'échelle internationale** (→ p. 343)

5020

NOTICE

The driver module of the serial interface can be damaged!

Disconnecting or connecting the serial interface while live can cause undefined states which damage the driver module.

- ► Do not disconnect or connect the serial interface while live.

© ifm electronic gmbh

2.2 What previous knowledge is required?

215

This document is intended for people with knowledge of control technology and PLC programming with IEC 61131-3.

To program the PLC, the people should also be familiar with the CODESYS software.

The document is intended for specialists. These specialists are people who are qualified by their training and their experience to see risks and to avoid possible hazards that may be caused during operation or maintenance of a product. The document contains information about the correct handling of the product.

Read this document before use to familiarise yourself with operating conditions, installation and operation. Keep the document during the entire duration of use of the device.

Adhere to the safety instructions.

2.3 Start-up behaviour of the controller

6827
15233
11575

⚠ WARNING

Danger due to unintentional and dangerous start of machine or plant sections!

- ► When creating the program, the programmer must ensure that no unintentional and dangerous start of machines or plant sections after a fault (e.g. e-stop) and the following fault elimination can occur!
  ⇨ Realise restart inhibit.
- ► In case of an error, set the outputs concerned to FALSE in the program!

A restart can, for example, be caused by:
- Voltage restoration after power failure
- Reset after the watchdog responded because the cycle time was too long
- Error elimination after an E-stop

To ensure safe controller behaviour:

- ► monitor the voltage supply in the application program.
- ► In case of an error switch off all relevant outputs in the application program.
- ► Additionally monitor actuators which can cause hazardous movements in the application program (feedback).
- ► Monitor relay contacts which can cause hazardous movements in the application program (feedback).
- ► If necessary, ensure that welded relay contacts in the application project cannot trigger or continue hazardous movements.

© ifm electronic gmbh www.ifm.com

2.4 Notes: serial number

20780

- ► In the user's production facility, draw a diagram of the controller network in the machine. Enter the serial number of each controller installed into the network diagram.
- ► Before downloading a software component, read out this serial number and check the network diagram to make sure that you are accessing the right controller.

2.5 Notes: TEST inputs

20781

- ► The TEST inputs of all the controllers in the machine should be wired individually and marked clearly so that they can be properly allocated to the controllers.
- ► During a service access only activate the TEST input of the controller to be accessed.

© ifm electronic gmbh www.ifm.com

3 System description

Contents

975

3.1 Information concerning the device

10415

This manual describes of the **ecomat*mobile*** family for mobile machines of **ifm electronic gmbh**:

- ClassicController: CR0033

3.2 Hardware description

Contents

14081

3.2.1 Hardware structure

Contents

15332

© ifm electronic gmbh www.ifm.com

Start conditions

19658

The device does not start until sufficient voltage is applied to the supply connection VBBs (e.g. supply of the relays on the standard side) and to clamp 15.
In vehicles clamp 15 is the plus cable switched by the ignition lock.

- permissible operating voltage = 8...32 V
- start condition: VBBs > 10 V

Relays

19661

The ClassicController has 2 internal output relays which can each separate 8 outputs from the terminal voltage VBBx (x = o | r).

The relays are only activated under the following condition:
- the global bit ERROR = FALSE

AND
- the bit RELAIS_VBBx = TRUE

In an active condition the relay contacts connect the outputs to the terminal voltage VBBx.

Activate the corresponding outputs no earlier than ≥ 45 ms after power-on of the relays!

Principle block diagram

19662

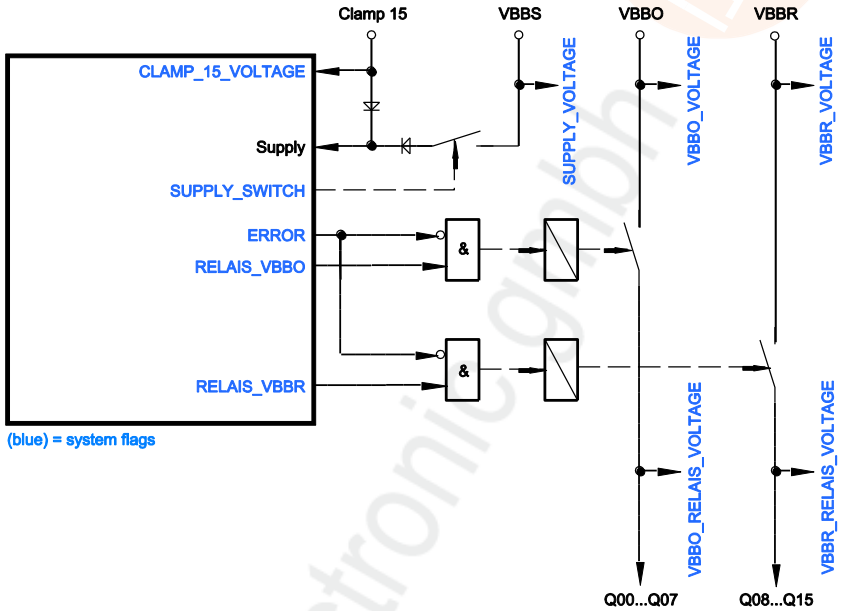


Figure: principle block diagram of supply and relays

© ifm electronic gmbh

Available memory

13736

FLASH-Speicher

8136

FLASH memory (non-volatile, slow memory) overall existing in the device	2 176 kByte

Thereof the following memory areas are reserved for ...

maximum size of the application program	1 280 kByte
data other than the application program user can write data such as files, bitmaps, fonts	128 kByte
data other than the application program read data with **FLASHREAD** (→ p. 207) or write data with **FLASHWRITE** (→ p. 208) (files: 128 bytes less for header)	64 kByte

The remaining rest of the memory is reserved for system internal purposes.

SRAM

8360

SRAM (volatile, fast memory) overall existing in the device SRAM indicates here all kinds of volatile and fast memories.	2 216 kByte

Thereof the following memory areas are reserved for ...

data reserved by the application program	192 kByte

The remaining rest of the memory is reserved for system internal purposes.

FRAM

20794

FRAM (non-volatile, fast memory) overall existing in the device FRAM indicates here all kinds of non-volatile and fast memories.	128 kByte

Thereof the following memory areas are reserved for ...

variables in the application program, declared as VAR_RETAIN	4 kByte
as remanent defined flags (from %MB0...) ▶ Set the end of the memory area by FB **MEMORY_RETAIN_PARAM** (→ p. 205)!	4 kByte

The remaining rest of the memory is reserved for system internal purposes.

© ifm electronic gmbh www.ifm.com

3.2.2 Operating principle of the delayed switch-off

993

If the ecomatmobile controllers are disconnected from the supply voltage (ignition off), all outputs are normally switched off at once, input signals are no longer read and processing of the controller software (runtime system and application program) is interrupted. This happens irrespective of the current program step of the controller.

If this is not requested, the controller must be switched off via the program. After switch-off of the ignition this enables, for example, saving of memory states.

The ClassicControllers can be switched off via the program by means of a corresponding connection of the supply voltage inputs and the evaluation of the related system flags. The block diagram in the chapter **Hardware structure** (→ p. 13) shows the context of the individual current paths.

Connect terminal VBB15 to the ignition switch

2418

The internal PLC electronics is initialised via the terminal VBB15 if at terminal VBBs supply voltage is applied.

These terminals VBB15 and VBBs are monitored internally. The applied terminal voltage VBB15 can be monitored via the system flag CLAMP_15_VOLTAGE. The applied terminal voltage VBBs can be monitored via the system flag SUPPLY_VOLTAGE.

Latching

2419

Power-on of the controller:

- voltage is applied to VBB15 (clamp 15*) by means of the ignition switch.
- The system flag CLAMP_15_VOLTAGE recognises the voltage that has been applied and activates the system flag SUPPLY_SWITCH.
- SUPPLY_SWITCH activates the connection to the potential VBBs.

> The ignition switch is bypassed. Latching of the control voltage is established.

Power-off of the controller via clamp 15:

- The system flag CLAMP_15_VOLTAGE recognises the switching off of the supply voltage on terminal VBB15.

► Reset the system flag SUPPLY_SWITCH in the application program.

> Latching via VBBs is removed and the controller switches off completely.

*) In vehicles clamp 15 is the plus cable switched by the ignition lock.

© ifm electronic gmbh

3.2.3 Relays: important notes!

12976

Assignment relays – potentials: → data sheet
Max. total current per relay contact (= per output group): → data sheet

NOTICE

Risk of destruction of the relay contacts!
In an emergency situation, "sticking" relay contacts can no longer separate the outputs from the power supply!

If VBBS (VBBrel) and clamp 15 are separated from the power supply at the same time, but the potentials VBBx stay connected to it, then the relays can drop even before the outputs are deactivated by the system.

In this case the relays separate the outputs from the power supply **under load**. This significantly reduces the life cycle of the relays.

► If VBBx is permanently connected to the power supply:
  • also connect VBBS (VBBrel) permanently and
  • switch off the outputs via the program with the help of clamp 15.

© ifm electronic gmbh

3.2.4 Monitoring concept

Contents

991

The controller monitors the supply voltages and the system error flags.
Depending on the status...

- the controller switches off the internal relays
  > the outputs are de-energised, but retain their logic state

or:

- the runtime system deactivates the controller
  > the program stops
  > the outputs change to logic "0"
  > the status LED goes out

© ifm electronic gmbh

Monitoring of the supply voltages VBBx

6752

In case of a fault we differentiate 2 scenarios:

Terminal voltage VBBx falls below the limit value of 5.25 V

15752

> The controller detects undervoltage. The outputs supplied by the terminal voltage VBBx are deactivated.
> If the terminal voltage recovers and returns to the normal range (> 10 V), the outputs are reactivated.

13975

⚠ WARNING

Dangerous restart possible!
Risk of personal injury! Risk of material damage to the machine/plant!

If in case of a fault an output is switched off via the hardware, the logic state generated by the application program is not changed.

► Remedy:
- Reset the output logic in the application program!
- Remove the fault!
- Reset the outputs depending on the situation.

Terminal voltage VBBs falls below the limit value of 10 V

20638

> The controller continues to operate until the voltage has dropped so far that the internal voltages created from it also drop.

Below 10 V no retain data is saved. → flag RETAIN_WARNING

> In case of a drop of the internal voltages the controller goes into reset.
Execution of the runtime and application programs is interrupted.
This happens irrespective of the current program step of the PLC.
> A restart of the controller is not carried out before the supply voltages are above the limit value again.

© ifm electronic gmbh www.ifm.com

Operating principle of the monitoring concept

2421

⚠ WARNING

Danger due to unintentional switch-off of all outputs!

If monitoring routines detect a system error:
> the device deactivates the energy for all outputs.

During program processing the output relays are completely controlled via the software by the user. So a parallel contact of the safety chain, for example, can be evaluated as an input signal and the output relay can be switched off accordingly. To be on the safe side, the corresponding applicable national regulations must be complied with.

If an error occurs during program processing, the relays can be switched off using the system flag bit ERROR to disconnect critical plant sections.

Manual setting of a flag bit ERROR_VBB... has NO effects on the relays!

11575

⚠ WARNING

Danger due to unintentional and dangerous start of machine or plant sections!

- When creating the program, the programmer must ensure that no unintentional and dangerous start of machines or plant sections after a fault (e.g. e-stop) and the following fault elimination can occur!
  ⇨ Realise restart inhibit.
- In case of an error, set the outputs concerned to FALSE in the program!

If a watchdog error occurs, ...
> the program processing is interrupted automatically
> the outputs become currentless and go to logical "0"
> the controller is reset
> the controller then starts again as after power on.

© ifm electronic gmbh www.ifm.com

Reference voltage output

13934

The reference voltage output is used to supply sensors with a stable voltage which is not subjected to the fluctuations of the supply voltage.

13402

NOTICE

Reference voltage output can get damaged!

► Do NOT apply any external voltage!

Via the binary system variables REFERENCE_VOLTAGE_5 or REFERENCE_VOLTAGE_10 the voltage is set on the reference voltage output [V_{REF} OUT]:

REFERENCE_VOLTAGE_10	REFERENCE_VOLTAGE_5	Reference voltage [V_{REF} OUT]
FALSE	FALSE	0 V
FALSE	TRUE	5 V
TRUE	FALSE	10 V
TRUE	TRUE	0 V

► If reference voltage = 10 V selected:
supply the controller with min. 13 V!

► Voltage monitoring on the reference voltage output with system variable REF_VOLTAGE.

© ifm electronic gmbh

3.2.5 Inputs (technology)

Contents

14090

Analogue inputs

15446

The analogue inputs can be configured via the application program. The measuring range can be set as follows:

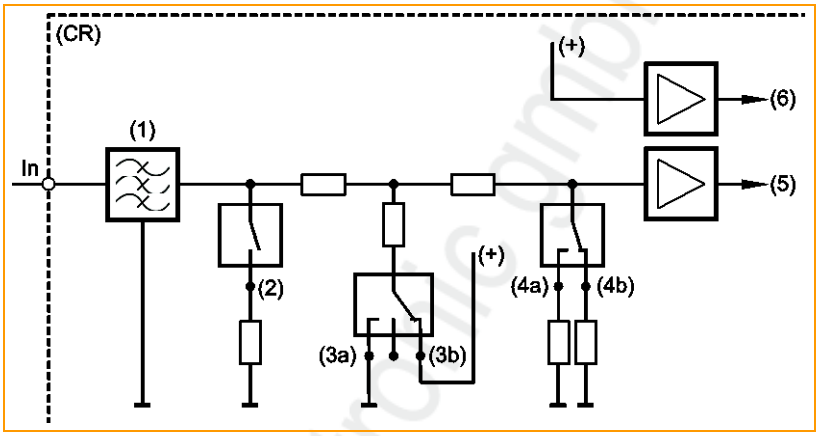
- current input 0...20 mA
- voltage input 0...10 V
- voltage input 0...32 V
- resistance measurement 16...30 000 Ω / 3...680 Ω (measurement to GND)

The voltage measurement can also be carried out ratiometrically (0...1000 ‰, adjustable via function blocks). This means potentiometers or joysticks can be evaluated without additional reference voltage. A fluctuation of the supply voltage has no influence on this measured value.

As an alternative, an analogue channel can also be evaluated binarily.

> In case of ratiometric measurement the connected sensors should be supplied with VBBs of the device. So, faulty measurements caused by offset voltage are avoided.

8971



In = pin multifunction input n
(CR) = device
(1) = input filter
(2) = analogue current measuring
(3a) = binary-input plus switching
(3b) = binary-input minus switching
(4a) = analogue voltage measuring 0...10 V
(4b) = analogue voltage measuring 0...32 V
(5) = voltage
(6) = reference voltage

Figure: principle block diagram multifunction input

8972

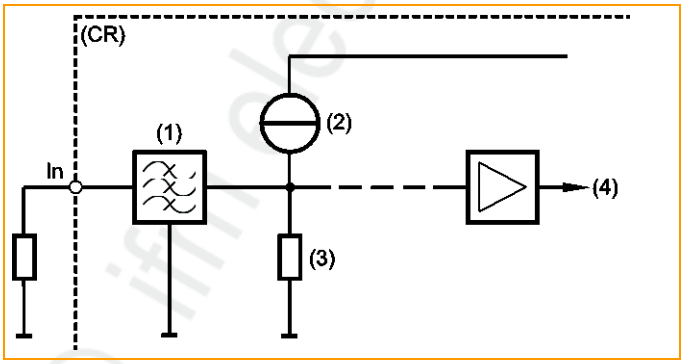


Figure: block diagram of the resistor survey input

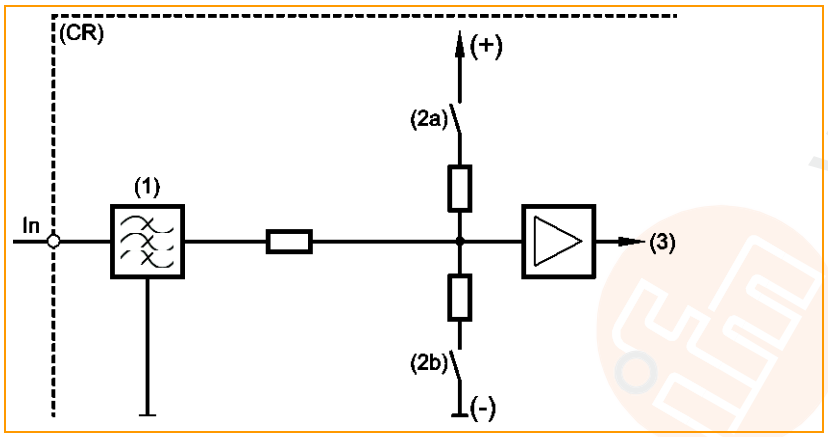
In = pin resistor survey input n
(CR) = device
(1) = input filter
(2) = constant-current source
(3) = internal resistance
(4) = voltage

Binary inputs

1015
7345

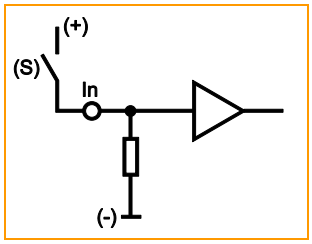
The binary input can be operated in following modes:
- binary input plus switching (BL) for positive sensor signal
- binary input minus switching (BH) for negative sensor signal

Depending on the device the binary inputs can configured differently. In addition to the protective mechanisms against interference, the binary inputs are internally evaluated via an analogue stage. This enables diagnosis of the input signals. But in the application software the switching signal is directly available as bit information



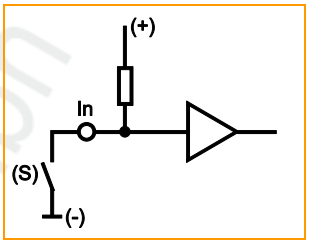
In = pin binary-input n
(CR) = device
(1) = input filter
(2a) = input minus switching
(2b) = input plus switching
(3) = voltage

Figure: basic circuit of binary input minus switching / plus switching for negative and positive sensor signals



In = pin binary input n
(S) = sensor

Basic circuit of binary input plus switching (BL)
for positive sensor signal:
Input = open ⇨ signal = low (GND)



In = pin binary input n
(S) = sensor

Basic circuit of binary input minus switching (BH)
for negative sensor signal:
Input = open ⇨ signal = high (supply)

For some of these inputs (→ data sheet) the potential can be selected to which it will be switched.

© ifm electronic gmbh www.ifm.com

Input group I00...I11

19315

These inputs are a group of multifunction channels.

These inputs can be used as follows (each input separately configurable):
- analogue input 0...20 mA
- analogue input 0...10 V
- analogue input 0...32 V
- voltage measurement ratiometric 0...1000 ‰
- binary input plus switching (BL) for positive sensor signal (with/without diagnosis)
- binary input minus switching (BH) for negative sensor signal
- fast input for e.g. incremental encoders and frequency or interval measurement

→ chapter **Possible operating modes inputs/outputs** (→ p. 243)

Sensors with diagnostic capabilities to NAMUR can be evaluated.

All inputs show the same behaviour concerning function and diagnosis. But:
I00...I07: multifunction input with supply voltage dependent levels for frequency measurement.
I08...I11: multifunction input with fixed levels for frequency measurement.

ⓘ Detailed description → chapter **Address assignment inputs / outputs**

► Configuration of each input is made via the application program:
- FB **INPUT_ANALOG** (→ p. 131) > input MODE or:
- FB **SET_INPUT_MODE** (→ p. 134) > input MODE
- Fast inputs with the following FBs:

FAST_COUNT (→ p. 142)	Counter block for fast input pulses
FREQUENCY (→ p. 144)	Measures the frequency of the signal arriving at the selected channel
FREQUENCY_PERIOD (→ p. 146)	Measures the frequency and the cycle period (cycle time) in [µs] at the indicated channel
INC_ENCODER (→ p. 148)	Up/down counter function for the evaluation of encoders
INC_ENCODER_HR (→ p. 150)	Up/down counter function for the high resolution evaluation of encoders
PERIOD (→ p. 152)	Measures the frequency and the cycle period (cycle time) in [µs] at the indicated channel
PHASE (→ p. 156)	Reads a pair of channels with fast inputs and compares the phase position of the signals

> If the analogue inputs are configured for current measurement, the device switches to the safe voltage measurement range (0...32V DC) and the corresponding error bit in the flag byte ERROR_CURRENT_Ix is set when the final value (> 21.7 mA) is exceeded.
The device checks once a second if the current value is again below the limit value. When the value is again below the limit value, the input automatically switches back to the current measurement range.

► For NAMUR: if the diagnosis function is to be used, additionally activate this mode:
- function block SET_INPUT_MODE > set input DIAGNOSTICS.

- For NAMUR: The useful signal of the switches or sensors should not be higher than the system supply voltage.

NAMUR diagnosis for binary signals of non-electronic switches:

► Equip the switch with an additional resistor connection!

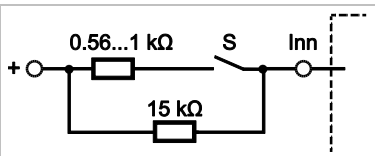


Figure: non-electronic switch S at input Inn

© ifm electronic gmbh

13956

> The result of the diagnostics is for example shown by the following system flags:

System flags (symbol name)	Type	Description
ERROR_BREAK_Ix (0...x, value depends on the device, → data sheet)	DWORD	input double word x: wire break error or (resistance input): short to supply [Bit 0 for input 0] ... [bit z for input z] of this group Bit = TRUE: error Bit = FALSE: no error
ERROR_SHORT_Ix (0...x, value depends on the device, → data sheet)	DWORD	input double word x: short circuit error only if input mode = IN_DIGITAL_H [Bit 0 for input 0] ... [bit z for input z] of this group Bit = TRUE: error Bit = FALSE: no error

> In the application program, the system variables ANALOG00...ANALOGxx can be used for customer-specific diagnostics.

© ifm electronic gmbh
www.ifm.com

Input group I12...I15

10424

These inputs are a group of multifunction channels.

These inputs can be used as follows (each input separately configurable):
- binary input plus switching (BL) for positive sensor signal (with/without diagnosis)
- input for resistance measurement (e.g. temperature sensors or fuel sensors)

→ chapter **Possible operating modes inputs/outputs** (→ p. 243)

Sensors with diagnostic capabilities to NAMUR can be evaluated.

► Configuration of each input is made via the application program:
- FB **INPUT_ANALOG** (→ p. 131) > input MODE or:
- FB **SET_INPUT_MODE** (→ p. 134) > input MODE

Resistance measurement

9773

Typical sensors on these inputs:
- tank level
- temperature (PT1000, NTC)

8972

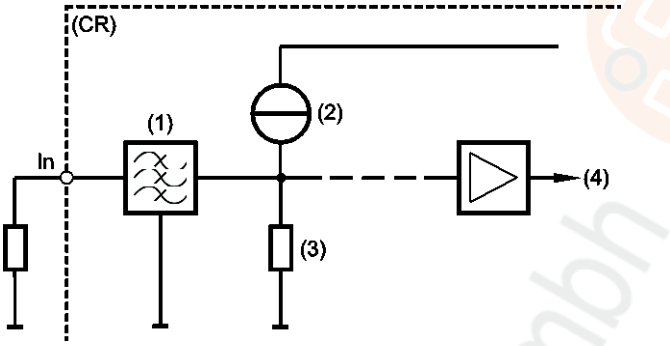


Figure: block diagram of the resistor survey input

In = pin resistor survey input n
(CR) = device
(1) = input filter
(2) = constant-current source
(3) = internal resistance
(4) = voltage

8970

The resistance for this device is not linearly dependent on the resistance value, → figure:

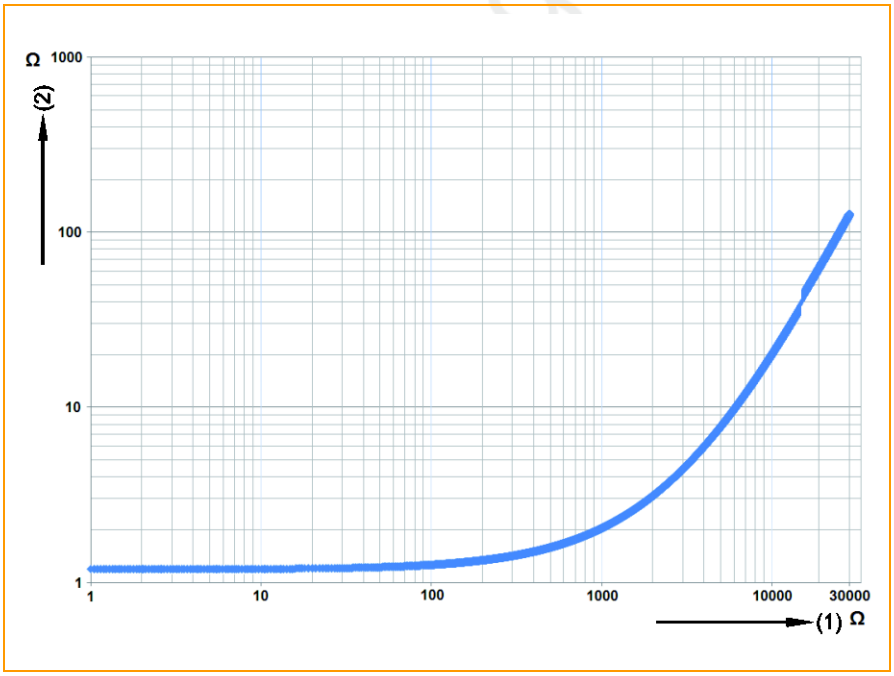


Figure: resolution dependent on the resistance value

(1) = resistance value at the input
(2) = resolution

By how many ohms does the measured value change when the signal of the A/D converter on the input changes by 1?
Examples:

- In the range of 1...100 Ω the resolution is 1.2 Ω.
- In the range of 1 kΩ the resolution is approx. 2 Ω.
- In the range of 2 kΩ the resolution is approx. 3 Ω.
- In the range of 3 kΩ the resolution is approx. 6 Ω.
- In the range of 6 kΩ the resolution is approx. 10 Ω.
- In the range of 10 kΩ the resolution is approx. 11 Ω
- In the range of 20 kΩ the resolution is approx. 60 Ω.

© ifm electronic gmbh
www.ifm.com

3.2.6 Outputs (technology)

Contents

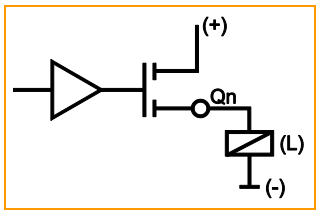
14093

Binary outputs

14094

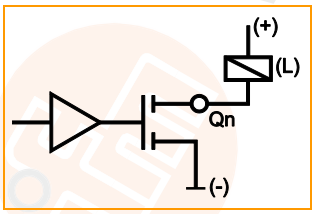
The following operating modes are possible for the device outputs (→ data sheet):
- binary output, plus switching (BH) with/without diagnostic function
- binary output minus switched (BL) without diagnostic function

15450



Qn = pin output n
(L) = load

Basic circuit of output plus switching (BH)
for positive output signal



Qn = pin output n
(L) = load

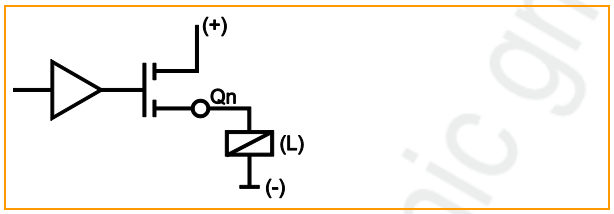
Basic circuit of output minus switching (BL)
for negative output signal

PWM outputs

14095

The following operating modes are possible for the device outputs (→ data sheet):
- PWM output, plus switching (BH) without diagnostic function

15451



Qn = pin output n
(L) = load

Basic circuit of output plus switching (BH)
for positive output signal

© ifm electronic gmbh

Protective functions of the outputs

15248

The outputs of this device are protected against overload and short circuit within specific ranges.
→ data sheet

Definition: overload

15249

Overload can only be detected on an output with current measurement.

Overload is defined as ...
"a nominal maximum current of 12.5 %".

Definition: short circuit

15250

A short circuit can be detected on all outputs with diagnostic capabilities.
Precondition: output is NOT configured for current measurement.

A short circuit is defined as ...
"a drop of the output voltage below 88% (± 2.5% of the measured value) of the corresponding supply voltage."

> A ground fault can only be detected in case of output = TRUE.

© ifm electronic gmbh www.ifm.com

Reaction of the outputs to overload or short circuit

15251

Self-protection of the output

15333

The hardware protects itself, irrespective of the operating mode of the output and of the fault detection. In case of a too high thermal load (caused by short circuit or overload), the output driver begins to clock.

The driver may be damaged in case of too long clocking of the output (several hours).

We therefore recommend:
that you operate device outputs with diagnostic capabilities in the following mode, since, in this case, the software protects the drivers additionally by switching off:
- function block **SET_OUTPUT_MODE** (→ p. 159) > input DIAGNOSTICS = TRUE and
- function block SET_OUTPUT_MODE > input PROTECTION = TRUE

Reaction according to the operating mode of the output

15479

In case of an overload or a short circuit, the behaviour of the output depends on its operating mode (→ FB **SET_OUTPUT_MODE** (→ p. 159) > inputs DIAGNOSTICS and PROTECTION):

- DIAGNOSTICS = FALSE and PROTECTION = FALSE:
  > the output continues to be operated.
- DIAGNOSTICS = TRUE and PROTECTION = FALSE:
  > error is detected and signalled as error code (→ chapter **Error codes** (→ p. 314)).
  This depends on the type of output and the current or voltage at the output.
  The programmer can react to the error in the program.
- DIAGNOSTICS = TRUE and PROTECTION = TRUE:
  > error is detected and signalled as error code (→ chapter Error codes).
  > the respective output is switched off.
  > The logic state of the output remains unaffected by this!
  > The controller checks every second if the error has been eliminated.
  If error eliminated: Controller switches the output on again.

Reaction when PWM1000, OUTPUT_CURRENT_CONTROL, OUTPUT_BRIDGE is/are used

15480

It is different when the following FBs are used:
- **PWM1000** (→ p. 171)
- **OUTPUT_CURRENT_CONTROL** (→ p. 168)
- **OUTPUT_BRIDGE** (→ p. 163)

There is no diagnostics.
The **Self-protection of the output** (→ p. 30) becomes active.

Reaction for outputs with current feedback

20641

► For outputs with current feedback:
Request the typical current for the output in the application program!
It is the responsibility of the application programmer to react to the event.

© ifm electronic gmbh www.ifm.com

Output group Q00...Q15

10445

These outputs are a group of multifunction channels.

These outputs provide several function options (each output separately configurable):
- binary output, plus switching (BH), partly also minus switching (BL)
- analogue current-controlled output (PWMi)
- analogue output with pulse-width modulation (partly as H-bridge)

→ chapter **Possible operating modes inputs/outputs** (→ p. 243)

► Configuration of each output is made via the application program:
→ function block **SET_OUTPUT_MODE** (→ p. 159) > input MODE
indicate the load currents → FB **OUTPUT_CURRENT** (→ p. 167)
PWM output: → FB **PWM1000** (→ p. 171)
PWMi output: → FB **OUTPUT_CURRENT_CONTROL** (→ p. 168)
control H-bridge → FB **OUTPUT_BRIDGE** (→ p. 163)

► Configure the current measuring range for outputs Q00…Q03 and Q08…Q11 (either 2 A or 4 A):
→ function block SET_OUTPUT_MODE > input CURRENT_RANGE

When using the H-bridge current control is not supported.

NOTE

To protect the internal measuring resistors, the overload protection should always be active (default setting). The following protection is given:
- current measuring range = 2 A: protection from 2.25 A
- current measuring range = 3 A: protection from 3.375 A
- current measuring range = 4 A: protection from 4.5 A.

The function is **not** supported in PWM mode.
The function can be deactivated if required.

For the limit values please make sure to adhere to the data sheet!

Wire break and short circuit detection are active when ...
- the output is configured as "binary plus switching" (BH) AND
- the output is switched ON.

► When using outputs switching to mass the supply voltage on the connected load must not be higher than the supply voltage(s) of the output group(s)!

► Should be avoided that an output group can continued to be operated in spite of deactivated relay, the load on an output switching against ground must only be supplied via an output switching against a supply of the same output group.

13976

Depending on the ambient temperature a short circuit cannot be reliably detected from a certain short circuit current since the output drivers temporarily deactivate themselves for protection against destruction.

© ifm electronic gmbh www.ifm.com

Diagnosis: binary outputs (via current measurement)

19398
19396

The diagnostics of these outputs is made via internal current measurement in the output:

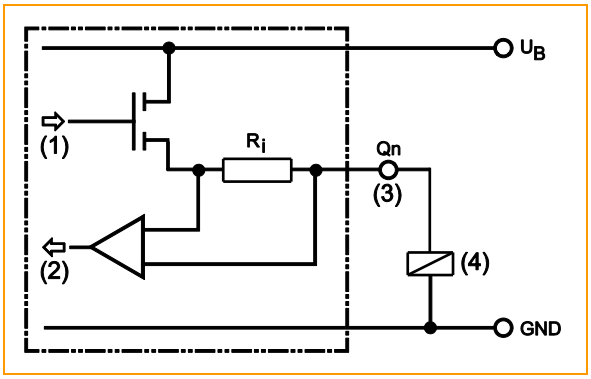


Figure: principle block diagram
(1) Output channel
(2) Read back channel for diagnostics
(3) Pin output n
(4) Load

Diagnosis: overload (via current measurement)

19437
15249

Overload can only be detected on an output with current measurement.
Overload is defined as ...
"a nominal maximum current of 12.5 %".

Diagnosis: wire break (via current measurement)

19400

Wire-break detection is done via the read back channel inside the output.

Prerequisite for diagnosis:	output = TRUE
Diagnosis = wire break:	no current flows on the resistor Ri (no voltage drops). Without wire break the load current flows through the series resistor Ri generating a voltage drop which is evaluated via the read back channel.

Diagnosis: short circuit (via current measurement)

19401

Wire-break detection is done via the read back channel inside the output.

Prerequisite for diagnosis:	output = TRUE
Diagnosis = short circuit against GND:	the supply voltage drops over the series resistor Ri

© ifm electronic gmbh

3.2.7 Note on wiring

1426

The wiring diagrams (→ installation instructions of the devices, chapter "Wiring") describe the standard device configurations. The wiring diagram helps allocate the input and output channels to the IEC addresses and the device terminals.

The individual abbreviations have the following meaning:

A	Analogue input
BH	Binary high side input: minus switching for negative sensor signal Binary high side output: plus switching for positive output signal
BL	Binary low side input: plus switching for positive sensor signal Binary low side output: minus switching for negative output signal
CYL	Input period measurement
ENC	Input encoder signals
FRQ	Frequency input
H bridge	Output with H-bridge function
PWM	**P**ulse-**w**idth **m**odulated signal
PWMi	PWM output with current measurement
IH	Pulse/counter input, high side: minus switching for negative sensor signal
IL	Pulse/counter input, low side: plus switching for positive sensor signal
R	Read back channel for one output

Allocation of the input/output channels: → Catalogue, mounting instructions or data sheet

3.2.8 Safety instructions about Reed relays

7348

For use of non-electronic switches please note the following:

6915

Contacts of Reed relays may be clogged (reversibly) if connected to the device inputs without series resistor.

- **Remedy:** Install a series resistor for the Reed relay:
  Series resistor = max. input voltage / permissible current in the Reed relay
  Example: 32 V / 500 mA = 64 Ohm
- The series resistor must not exceed 5 % of the input resistance RE of the device input (→ data sheet). Otherwise, the signal will not be detected as TRUE.
  Example:
  RE = 3 000 Ohm
  ⇒ max. series resistor = 150 Ohm

© ifm electronic gmbh www.ifm.com

3.2.9 Feedback in case of externally supplied outputs

2422

In some applications actuators are not only controlled by outputs of the PLC but additionally by external switches. In such cases the externally supplied outputs must be protected with blocking diodes (→ see graphics below).

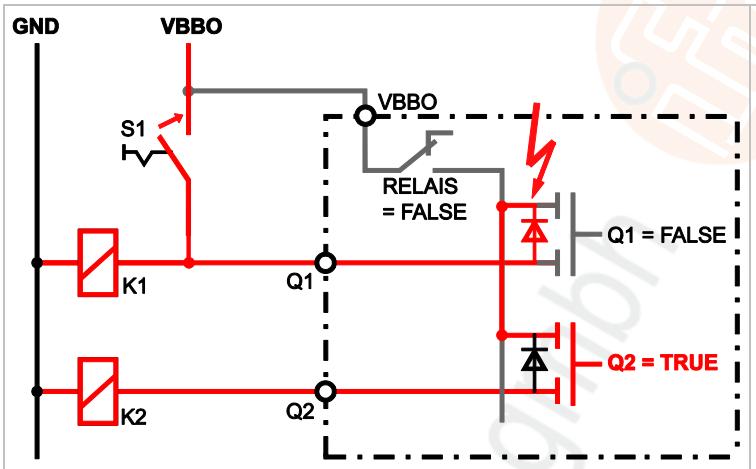
NOTICE

Destruction of outputs if there is inadmissible feedback!

If actuators are externally controlled, the corresponding potential bar of the same output group must not become potential-free (e.g. for RELAIS = FALSE).

Otherwise the terminal voltage VBBx is fed back to the potential bar of the output group via the protective diode integrated in the output driver of the external connected output. A possibly other set output of this group thus triggers its connected load. The load current destroys the output which feeds back.

► Protect externally supplied outputs by means of blocking diodes!

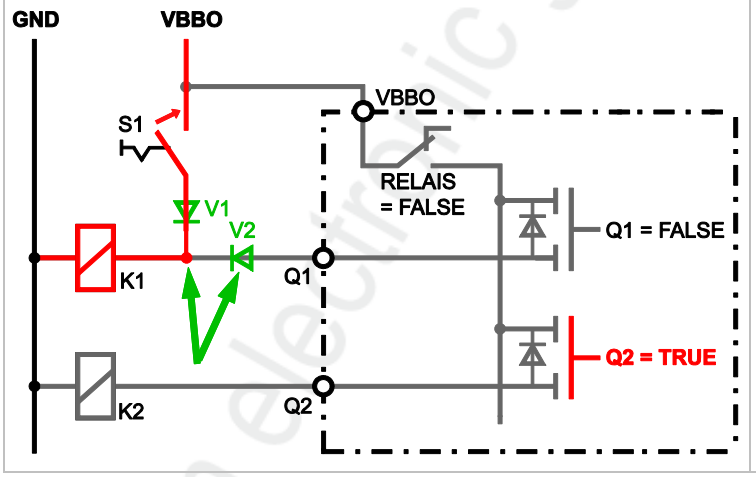


Example:

The flag RELAIS switches off the supply VBBo of the output group.

Without blocking diodes the external switch S1 feeds the supply VBBo via the internal protective diode (red) from output Q1 to the internal potential bar of the outputs.

If output Q2 = TRUE (→ graphic), K2 will receive voltage via the protective diode Q1 despite RELAIS = FALSE (red lines). Due to overload this protective diode burns out and the output Q1 is destroyed!



Graphic: example wiring with blocking diodes due to the danger of feedback

Remedy:

Insert the blocking diodes V1 and V2 (→ green arrows)!

Successful:

If RELAIS = FALSE, K2 remains switched off, even if Q2 = TRUE.

© ifm electronic gmbh

NOTE

Help for externally supplied outputs

► The externally supplied outputs must be decoupled via diodes so that no external voltage is applied to the output terminal.

© ifm electronic gmbh
www.ifm.com

3.2.10 Status LED

20774

The operating states are indicated by the integrated status LED (default setting).

LED colour	Display	Description
Off	Permanently off	No operating voltage
		
Yellow	Briefly on	Initialisation or reset checks
	 (time frame = 200 ms)	
Orange	Flashing with 0.2 Hz	TEST=FALSE: no runtime system loaded
	 (time frame = 1 s)	
Green	Flashing with 5 Hz	TEST=TRUE: no runtime system loaded
	 (time frame = 200 ms)	
Green	Flashing with 2 Hz	Application = RUN
	 (time frame = 200 ms)	
Green	Permanently on	Application = STOP or: no application loaded
		
Red	Flashing with 2 Hz	TEST=TRUE: Application = RUN and ERROR STOP / FATAL ERROR
	 (time frame = 200 ms)	
Red	Briefly on	Trap error or: TEST=FALSE: FATAL ERROR
	 (time frame = 200 ms)	
Red	Permanently on	TEST=FALSE: ERROR STOP
		

The status LED can be changed by the programming system for the operating states STOP and RUN.

© ifm electronic gmbh www.ifm.com

Control the LED in the application program

20775

With this device the status LED can also be set by the application program. To do so, the following system variables are used (→ chapter **System flags** (→ p. 231)):

System flags (symbol name)	Type	Description
LED	WORD	LED color for "LED switched on": 0x0000 = LED_GREEN (preset) 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 color for "LED switched off": 0x0000 = LED_GREEN 0x0001 = LED_BLUE 0x0002 = LED_RED 0x0003 = LED_WHITE 0x0004 = LED_BLACK (preset) 0x0005 = LED_MAGENTA 0x0006 = LED_CYAN 0x0007 = LED_YELLOW
LED_MODE	WORD	LED flashing frequency: 0x0000 = LED_2HZ (flashes at 2 Hz; preset) 0x0001 = LED_1HZ (flashes at 1 Hz) 0x0002 = LED_05HZ (flashes at 0.5 Hz) 0x0003 = LED_0HZ (lights permanently with value in LED)

NOTE

► Do NOT use the LED color RED in the application program.

> In case of an error the LED color RED is set by the runtime system.
BUT: If the colors and/or flashing modes are changed in the application program, the above table with the default setting is no longer valid.

© ifm electronic gmbh www.ifm.com

3.3 Interface description

Contents

14098

3.3.1 Serial interface

14099

This device features a serial interface.

The serial interface can generally be used in combination with the following functions:
- program download
- debugging
- free use of the application

12998

NOTE

The serial interface is not available to the user by default, because it is used for program download and debugging.

The interface can be freely used if the user sets the system flag bit SERIAL_MODE=TRUE. Debugging of the application program is then only possible via any of the CAN interfaces.

Connections and data → data sheet

3.3.2 USB interface

14100

This device features a USB interface for program download and debugging.

Connections and data → data sheet
Install the USB driver on the PC → installation instructions / operating instructions
Settings in CODESYS for [Online] > [Communication Parameters...] via USB:

Device	Runtime system version	Parameter	Value
CR0032	< V03.00.00	Baud rate	115200
CR0032	≥ V03.00.01	Baud rate	4800...57600
CR0033, CR0133	≤ V02.00.01	Baud rate	115200
CR0033, CR0133	≥ V02.00.02	Baud rate	4800...57600
CR0232, CR0233	all	Baud rate	115200
CR0234, CR0235	all	Baud rate	4800...57600
CR7n32	≤ V01.00.04	Baud rate	115200
CR7n32	≥ V01.00.05	Baud rate	4800...57600
CR0n3n, CR7n32	all	Motorola byteorder	No
CR0n3n, CR7n32	all	Flow Control	On

© ifm electronic gmbh

3.3.3 CAN interfaces

Contents

14101

Connections and data → data sheet

CAN: interfaces and protocols

13820
14587

The devices are equipped with several CAN interfaces depending on the hardware design. Basically, all interfaces can be used with the following functions independently of each other:
- Layer 2: CAN at level 2 (→ chapter **Function elements: CAN layer 2** (→ p. 76))
- CANopen master (→ chapter **Function elements: CANopen master** (→ p. 85))
- CANopen slave (→ chapter **Function elements: CANopen slave** (→ p. 95))
- CANopen network variables (via CODESYS)
- SAE J1939 (for drive management, → chapter **Function elements: SAE J1939** (→ p. 108))
- bus load detection
- error frame counter
- download interface
- 100 % bus load without package loss

11793

The following CAN interfaces and CAN protocols are available in this ecomat*mobile* device:

CAN interface	CAN 1	CAN 2	CAN 3	CAN 4
Default download ID	ID 127	ID 126	ID 125	ID 124
CAN protocols	CAN Layer 2	CAN Layer 2	CAN Layer 2	CAN Layer 2
	CANopen	CANopen	CANopen	CANopen
	SAE J1939	SAE J1939	SAE J1939	SAE J1939

Standard baud rate = 125 Kbits/s

Which CANopen compatible interface works with which CANopen protocol is decided by the order in which you append the subelements in the PLC configuration:
CODESYS > [PLC Configuration] > [CR0033 Configuration Vxx] > [Append subelement] > [CANopen master] or [CANopen slave]

© ifm electronic gmbh

3.4 Software description

Contents

14107

3.4.1 Software modules for the device

Contents

14110

The software in this device communicates with the hardware as below:

software module	Can user change the module?	By means of what tool?
Application program with libraries	yes	CODESYS, MaintenanceTool
Runtime system *)	Upgrade yes Downgrade yes	MaintenanceTool
Bootloader	no	---
(Hardware)	no	---

*) The runtime system version number must correspond to the target version number in the CODESYS target system setting.
→ chapter Set up the target (→ p. 56)

Below we describe this software module:

© ifm electronic gmbh

Bootloader

14111

On delivery **ecomatmobile** controllers only contain the boot loader.
The boot loader is a start program that allows to reload the runtime system and the application program on the device.
The boot loader contains basic routines...
- for communication between hardware modules,
- for reloading the operating system.

The boot loader is the first software module to be saved on the device.

Runtime system

14112

Basic program in the device, establishes the connection between the hardware of the device and the application program.
→ chapter **Software modules for the device** (→ p. 40)

On delivery, there is normally no runtime system loaded in the controller (LED flashes green at 5 Hz). Only the bootloader is active in this operating mode. It provides the minimum functions for loading the runtime system, among others support of the interfaces (e.g. CAN).

Normally it is necessary to download the runtime system only once. Then, the application program can be loaded into the controller (also repeatedly) without affecting the runtime system.

The runtime system is provided with this documentation on a separate data carrier. In addition, the current version can be downloaded from the website of **ifm electronic gmbh**:
→ **ifm weltweit • ifm worldwide • ifm à l'échelle internationale** (→ p. 343)

Application program

14118

Software specific to the application, implemented by the machine manufacturer, generally containing logic sequences, limits and expressions that control the appropriate inputs, outputs, calculations and decisions.

8340

⚠ WARNING

The user is responsible for the reliable function of the application programs he designed. If necessary, he must additionally carry out an approval test by corresponding supervisory and test organisations according to the national regulations.

© ifm electronic gmbh www.ifm.com

Libraries

14117

ifm electronic offers several libraries (*.LIB) to match each device containing program modules for the application program. Examples:

Library	Use
ifm_CR0033_Vxxyyzz.LIB	device-specific library Must always be contained in the application program!
ifm_CR0033_CANopenxMaster_Vxxyyzz.LIB x = 1...4 = number of the CAN interface	(optional) if a CAN interface of the device is to be operated as a CANopen master
ifm_CR0033_CANopenxSlave_Vxxyyzz.LIB x = 1...4 = number of the CAN interface	(optional) if a CAN interface of the device is to be operated as a CANopen slave
ifm_CR0033_J1939_Vxxyyzz.LIB	(optional) if a CAN interface of the device is to communicate with a Diesel engine

Details: → chapter ifm libraries for the device CR0033 (→ p. 71)

© ifm electronic gmbh

3.4.2 Programming notes for CODESYS projects

Contents

7426

Here you receive tips how to program the device.

► See the notes in the CODESYS programming manual.

FB, FUN, PRG in CODESYS

8473

In CODESYS we differentiate between the following types of function elements:

FB = function block
- An FB can have several inputs and several outputs.
- An FB may be called several times in a project.
- An instance must be declared for each call.
- Permitted: Call FB and FUN in FB.

FUN = function
- A function can have several inputs but only one output.
- The output is of the same data type as the function itself.

PRG = program
- A PRG can have several inputs and several outputs.
- A PRG may only be called once in a project.
- Permitted: Call PRG, FB and FUN in PRG.

NOTE

Function blocks must NOT be called in functions!
Otherwise: During execution the application program will crash.

All function elements must NOT be called recursively, nor indirectly!

An IEC application must contain max. 8,000 function elements!

Background:

All variables of functions...
- are initialised when called and
- become invalid after return to the caller.

Function blocks have 2 calls:
- an initialisation call and
- the actual call to do something.

Consequently that means for the FB call in a function:
- every time there is an additional initialisation call and
- the data of the last call gets lost.

© ifm electronic gmbh www.ifm.com

Calculations and conversions in the application program

20779

NOTE

If the following elements are required in the application program:
- mathematical functions (e.g. ATAN),
- calculations,
- conversions (e.g. REAL_TO_BYTE),

then the following applies to the values at the inputs and outputs of the corresponding operators:

► Strictly observe the admissible value range in each individual case!

> Otherwise, this may cause an FPU error in the controller.

Examples:

20777

The value of the target format that can max. represented is exceeded.
Example:

```
REAL_TO_INT (12345678.3)
```

> INT is limited to -32768...+32767 (only integers)

20778

An existing real number is obviously in the value range of the target format.
In reality, however, the number is outside the target format (because of the internal representation of the real number).
Example:

```
DW := REAL_TO_DWORD (4294967295.0);
```

> The most accurate representation of 4294967295 in REAL is 4.294967296E9
> Therefore the value exceeds the max. permissible value of the target format by 1.
> DWORD is limited to 0...4294967295.

Note the cycle time!

8006

For the programmable devices from the controller family **ecomat*mobile*** numerous functions are available which enable use of the devices in a wide range of applications.

As these units use more or fewer system resources depending on their complexity it is not always possible to use all units at the same time and several times.

NOTICE

Risk that the device acts too slowly!
Cycle time must not become too long!

► When designing the application program the above-mentioned recommendations must be complied with and tested.

► If necessary, the cycle time must be optimised by restructuring the software and the system set-up.

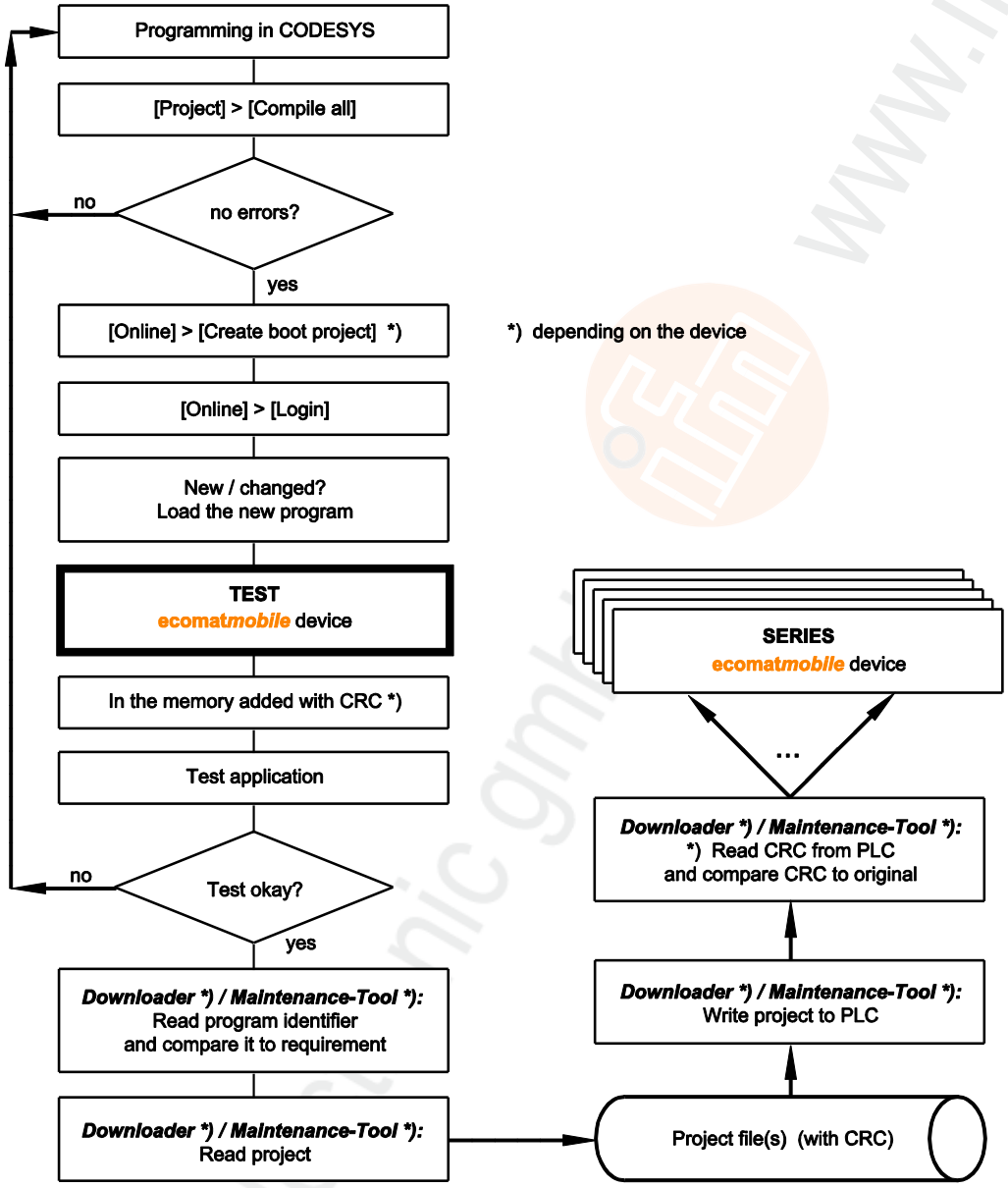
© ifm electronic gmbh www.ifm.com

Creating application program

8007

The application program is generated by the CODESYS 2.3 programming system and loaded in the controller several times during the program development for testing:
In CODESYS: [Online] > [Login] > load the new program.

For each such download via CODESYS 2.3 the source code is translated again. The result is that each time a new checksum is formed in the controller memory. This process is also permissible for safety controllers until the release of the software.



Graphics: Creation and distribution of the software

© ifm electronic gmbh www.ifm.com

Save boot project

7430

Always save the related boot project together with your application project in the device. Only then will the application program be available after a power failure in the device.

NOTE

Note: The boot project is slightly larger than the actual program.

However: Saving the boot project in the device will fail if the boot project is larger than the available IEC code memory range. After power-on the boot project is deleted or invalid.

► CODESYS menu [Online] > [Create boot project]
This is necessary after each change!

> After a reboot, the device starts with the boot project last saved.

> If NO boot project was saved:
- The device remains in the STOP operation after reboot.
- The application program is not (no longer) available.
- The LED lights green.

Using ifm downloader

8008

The ifm downloader serves for easy transfer of the program code from the programming station to the controller. As a matter of principle each application software can be copied to the controllers using the ifm downloader. Advantage: A programming system with CODESYS licence is not required.

Here you will find the current ifm downloader (min. V06.18.26):
Homepage → **ifm weltweit • ifm worldwide • ifm à l'échelle internationale** (→ p. 343)

Using ifm maintenance tool

8492

The ifm Maintenance Tool serves for easy transfer of the program code from the programming station to the controller. As a matter of principle each application software can be copied to the controllers using the ifm Maintenance Tool. Advantage: A programming system with CODESYS licence is not required.

Here you will find the current ifm Maintenance Tool:
Homepage → **ifm weltweit • ifm worldwide • ifm à l'échelle internationale** (→ p. 343)

© ifm electronic gmbH www.ifm.com

3.4.3 Operating states

Contents

14120

After power on the ecomatmobile device can be in one of five possible operating states:
- BOOTLOADER
- INIT
- STOP
- RUN
- SYSTEM STOP (after ERROR STOP)

Operating states: runtime system is not available

19217

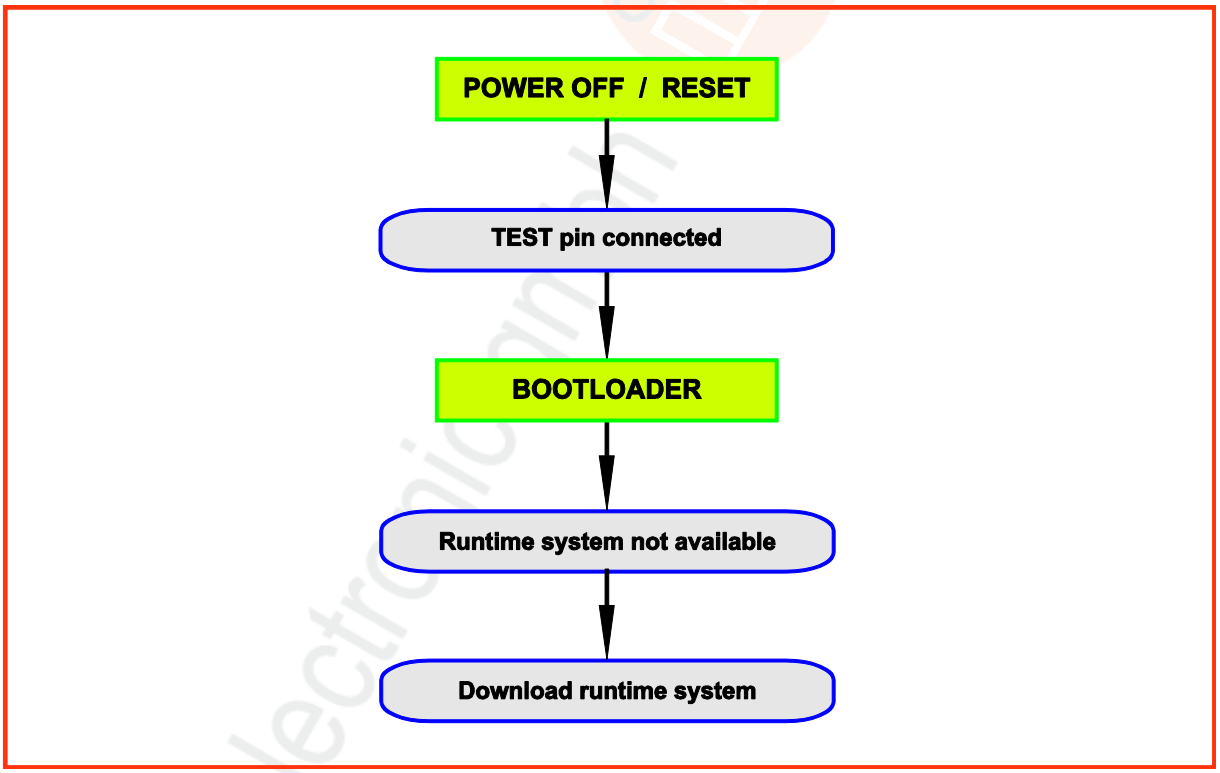


Figure: operating states (here: runtime system is not available)

© ifm electronic gmbh

Operating states: application program is not available

19218

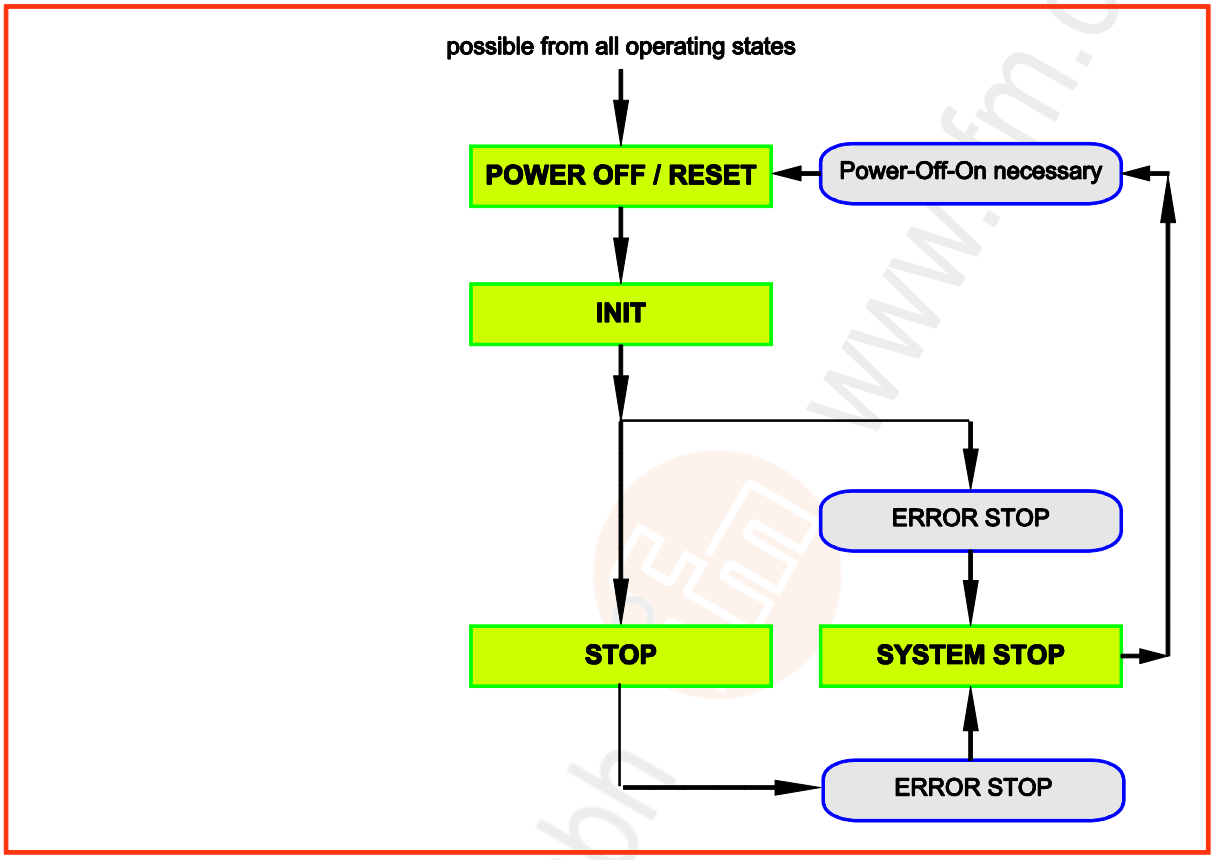


Figure: operating states (here: application program is not available)

© ifm electronic gmbh www.ifm.com

Operating states: application program is available

19219

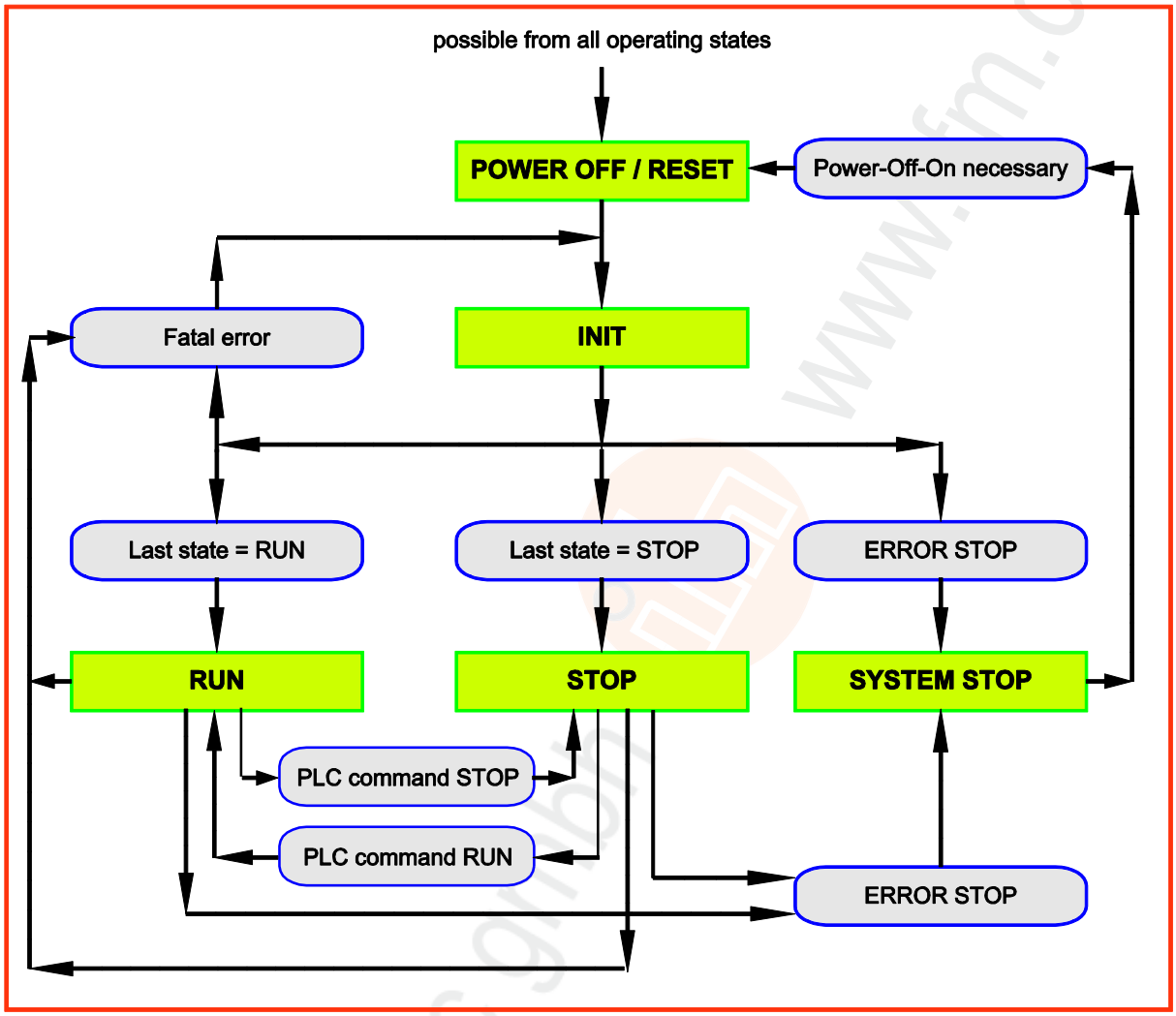


Figure: operating states (here: application program is available)

© ifm electronic gmbh

Bootloader state

1080

No runtime system was loaded. The **ecomat*mobile*** controller is in the boot loading state. Before loading the application software the runtime system must be downloaded.

> The LED flashes green (5 Hz).

INIT state (Reset)

1076

Premise: a valid runtime system is installed.

This state is passed through after every power on reset:

> The runtime system is initialised.

> Various checks are carried out, e.g. waiting for correctly power supply voltage.

> This temporary state is replaced by the RUN or STOP state.

> The LED lights yellow.

Change out of this state possible into one of the following states:
- RUN
- STOP

STOP state

1078

This state is reached in the following cases:

- From the RESET state if:
  - no program is loaded or
  - the last state before the RESET state was the STOP state
- From the RUN state by the STOP command
  - only for the operating mode = Test (→ chapter **TEST mode** (→ p. 51))

> The LED lights green.

RUN state

1077

This state is reached in the following cases:

- From the RESET state if:
  - the last state before the RESET state was the RUN state
- From the STOP state by the RUN command
  - only for the operating mode = Test (→ chapter **TEST mode** (→ p. 51))

> The LED flashes green (2 Hz).

SYSTEM STOP state

19222

The **ecomat*mobile*** controller goes to this state if a non tolerable error (ERROR STOP) was found. This state can only be left by a power-off-on reset.

> The LED lights red.

© ifm electronic gmbh www.ifm.com

3.4.4 Operating modes

1083

Independent of the operating states the controller can be operated in different modes.

TEST mode

1084

NOTICE

Loss of the stored software possible!
In the test mode there is no protection of the stored runtime system and application software.

14892

NOTE

> Connect the TEST connection to the supply voltage only AFTER you have connected the OPC client!

This operating mode is reached by applying supply voltage to the test input
(→ installation instructions > chapter "Technical data" > chapter "Wiring").

The ecomat*mobile* controller can now receive commands via one of the interfaces in the RUN or STOP mode and, for example, communicate with the programming system.

Only in the TEST mode the software can be downloaded to the controller.

The state of the application program can be queried via the flag TEST.

Summary Test input is active:
- Programming mode is enabled
- Software download is possible
- Status of the application program can be queried
- Protection of stored software is not possible

Notes: TEST inputs

20781

► The TEST inputs of all the controllers in the machine should be wired individually and marked clearly so that they can be properly allocated to the controllers.

► During a service access only activate the TEST input of the controller to be accessed.

© ifm electronic gmbh www.ifm.com

SERIAL_MODE

2548

The serial interface is available for the exchange of data in the application. Debugging the application software is then only possible via all 4 CAN interfaces.

This function is switched off as standard (FALSE). Via the flag SERIAL_MODE the state can be controlled and queried via the application program or the programming system.

→ chapter **Function elements: serial interface** (→ p. 120)

DEBUG mode

1086

If the input DEBUG of **SET_DEBUG** (→ p. 218) is set to TRUE, the programming system or the downloader, for example, can communicate with the controller and execute some special system commands (e.g. for service functions via the GSM modem CANremote).

In this operating mode a software download is not possible because the test input (→ chapter **TEST mode** (→ p. 51)) is not connected to supply voltage.

3.4.5 Performance limits of the device

7358

Note the limits of the device! → Data sheet

Watchdog behaviour

11786

In this device, a watchdog monitors the program runtime of the CODESYS application.

If the maximum watchdog time (approx. 100 ms) is exceeded:
> the device performs a reset and reboots.

This you can read in the flag LAST_RESET.

CODESYS functions

2254

You should note the following limits:

- Up to 2 048 blocks (PB, FB...) are supported.
- Flags available for user → chapter **Available memory** (→ p. 15).
  Description of the retain flags → for the corresponding FBs.

© ifm electronic gmbh www.ifm.com

4 Configurations

Contents

18065
1016

The device configurations described in the corresponding installation instructions or in the **Appendix** (→ p. 231) to this documentation are used for standard devices (stock items). They fulfil the requested specifications of most applications.

Depending on the customer requirements for series use it is, however, also possible to use other device configurations, e.g. with respect to the inputs/outputs and analogue channels.

16420

NOTE

These instructions are valid for the device without and with integrated I/O module.

► In both cases, make sure to set up the PLC configuration for the device CR0033!

You can find more information about the integrated I/O module in:
→ chapter **Integrated I/O module: Description** (→ p. 248) in the appendix of this documentation.

4.1 Set up the runtime system

Contents

14091

© ifm electronic gmbh

4.1.1 Reinstall the runtime system

14092
2733

On delivery of the ecomat*mobile* device no runtime system is normally loaded (LED flashes green at 5 Hz). Only the bootloader is active in this operating mode. It provides the minimum functions for loading the runtime system (e.g. RS232, CAN).

Normally it is necessary to download the runtime system only once. The application program can then be loaded to the device (also several times) without influencing the runtime system.

The runtime system is provided with this documentation on a separate data carrier. In addition, the current version can be downloaded from the website of **ifm electronic gmbh**:
→ **ifm weltweit • ifm worldwide • ifm à l'échelle internationale** (→ p. 343)

2689

NOTE

The software versions suitable for the selected target must always be used:
- runtime system (`ifm_CR0033_Vxxyyzz.H86`),
- PLC configuration (`ifm_CR0033_Vxx.CFG`),
- device library (`ifm_CR0033_Vxxyyzz.LIB`) and
- the further files.

V	version
xx: 00...99	target version number
yy: 00...99	release number
zz: 00...99	patch number

The basic file name (e.g. "`CR0033`") and the software version number "xx" (e.g. "`02`") must always have the same value! Otherwise the device goes to the STOP mode.

The values for "yy" (release number) and "zz" (patch number) do **not** have to match.

4368

The following files must also be loaded:
- the internal libraries (created in IEC 1131) required for the project,
- the configuration files (`*.CFG`) and
- the target files (`*.TRG`).

It may happen that the target system cannot or only partly be programmed with your currently installed version of CODESYS. In such a case, please contact the technical support department of **ifm electronic gmbh**.
Contact → **ifm weltweit • ifm worldwide • ifm à l'échelle internationale** (→ p. 343)

The runtime system is transferred to the device using the separate program "**ifm** downloader".
The software can be downloaded from **ifm's** website, if necessary:
→ **ifm weltweit • ifm worldwide • ifm à l'échelle internationale** (→ p. 343)

Normally the application program is loaded to the device via the programming system. But it can also be loaded using the **ifm** downloader if it was first read from the device (→ upload).

© ifm electronic gmbh www.ifm.com

4.1.2 Update the runtime system

13269

An older runtime system is already installed on the device. Now, you would like to update the runtime system on the device?

14158

NOTICE

Risk of data loss!

When deleting or updating the runtime system all data and programs on the device are deleted.

► Save all required data and programs before deleting or updating the runtime system!

For this operation, the same instructions apply as in the previous chapter 'Reinstall the runtime system'.

4.1.3 Verify the installation

19517
14406

► After loading of the runtime system into the controller:
- check whether the runtime system was transmitted correctly!
- check whether the right runtime system is on the controller!

► 1st check:
use the ifm downloader or the maintenance tool to verify whether the correct version of the runtime system was loaded:
- read out the name, version and CRC of the runtime system in the device!
- Manually compare this information with the target data!

► 2nd check (optional):
verify in the application program whether the correct version of the runtime system was loaded:
- read out the name and version of the runtime system in the device!
- Compare this data with the specified values!

The following FB serves for reading the data:

GET_IDENTITY (→ p. 217)	Reads the specific identifications stored in the device: • hardware name and hardware version of the device • serial number of the device • name of the runtime system in the device • version and revision no. of the runtime system in the device • name of the application (has previously been saved by means of **SET_IDENTITY** (→ p. 219))

► If the application detects an incorrect version of a runtime system:
bring all safety functions into the safe state.

© ifm electronic gmbh

4.2 Set up the programming system

Contents

3968

4.2.1 Set up the programming system manually

Contents

3963

Set up the target

2687
11379

When creating a new project in CODESYS the target file corresponding to the device must be loaded.

► Select the requested target file in the dialogue window [Target Settings] in the menu [Configuration].

> The target file constitutes the interface to the hardware for the programming system.

> At the same time, several important libraries and the PLC configuration are loaded when selecting the target.

► If necessary, in the window [Target settings] > tab [Network functionality] > activate [Support parameter manager] and / or activate [Support network variables].

► If necessary, remove the loaded (3S) libraries or complement them by further (ifm) libraries.

► Always complement the appropriate device library `ifm_CR0033_Vxxyyzz.LIB` manually!

© ifm electronic gmbh

Activate the PLC configuration (e.g. CR0033)

15824

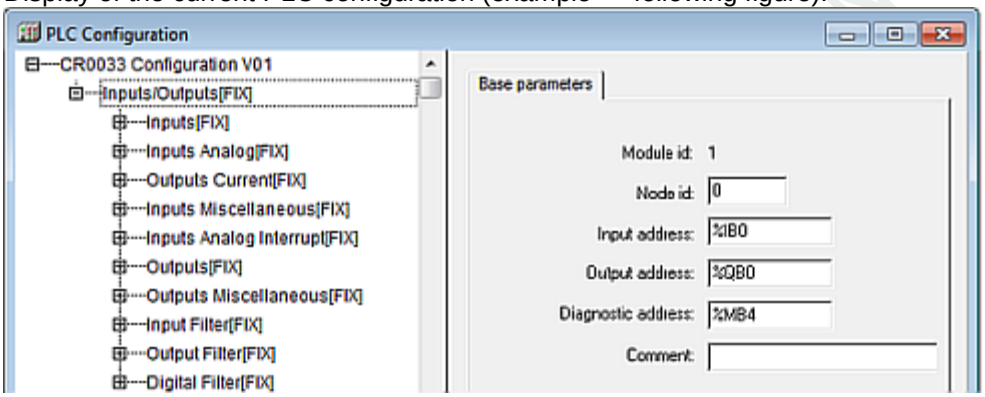
During the configuration of the programming system (→ previous section) the PLC configuration was also carried out automatically.

► The menu item [PLC Configuration] is reached via the tab [Resources].
Double-click on [PLC Configuration] to open the corresponding window.

► Click on the tab [Resources] in CODESYS:

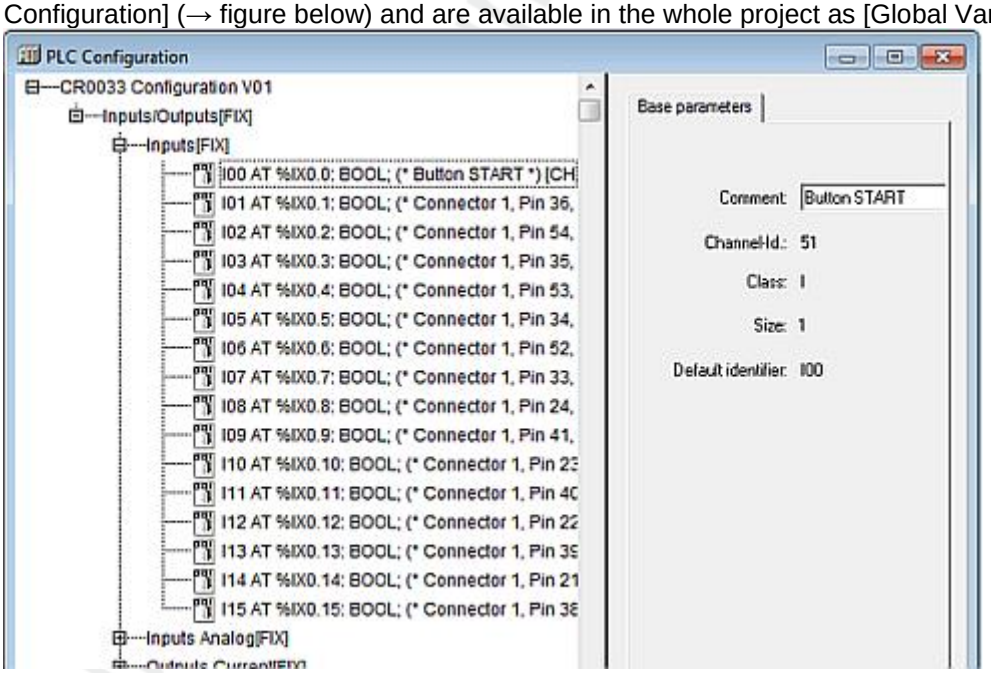
► In the left column double-click on [PLC Configuration].

> Display of the current PLC configuration (example → following figure):



Based on the configuration the user can find the following in the program environment:

- all important system and error flags
  Depending on the application and the application program, these flags must be processed and evaluated. Access is made via their symbolic names.
- The structure of the inputs and outputs
  These can directly be designated symbolically (highly recommended!) in the window [PLC Configuration] (→ figure below) and are available in the whole project as [Global Variables].



4.2.2 Set up the programming system via templates

13745

ifm offers ready-to-use templates (program templates), by means of which the programming system can be set up quickly, easily and completely.

970

When installing the **ecomat*mobile*** DVD "Software, tools and documentation", projects with templates have been stored in the program directory of your PC:
`…\ifm electronic\CoDeSys V…\Projects\Template_DVD_V…`

► Open the requested template in CODESYS via:
[File] > [New from template…]

> CODESYS creates a new project which shows the basic program structure. It is strongly recommended to follow the shown procedure.

© ifm electronic gmbh

4.3 Function configuration in general

3971

4.3.1 Configuration of the inputs and outputs (default setting)

20784

Inputs (preset)

20785

- in binary mode (plus switching)
- diagnostic function is not active

Outputs with current measurement (default setting)

20786

- in binary mode (plus switching)
- diagnostic function is active
- overload protection is active

Outputs without current measurement (default setting)

20787

- in binary mode (plus switching)
- diagnostic function is not active
- there is no overload protection

4.3.2 System variables

13519
15576

All system variables (→ chapter **System flags** (→ p. 231)) have defined addresses which cannot be shifted.

> To indicate and process a watchdog error or causes of a new start the system variable LAST_RESET is set.

© ifm electronic gmbh www.ifm.com

4.4 Function configuration of the inputs and outputs

Contents

1394

For some devices of the ecomatmobile controller family, additional diagnostic functions can be activated for the inputs and outputs. So, the corresponding input and output signal can be monitored and the application program can react in case of a fault.

Depending on the input and output, certain marginal conditions must be taken into account when using the diagnosis:

► It must be checked by means of the data sheet if the device used has the described input and output groups (→ data sheet).

- Constants are predefined (e.g. IN_DIGITAL_H) in the device libraries (`ifm_CR0033_Vxxyyzz.LIB`) for the configuration of the inputs and outputs.
  For details → **Possible operating modes inputs/outputs** (→ p. 243).

4.4.1 Configure inputs

Contents

3973

Valid operating modes → chapter **Possible operating modes inputs/outputs** (→ p. 243)

Safety instructions about Reed relays

7348

For use of non-electronic switches please note the following:

6915

Contacts of Reed relays may be clogged (reversibly) if connected to the device inputs without series resistor.

► **Remedy:** Install a series resistor for the Reed relay:
Series resistor = max. input voltage / permissible current in the Reed relay
Example: 32 V / 500 mA = 64 Ohm

► The series resistor must not exceed 5 % of the input resistance RE of the device input (→ data sheet). Otherwise, the signal will not be detected as TRUE.
Example:
RE = 3 000 Ohm
⇒ max. series resistor = 150 Ohm

© ifm electronic gmbh

Configure the software filters of the inputs

6883

A software filter that filters the measured input voltage on the analogue inputs can be configured via the system variables Ixx_FILTER. In case of a step response the filter behaves like a conventional low-pass filter, the limit frequency is set by the value entered in the system variable. Values of 0...8 are possible.

Table: limit frequency software low-pass filter on analogue input

Ixx_FILTER	Filter frequency [Hz]	Signal rise time	Remarks
0	Filter deactivated		
1	390	1 ms	
2	145	2.5 ms	
3	68	5 ms	
4	34	10 ms	Recommended, default setting
5	17	21 ms	
6	8	42 ms	
7	4	84 ms	
8	2	169 ms	
≥ 9	34	10 ms	→ Default setting

12969

After changing the filter setting, the value of this input or output is not output correctly at once. Only after the signal rise time (→ table) will the value be correct again.

The signal rise time is the time taken by a signal at the output of the filter to rise from 10 % to 90 % of the final value if an input step is applied. The signal fall time is the time taken by a signal to decrease from 90 % to 10 %.

Analogue inputs: configuration and diagnosis

13960

► Configuration of each input is made via the application program:
  • FB **INPUT_ANALOG** (→ p. 131) > input MODE or:
  • FB **SET_INPUT_MODE** (→ p. 134) > input MODE

> If the analogue inputs are configured for current measurement, the device switches to the safe voltage measurement range (0...32V DC) and the corresponding error bit in the flag byte ERROR_CURRENT_Ix is set when the final value (> 21.7 mA) is exceeded.
The device checks once a second if the current value is again below the limit value. When the value is again below the limit value, the input automatically switches back to the current measurement range.

> In the application program, the system variables ANALOG00...ANALOGxx can be used for customer-specific diagnostics.

18414

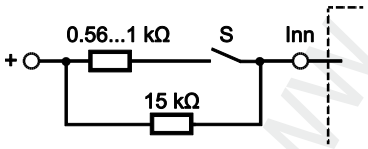
If inputI15 is not used:
► Configure input I15 as binary input!

Binary inputs: configuration and diagnosis

14516

► Configuration of each input is made via the application program:
  • FB **INPUT_ANALOG** (→ p. 131) > input MODE or:
  • FB **SET_INPUT_MODE** (→ p. 134) > input MODE

► For NAMUR: if the diagnosis function is to be used, additionally activate this mode:
  • function block SET_INPUT_MODE > set input DIAGNOSTICS.

NAMUR diagnosis for binary signals of non-electronic switches: ► Equip the switch with an additional resistor connection!	 Figure: non-electronic switch S at input Inn

13956

> The result of the diagnostics is for example shown by the following system flags:

System flags (symbol name)	Type	Description
ERROR_BREAK_Ix (0...x, value depends on the device, → data sheet)	DWORD	input double word x: wire break error or (resistance input): short to supply [Bit 0 for input 0] ... [bit z for input z] of this group Bit = TRUE: error Bit = FALSE: no error
ERROR_SHORT_Ix (0...x, value depends on the device, → data sheet)	DWORD	input double word x: short circuit error only if input mode = IN_DIGITAL_H [Bit 0 for input 0] ... [bit z for input z] of this group Bit = TRUE: error Bit = FALSE: no error

> In the application program, the system variables ANALOG00...ANALOGxx can be used for customer-specific diagnostics.

© ifm electronic gmbh www.ifm.com

Fast inputs

19318

The devices dispose of fast counting/pulse inputs for an input frequency up to 30 kHz (→ data sheet).
The input resistance of the fast inputs switches automatically depending on the applied mode or function block:

Input resistance	for mode / FB
3.2 kohms	(standard) FAST_COUNT, FREQUENCY, INC_ENCODER, PERIOD and similar FBs
50.7 kohms	input with fixed switching level 32 V

23900

The internal resistance R_i of the signal source must be substantially lower than the input resistance R_{input} of the used input (prinziple voltage alignment).
Otherwise the input signal of the fast input can be distort (low-pass characteristic).

14677

If, for example, mechanical switches are connected to these inputs, there may be faulty signals in the controller due to contact bouncing.
▶ If necessary, filter these "false signals" using the filters Ixx_DFILTER.
(→ chapter **System flags** (→ p. 231)) (not available for all inputs)

Appropriate function blocks are e.g.:

FAST_COUNT (→ p. 142)	Counter block for fast input pulses
FREQUENCY (→ p. 144)	Measures the frequency of the signal arriving at the selected channel
FREQUENCY_PERIOD (→ p. 146)	Measures the frequency and the cycle period (cycle time) in [µs] at the indicated channel
INC_ENCODER (→ p. 148)	Up/down counter function for the evaluation of encoders
INC_ENCODER_HR (→ p. 150)	Up/down counter function for the high resolution evaluation of encoders
PERIOD (→ p. 152)	Measures the frequency and the cycle period (cycle time) in [µs] at the indicated channel
PERIOD_RATIO (→ p. 154)	Measures the frequency and the cycle period (cycle time) in [µs] during the indicated periods at the indicated channel. In addition, the mark-to-space ratio is indicated in [‰].
PHASE (→ p. 156)	Reads a pair of channels with fast inputs and compares the phase position of the signals

When using these units, the parameterised inputs and outputs are automatically configured, so the programmer of the application does not have to do this.

© ifm electronic gmbh www.ifm.com

Configure the hardware filter

19320

A digital hardware filter can be configured on the fast counter and pulse inputs via the system variable Ixx_DFILTER. The value in µs (max. 100 000) indicates how long a binary level must be applied without interruption before it is adopted. Default = 0 µs.

The level change of the input signal is delayed by the value set in the filter.

The filter has an effect on the detected signals only for the following function blocks:

FAST_COUNT (→ p. 142)	Counter block for fast input pulses
FREQUENCY (→ p. 144)	Measures the frequency of the signal arriving at the selected channel
FREQUENCY_PERIOD (→ p. 146)	Measures the frequency and the cycle period (cycle time) in [µs] at the indicated channel
INC_ENCODER (→ p. 148)	Up/down counter function for the evaluation of encoders
INC_ENCODER_HR (→ p. 150)	Up/down counter function for the high resolution evaluation of encoders
PERIOD (→ p. 152)	Measures the frequency and the cycle period (cycle time) in [µs] at the indicated channel
PERIOD_RATIO (→ p. 154)	Measures the frequency and the cycle period (cycle time) in [µs] during the indicated periods at the indicated channel. In addition, the mark-to-space ratio is indicated in [‰].

Digital filters are not available for all fast counter and pulse inputs.

Use as binary inputs

3804

The permissible high input frequencies also ensure the detection of faulty signals, e.g. bouncing contacts of mechanical switches.

► If required, suppress the faulty signals in the application program!

© ifm electronic gmbh www.ifm.com

4.4.2 Configure outputs

Contents

3976

Valid operating modes → chapter **Possible operating modes inputs/outputs** (→ p. 243)

Configure the software filters of the outputs

6882

Via the system variables Qxx_FILTER a software filter which filters the measured current values can be configured.

- In case of a step response the filter behaves like a conventional low-pass filter, the limit frequency is set by the value entered in the system variable.
- During current measuring the filter setting affects the diagnosis time.

Table: Limit frequency software low-pass filter for the current measurement on the output

Qxx_FILTER	Filter frequency [Hz]	Signal rise time	Remarks
0	Filter deactivated		
1	580	0.6 ms	
2	220	1.6 ms	
3	102	3.5 ms	
4	51	7 ms	Recommended, default setting
5	25	14 ms	
6	12	28 ms	
7	6	56 ms	
8	3	112 ms	
≥ 9	51	7 ms	→ Default setting

12969

After changing the filter setting, the value of this input or output is not output correctly at once. Only after the signal rise time (→ table) will the value be correct again.

The signal rise time is the time taken by a signal at the output of the filter to rise from 10 % to 90 % of the final value if an input step is applied. The signal fall time is the time taken by a signal to decrease from 90 % to 10 %.

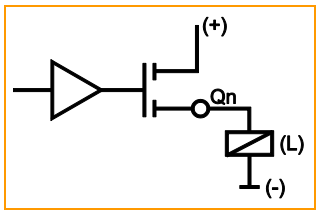
© ifm electronic gmbh

Binary outputs: configuration and diagnosis

15754

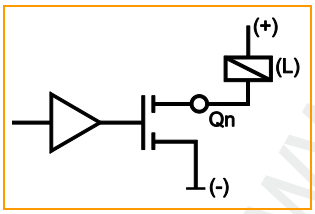
The following operating modes are possible for the device outputs (→ data sheet):
- binary output, plus switching (BH) with/without diagnostic function
- binary output minus switched (BL) without diagnostic function

15450



Qn = pin output n
(L) = load

Basic circuit of output plus switching (BH)
for positive output signal



Qn = pin output n
(L) = load

Basic circuit of output minus switching (BL)
for negative output signal

13975

⚠ WARNING

Dangerous restart possible!
Risk of personal injury! Risk of material damage to the machine/plant!

If in case of a fault an output is switched off via the hardware, the logic state generated by the application program is not changed.

► Remedy:
- Reset the output logic in the application program!
- Remove the fault!
- Reset the outputs depending on the situation.

Binary outputs: configuration

15868

► Configuration of each output is made via the application program:
→ function block **SET_OUTPUT_MODE** (→ p. 159) > input MODE
Permissible values → chapter **Possible operating modes inputs/outputs** (→ p. 243)

Binary outputs: Diagnostics

15762

If the diagnostics is to be used, it needs to be activated additionally.
Prerequisite: configuration of the binary output as plus switching

► Use output as binary output with diagnostics (→ data sheet):
→ FB SET_OUTPUT_MODE > input DIAGNOSTICS = TRUE

> The diagnosis messages appear in the system flag ERRORCODE:

ERRORCODE	DWORD	Last error written to internal error list The list contains all error codes that occurred.

More messages are indicated via various error flags.
→ chapter **System flags** (→ p. 231)

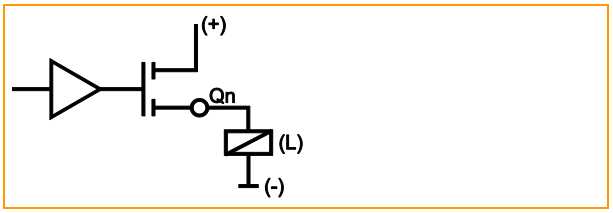
© ifm electronic GmbH

PWM outputs

14717

The following operating modes are possible for the device outputs (→ data sheet):
- PWM output, plus switching (BH) without diagnostic function
- PWM output pair H-bridge without diagnostic function

15451



Qn = pin output n
(L) = load

Basic circuit of output plus switching (BH)
for positive output signal

16253

⚠ WARNING

Property damage or bodily injury possible due to malfunctions!

For outputs in PWM mode:
- There are no diagnostic functions

9980

NOTE

PWM outputs must NOT be operated in parallel, e.g. in order to increase the max. output current. The outputs do not operate synchronously.

Otherwise the entire load current could flow through only one output. The current measurement would no longer function.

- PWM outputs can be operated with and without current control function.
  Current-controlled PWM outputs are mainly used for triggering proportional hydraulic functions.
  The medium current across a PWM signal can only be correctly determined via the FB OUTPUT_CURRENT if the current flowing in the switched-on state is within the measuring range.

Availability of PWM

15885

Outputs with PWM function → Data sheet

FBs for PWM functions

14710

The following function blocks are available for the PWM function:

OUTPUT_BRIDGE (→ p. 163)	H-bridge on a PWM channel pair
OUTPUT_CURRENT (→ p. 167)	Measures the current (average via dither period) on an output channel
OUTPUT_CURRENT_CONTROL (→ p. 168)	Current controller for a PWMi output channel
PWM1000 (→ p. 171)	Initialises and configures a PWM-capable output channel the mark-to-space ratio can be indicated in steps of 1 ‰

Possible special functions of the outputs:
→ chapter **Function elements: hydraulic control** (→ p. 173)
→ chapter **Function elements: controllers** (→ p. 188)

Current control with PWM (= PWMi)

13829

Current measurement of the coil current can be carried out via the current measurement channels integrated in the controller. This allows for example that the current can be re-adjusted if the coil heats up. Thus the hydraulic relationships in the system remain the same.

In principle, the current-controlled outputs are protected against short circuit.

© ifm electronic gmbh www.ifm.com

4.5 Variables

Contents

3130

In this chapter you will learn more about how to handle variables.

14486

The device supports the following types of variables:

Variable	Declaration place	Validity area	Memory behaviour
local	in the declaration part of the function element (POU)	Only valid in the function element (POU) where it was configured.	volatile
local retain			nonvolatile
global	In [Resources] > [Global Variables] > [Globale_Variables]:	Valid in all function elements of this CODESYS project.	volatile
global retain			nonvolatile
Network	In [Resources] > [Global Variables] > declaration list	Values are available to all CODESYS projects in the whole network if the variable is contained in its declaration lists.	volatile
Network retain			nonvolatile

→ CODESYS programming manual

© ifm electronic gmbh

4.5.1 Retain variables

15454

Variables declared as `RETAIN` generate remanent data. Retain variables keep the values saved in them when the device is switched on/off or when an online reset is made.

> The contents of the retain variables are lost if the device is in the STOP state during power-off!

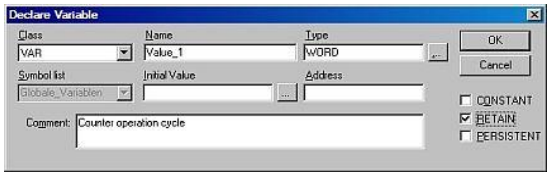
14166

Typical applications for retain variables are for example:
- operating hours which are counted up and retained while the machine is in operation,
- position values of incremental encoders,
- preset values entered in the monitor,
- machine parameters,

i.e. all variables whose values must not get lost when the device is switched off.

All variable types, also complex structures (e.g. timers), can be declared as retain.

► To do so, activate the control field [RETAIN] in the variable declaration (→ window).



4.5.2 Network variables

9856

Global network variables are used for data exchange between controllers in the network. The values of global network variables are available to all CODESYS projects in the whole network if the variables are contained in their declaration lists.

► Integrate the following library/libraries into the CODESYS project:
- `3S_CANopenNetVar.lib`

© ifm electronic gmbh www.ifm.com

5 ifm function elements

Contents

13586

All CODESYS function elements (FBs, PRGs, FUNs) are stored in libraries. Below you will find a list of all the ifm libraries you can use with this device.

This is followed by a description of the function elements, sorted by topic.

5.1 ifm libraries for the device CR0033

Contents

14235

© ifm electronic gmbh

5.1.1 Library ifm_CR0033_V02yyzz.LIB

19426

This is the device library. This ifm library contains the following function blocks:

Function element	Short description
CANx (→ p. 77)	Initialises CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_BAUDRATE (→ p. 78)	Sets the transmission rate for the bus participant on CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_BUSLOAD (→ p. 79)	Determines the current bus load on CAN interface x and counts the occurred error frames x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_DOWNLOADID (→ p. 80)	Sets the download identifier for CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_ERRORHANDLER (→ p. 81)	Executes a "manual" bus recovery on CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_RECEIVE (→ p. 82)	CAN interface x: Configures a data receive object and reads out the receive buffer of the data object x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_SDO_READ (→ p. 104)	CAN interface x: Reads the SDO with the indicated indices from the node x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_SDO_WRITE (→ p. 106)	CAN interface x: writes the SDO with the indicated indices to the node x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_TRANSMIT (→ p. 84)	Transfers a CAN data object (message) to the CAN interface x for transmission at each call x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CHECK_DATA (→ p. 215)	Generates a checksum (CRC) for a configurable memory area and checks the data of the memory area for undesired changes
DELAY (→ p. 189)	Delays the output of the input value by the time T (dead-time element)
ERROR_REPORT (→ p. 222)	Reports an application-specific error to the system
ERROR_RESET (→ p. 223)	Resets upcoming error messages
FAST_COUNT (→ p. 142)	Counter block for fast input pulses
FLASHREAD (→ p. 207)	transfers different data types directly from the flash memory to the RAM
FLASHWRITE (→ p. 208)	writes different data types directly into the flash memory
FRAMREAD (→ p. 210)	transfers different data types directly from the FRAM memory to the RAM FRAM indicates here all kinds of non-volatile and fast memories.
FRAMWRITE (→ p. 211)	writes different data types directly into the FRAM memory FRAM indicates here all kinds of non-volatile and fast memories.
FREQUENCY (→ p. 144)	Measures the frequency of the signal arriving at the selected channel
FREQUENCY_PERIOD (→ p. 146)	Measures the frequency and the cycle period (cycle time) in [µs] at the indicated channel
GET_IDENTITY (→ p. 217)	Reads the specific identifications stored in the device: • hardware name and hardware version of the device • serial number of the device • name of the runtime system in the device • version and revision no. of the runtime system in the device • name of the application (has previously been saved by means of **SET_IDENTITY** (→ p. 219))
INC_ENCODER (→ p. 148)	Up/down counter function for the evaluation of encoders
INC_ENCODER_HR (→ p. 150)	Up/down counter function for the high resolution evaluation of encoders
INPUT_ANALOG (→ p. 131)	analogue input channel: alternatively measurement of ... • current • voltage
MEMCPY (→ p. 212)	Writes and reads different data types directly in the memory
MEMORY_RETAIN_PARAM (→ p. 205)	Determines the remanent data behaviour for various events
MEMSET (→ p. 213)	Writes in a specified data area

© ifm electronic gmbh www.ifm.com

Function element	Short description
NORM (→ p. 137)	Normalises a value [WORD] within defined limits to a value with new limits
NORM_DINT (→ p. 139)	Normalises a value [DINT] within defined limits to a value with new limits
NORM_REAL (→ p. 140)	Normalises a value [REAL] within defined limits to a value with new limits
OUTPUT_BRIDGE (→ p. 163)	H-bridge on a PWM channel pair
OUTPUT_CURRENT (→ p. 167)	Measures the current (average via dither period) on an output channel
OUTPUT_CURRENT_CONTROL (→ p. 168)	Current controller for a PWMi output channel
PACK_ERRORCODE (→ p. 225)	Helps to build an ERRORCODE from the bytes for: • error class • application-specific error • error source • error cause
PERIOD (→ p. 152)	Measures the frequency and the cycle period (cycle time) in [µs] at the indicated channel
PERIOD_RATIO (→ p. 154)	Measures the frequency and the cycle period (cycle time) in [µs] during the indicated periods at the indicated channel. In addition, the mark-to-space ratio is indicated in [‰].
PHASE (→ p. 156)	Reads a pair of channels with fast inputs and compares the phase position of the signals
PID1 (→ p. 190)	PID controller
PID2 (→ p. 192)	PID controller
PT1 (→ p. 194)	Controlled system with first-order delay
PWM1000 (→ p. 171)	Initialises and configures a PWM-capable output channel the mark-to-space ratio can be indicated in steps of 1 ‰
SERIAL_PENDING (→ p. 121)	Determines the number of data bytes stored in the serial receive buffer
SERIAL_RX (→ p. 122)	Reads a received data byte from the serial receive buffer at each call
SERIAL_SETUP (→ p. 123)	Initialises the serial RS232 interface
SERIAL_TX (→ p. 124)	Transmits one data byte via the serial RS232 interface
SET_DEBUG (→ p. 218)	organises the DEBUG mode or the monitoring mode (depending on the TEST input)
SET_IDENTITY (→ p. 219)	Sets an application-specific program identification
SET_INPUT_MODE (→ p. 134)	Assigns an operating mode to an input channel
SET_INTERRUPT_I (→ p. 126)	Conditional execution of a program part after an interrupt request via a defined input channel
SET_INTERRUPT_XMS (→ p. 128)	Conditional execution of a program part at an interval of x milliseconds
SET_OUTPUT_MODE (→ p. 159)	Sets the operating mode of the selected output channel
SET_PASSWORD (→ p. 220)	Sets a user password for access control to program and memory upload
SHOW_ERROR_LIST (→ p. 226)	Reads the present error code
SOFTRESET (→ p. 196)	leads to a complete reboot of the device
TEMPERATURE (→ p. 201)	Reads the current temperature in the device
TIMER_READ (→ p. 198)	Reads out the current system time in [ms] Max. value = 49d 17h 2min 47s 295ms
TIMER_READ_US (→ p. 199)	Reads out the current system time in [µs] Max. value = 1h 11min 34s 967ms 295µs
UNPACK_ERRORCODE (→ p. 227)	Helps to unpack an ERRORCODE into the bytes for: • error class • application-specific error • error source • error cause

5.1.2 Library ifm_CR0033_CANopenxMaster_Vxxyyzz.LIB

13707

x = 1...4 = number of the CAN interface

This library contains function blocks for operation of the device as a CANopen master.

This ifm library contains the following function blocks:

Function element	Short description
CANx_MASTER_EMCY_HANDLER (→ p. 86)	Handles the device-specific error status of the CANopen master on CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_MASTER_SEND_EMERGENCY (→ p. 87)	Sends application-specific error status of the CANopen master on CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_MASTER_STATUS (→ p. 89)	Status indication on CAN interface x of the device used as CANopen master x = 1...n = number of the CAN interface (depending on the device, → data sheet)

5.1.3 Library ifm_CR0033_CANopenxSlave_Vxxyyzz.LIB

13709

x = 1...4 = number of the CAN interface

This library contains function blocks for operation of the device as a CANopen slave.

This ifm library contains the following function blocks:

Function element	Short description
CANx_SLAVE_EMCY_HANDLER (→ p. 96)	Handles the device-specific error status of the CANopen slave on CAN interface x: • error register (index 0x1001) and • error field (index 0x1003) of the CANopen object directory x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_SLAVE_NODEID (→ p. 97)	Enables setting of the node ID of a CANopen slave on CAN interface x at runtime of the application program x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_SLAVE_SEND_EMERGENCY (→ p. 98)	Sends application-specific error status of the CANopen slave on CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_SLAVE_SET_PREOP (→ p. 100)	Switches the operating mode of this CANopen slave from "OPERATIONAL" to "OPERATIONAL" on CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_SLAVE_STATUS (→ p. 101)	Shows the status of the device used as CANopen slave on CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)

© ifm electronic gmbh

5.1.4 Library ifm_CR0033_J1939_Vxxyyzz.LIB

13711

This library contains function blocks for engine control.

This ifm library contains the following function blocks:

Function element	Short description
J1939_x (→ p. 109)	CAN interface x: protocol handler for the communication profile SAE J1939 x = 1...n = number of the CAN interface (depending on the device, → data sheet)
J1939_x_GLOBAL_REQUEST (→ p. 110)	CAN interface x: handles global requesting and receipt of data from the J1939 network participants x = 1...n = number of the CAN interface (depending on the device, → data sheet)
J1939_x_RECEIVE (→ p. 112)	CAN interface x: Receives a single message or a message block x = 1...n = number of the CAN interface (depending on the device, → data sheet)
J1939_x_RESPONSE (→ p. 114)	CAN interface x: handles the automatic response to a request message x = 1...n = number of the CAN interface (depending on the device, → data sheet)
J1939_x_SPECIFIC_REQUEST (→ p. 116)	CAN interface x: automatic requesting of individual messages from a specific J1939 network participant x = 1...n = number of the CAN interface (depending on the device, → data sheet)
J1939_x_TRANSMIT (→ p. 118)	CAN interface x: sends individual messages or message blocks x = 1...n = number of the CAN interface (depending on the device, → data sheet)

5.1.5 Library ifm_hydraulic_32bit_Vxxyyzz.LIB

13729

This library contains function blocks for hydraulic controls.

This ifm library contains the following function blocks:

Function element	Short description
CONTROL_OCC (→ p. 174)	OCC = **O**utput **C**urrent **C**ontrol Scales the input value [WORD] to an indicated current range
JOYSTICK_0 (→ p. 176)	Scales signals [INT] from a joystick to clearly defined characteristic curves, standardised to 0... 1000
JOYSTICK_1 (→ p. 179)	Scales signals [INT] from a joystick D standardised to 0... 1000
JOYSTICK_2 (→ p. 183)	Scales signals [INT] from a joystick to a configurable characteristic curve; free selection of the standardisation
NORM_HYDRAULIC (→ p. 186)	Normalises a value [DINT] within defined limits to a value with new limits

© ifm electronic gmbh

5.2 ifm function elements for the device CR0033

Contents

13988
3826

Here you will find the description of the ifm function elements suitable for this device, sorted by topic.

5.2.1 Function elements: CAN layer 2

Contents

13754

Here, the CAN function blocks (layer 2) for use in the application program are described.

© ifm electronic gmbh

CANx

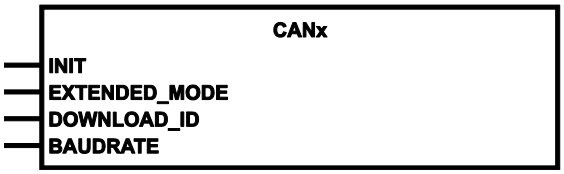
2159

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2162

CANx initialises the xth CAN interface
x = 1...n = number of the CAN interface (depending on the device, → data sheet).

The download ID must be different for every interface.

The baud rates of the individual CANx can be set to different values.

► The input INIT is only set for one cycle during reboot or restart of the interface!

A change of the download ID and/or baud rate only becomes valid after power off/on.

If the unit is not executed, the interface works with 11-bit identifiers.

Parameters of the inputs

2163

Parameter	Data type	Description
INIT	BOOL	TRUE (in the 1st cycle): Function block is initialised FALSE: during further processing of the program
EXTENDED_MODE	BOOL := FALSE	TRUE: identifier of the CAN interface operates with 29 bits FALSE: identifier of the CAN interface operates with 11 bits
DOWNLOAD_ID	BYTE	Download ID of CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet) valid = 1...127 preset = 127 - (x-1)
BAUDRATE	WORD := 125	Baud rate [kbits/s] valid = 20, 50, 100, 125, 250, 500, 1000

© ifm electronic gmbh www.ifm.com

CANx_BAUDRATE

11834

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

11839

CANx_BAUDRATE sets the transmission rate for the bus participant.

The function block is used to set the transmission rate for the device. To do so, the corresponding value in Kbits/s is entered at the input BAUDRATE.

The new value will become effective on RESET (voltage OFF/ON or soft reset).

Parameters of the inputs

655

Parameter	Data type	Description
ENABLE	BOOL	TRUE (in the 1st cycle): Adopt and activate parameters else: this function is not executed
BAUDRATE	WORD := 125	Baud rate [kbits/s] valid = 20, 50, 100, 125, 250, 500, 1000

© ifm electronic gmbh www.ifm.com

CANx_BUSLOAD

2178

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

2180

Determines the current bus load on the CAN bus and counts the occurred error frames.

CANx_BUSLOAD determines the bus load via the number and length of the messages transferred via the CAN bus during the time indicated in PERIOD by taking the current baud rate into account. The value BUSLOAD is updated after the time indicated in PERIOD has elapsed.

If the bit RESET is permanently FALSE, the number of the error frames occurred since the last RESET is indicated.

> **NOTE**
>
> If the communication on the CAN bus is carried out via the CANopen protocol, it is useful to set the value of PERIOD to the duration of the SYNC cycle.
>
> The measurement period is not synchronised with the CANopen SYNC cycle.

Parameters of the inputs

2181

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
INIT	BOOL	TRUE (only for 1 cycle): configuration of the measurement duration PERIOD FALSE: during further processing of the program
RESET	BOOL	TRUE: Set ERRORFRAME to "0" FALSE: function element is not executed
PERIOD	WORD	Time in [ms] to determine the bus load allowed = 20...1 000 ms

Parameters of the outputs

2182

Parameter	Data type	Description
ERRORFRAMES	WORD	Number of error frames occurred on the CAN bus since the last reset
BUSLOAD	BYTE	Current bus load in [%]

© ifm electronic gmbH www.ifm.com

CANx_DOWNLOADID

11841

= CANx Download-ID

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

11846

CANx_DOWNLOADID sets the download identifier for the CAN interface x.

The function block can be used to set the communication identifier for program download and debugging. The new value is entered when the input ENABLE is set to TRUE.

The new value will become effective on RESET (voltage OFF/ON or soft reset).

Parameters of the inputs

649

Parameter	Data type	Description
ENABLE	BOOL	TRUE (in the 1st cycle): Adopt and activate parameters else: this function is not executed
ID	BYTE	Set download ID of CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet) allowed = 1...127 preset = 127 - (x-1)

© ifm electronic gmbh www.ifm.com

CANx_ERRORHANDLER

2174

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

2329
13991

If the automatic bus recover function is to be used (default setting) the function CANx_ERRORHANDLER must **not** be integrated and instanced in the program!

CANx_ERRORHANDLER executes a "manual" bus recovery on the CAN interface x.

► After a recognised CAN bus-off, call the function block for one cycle with BUSOFF_RECOVER = TRUE to make sure that the controller can send and receive on the CAN bus again.

► Then reset the error bit CANx_BUSOFF for this CAN interface in the application program.

> The CAN interface is operative again.

Parameters of the inputs

2177

Parameter	Data type	Description
BUSOFF_RECOVER	BOOL	TRUE (only 1 cycle): > remedy 'bus off' status > reboot of the CAN interfacex FALSE: function element is not executed

© ifm electronic gmbh www.ifm.com

CANx_RECEIVE

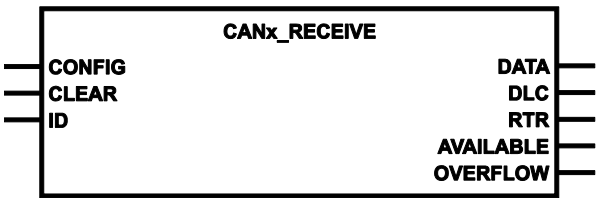
627

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

13338

CANx_RECEIVE configures a data receive object and reads the receive buffer of the data object.

- ► The FB must be called once for each data object during initialisation to inform the CAN controller about the identifiers of the data objects.
- ► In the further program cycle CANx_RECEIVE is called for reading the corresponding receive buffer, also repeatedly in case of long program cycles.
- ► Depending on the CAN interface max. 256 instances are possible for the FB CANx_RECEIVE.
- ► In the Standard Mode all 2048 IDs can be used simultaneously
  In the Extended Mode only 256 (any) IDs can be used simultaneously.
- ► Each ID (Standard or Extended) can be allotted to only one FB instance.
  For multiple use of an ID: the last instance called.
- ► Set in FB CANx if CANx_RECEIVE should receive normal or extended frames.
- > If CANx_RECEIVE is configured for the reception of a normal frame, the frame with this ID will not be transferred to a CANopen Stack (if available).
- > If an ID is set outside the permissible range (depending on the setting in CANx), the function block will not be executed.
- ► Evaluate the output AVAILABLE so that newly received data objects are read from the butter and processed in time.
  Receive buffer: max. 16 software buffers per identifier.
- > Each call of the FB decrements the byte AVAILABLE by 1.
  If AVAILABLE = 0, there is no data in the buffer.
- ► Evaluate the output OVERFLOW to detect an overflow of the data buffer.
  If OVERFLOW = TRUE, at least 1 data object has been lost.

© ifm electronic gmbh www.ifm.com

Parameters of the inputs

2172

Parameter	Data type	Description
CONFIG	BOOL	TRUE (in the 1st cycle): configure data object FALSE: during further processing of the program
CLEAR	BOOL	TRUE: delete receive buffer FALSE: function element is not executed
ID	DWORD	Number of the data object identifier: 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

Parameters of the outputs

19810

Parameter	Data type	Description
DATA	ARRAY [0..7] OF BYTE	received data, (1...8 bytes)
DLC	BYTE	Number of the bytes of the CAN telegram read from the receive buffer allowed: 0...8
RTR	BOOL = FALSE	Received message was a **R**emote **T**ransmission **R**equest (wird hier nicht unterstützt)
AVAILABLE	BYTE	Number of the CAN telegrams received but not yet read from the receive buffer (before the FB is called). Possible values = 0...16 0 = no valid data available
OVERFLOW	BOOL	TRUE: Overflow of the data buffer ⇨ loss of data! FALSE: Data buffer is without data loss

© ifm electronic gmbh www.ifm.com

CANx_TRANSMIT

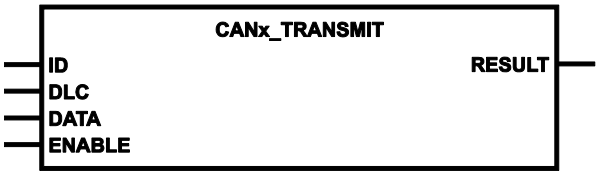
609

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2166

CANx_TRANSMIT transmits a CAN data object (message) to the CAN controller for transmission.

The FB is called for each data object in the program cycle, also repeatedly in case of long program cycles. The programmer must ensure by evaluating the output RESULT that his transmit order was accepted. Simplified it can be said that at 125 kbits/s one transmit order can be executed per 1 ms.

The execution of the FB can be temporarily blocked (ENABLE = FALSE) via the input ENABLE. So, for example a bus overload can be prevented.

ⓘ To put it simply, at 125 kbits/s one transmit order can be executed per 1 ms.

Several data objects with the same or with different ID can be transmitted virtually at the same time if a flag is assigned to each data object and controls the execution of the FB via the ENABLE input.

Transmit buffer: max. 16 software buffers and 1 hardware buffer for all identifiers together.

Parameters of the inputs

2167

Parameter	Data type	Description
ID	DWORD	Number of the data object identifier: 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	Number of bytes received in the DATA array with SRDO allowed: 0...8
DATA	ARRAY [0..7] OF BYTE	data to be sent (1...8 bytes)
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified

Parameters of the outputs

2168

Parameter	Data type	Description
RESULT	BOOL	TRUE (only for 1 cycle): Function block accepted transmit order FALSE: Transmit order was not accepted

© ifm electronic gmbh www.ifm.com

5.2.2 Function elements: CANopen master

Contents

1870

ifm electronic provides a number of FBs for the CANopen master which will be explained below.

© ifm electronic gmbh

CANx_MASTER_EMCY_HANDLER

2006

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_CANopenxMaster_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

2009

CANx_MASTER_EMCY_HANDLER manages the device-specific error status of the master. The FB must be called in the following cases:

- the error status is to be transmitted to the network and
- the error messages of the application are to be stored in the object directory.

The current values from the error register (index 0x1001/01) and error field (index 0x1003/0-5) of the CANopen object directory can be read via the FB.

If application-specific error messages are to be stored in the object directory, CANx_MASTER_EMCY_HANDLER must be called **after** (repeatedly) calling **CANx_MASTER_SEND_EMERGENCY** (→ p. 87).

Parameters of the inputs

2010

Parameter	Data type	Description
CLEAR_ERROR_FIELD	BOOL	FALSE ⇨ TRUE (edge): • transmit content of ERROR_FIELD to function block output • delete content of ERROR_FIELD in object directory else: this function is not executed

Parameters of the outputs

2011

Parameter	Data type	Description
ERROR_REGISTER	BYTE	Shows content of OBV index 0x1001 (error register)
ERROR_FIELD	ARRAY [0..5] OF WORD	Shows the content of the OBV index 0x1003 (error field) ERROR_FIELD[0]: number of stored errors ERROR_FIELD[1...5]: Stored errors, the most recent error is shown on index [1]

© ifm electronic gmbh

CANx_MASTER_SEND_EMERGENCY

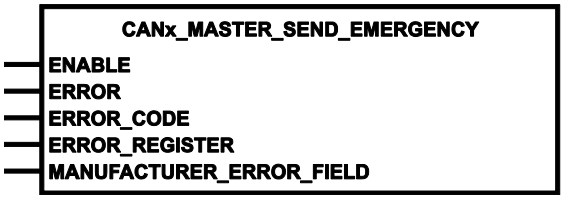
2012

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_CANopenxMaster_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2015

CANx_MASTER_SEND_EMERGENCY transmits application-specific error states. The FB is called if the error status is to be transmitted to other devices in the network.

If application-specific error messages are to be stored in the object directory, **CANx_MASTER_EMCY_HANDLER** (→ p. 86) must be called **after** (repeatedly) calling CANx_MASTER_SEND_EMERGENCY.

Parameters of the inputs

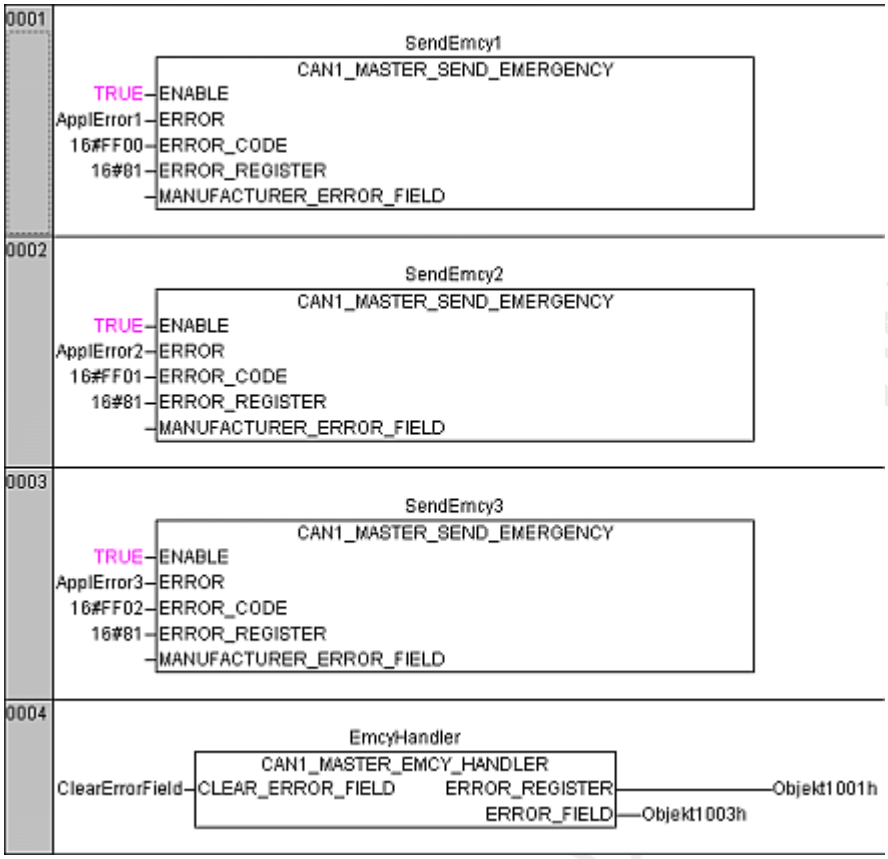
2016

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
ERROR	BOOL	Using this input, the information whether the error associated to the configured error code is currently present is transmitted. FALSE ⇨ TRUE (edge): sends the next error code if input was not TRUE in the last second TRUE ⇨ FALSE (edge) AND the fault is no longer indicated: after a delay of approx. 1 s: > zero error message is sent else: this function is not executed
ERROR_CODE	WORD	The error code provides detailed information about the detected error. The values should be entered according to the CANopen specification.
ERROR_REGISTER	BYTE	ERROR_REGISTER indicates the error type. The value indicated here is linked by a bit-by-bit OR operation with all the other error messages that are currently active. The resulting value is written into the error register (index 1001_{16}/00) and transmitted with the EMCY message. The values should be entered according to the CANopen specification.
MANUFACTURER_ERROR_FIELD	ARRAY [0..4] OF BYTE	Here, up to 5 bytes of application-specific error information can be entered. The format can be freely selected.

© ifm electronic gmbh www.ifm.com

Example: CANx_MASTER_SEND_EMERGENCY

2018



In this example 3 error messages will be generated subsequently:

1. ApplError1, Code = 0xFF00 in the error register 0x81
2. ApplError2, Code = 0xFF01 in the error register 0x81
3. ApplError3, Code = 0xFF02 in the error register 0x81

CAN1_MASTER_EMCY_HANDLER sends the error messages to the error register "Object 0x1001" in the error array "Object 0x1003".

© ifm electronic gmbh www.ifm.com

CANx_MASTER_STATUS

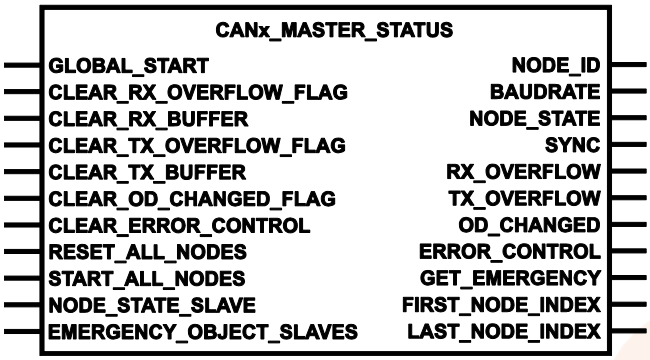
2692

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_CANopenxMaster_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2024

Status indication of the device used with CANopen.

CANx_MASTER_STATUS shows the status of the device used as CANopen master. Further possibilities:

- monitoring the network status
- monitoring the status of the connected slaves
- resetting or starting the slaves in the network.

The FB simplifies the use of the CODESYS CANopen master libraries. We urgently recommend to carry out the evaluation of the network status and of the error messages via this FB.

© ifm electronic gmbh www.ifm.com

Parameters of the inputs

19861

Parameter	Data type	Description
GLOBAL_START	BOOL	TRUE: All connected network participants (slaves) are started simultaneously during network initialisation (⇨ state OPERATIONAL). FALSE: The connected network participants are started one after the other.
CLEAR_RX_OVERFLOW_FLAG	BOOL	FALSE ⇨ TRUE (edge): Clear error flag RX_OVERFLOW else: this function is not executed
CLEAR_RX_BUFFER	BOOL	FALSE ⇨ TRUE (edge): Delete data in the receive buffer else: this function is not executed
CLEAR_TX_OVERFLOW_FLAG	BOOL	FALSE ⇨ TRUE (edge): Clear error flag TX_OVERFLOW else: this function is not executed
CLEAR_TX_BUFFER	BOOL	FALSE ⇨ TRUE (edge): Delete data in the transmit buffer else: this function is not executed
CLEAR_OD_CHANGED_FLAG	BOOL	FALSE ⇨ TRUE (edge): Delete flag OD_CHANGED else: this function is not executed
CLEAR_ERROR_CONTROL	BOOL	FALSE ⇨ TRUE (edge): Delete the guard error list (ERROR_CONTROL) else: this function is not executed
RESET_ALL_NODES	BOOL	FALSE ⇨ TRUE (edge): All connected network participants (slaves) are reset via NMT command else: this function is not executed
START_ALL_NODES	BOOL	FALSE ⇨ TRUE (edge): All connected network participants (slaves) are started via NMT command else: this function is not executed
NODE_STATE_SLAVES	DWORD	Pointer address to a array [0.. MAX_NODEINDEX] of CANx_NODE_STATE The status information of the slaves in the CANopen network is to be written into this array. The behaviour of the slaves can be controlled by means of access to certainly values. MAX_NODEINDEX is a constant, which is calculated by CODESYS during the compiling of the application. Determine the address by means of the operator ADR and assigne it to the POU! Example code → chapter **Example: CANx_MASTER_STATUS** (→ p. 93)
EMERGENCY_OBJECT_SLAVES	DWORD	Pointer address to a array [0.. MAX_NODEINDEX] of CANx_EMERGENCY_MESSAGE Shows the last error messages of all network nodes. Determine the address by means of the operator ADR and assigne it to the POU!

© ifm electronic gmbh

Parameters of the outputs

2696

Parameter	Data type	Description
NODE_ID	BYTE	current node ID of the CANopen slave
BAUDRATE	WORD	current baudrate of the CANopen node in [kBaud]
NODE_STATE	INT	Current status of CANopen master
SYNC	BOOL	SYNC signal of the CANopen master TRUE: In the last cycle a SYNC signal was sent FALSE: In the last cycle no SYNC signal was sent
RX_OVERFLOW	BOOL	TRUE: Error: receive buffer overflow FALSE: no overflow
TX_OVERFLOW	BOOL	TRUE: Error: transmission buffer overflow FALSE: no overflow
OD_CHANGED	BOOL	TRUE: Data in the object directory of the CANopen master have been changed FALSE: no data change
ERROR_CONTROL	ARRAY [0..7] OF BYTE	The array contains the list (max. 8) of missing network nodes (guard or heartbeat error)
GET_EMERGENCY	STRUCT CANx_EMERGENY_MESSAGE	At the output the data for the structure CANx_EMERGENCY_MESSAGE are available. The last received EMCY message in the CANopen network is always displayed. To obtain a list of all occurred errors, the array EmergencyObjectSlavesArray must be evaluated!
FIRST_NODE_INDEX	INT	Section where the node numbers of the nodes (slaves) connected to this CAN bus are located
LAST_NODE_INDEX	INT	

© ifm electronic gmbh www.ifm.com

Internal structure parameters

2698

Here you can see the structures of the arrays used in this function block.
Using Controller CR0032 as an example, the following code fragments show the use of the function block CANx_MASTER_STATUS→ chapter **Example: CANx_MASTER_STATUS** (→ p. 93).

Structure of CANx_EMERGENCY_MESSAGE

13996

The structure is defined by the global variables of the library `ifm_CR0033_CANopenMaster_Vxxyyzz.LIB`.

Parameter	Data type	Description
NODE_ID	BYTE	Node ID of the participant the EMCY has been received from
ERROR_CODE	WORD	Error code indicating which error has occurred. → CANopen specification CiA Draft Standard 301 Version 4
ERROR_REGISTER	BYTE	Value in the error register (index 0x1001/00) of the sending participant
MANUFACTURER_ERROR_FIELD	ARRAY [0..4] OF BYTE	Manufacturer-specific data field in EMCY message

Structure of CANx_NODE_STATE

13997

The structure is defined by the global variables of the library `ifm_CR0033_CANopenMaster_Vxxyyzz.LIB`.

Parameter	Data type	Description
NODE_ID	BYTE	Node ID of the CANopen slave the status information and configuration flags in the structure belong to
NODE_STATE	BYTE	Current state of the CANopen slave seen from the perspective of the CANopen stack of the CANopen master
LAST_STATE	BYTE	The last known state of the CANopen slave 0 = receive bootup message from CANopen slave 4 = CANopen slave in PRE-OPERATIONAL state and is configured via SDO access 5 = CANopen slave in OPERATIONAL state 127 = CANopen slave in PRE-OPERATIONAL state
RESET_NODE	BOOL	Flag for manual reset of CANopen slave (NMT command = `Reset_Node`)
START_NODE	BOOL	Flag for manual start of CANopen slave (NMT command = `start`)
PREOP_NODE	BOOL	Flag to manually set the CANopen slave to the PRE-OPERATIONAL state NMT command = `enter PRE-OPERATIONAL`)
SET_TIMEOUT_STATE	BOOL	Flag used to manually skip initialisation of a CANopen slave if the following applies: • slave does not exist in network • **and** slave is not configured as optional
SET_NODE_STATE	BOOL	Flag for manual initialisation of a CANopen slave When accessing object 0x1000, the slave had identified itself as a device type other than the one indicated in the EDS file incorporated in the CODESYS configuration of the controller.

© ifm electronic gmbh www.ifm.com

Example: CANx_MASTER_STATUS

2031

Slave information

2699

To be able to access the information of the individual CANopen nodes, you must create an array for the corresponding structure. The structures are contained in the library. You can see them under [Data types] in the library manager.

The number of the array elements is determined by the global variable MAX_NODEINDEX which is automatically generated by the CANopen stack. It contains the number of the slaves minus 1 indicated in the network configurator.

> The numbers of the array elements do **not** correspond to the node ID. The identifier can be read from the corresponding structure under NODE_ID.

Program example to CAN1_MASTER_STATUS

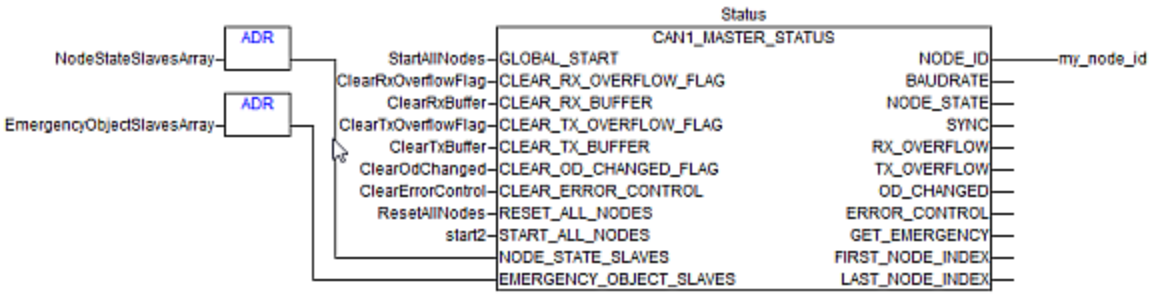
20651

Declaration of the variables:

```
VAR
    Status: CAN1_MASTER_STATUS;


    LedStatus: BOOL:= TRUE;
    StartAllNodes: BOOL:= TRUE;
    ClearRxOverflowFlag: BOOL;
    ClearRxBuffer: BOOL;
    ClearTxOverflowFlag: BOOL;
    ClearTxBuffer: BOOL;
    ClearOdChanged: BOOL;
    ClearErrorControl: BOOL;
    ResetAllNodes: BOOL;
    NodeStateSlavesArray: ARRAY [0..MAX_NODEINDEX] OF CAN1_NODE_STATE;
    EmergencyObjectSlavesArray: ARRAY[0..MAX_NODEINDEX] OF CAN1_EMERGENCY_MESSAGE;
    my_node_id: BYTE;
    my_baudrate: WORD;
    my_node_state: INT;
    Sync: BOOL;
    RxOverflow: BOOL;
    TxOverflow: BOOL;
    OdChanged: BOOL;
    GuardHeartbeatErrorArray: ARRAY[0..7] OF BYTE;
    GetEmergency: CAN1_EMERGENCY_MESSAGE;
    start2: BOOL;
    Ency_handler: CAN1_MASTER_EMCY_HANDLER;
    reset_emcy: BOOL;
END_VAR
```

Example of the program:



Structure node 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
```

Structure 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
```

© ifm electronic gmbh

5.2.3 Function elements: CANopen slave

Contents

1874

ifm electronic provides a number of FBs for the CANopen slave which will be explained below.

© ifm electronic gmbh

CANx_SLAVE_EMCY_HANDLER

2050

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_CANopenxSlave_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

2053

CANx_SLAVE_EMCY_HANDLER handles the device-specific error status of the CANopen slave:
- error register (index 0x1001) and
- error field (index 0x1003) of the CANopen object directory.

► Call the function block in the following cases:
  - the error status is to be transmitted to the CAN network and
  - the error messages of the application program are to be stored in the object directory.

Do you want to store the error messages in the object directory?

► **After** (repeated) handling of **CANx_SLAVE_SEND_EMERGENCY** (→ p. 98) call CANx_SLAVE_EMCY_HANDLER once!

Parameters of the inputs

2054

Parameter	Data type	Description
CLEAR_ERROR_FIELD	BOOL	FALSE ⇨ TRUE (edge): • transmit content of ERROR_FIELD to function block output • delete content of ERROR_FIELD in object directory else: this function is not executed

Parameters of the outputs

2055

Parameter	Data type	Description
ERROR_REGISTER	BYTE	Shows content of OBV index 0x1001 (error register)
ERROR_FIELD	ARRAY [0..5] OF WORD	Shows the content of the OBV index 0x1003 (error field) ERROR_FIELD[0]: number of stored errors ERROR_FIELD[1...5]: Stored errors, the most recent error is shown on index [1]

© ifm electronic gmbh www.ifm.com

CANx_SLAVE_NODEID

2044

= CANx Slave Node-ID

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_CANopenxSlave_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

2049

CANx_SLAVE_NODEID enables the setting of the node ID of a CANopen slave at runtime of the application program.

Normally, the FB is called once during initialisation of the controller, in the first cycle. Afterwards, the input ENABLE is set to FALSE again.

Parameters of the inputs

2047

Parameter	Data type	Description
ENABLE	BOOL	FALSE ⇨ TRUE (edge): Adopt and activate parameters else: this function is not executed
NODEID	BYTE	node ID = ID of the node permissible values = 1...127

© ifm electronic gmbh www.ifm.com

CANx_SLAVE_SEND_EMERGENCY

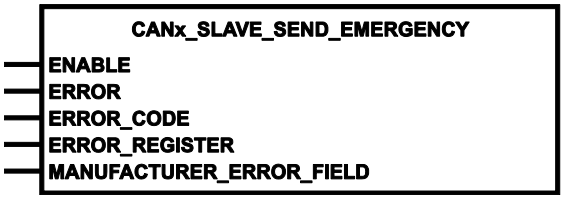
2056

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_CANopenxSlave_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2059

CANx_SLAVE_SEND_EMERGENCY transmits application-specific error states. These are error messages which are to be sent in addition to the device-internal error messages (e.g. short circuit on the output).

► Call the FB if the error status is to be transmitted to other devices in the network.

Parameters of the inputs

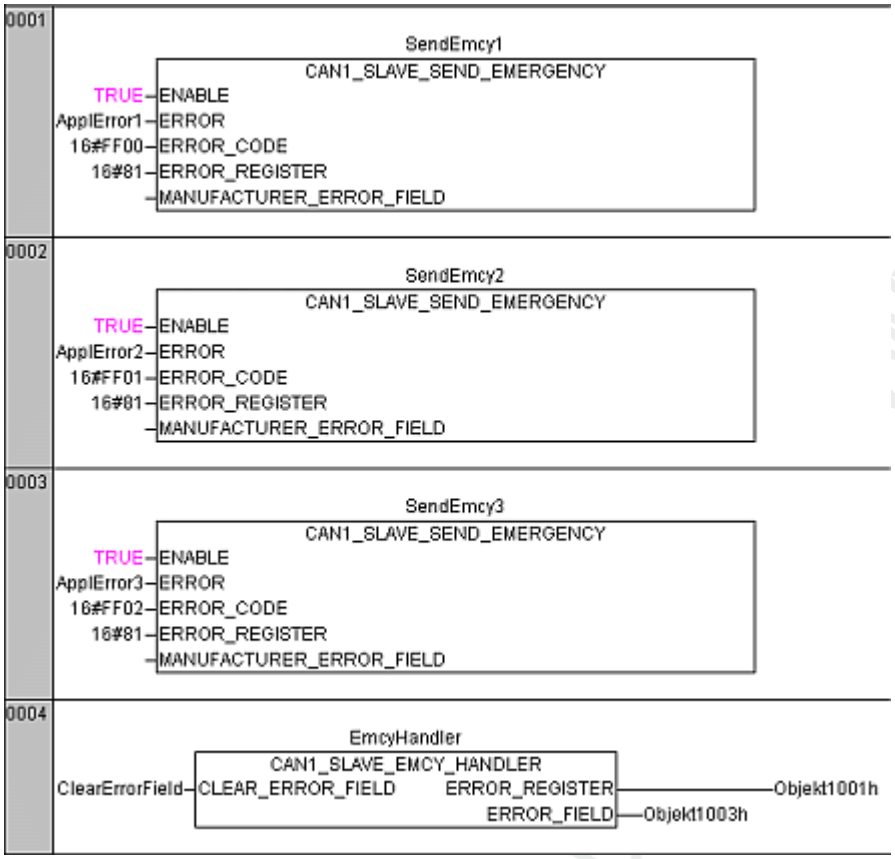
2060

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
ERROR	BOOL	Using this input, the information whether the error associated to the configured error code is currently present is transmitted. FALSE ⇨ TRUE (edge): sends the next error code if input was not TRUE in the last second TRUE ⇨ FALSE (edge) AND the fault is no longer indicated: after a delay of approx. 1 s: > zero error message is sent else: this function is not executed
ERROR_CODE	WORD	The error code provides detailed information about the detected error. The values should be entered according to the CANopen specification.
ERROR_REGISTER	BYTE	ERROR_REGISTER indicates the error type. The value indicated here is linked by a bit-by-bit OR operation with all the other error messages that are currently active. The resulting value is written into the error register (index 1001_{16}/00) and transmitted with the EMCY message. The values should be entered according to the CANopen specification.
MANUFACTURER_ERROR_FIELD	ARRAY [0..4] OF BYTE	Here, up to 5 bytes of application-specific error information can be entered. The format can be freely selected.

© ifm electronic gmbh www.ifm.com

Example: CANx_SLAVE_SEND_EMERGENCY

2062



In this example 3 error messages will be generated subsequently:

1. ApplError1, Code = 0xFF00 in the error register 0x81
2. ApplError2, Code = 0xFF01 in the error register 0x81
3. ApplError3, Code = 0xFF02 in the error register 0x81

CAN1_SLAVE_EMCY_HANDLER sends the error messages to the error register "Object 0x1001" in the error array "Object 0x1003".

© ifm electronic GmbH
www.ifm.com

CANx_SLAVE_SET_PREOP

2700

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_CANopenxSlave_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

2703

CANx_SLAVE_SET_PREOP switches the operating mode of this CANopen slave from "OPERATIONAL" to "PRE-OPERATIONAL".

Normally, in case of a fault the controller switches as follows:
- a FATAL ERROR results in SOFT RESET of the controller
- an ERROR STOP results in a SYSTEM STOP

Under certain conditions it might be necessary that the application sets the operating mode of the device working as a slave to "PRE-OPERATIONAL". This is done via the FB described here.

Parameters of the inputs

2704

Parameter	Data type	Description
ENABLE	BOOL	FALSE ⇨ TRUE (edge): Set slave to PRE-OPERATIONAL else: this function is not executed

© ifm electronic gmbh www.ifm.com

CANx_SLAVE_STATUS

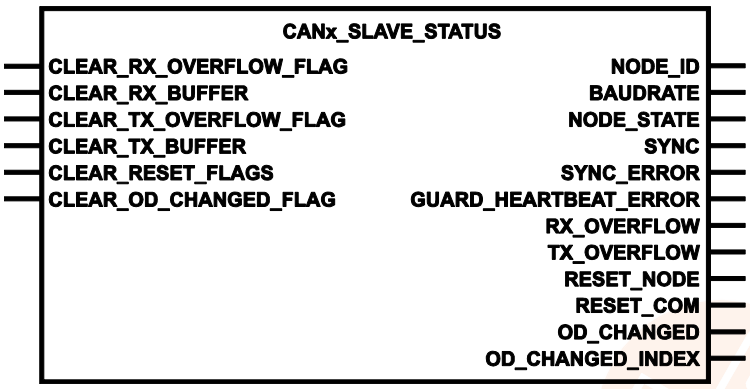
2706

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_CANopenxSlave_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2707

CANx_SLAVE_STATUS shows the status of the device used as CANopen slave.

We urgently recommend carrying out the evaluation of the network status via this function block.

Parameters of the inputs

2708

Parameter	Data type	Description
CLEAR_RX_OVERFLOW_FLAG	BOOL	FALSE ⇨ TRUE (edge): Clear error flag RX_OVERFLOW else: this function is not executed
CLEAR_RX_BUFFER	BOOL	FALSE ⇨ TRUE (edge): Delete data in the receive buffer else: this function is not executed
CLEAR_TX_OVERFLOW_FLAG	BOOL	FALSE ⇨ TRUE (edge): Clear error flag TX_OVERFLOW else: this function is not executed
CLEAR_TX_BUFFER	BOOL	FALSE ⇨ TRUE (edge): Delete data in the transmit buffer else: this function is not executed
CLEAR_RESET_FLAGS	BOOL	FALSE ⇨ TRUE (edge): Clear flag RESET_NODE Clear flag RESET_COM else: this function is not executed
CLEAR_OD_CHANGED_FLAGS	BOOL	FALSE ⇨ TRUE (edge): Clear flag OD_CHANGED Clear flag OD_CHANGED-_INDEX else: this function is not executed

© ifm electronic gmbh www.ifm.com

Parameters of the outputs

2068

Parameter	Data type	Description
NODE_ID	BYTE	current node ID of the CANopen slave
BAUDRATE	WORD	current baudrate of the CANopen node in [kBaud]
NODE_STATE	BYTE	Current status of CANopen slave 0 = Bootup message sent 4 = CANopen slave in PRE-OPERATIONAL state and is configured via SDO access 5 = CANopen slave in OPERATIONAL state 127 = CANopen slave in PRE-OPERATIONAL state
SYNC	BOOL	SYNC signal of the CANopen master TRUE: In the last cycle a SYNC signal was received FALSE: In the last cycle no SYNC signal was received
SYNC_ERROR	BOOL	TRUE: Error: the SYNC signal of the master was not received or received too late (after expiration of ComCyclePeriod) FALSE: no SYNC error
GUARD_HEARTBEAT_ERROR	BOOL	TRUE: Error: the guarding or heartbeat signal of the master was not received or received too late FALSE: no guarding or heartbeat error
RX_OVERFLOW	BOOL	TRUE: Error: receive buffer overflow FALSE: no overflow
TX_OVERFLOW	BOOL	TRUE: Error: transmission buffer overflow FALSE: no overflow
RESET_NODE	BOOL	TRUE: the CANopen stack of the slave was reset by the master FALSE: the CANopen stack of the slave was not reset
RESET_COM	BOOL	TRUE: the communication interface of the CAN stack was reset by the master FALSE: the communication interface was not reset
OD_CHANGED	BOOL	TRUE: Data in the object directory of the CANopen master have been changed FALSE: no data change
OD_CHANGED_INDEX	INT	Index of the object directory entry changed last

© ifm electronic gmbh www.ifm.com

5.2.4 Function elements: CANopen SDOs

Contents

2071

Here you will find ifm function elements for CANopen handling of Service Data Objects (SDOs).

© ifm electronic gmbh

CANx_SDO_READ

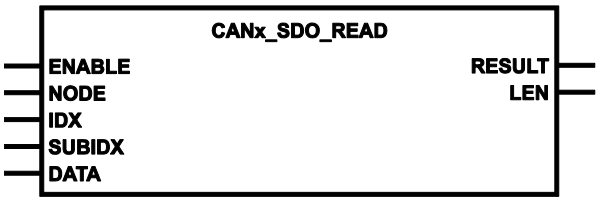
621

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

624

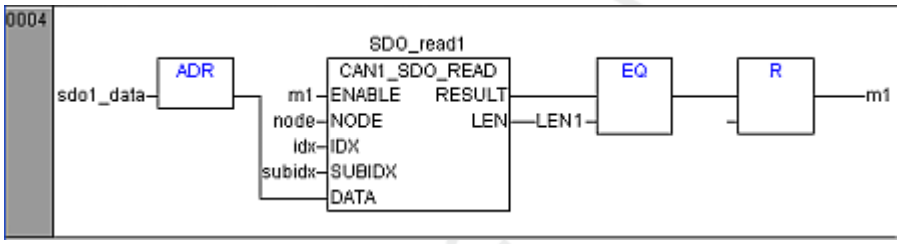
CANx_SDO_READ reads the →**SDO** (→ p. 333) with the indicated indexes from the node.

Prerequisite: Node must be in the mode "PRE-OPERATIONAL" or "OPERATIONAL".

By means of these, the entries in the object directory can be read. So it is possible to selectively read the node parameters.

> Danger of data loss!
> Allocate enough memory space for the requested SDO!
> Otherwise the data following behind will be overwritten.

Example:



© ifm electronic gmbh www.ifm.com

Parameters of the inputs

625

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
NODE	BYTE	ID of the node permissible values = 1...127 = 0x01...0x7F
IDX	WORD	index in object directory
SUBIDX	BYTE	sub-index referred to the index in the object directory
DATA	DWORD	Addresse of the receive data array valid length = 0...255 ! Determine the address by means of the operator ADR and assigne it to the POU!

Parameters of the outputs

626

Parameter	Data type	Description
RESULT	BYTE	feedback of the function block (possible messages → following table)
LEN	WORD	length of the entry in "number of bytes" The value for LEN must not be greater than the size of the receive array. Otherwise any data is overwritten in the application.

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
2	02	function block is active (action not yet completed)
3	03	Error, no data received during monitoring time

© ifm electronic gmbh www.ifm.com

CANx_SDO_WRITE

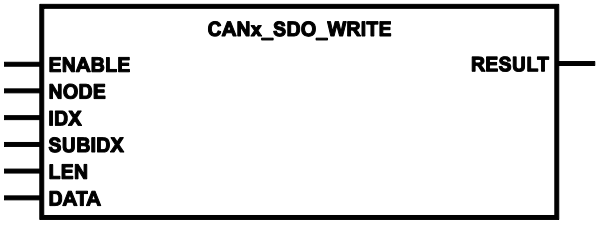
615

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

618

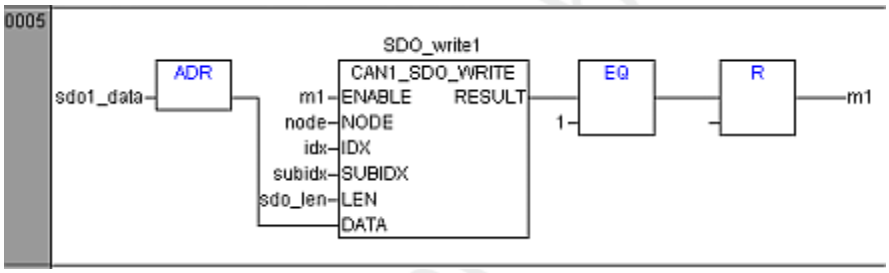
CANx_SDO_WRITE writes the →**SDO** (→ p. 333) with the specified indexes to the node.

Prerequisite: the node must be in the state "PRE-OPERATIONAL" or "OPERATIONAL".

Using this FB, the entries can be written to the object directory. So it is possible to selectively set the node parameters.

> The value for LEN must be lower than the length of the transmit array. Otherwise, random data will be sent.

Example:



© ifm electronic gmbh www.ifm.com

Parameters of the inputs

619

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
NODE	BYTE	ID of the node permissible values = 1...127 = 0x01...0x7F
IDX	WORD	index in object directory
SUBIDX	BYTE	sub-index referred to the index in the object directory
LEN	WORD	length of the entry in "number of bytes" The value for LEN must not be greater than the size of the transmit array. Otherwise any data is sent.
DATA	DWORD	Address of the transmit data array permissible length = 0...255 Determine the address by means of the operator ADR and assigne it to the POU!

Parameters of the outputs

620

Parameter	Data type	Description
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	Value hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
2	02	function block is active (action not yet completed)
3	03	Error, data cannot be transmitted

© ifm electronic gmbh www.ifm.com

5.2.5 Function elements: SAE J1939

Contents

2273

For SAE J1939, ifm electronic provides a number of function elements which will be explained in the following.

© ifm electronic gmbh

J1939_x

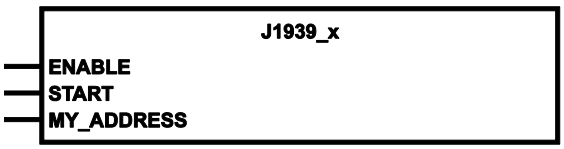
2274

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_J1939_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2276

J1939_x serves as protocol handler for the communication profile SAE J1939.

To handle the communication, the protocol handler must be called in each program cycle. To do so, the input ENABLE is set to TRUE.

Once set, ENABLE must remain TRUE!

The protocol handler is started if the input START is set to TRUE for one cycle.

Using MY_ADDRESS, a device address is assigned to the controller. It must differ from the addresses of the other J1939 bus participants. It can then be read by other bus participants.

Parameters of the inputs

469

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
START	BOOL	TRUE (only for 1 cycle): Start J1939 protocol at CAN interface x FALSE: during further processing of the program
MY_ADDRESS	BYTE	J1939 address of the device

© ifm electronic gmbh www.ifm.com

J1939_x_GLOBAL_REQUEST

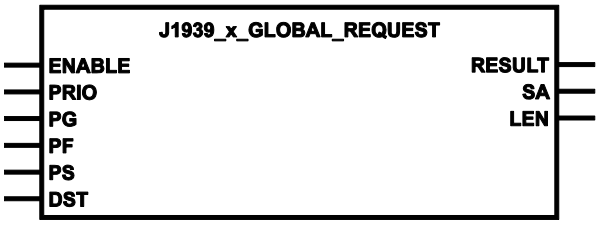
2282

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_J1939_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2301

J1939_x_GLOBAL_REQUEST is responsible for the automatic requesting of individual messages from all (global) active J1939 network participants. To do so, the parameters PG, PF, PS and the address of the array DST in which the received data is stored are assigned to the FB.

Info

PGN = [Page] + [PF] + [PS]
PDU = [PRIO] + [PGN] + [J1939 address] + [data]

13790

NOTICE

Risk of inadmissible overwriting of data!

- ► Create a receiver array with a size of 1 785 bytes.
  This is the maximum size of a J1939 message.
- ► Check the amount of received data:
  the value must not exceed the size of the array created to receive data!

- ► For every requested message use an own instance of the FB!
- ► To the destination address DST applies:
  Determine the address by means of the operator ADR and assigne it to the POU!
- ► In addition, the priority (typically 3, 6 or 7) must be assigned.

- ► Given that the request of data can be handled via several control cycles, this process must be evaluated via the RESULT byte.
- RESULT = 2: the POU is waiting for data of the participants.
- RESULT = 1: data was received by a participant.
  The output LEN indicates how many data bytes have been received.
  Store / evaluate this new data immediately!
  When a new message is received, the data in the memory address DST is overwritten.
- RESULT = 0: no participant on the bus sends a reply within 1.25 seconds.
  The FB returns to the non-active state.
  Only now may ENABLE be set again to FALSE!
- ► For the reception of data from several participants at short intervals:
  call the POU several times in the same PLC cycle and evaluate it at once!

Parameters of the inputs

463

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
PRIO	BYTE	message priority (0…7)
PG	BYTE	Data page Value of defined PGN (Parameter Group Number) allowed = 0...1 (normally = 0)
PF	BYTE	PDU format byte Value of defined PGN (Parameter Group Number) PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Value of defined PGN (Parameter Group Number) GE (Group Extension) = 0...255
DST	DWORD	destination address Determine the address by means of the operator ADR and assigne it to the POU!

Info

PGN = [Page] + [PF] + [PS]
PDU = [PRIO] + [PGN] + [J1939 address] + [data]

Parameters of the outputs

20789

Parameter	Data type	Description
RESULT	BYTE	feedback of the function block (possible messages → following table)
SA	BYTE	J1939 address of the answering device
LEN	WORD	number of received bytes

Possible results for RESULT:

Value dec	Value hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
2	02	function block is active (action not yet completed)

© ifm electronic gmbh

J1939_x_RECEIVE

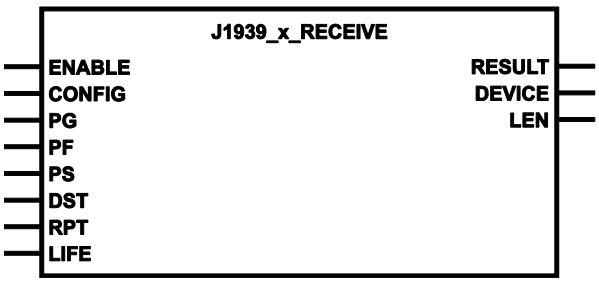
2278

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_J1939_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2288

J1939_x_RECEIVE serves for receiving one individual message or a block of messages.

To do so, the FB must be initialised for one cycle via the input CONFIG. During initialisation, the parameters PG, PF, PS, RPT, LIFE and the memory address of the data array DST are assigned.

Once the following parameters have been configured they can no longer be modified in the running application program: PG, PF, PS, RPT, LIFE, DST.

13790

NOTICE

Risk of inadmissible overwriting of data!

- ► Create a receiver array with a size of 1 785 bytes.
  This is the maximum size of a J1939 message.
- ► Check the amount of received data:
  the value must not exceed the size of the array created to receive data!

- ► To the destination address DST applies:
  Determine the address by means of the operator ADR and assigne it to the POU!

Once RPT has been set it can no longer be modified!

- ► The receipt of data must be evaluated via the RESULT byte. If RESULT = 1 the data can be read from the memory address assigned via DST and can be further processed.
- > When a new message is received, the data in the memory address DST is overwritten.
- > The number of received message bytes is indicated via the output LEN.
- > If RESULT = 3, no valid messages have been received in the indicated time window (LIFE • RPT).

This block must also be used if the messages are requested using the FBs J1939_..._REQUEST.

© ifm electronic gmbh www.ifm.com

Parameters of the inputs

457

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
CONFIG	BOOL	TRUE (in the 1st cycle): configure data object FALSE: during further processing of the program
PG	BYTE	Data page Value of defined PGN (Parameter Group Number) allowed = 0...1 (normally = 0)
PF	BYTE	PDU format byte Value of defined PGN (Parameter Group Number) PDU1 (specific) = 0...239 PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Value of defined PGN (Parameter Group Number) If PF = PDU1 ⇨ PS = DA (Destination Address) (DA = J1939 address of external device) If PF = PDU2 ⇨ PS = GE (Group Extension)
DST	DWORD	destination address Determine the address by means of the operator ADR and assigne it to the POU!
RPT	TIME	Monitoring time Within this time window the messages must be received cyclically. > Otherwise, there will be an error message. RPT = T#0s ⇨ no monitoring Once RPT has been set it can no longer be modified!
LIFE	BYTE	tolerated number of J1939 messages not received

Parameters of the outputs

458

Parameter	Data type	Description
RESULT	BYTE	feedback of the function block (possible messages → following table)
DEVICE	BYTE	J1939 address of the sender
LEN	WORD	number of received bytes

Possible results for RESULT:

Value dec	Value hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
3	03	Error, no data received during monitoring time

© ifm electronic gmbh www.ifm.com

J1939_x_RESPONSE

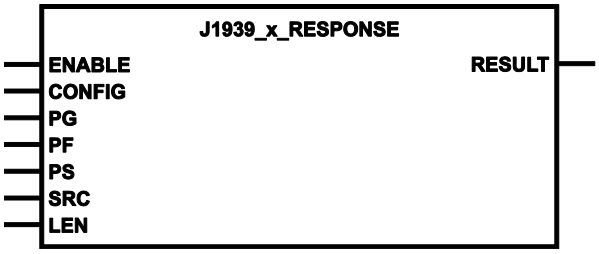
2280

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_J1939_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2299

J1939_x_RESPONSE handles the automatic response to a request message.

This FB is responsible for the automatic sending of messages to "Global Requests" and "Specific Requests". To do so, the FB must be initialised for one cycle via the input CONFIG.

The parameters PG, PF, PS, RPT and the address of the data array SRC are assigned to the FB.

► To the source address SRC applies:
[!] Determine the address by means of the operator ADR and assigne it to the POU!

► In addition, the number of data bytes to be transmitted is assigned.

Parameters of the inputs

451

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
CONFIG	BOOL	TRUE (in the 1st cycle): configure data object FALSE: during further processing of the program
PG	BYTE	Data page Value of defined PGN (Parameter Group Number) allowed = 0...1 (normally = 0)
PF	BYTE	PDU format byte Value of defined PGN (Parameter Group Number) PDU1 (specific) = 0...239 PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Value of defined PGN (Parameter Group Number) If PF = PDU1 ⇨ PS = DA (Destination Address) (DA = J1939 address of external device) If PF = PDU2 ⇨ PS = GE (Group Extension)
SRC	DWORD	start address in source memory [!] Determine the address by means of the operator ADR and assigne it to the POU!
LEN	WORD	number (≥ 1) of the data bytes to be transmitted

© ifm electronic gmbh www.ifm.com

Parameters of the outputs

13993

Parameter	Data type	Description
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	Value hex	Description
0	00	FB is inactive
1	01	Data transfer completed without errors
2	02	function block is active (action not yet completed)
3	03	Error, data cannot be transmitted

© ifm electronic gmbh

J1939_x_SPECIFIC_REQUEST

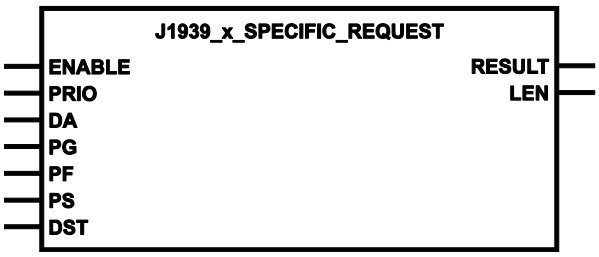
2281

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_J1939_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2300

J1939_x_SPECIFIC_REQUEST is responsible for the automatic requesting of individual messages from a specific J1939 network participant. To do so, the logical device address DA, the parameters PG, PF, PS and the address of the array DST in which the received data is stored are assigned to the FB.

Info

PGN = [Page] + [PF] + [PS]
PDU = [PRIO] + [PGN] + [J1939 address] + [data]

13790

NOTICE

Risk of inadmissible overwriting of data!

- ► Create a receiver array with a size of 1 785 bytes.
  This is the maximum size of a J1939 message.
- ► Check the amount of received data:
  the value must not exceed the size of the array created to receive data!

- ► To the destination address DST applies:
  Determine the address by means of the operator ADR and assigne it to the POU!
- ► In addition, the priority (typically 3, 6 or 7) must be assigned.
- ► Given that the request of data can be handled via several control cycles, this process must be evaluated via the RESULT byte. All data has been received if RESULT = 1.
- > The output LEN indicates how many data bytes have been received.

© ifm electronic gmbh

Parameters of the inputs

445

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
PRIO	BYTE	message priority (0…7)
DA	BYTE	J1939 address of the requested device
PG	BYTE	Data page Value of defined PGN (Parameter Group Number) allowed = 0...1 (normally = 0)
PF	BYTE	PDU format byte Value of defined PGN (Parameter Group Number) PDU1 (specific) = 0...239 PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Value of defined PGN (Parameter Group Number) If PF = PDU1 ⇨ PS = DA (Destination Address) (DA = J1939 address of external device) If PF = PDU2 ⇨ PS = GE (Group Extension)
DST	DWORD	destination address Determine the address by means of the operator ADR and assigne it to the POU!

Info

PGN = [Page] + [PF] + [PS]
PDU = [PRIO] + [PGN] + [J1939 address] + [data]

Parameters of the outputs

446

Parameter	Data type	Description
RESULT	BYTE	feedback of the function block (possible messages → following table)
LEN	WORD	number of received bytes

Possible results for RESULT:

Value dec	Value hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
2	02	function block is active (action not yet completed)
3	03	Error

© ifm electronic gmbh www.ifm.com

J1939_x_TRANSMIT

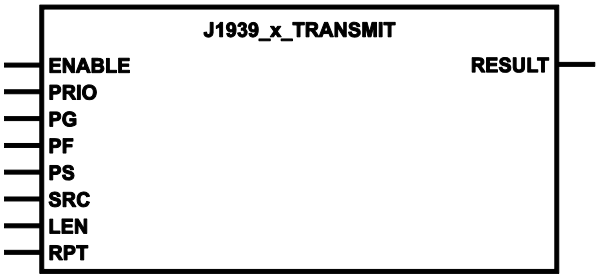
279

x = 1...n = number of the CAN interface (depending on the device, → data sheet)

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_J1939_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2298

J1939_x_TRANSMIT is responsible for transmitting individual messages or blocks of messages. To do so, the parameters PG, PF, PS, RPT and the address of the data array SRC are assigned to the FB.

Info

PGN = [Page] + [PF] + [PS]
PDU = [PRIO] + [PGN] + [J1939 address] + [data]

- ► To the source address SRC applies:
  Determine the address by means of the operator ADR and assigne it to the POU!
- ► In addition, the number of data bytes to be transmitted and the priority (typically 3, 6 or 7) must be assigned.
- ► Given that the transmission of data is processed via several control cycles, the process must be evaluated via the RESULT byte. All data has been transmitted if RESULT = 1.

If more than 8 bytes are to be sent, a "multi package transfer" is carried out.

© ifm electronic gmbh www.ifm.com

Parameters of the inputs

439

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
PRIO	BYTE	message priority (0…7)
PG	BYTE	Data page Value of defined PGN (Parameter Group Number) allowed = 0...1 (normally = 0)
PF	BYTE	PDU format byte Value of defined PGN (Parameter Group Number) PDU1 (specific) = 0...239 PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Value of defined PGN (Parameter Group Number) If PF = PDU1 ⇨ PS = DA (Destination Address) (DA = J1939 address of external device) If PF = PDU2 ⇨ PS = GE (Group Extension)
SRC	DWORD	start address in source memory Determine the address by means of the operator ADR and assigne it to the POU!
LEN	WORD	number of data bytes to be transmitted allowed = 1...1 785 = 0x0001...0x06F9
RPT	TIME	Repeat time during which the data messages are to be transmitted cyclically RPT = T#0s ⇨ sent only once

Info

PGN = [Page] + [PF] + [PS]
PDU = [PRIO] + [PGN] + [J1939 address] + [data]

Parameters of the outputs

440

Parameter	Data type	Description
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	Value hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
2	02	function block is active (action not yet completed)
3	03	Error, data cannot be transmitted

© ifm electronic gmbh www.ifm.com

5.2.6 Function elements: serial interface

Contents

13011
12998

NOTE

The serial interface is not available to the user by default, because it is used for program download and debugging.

The interface can be freely used if the user sets the system flag bit SERIAL_MODE=TRUE. Debugging of the application program is then only possible via any of the CAN interfaces.

The function blocks listed below allow you to use the serial interface in the application program.

© ifm electronic gmbh

SERIAL_PENDING

314

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

12994

SERIAL_PENDING determines the number of data bytes stored in the serial receive buffer.

In contrast to SERIAL_RX the content of the buffer remains unchanged after this function has been called.

The SERIAL FBs form the basis for the creation of an application-specific protocol for the serial interface.
To do so, set the system flag bit SERIAL_MODE=TRUE!

12998

NOTE

The serial interface is not available to the user by default, because it is used for program download and debugging.

The interface can be freely used if the user sets the system flag bit SERIAL_MODE=TRUE. Debugging of the application program is then only possible via any of the CAN interfaces.

Parameters of the outputs

12996

Parameter	Data type	Description
NUMBER	WORD	Number of data bytes received (1...1 000)

© ifm electronic gmbh www.ifm.com

SERIAL_RX

308

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

12997

SERIAL_RX reads a received data byte from the serial receive buffer at each call.

If more than 1000 data bytes are received, the buffer overflows and data is lost. This is indicated by the bit OVERFLOW.

If 7-bit data transmission is used, the 8th bit contains the parity and must be suppressed by the user if necessary.

The SERIAL FBs form the basis for the creation of an application-specific protocol for the serial interface.
To do so, set the system flag bit SERIAL_MODE=TRUE!

12998

NOTE

The serial interface is not available to the user by default, because it is used for program download and debugging.

The interface can be freely used if the user sets the system flag bit SERIAL_MODE=TRUE. Debugging of the application program is then only possible via any of the CAN interfaces.

Parameters of the inputs

312

Parameter	Data type	Description
CLEAR	BOOL	TRUE: delete receive buffer FALSE: function element is not executed

Parameters of the outputs

12931

Parameter	Data type	Description
Rx	BYTE	Byte data received from the receive buffer
AVAILABLE	WORD	Number of received bytes available in the receive buffer BEFORE the call of the function block: 0 = no data received 1...1 000 = number of bytes in the receive buffer
OVERFLOW	BOOL	TRUE: Overflow of the data buffer ⇨ loss of data! FALSE: Data buffer is without data loss

© ifm electronic gmbh www.ifm.com

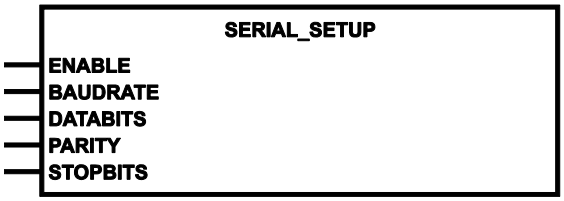
SERIAL_SETUP

302

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

13000

SERIAL_SETUP initialises the serial RS232 interface.

The function block does not necessarily need to be executed in order to be able to use the serial interface. Without function block call the last preset value applies.

Using ENABLE=TRUE for one cycle, the function block sets the serial interface to the indicated parameters. The changes made with the help of the function block are saved non-volatily.

12998

NOTE

The serial interface is not available to the user by default, because it is used for program download and debugging.

The interface can be freely used if the user sets the system flag bit SERIAL_MODE=TRUE. Debugging of the application program is then only possible via any of the CAN interfaces.

Parameters of the inputs

13002

Parameter	Data type	Description
ENABLE	BOOL	TRUE (only for 1 cycle): Initialise interface FALSE: during further processing of the program
BAUD RATE	DWORD	Baud rate permissible values → data sheet preset value → data sheet
DATABITS	BYTE := 8	Number of data bits allowed = 7 or 8
PARITY	BYTE := 0	Parity allowed: 0=none, 1=even, 2=odd With parameter setting DATABITS = 7 and PARITY = 0: function block operates with PARITY = 1
STOPBITS	BYTE := 1	Number of stop bits allowed = 1 or 2

© ifm electronic gmbh

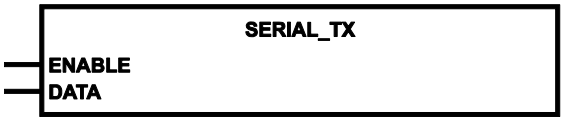
SERIAL_TX

296

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

13003

SERIAL_TX transmits one data byte via the serial RS232 interface.

The FiFo transmission memory contains 1 000 bytes.

Using the input ENABLE the transmission can be enabled or disabled.

The SERIAL FBs form the basis for the creation of an application-specific protocol for the serial interface.
To do so, set the system flag bit SERIAL_MODE=TRUE!

12998

NOTE

The serial interface is not available to the user by default, because it is used for program download and debugging.

The interface can be freely used if the user sets the system flag bit SERIAL_MODE=TRUE. Debugging of the application program is then only possible via any of the CAN interfaces.

Parameters of the inputs

300

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
DATA	BYTE	value to be transmitted

© ifm electronic gmbh www.ifm.com

5.2.7 Function elements: Optimising the PLC cycle via processing interrupts

Contents

20965
8609

Here we show you functions to optimise the PLC cycle.

1599

The PLC cyclically processes the stored application program in its full length. The cycle time can vary due to program branchings which depend e.g. on external events (= conditional jumps). This can have negative effects on certain functions.

By means of systematic interrupts of the cyclic program it is possible to call time-critical processes independently of the cycle in fixed time periods or in case of certain events.

Since interrupt functions are principally not permitted for SafetyControllers, they are thus not available.

© ifm electronic gmbh

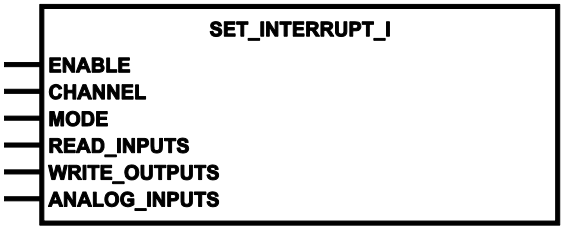
SET_INTERRUPT_I

2381

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

19234
11573

SET_INTERRUPT_I handles the execution of a program part by an interrupt request via an input channel.

In the conventional PLC the cycle time is decisive for real-time monitoring. So the PLC is at a disadvantage as compared to customer-specific controllers. Even a "real-time operating system" does not change this fact when the whole application program runs in one single block which cannot be changed.

A possible solution would be to keep the cycle time as short as possible. This often leads to splitting the application up to several control cycles. This, however, makes programming complex and difficult.

Another possibility is to call a certain program part only upon request by an input pulse independently of the control cycle:

The time-critical part of the application is integrated by the user in a block of the type PROGRAM (PRG). This block is declared as the interrupt routine by calling SET_INTERRUPT_I once (during initialisation). As a consequence, this program block will always be executed if an edge is detected on the input CHANNEL. If inputs and outputs are used in this program part, these are also read and written in the interrupt routine, triggered by the input edge. Reading and writing can be stopped via the FB inputs READ_INPUTS, WRITE_OUTPUTS and ANALOG_INPUTS.

So in the program block all time-critical events can be processed by linking inputs or global variables and writing outputs. So FBs can only be executed if actually called by an input signal.

NOTE

The program block should be skipped in the cycle (except for the initialisation call) so that it is not cyclically called, too.

The input (CHANNEL) monitored for triggering the interrupt cannot be initialised and further processed in the interrupt routine.

The runtime of the main cycle plus the sum of the duration of all program parts called via interrupt must always be within the max. permissible cycle time!

The user is responsible for data consistency between the main program and the program parts running in the interrupt mode!

© ifm electronic gmbh www.ifm.com

19866

Interrupt priorities:

- All program parts called via interrupt have the same priority of execution. Several simultaneous interrupts are processed sequentially in the order of their occurrence.
- If a further edge is detected on the same input during execution of the program part called via interrupt, the interrupt is listed for processing and the program is directly called again after completion. As an option, interfering multiple pulses can be filtered out by setting the glitch filter.
- The program running in the interrupt mode can be disrupted by interrupts with a higher priority (e.g. CAN).
- If several interrupts are present on the same channel, the last initialised FB (or the PRG) will be assigned the channel. The previously defined FB (or the PRG) is then no longer called and no longer provides data.

Do **not** use this function block on one input together with one of the following function blocks!
• **INC_ENCODER_HR** (→ p. 150)

Parameters of the inputs

2383

Parameter	Data type	Description
ENABLE	BOOL	TRUE (only for 1 cycle): initialisation of the function block FALSE: unit is not executed
CHANNEL	BYTE	Number of interrupt input 0...7 for the inputs IN00...IN07
MODE	BYTE	Type of edge at the input CHANNEL which triggers the interrupt 1 = rising edge (standard value) 2 = falling edge 3 = rising and falling edge > 3 = standard value
READ_INPUTS	BOOL	TRUE: read the inputs 0...7 before calling the program and write into the input flags I00...I07 FALSE: only read the channel indicated under CHANNEL and write to the corresponding input flag Ixx
WRITE_OUTPUTS	BOOL	TRUE: write the current values of the output flags Q00...Q07 to the outputs after completion of the program sequence FALSE: do not write outputs
ANALOG_INPUTS	BOOL	TRUE: read inputs 0...7 and write the unfiltered, uncalibrated analogue values to the flags ANALOG_IRQ00...07 FALSE: do not write flags ANALOG_IRQ00...07

© ifm electronic gmbh

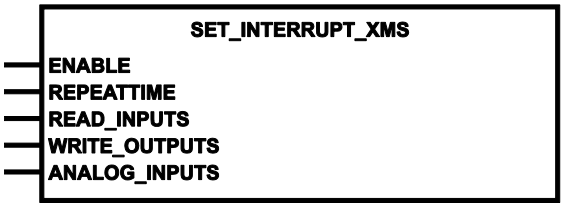
SET_INTERRUPT_XMS

272

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

19363

SET_INTERRUPT_XMS handles the execution of a program part at an interval of x ms.

In the conventional PLC the cycle time is decisive for real-time monitoring. So, the PLC is at a disadvantage as compared to customer-specific controllers. Even a "real-time operating system" does not change this fact when the whole application program runs in one single block which cannot be changed.

A possible solution would be to keep the cycle time as short as possible. This often leads to splitting the application up to several control cycles. This, however, makes programming complex and difficult.

Another possibility is to call a certain program part at fixed intervals (every x ms) independently of the control cycle:

The time-critical part of the application is integrated by the user in a block of the type PROGRAM (PRG). This block is declared as the interrupt routine by calling SET_INTERRUPT_XMS once (during initialisation). As a consequence, this program block is always processed after the REPEATTIME has elapsed (every x ms). If inputs and outputs are used in this program part, they are also read and written in the defined cycle. Reading and directly writing can be stopped via the FB inputs READ_INPUTS, WRITE_OUTPUTS and ANALOG_INPUTS.

So, in the program block all time-critical events can be processed by linking inputs or global variables and writing outputs. So, timers can be monitored more precisely than in a "normal cycle".

NOTE

- In the whole program call SET_INTERRUPT_XMS only once and only in the first cycle!
- To avoid that the program block called by interrupt is additionally called cyclically, it should be skipped in the cycle (with the exception of the initialisation call).

Several timer interrupt blocks can be active.

Restrictions for time permitted:

- Runtime of the program part running in interrupt < REPEATTIME
- The runtime of the main cycle plus the sum of the duration of all program parts called via interrupt must always be within the max. permissible cycle time!

The user is responsible for data consistency between the main program and the program parts running in the interrupt mode!

Please note: In case of a high CAN bus activity the set REPEATTIME may fluctuate.

The count of per interrupt running program parts (as SET_INTERRUPT_XMS) is limited to 16. More definitions of program parts running per interrupt than SET_INTERRUPT_XMS will be ignored and not executed.

© ifm electronic gmbh www.ifm.com

Interrupt priorities:

- All program parts called via interrupt have the same priority of execution. Several simultaneous interrupts are processed sequentially in the order of their occurrence.
- The program running in the interrupt mode can be disrupted by interrupts with a higher priority (e.g. CAN).

Parameters of the inputs

2382

Parameter	Data type	Description
ENABLE	BOOL	TRUE (only for 1 cycle): initialisation of the function block FALSE: unit is not executed
REPEATTIME	TIME	Duration in [ms] between end of program and reboot The duration between two calls is determined as the sum of REPEATTIME and runtime of the program called via interrupt.
READ_INPUTS	BOOL	TRUE: read the inputs 0...7 before calling the program and write into the input flags I00...I07 FALSE: no update of the inputs
WRITE_OUTPUTS	BOOL	TRUE: write the current values of the output flags Q00...Q07 to the outputs after completion of the program sequence FALSE: do not write outputs
ANALOG_INPUTS	BOOL	TRUE: read inputs 0...7 and write the unfiltered, uncalibrated analogue values to the flags ANALOG_IRQ00...07 FALSE: do not write flags ANALOG_IRQ00...07

© ifm electronic gmbh

5.2.8 Function elements: processing input values

Contents

1602
1302

In this chapter we show you ifm FBs which allow you to read and process the analogue or digital signals at the device input.

NOTE

The analogue raw values shown in the PLC configuration of CODESYS directly come from the ADC. They are not yet corrected!

Therefore different raw values can appear in the PLC configuration for identical devices.

Error correction and normalisation are only carried out by ifm function blocks. The function blocks provide the corrected value.

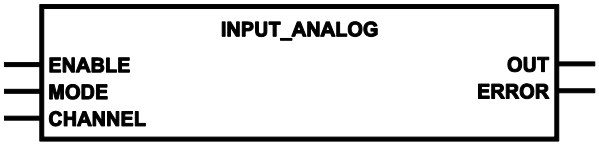
© ifm electronic gmbh

INPUT_ANALOG

15916

Unit type = function block (FB)
Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

12912
12916

INPUT_ANALOG enables the following operating modes at the input channels.
Details → chapter **Possible operating modes inputs/outputs** (→ p. 243)

The function block provides the current analogue value at the selected analogue channel. The analogue values are provided as standardised values. At the same time the uncalibrated raw values are provided via the system flags ANALOGxx.

► For frequency and period measurements as well as counter functions: set MODE=1 (= IN_DIGITAL_H)!

The measurement and the output value results from the operating mode indicated via MODE:

19260

MODE dec	MODE hex	Input operating mode	Global variable	FB Output OUT	Unit
0	0000	deactivated	IN_NOMODE	---	---
1	0001	binary input, minus switching (BH)	IN_DIGITAL_H	0 / 1	---
2	0002	binary input, plus switching (BL)	IN_DIGITAL_L	0 / 1	---
4	0004	current input	IN_CURRENT	0…20 000	µA
8	0008	voltage input	IN_VOLTAGE10	0…10 000	mV
16	0010	voltage input	IN_VOLTAGE30	0…32 000	mV
32	0020	voltage input, ratiometric	IN_RATIO	0…1 000	‰
64	0040	---	---	---	---
128	0080	---	---	---	---
256	0100	---	---	---	---
512	0200	resistance input	IN_RESISTANCE	3...680 16...30 000	Ω

20790

These and other operating modes of the inputs can also be configured with ...

SET_INPUT_MODE (→ p. 134)	Assigns an operating mode to an input channel

These and other operating modes of the inputs: → FB **SET_INPUT_MODE** (→ p. 134).

- If VBBs < 4 V (for the modes 1, 2, 32): no values are read
- Values which are higher than indicated are also recorded (incl. ratio)
- Overload protection is always active during current measurement

18414

If inputI15 is not used:
► Configure input I15 as binary input!

13020

NOTE

When switching into another mode during runtime, it takes a few cycles before the output value is correct again.

If the same input channel was configured differently during runtime, the most recent configuration will apply.

Parameters of the inputs

11937

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
MODE	WORD	Operating mode of the input channel CHANNEL: 0 = 0x0000 IN_NOMODE (off; preset is active) 1 = 0x0001 IN_DIGITAL_H preset 2 = 0x0002 IN_DIGITAL_L 4 = 0x0004 IN_CURRENT 0…20 000 µA 8 = 0x0008 IN_VOLTAGE10 0…10 000 mV 16 = 0x0010 IN_VOLTAGE30 0…32 000 mV 32 = 0x0020 IN_RATIO 0...1 000 ‰ 64 = 0x0040 --- 128 = 0x0080 --- 256 = 0x0100 --- 512 = 0x0200 IN_RESISTANCE 3...680 Ω / 16...30 000 Ω
CHANNEL	BYTE	Number of input channel 0...15 for the inputs I00...I15 For the function block xxx_**E** (if available) applies: 0...23 for the inputs I00_E...I23_E

© ifm electronic gmbh www.ifm.com

Parameters of the outputs

11938

Parameter	Data type	Description
OUT	WORD	Output value according to MODE in case of an invalid setting: OUT = "0"
ERROR	DWORD	Error code from this function block call → **Error codes** (→ p. 314) (possible messages → following table)

Possible results for ERROR (n=any desired value):

The 32-bit error code consists of four 8-bit values (DWORD).

4th byte	3rd byte	2nd byte	1st byte
Error class	Application-specific error code	Error source	Error cause

Value [hex]	Description
00 00 00 00	no error
01 00 10+nn 01	Wire break on input channel Inn (nn = HEX value) Example: channel I15 ⇨ 10+nn = 1F
01 00 10+nn 02	Short cirquit on input channel Inn (nn = HEX value) Example: channel I15 ⇨ 10+nn = 1F (only if IN_DIGITAL_H and DIAGNOSTICS = TRUE)
01 00 10+nn 04	Overload on input channel Inn (nn = HEX value) Example: channel I15 ⇨ 10+nn = 1F (only if current measurement or resistance measurement)
01 00 00 F8	Wrong parameter ⇨ general error

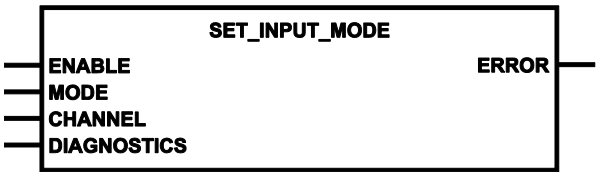
© ifm electronic gmbh

SET_INPUT_MODE

15918

Unit type = function block (FB)
Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

11944

Use SET_INPUT_MODE to assign operating modes to the input channels.
→ chapter **Possible operating modes inputs/outputs** (→ p. 243)

- Values which are higher than indicated are also recorded (incl. ratio)
- Overload protection is always active during current measurement

The operating mode should not be changed during operation.

The function block **INPUT_ANALOG** (→ p. 131) can also be used to configure the operating mode on an input.

18414

If inputI15 is not used:
▶ Configure input I15 as binary input!

13020

NOTE

When switching into another mode during runtime, it takes a few cycles before the output value is correct again.

If the same input channel was configured differently during runtime, the most recent configuration will apply.

© ifm electronic gmbh www.ifm.com

Parameters of the inputs

11945

Parameter	Data type	Description
ENABLE	BOOL	FALSE ⇨ TRUE (edge): Initialise block (only 1 cycle) > Read block inputs TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
MODE	WORD	Operating mode of the input channel CHANNEL: 0 = 0x0000 IN_NOMODE (off; preset is active) 1 = 0x0001 IN_DIGITAL_H preset 2 = 0x0002 IN_DIGITAL_L 4 = 0x0004 IN_CURRENT 0…20 000 µA 8 = 0x0008 IN_VOLTAGE10 0…10 000 mV 16 = 0x0010 IN_VOLTAGE30 0…32 000 mV 32 = 0x0020 IN_RATIO 0...1 000 ‰ 64 = 0x0040 --- 128 = 0x0080 --- 256 = 0x0100 --- 512 = 0x0200 IN_RESISTANCE 3...680 Ω / 16...30 000 Ω
CHANNEL	BYTE	Number of input channel 0...15 for the inputs I00...I15 For the function block xxx_**E** (if available) applies: 0...15 for the inputs I00_E...I15_E
DIAGNOSTICS	BOOL	TRUE: Channel with diagnostic function only effective for IN_DIGITAL_H or IN_RESISTANCE Error messages: • wire break or short to ground if input voltage < 1V for > 100 ms • short to supply if input voltage < 95 % VBBS for > 100 ms • resistor value > 31 kOhm (I12...I14) or > 700 Ohm (I15)

Parameters of the outputs

11947

Parameter	Data type	Description
ERROR	DWORD	Error code from this function block call → **Error codes** (→ p. 314) (possible messages → following table)

Possible results for ERROR (n=any desired value):

The 32-bit error code consists of four 8-bit values (DWORD).

4th byte	3rd byte	2nd byte	1st byte
Error class	Application-specific error code	Error source	Error cause

Value [hex]	Description
00 00 00 00	no error
01 00 00 F8	Wrong parameter ⇨ general error

© ifm electronic gmbh www.ifm.com

5.2.9 Function elements: adapting analogue values

Contents

1603

If the values of analogue inputs or the results of analogue functions must be adapted, the following FBs will help you.

© ifm electronic gmbh

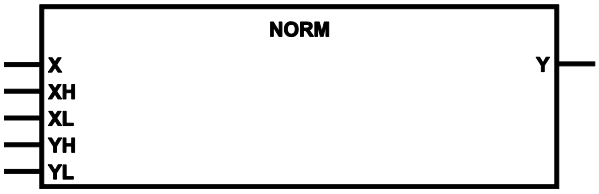
NORM

401

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

404

NORM normalises a value within defined limits to a value with new limits.

The FB normalises a value of type WORD within the limits of XH and XL to an output value within the limits of YH and YL. This FB is for example used for generating PWM values from analogue input values.

NOTE

- ► The value for X must be in the defined input range between XL and XH!
  There is no internal plausibility check of the value X.
- > Due to rounding errors the normalised value can deviate by 1.
- > If the limits (XH/XL or YH/YL) are defined in an inverted manner, normalisation is also done in an inverted manner.

Parameters of the inputs

405

Parameter	Data type	Description
X	WORD	input value
XH	WORD	Upper limit of input value range [increments]
XL	WORD	Lower limit of input value range [increments]
YH	WORD	Upper limit of output value range
YL	WORD	Lower limit of output value range

Parameters of the outputs

406

Parameter	Data type	Description
Y	WORD	output value

© ifm electronic gmbh www.ifm.com

Example: NORM (1)

407

lower limit value input	0	XL
upper limit value input	100	XH
lower limit value output	0	YL
upper limit value output	2000	YH

then the FB converts the input signal for example as follows:

from X =	50	0	100	75
	↓	↓	↓	↓
to Y =	1000	0	2000	1500

Example: NORM (2)

408

lower limit value input	2000	XL
upper limit value input	0	XH
lower limit value output	0	YL
upper limit value output	100	YH

then the FB converts the input signal for example as follows:

from X =	1000	0	2000	1500
	↓	↓	↓	↓
to Y =	50	100	0	25

© ifm electronic gmbh www.ifm.com

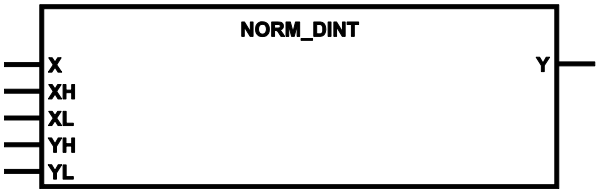
NORM_DINT

2217

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2355

NORM_DINT normalises a value within defined limits to a value with new limits.

The FB normalises a value of type DINT within the limits of XH and XL to an output value within the limits of YH and YL. This FB is for example used for generating PWM values from analogue input values.

NOTE

- ► The value for X must be in the defined input range between XL and XH!
  There is no internal plausibility check of the value X.
- ► The result of the calculation (XH-XL)•(YH-YL) must remain in the value range of data type DINT (-2 147 483 648...2 147 483 647)!
- > Due to rounding errors the normalised value can deviate by 1.
- > If the limits (XH/XL or YH/YL) are defined in an inverted manner, normalisation is also done in an inverted manner.

Parameters of the inputs

2359

Parameter	Data type	Description
X	DINT	current input value
XH	DINT	upper limit of input value range
XL	DINT	lower limit of input value range
YH	DINT	upper limit of output value range
YL	DINT	lower limit of output value range

Parameters of the outputs

2360

Parameter	Data type	Description
Y	DINT	output value

© ifm electronic gmbh www.ifm.com

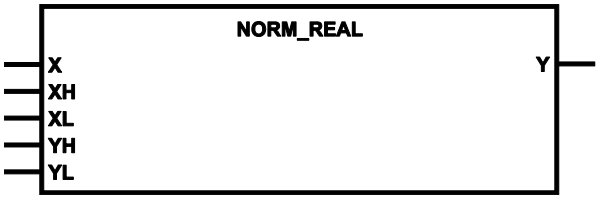
NORM_REAL

2218

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2358

NORM_REAL normalises a value within defined limits to a value with new limits.

The FB normalises a value of type REAL within the limits of XH and XL to an output value within the limits of YH and YL. This FB is for example used for generating PWM values from analogue input values.

NOTE

- ► The value for X must be in the defined input range between XL and XH!
  There is no internal plausibility check of the value X.
- ► The result of the calculation (XH-XL)•(YH-YL) must remain in the value range of data type REAL ($-3{,}402823466 \cdot 10^{38}$...$3{,}402823466 \cdot 10^{38}$)!
- > Due to rounding errors the normalised value can deviate by 1.
- > If the limits (XH/XL or YH/YL) are defined in an inverted manner, normalisation is also done in an inverted manner.

Parameters of the inputs

2356

Parameter	Data type	Description
X	REAL	Input value
XH	REAL	Upper limit of output value range
XL	REAL	Lower limit of the input value range
YH	REAL	Upper limit of the output value range
YL	REAL	Lower limit of output value range

Parameters of the outputs

2357

Parameter	Data type	Description
Y	REAL	Output value

© ifm electronic gmbh www.ifm.com

5.2.10 Function elements: counter functions for frequency and period measurement

Contents

19285

Depending on the ecomatmobile device up to 16*) fast inputs are supported which can process input frequencies of up to 30 kHz. In addition to frequency measurement, the inputs can also be used for the evaluation of incremental encoders (counter function).
*) ExtendedController: up to 32 fast inputs

Due to the different measuring methods errors can occur when the frequency is determined.

The following FBs are available for easy evaluation:

Function element	Permissible values	Explanation
FREQUENCY	0.1...30 000 Hz	Measurement of the frequency on the indicated channel. Measurement error is reduced in case of high frequencies
PERIOD	0.1...5 000 Hz	Measurement of frequency and period duration (cycle time) on the indicated channel
PERIOD_RATIO	0.1...5 000 Hz	Measurement of frequency and period duration (cycle time) as well as mark-to-space ratio [‰] on the indicated channel
FREQUENCY_PERIOD	0.1...30 000 Hz	The FB combines the two FBs FREQUENCY and PERIOD or PERIOD_RATIO. Automatic selection of the measuring method at 5 kHz
PHASE	0.1...5 000 Hz	Reading of a channel pair and comparison of the phase position of the signals
INC_ENCODER	0.1...30 000 Hz	Up/down counter function for the evaluation of encoders
INC_ENCODER_HR	0.1...5 000 Hz	Up/down counter function for the high resolution evaluation of encoders
FAST_COUNT	0.1...30 000 Hz	Counting of fast pulses

Important when using the fast inputs as "normal" digital inputs:

- ► The increased sensitivity to noise pulses must be taken into account (e.g. contact bouncing for mechanical contacts).
- ► The input signal must be debounced, if necessary! → chapter **Configure the hardware filter** (→ p. 64)
- • The standard digital input can evaluate signals up to 50 Hz (depending on IEC cycle and filter settings).

© ifm electronic gmbh

FAST_COUNT

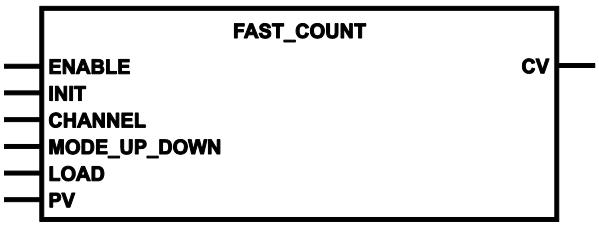
15922

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

For the extended side of the ExtendedControllers the FB name ends with '_E'. (not for CR0133)

Symbol in CODESYS:



Description

20653

FAST_COUNT operates as counter block for fast input pulses.

During ENABLE=TRUE the function block detects rising edges at the FRQ input channels.
Maximum input frequency → data sheet.

If resetting and newly setting ENABLE, the counter continues to count from the value that was valid at the last reset of ENABLE.

When setting INIT (rising edge) the count value CV is set to 0.
When resetting the parameter INIT the counter counts from 0.

22690

The following applies to the standard side of the device:
This function block may be used at the same input together with the function block **INC_ENCODER** (→ p. 148).

22692

► Initialise the jointly operated function blocks at the same time.

> Subsequent loading of function block FAST_COUNT may influence the values of the function block INC_ENCODER.

Do **not** use this function block on one input together with one of the following function blocks!
- **FREQUENCY** (→ p. 144)
- **FREQUENCY_PERIOD** (→ p. 146)
- **INC_ENCODER_HR** (→ p. 150)
- **PERIOD** (→ p. 152)
- **PERIOD_RATIO** (→ p. 154)
- **PHASE** (→ p. 156)

14888

NOTE

In case of higher frequencies (higher than those guaranteed by ifm) the following problems may occur:

- The switch-on and switch-off times of the outputs become more important.
- Undue heating of the components may occur.

The influences mentioned above depend on the components used in the individual case.
These possible influences cannot be exactly predicted.

© ifm electronic gmbh www.ifm.com

Parameters of the inputs

19869

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > counter stopped
INIT	BOOL	FALSE ⇨ TRUE (edge): unit is initialised FALSE: during further processing of the program
CHANNEL	BYTE	number of the fast input channel 0...11 for the inputs I00...I11 For the function block xxx_**E** (if available) applies: 0...15 for the inputs I00_E...I15_E
MODE_UP_DOWN	BOOL	TRUE: counter counts downwards FALSE: counter counts upwards
LOAD	BOOL	TRUE: start value PV is loaded in CV FALSE: function element is not executed
PV	DWORD	Start value (preset value) for the counter

Parameters of the outputs

572

Parameter	Data type	Description
CV	DWORD	current counter value Behaviour in case of overflow: • the counter stops at 0 when counting downwards • there is an overflow when counting upwards

© ifm electronic gmbh www.ifm.com

FREQUENCY

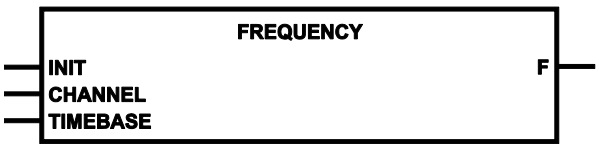
15924

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

For the extended side of the ExtendedControllers the FB name ends with '_E'. (not for CR0133)

Symbol in CODESYS:



Description

20656
20675

FREQUENCY measures the frequency of the signal arriving at the selected CHANNEL. The FB evaluates the positive edge of the signal.

Depending on the TIMEBASE, frequency measurements can be carried out in a wide value range.
- high frequencies require a short timebase
- low frequencies require a longer timebase

Limit values:

TIMEBASE	permissible, measurable frequency
57 000 ms (= maximum value)	1 149 Hz
2 184 ms	30 000 Hz (= maximum value)

The longer the timebase for the frequency to be measured, the more precise the measured value determined.
Example of a frequency = 1 Hz:

TIMEBASE [ms]	max. errors [%]	Measurement [Hz]
1 000	100	0...2
10 000	10	0.9...1.1

The frequency is provided directly in [Hz].

14888

NOTE

In case of higher frequencies (higher than those guaranteed by ifm) the following problems may occur:

- The switch-on and switch-off times of the outputs become more important.
- Undue heating of the components may occur.

The influences mentioned above depend on the components used in the individual case.
These possible influences cannot be exactly predicted.

7321

For frequency measuring: ensure that the function block does not receive more than 65 535 positive edges within the value of TIMEBASE!
Otherwise, the internal counting register may overflow and lead to incorrect results.

© ifm electronic gmbh www.ifm.com

22690

The following applies to the standard side of the device:
This function block may be used at the same input together with the function block **INC_ENCODER** (→ p. 148).

22691

► Initialise the jointly operated function blocks at the same time.

Do **not** use this function block on one input together with one of the following function blocks!
- **FAST_COUNT** (→ p. 142)
- **FREQUENCY_PERIOD** (→ p. 146)
- **INC_ENCODER_HR** (→ p. 150)
- **PERIOD** (→ p. 152)
- **PERIOD_RATIO** (→ p. 154)
- **PHASE** (→ p. 156)

Parameters of the inputs

19871

Parameter	Data type	Description
INIT	BOOL	TRUE (only for 1 cycle): Function block and interface are initialised FALSE: measurement in process or:measurement begins if previously INIT=TRUE
CHANNEL	BYTE	number of the fast input channel 0...11 for the inputs I00...I11 For the function block xxx_**E** (if available) applies: 0...15 for the inputs I00_E...I15_E
TIMEBASE	TIME	Time basis for frequency measurement (max. 57 s)

Parameters of the outputs

542

Parameter	Data type	Description
F	REAL	frequency of the input signal in [Hz]

© ifm electronic gmbh

FREQUENCY_PERIOD

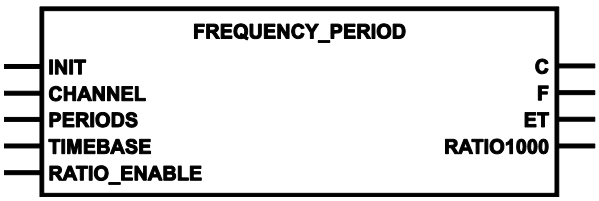
15926

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

For the extended side of the ExtendedControllers the FB name ends with '_E'. (not for CR0133)

Symbol in CODESYS:



Description

20659
20676

FREQUENCY_PERIOD measures the frequency and period duration (cycle time) in [µs] on the indicated channel (allowed for all inputs). Maximum input frequency → data sheet.

The FB combines PERIOD/PERIOD_RATIO and FREQUENCY in a common function. The measuring method is automatically selected at approx. 5 kHz:
- below 5.2 kHz the FB behaves like PERIOD or PERIOD_RATIO
- above 5.5 kHz the FB behaves like FREQUENCY.

This FB measures the frequency and the cycle time of the signal at the selected CHANNEL. To calculate, all positive edges are evaluated and the average value is determined by means of the number of indicated PERIODS.

For an input frequency > 5 kHz and an active FREQUENCY mode the ratio cannot be measured.

The maximum measuring range is approx. 15 min.

14888

NOTE

In case of higher frequencies (higher than those guaranteed by ifm) the following problems may occur:

- The switch-on and switch-off times of the outputs become more important.
- Undue heating of the components may occur.

The influences mentioned above depend on the components used in the individual case.
These possible influences cannot be exactly predicted.

7321

For frequency measuring: ensure that the function block does not receive more than 65 535 positive edges within the value of TIMEBASE!
Otherwise, the internal counting register may overflow and lead to incorrect results.

22690

The following applies to the standard side of the device:
This function block may be used at the same input together with the function block **INC_ENCODER** (→ p. 148).

22691

► Initialise the jointly operated function blocks at the same time.

NOTE

Do **not** use this function block on one input together with one of the following function blocks!
- **FAST_COUNT** (→ p. 142)
- **FREQUENCY** (→ p. 144)
- **INC_ENCODER_HR** (→ p. 150)
- **PERIOD** (→ p. 152)
- **PERIOD_RATIO** (→ p. 154)
- **PHASE** (→ p. 156)

Parameters of the inputs

19872

Parameter	Data type	Description
INIT	BOOL	TRUE (only for 1 cycle): Function block and interface are initialised FALSE: measurement in process or:measurement begins if previously INIT=TRUE
CHANNEL	BYTE	number of the fast input channel 0...11 for the inputs I00...I11 For the function block xxx_**E** (if available) applies: 0...15 for the inputs I00_E...I15_E
PERIODS	BYTE	Number of periods to be averaged (1...16) 0 : Outputs C and F are not updated > 16 : is limited to 16
TIMEBASE	TIME	Time basis for frequency measurement (max. 57 s)
RATIO_ENABLE	BOOL	TRUE: Ratio measurement provided to RATIO1000 FALSE: No ratio measurement provided

Parameters of the outputs

2337

Parameter	Data type	Description
C	DWORD	Cycle time of the detected periods in [µs] permissible = 33...10 000 000 = 0x21...0x989680
F	REAL	frequency of the input signal in [Hz]
ET	TIME	for measuring the interval: (can be used for very slow signals) RATIO_ENABLE = TRUE: time elapsed since the last change of edge on the input RATIO_ENABLE = FALSE: time elapsed since the last rising edge on the input for other measurements: ET = 0
RATIO1000	WORD	Mark-to-space ratio in [‰] permissible = 1...999 = 0x1...0x3E7 Preconditions: • for measuring the interval • pulse duration ≥ 100 µs • frequency < 5 kHz

© ifm electronic gmbh www.ifm.com

INC_ENCODER

15928

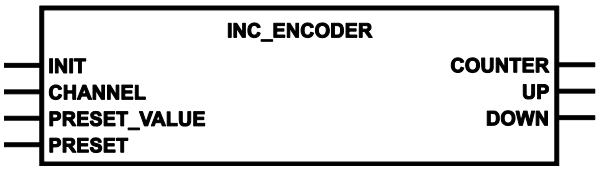
= Incremental Encoder

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

For the extended side of the ExtendedControllers the FB name ends with '_E'. (not for CR0133)

Symbol in CODESYS:



Description

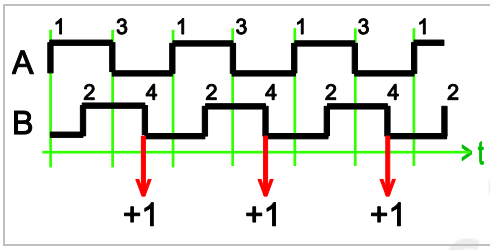
19302

INC_ENCODER offers up/down counter functions for the evaluation of encoders.

Each input pair CHANNEL to be evaluated by means of the function block is formed by two neighbouring frequency inputs.

Limit frequency = 30 kHz
max. number of units to be connected: 4 encoders (ExtendedController: max. 8 encoders)

Set preset value:
1. Enter value in PRESET_VALUE
2. Set PRESET to TRUE for one cycle
3. Reset PRESET to FALSE



The function block counts every 4th pulse at the input pair (corresponds to the resolution of the encoder), as long as INIT=FALSE and PRESET=FALSE.

The current counter value is available at the output COUNTER.

The outputs UP and DOWN indicate the current counting direction of the counter. The outputs are TRUE if the counter has counted in the corresponding direction in the preceding program cycle (even in case of overflow). If the counter stops, the direction output in the following program cycle is also reset.

22616

The following applies to the standard side of the device:
For each physical input of the function block **INC_ENCODER** (→ p. 148), **one** of the following POUs may be used in addition:

- **FAST_COUNT** (→ p. 142)
- **FREQUENCY** (→ p. 144)
- **FREQUENCY_PERIOD** (→ p. 146)
- **PERIOD** (→ p. 152)
- **PERIOD_RATIO** (→ p. 154)
- **PHASE** (→ p. 156)

22691

► Initialise the jointly operated function blocks at the same time.

The following applies to the extended side of the device:
For each physical input of the function block INC_ENCODER_E, **no** further function blocks may be used in addition.

Parameters of the inputs

529

Parameter	Data type	Description
INIT	BOOL	TRUE (only for 1 cycle): Function block is initialised FALSE: during further processing of the program
CHANNEL	BYTE	Number of the input channel pair 0 = channel pair 0 = inputs I00 + I01 ... 3 = channel pair 3 = inputs I06 + I07 For the function block xxx_**E** (if available) applies: 0 = channel pair 0 = inputs I00_E + I01_E ... 3 = channel pair 3 = inputs I06_E + I07_E
PRESET_VALUE	DINT	counter start value
PRESET	BOOL	FALSE ⇨ TRUE (edge): PRESET_VALUE is loaded to COUNTER TRUE: Counter ignores the input pulses FALSE: Counter counts the input pulses

Parameters of the outputs

530

Parameter	Data type	Description
COUNTER	DINT	Current counter value
UP	BOOL	TRUE: counter counts upwards in the last cycle FALSE: counter counts not upwards in the last cycle
DOWN	BOOL	TRUE: counter counts downwards in the last cycle FALSE: counter counts not downwards in the last cycle

© ifm electronic gmbh

INC_ENCODER_HR

19225

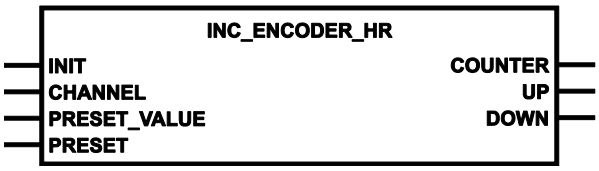
= Incremental Encoder high resolution

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

For the extended side of the ExtendedControllers the FB name ends with '_E'.

Symbol in CODESYS:



Description

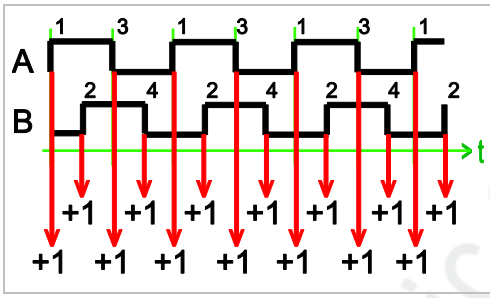
19231

INC_ENCODER offers up/down counter functions for the evaluation of encoders.

Each input pair to be evaluated by means of the function block is formed by two neighbouring frequency inputs.

Limit frequency = 5 kHz
max. number of units to be connected: 4 encoders (ExtendedController: max. 8 encoders)

Set preset value:
1. Enter value in PRESET_VALUE
2. Set PRESET to TRUE for one cycle
3. Reset PRESET to FALSE



The function block counts every edge at the input pair (corresponding to 4-fold resolution of the encoder), as long as INIT=FALSE and PRESET=FALSE.

The current counter value is available at the output COUNTER.

The outputs UP and DOWN indicate the current counting direction of the counter:

- Output = TRUE if the counter has counted in the corresponding direction in the preceding program cycle (even in case of overflow).
- If the counter stops, the direction output in the following program cycle is also reset.
- Different in the event of overflow:
  When the counter stops, the direction output = TRUE until an opposite count has been made.

Do **not** use this function block on one input together with one of the following function blocks!
- **FAST_COUNT** (→ p. 142)
- **FREQUENCY** (→ p. 144)
- **FREQUENCY_PERIOD** (→ p. 146)
- **INC_ENCODER** (→ p. 148)
- **PERIOD** (→ p. 152)
- **PERIOD_RATIO** (→ p. 154)
- **PHASE** (→ p. 156)
- **SET_INTERRUPT_I** (→ p. 126)

Parameters of the inputs

529

Parameter	Data type	Description
INIT	BOOL	TRUE (only for 1 cycle): Function block is initialised FALSE: during further processing of the program
CHANNEL	BYTE	Number of the input channel pair 0 = channel pair 0 = inputs I00 + I01 ... 3 = channel pair 3 = inputs I06 + I07 For the function block xxx_**E** (if available) applies: 0 = channel pair 0 = inputs I00_E + I01_E ... 3 = channel pair 3 = inputs I06_E + I07_E
PRESET_VALUE	DINT	counter start value
PRESET	BOOL	FALSE ⇨ TRUE (edge): PRESET_VALUE is loaded to COUNTER TRUE: Counter ignores the input pulses FALSE: Counter counts the input pulses

Parameters of the outputs

530

Parameter	Data type	Description
COUNTER	DINT	Current counter value
UP	BOOL	TRUE: counter counts upwards in the last cycle FALSE: counter counts not upwards in the last cycle
DOWN	BOOL	TRUE: counter counts downwards in the last cycle FALSE: counter counts not downwards in the last cycle

© ifm electronic gmbh www.ifm.com

PERIOD

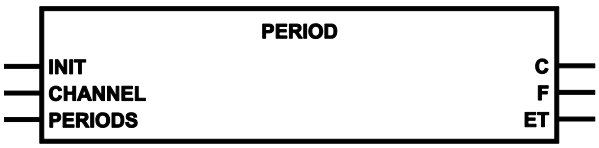
15930

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

For the extended side of the ExtendedControllers the FB name ends with '_E'. (not for CR0133)

Symbol in CODESYS:



Description

20662
20677

PERIOD measures the frequency and period duration (cycle time) in [µs] on the indicated channel (allowed for all inputs). Maximum input frequency → data sheet.

This FB measures the frequency and the cycle time of the signal at the selected CHANNEL. To calculate, all positive edges are evaluated and the average value is determined by means of the number of indicated PERIODS.

In case of low frequencies there will be inaccuracies when using **FREQUENCY** (→ p. 144). To avoid this, PERIOD can be used. The cycle time is directly indicated in [µs].

The maximum measuring range is 10 seconds.

14888

NOTE

In case of higher frequencies (higher than those guaranteed by ifm) the following problems may occur:

- The switch-on and switch-off times of the outputs become more important.
- Undue heating of the components may occur.

The influences mentioned above depend on the components used in the individual case.
These possible influences cannot be exactly predicted.

22690

The following applies to the standard side of the device:
This function block may be used at the same input together with the function block **INC_ENCODER** (→ p. 148).

22691

► Initialise the jointly operated function blocks at the same time.

Do **not** use this function block on one input together with one of the following function blocks!
- **FAST_COUNT** (→ p. 142)
- **FREQUENCY** (→ p. 144)
- **FREQUENCY_PERIOD** (→ p. 146)
- **INC_ENCODER_HR** (→ p. 150)
- **PERIOD_RATIO** (→ p. 154)
- **PHASE** (→ p. 156)

© ifm electronic gmbh www.ifm.com

Parameters of the inputs

19874

Parameter	Data type	Description
INIT	BOOL	FALSE ⇨ TRUE (edge): unit is initialised FALSE: during further processing of the program
CHANNEL	BYTE	number of the fast input channel 0...11 for the inputs I00...I11 For the function block xxx_**E** (if available) applies: 0...15 for the inputs I00_E...I15_E
PERIODS	BYTE	Number of periods to be averaged (1...16) 0 : Outputs C and F are not updated > 16 : is limited to 16

Parameters of the outputs

375

Parameter	Data type	Description
C	DWORD	Cycle time of the detected periods in [µs] allowed = 200...10 000 000 = 0xC8...0x989680 (= 10 seconds)
F	REAL	frequency of the input signal in [Hz]
ET	TIME	time elapsed since the last rising edge on the input (can be used for very slow signals)

© ifm electronic gmbh

PERIOD_RATIO

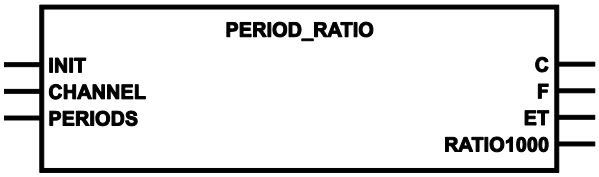
15932

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

For the extended side of the ExtendedControllers the FB name ends with '_E'. (not for CR0133)

Symbol in CODESYS:



Description

20665
20678

PERIOD_RATIO measures the frequency and the period duration (cycle time) in [µs] over the indicated periods on the indicated channel (allowed for all inputs). In addition, the mark-to-period ratio is indicated in [‰]. Maximum input frequency → data sheet.

This FB measures the frequency and the cycle time of the signal at the selected CHANNEL. To calculate, all positive edges are evaluated and the average value is determined by means of the number of indicated PERIODS. In addition, the mark-to-period ratio is indicated in [‰].
For example: In case of a signal ratio of 25 ms high level and 75 ms low level the value RATIO1000 is provided as 250 ‰.

In case of low frequencies there will be inaccuracies when using **FREQUENCY** (→ p. 144). To avoid this, PERIOD_RATIO can be used. The cycle time is directly indicated in [µs].

The maximum measuring range is 10 seconds.

14888

NOTE

In case of higher frequencies (higher than those guaranteed by ifm) the following problems may occur:

- The switch-on and switch-off times of the outputs become more important.
- Undue heating of the components may occur.

The influences mentioned above depend on the components used in the individual case.
These possible influences cannot be exactly predicted.

22690

The following applies to the standard side of the device:
This function block may be used at the same input together with the function block **INC_ENCODER** (→ p. 148).

22691

► Initialise the jointly operated function blocks at the same time.

Do **not** use this function block on one input together with one of the following function blocks!
- **FAST_COUNT** (→ p. 142)
- **FREQUENCY** (→ p. 144)
- **FREQUENCY_PERIOD** (→ p. 146)
- **INC_ENCODER_HR** (→ p. 150)
- **PERIOD** (→ p. 152)
- **PHASE** (→ p. 156)

© ifm electronic gmbh www.ifm.com

Parameters of the inputs

19873

Parameter	Data type	Description
INIT	BOOL	FALSE ⇨ TRUE (edge): unit is initialised FALSE: during further processing of the program
CHANNEL	BYTE	number of the fast input channel 0...11 for the inputs I00...I11 ℹ For the function block xxx_**E** (if available) applies: 0...15 for the inputs I00_E...I15_E
PERIODS	BYTE	Number of periods to be averaged (1...16) 0 : Outputs C and F are not updated > 16 : is limited to 16

Parameters of the outputs

369

Parameter	Data type	Description
C	DWORD	Cycle time of the detected periods in [µs] allowed = 200...10 000 000 = 0xC8...0x989680 (= 10 seconds)
F	REAL	frequency of the input signal in [Hz]
ET	TIME	Time passed since the last change of state on the input (can be used in case of very slow signals)
RATIO1000	WORD	Mark-to-space ratio in [‰] permissible = 1...999 = 0x1...0x3E7 Preconditions: • for measuring the interval • pulse duration ≥ 100 µs • frequency < 5 kHz

© ifm electronic gmbh www.ifm.com

PHASE

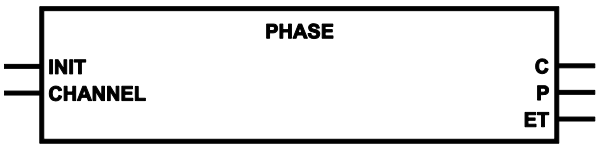
15934

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

For the extended side of the ExtendedControllers the FB name ends with '_E'. (not for CR0133)

Symbol in CODESYS:



Description

20668
20679

PHASE reads a pair of channels with fast inputs and compares the phase position of the signals. Maximum input frequency → data sheet.

This FB compares a pair of channels with fast inputs so that the phase position of two signals towards each other can be evaluated. An evaluation of the cycle period is possible even in the range of seconds (max. 10 seconds).

14888

NOTE

In case of higher frequencies (higher than those guaranteed by ifm) the following problems may occur:

- The switch-on and switch-off times of the outputs become more important.
- Undue heating of the components may occur.

The influences mentioned above depend on the components used in the individual case.
These possible influences cannot be exactly predicted.

22690

The following applies to the standard side of the device:
This function block may be used at the same input together with the function block **INC_ENCODER** (→ p. 148).

22691

► Initialise the jointly operated function blocks at the same time.

Do **not** use this function block on one input together with one of the following function blocks!

- **FAST_COUNT** (→ p. 142)
- **FREQUENCY** (→ p. 144)
- **FREQUENCY_PERIOD** (→ p. 146)
- **INC_ENCODER_HR** (→ p. 150)
- **PERIOD** (→ p. 152)
- **PERIOD_RATIO** (→ p. 154)

© ifm electronic gmbh www.ifm.com

Parameters of the inputs

19875

Parameter	Data type	Description
INIT	BOOL	TRUE (only for 1 cycle): Function block and interface are initialised FALSE: during further processing of the program
CHANNEL	BYTE	Number of the input channel pair 0 = channel pair 0 = inputs I00 + I01 ... 5 = channel pair 5 = inputs I10 + I11 For the function block xxx_**E** (if available) applies: 0 = channel pair 0 = inputs I00_E + I01_E ... 7 = channel pair 7 = inputs I14_E + I15_E

Parameters of the outputs

363

Parameter	Data type	Description
C	DWORD	period duration of the first input's signal of the channel pair in [µs]
P	INT	angle of the phase shaft valid measurement: 1...358 °
ET	TIME	Time elapsed since the last positive edge at the second pulse input of the channel pair

© ifm electronic gmbh

5.2.11 Function elements: output functions in general

Contents

10462

For this device you can set the mode of some or all outputs. Here we show you a couple of function elements to it.

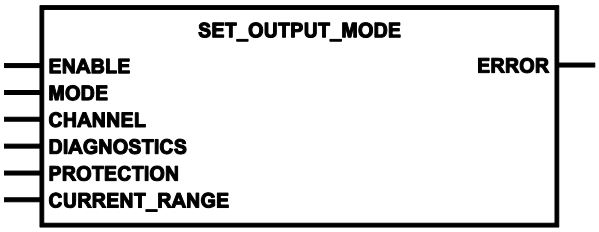
© ifm electronic gmbh

SET_OUTPUT_MODE

15937

Unit type = function block (FB)
Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

12094

SET_OUTPUT_MODE sets the operating mode of the selected output channel.
Allowed operating modes (→ data sheet):

MODE	Config. value	
	hex	dec
---	---	---
OUT_DIGITAL_H (positive)	0001	1
OUT_DIGITAL_L (negative)	0002	2

CURRENT_RANGE	Config. value	
	hex	dec
no current measurement for MODE = 0	00	0
current measurement 2 A (3 A) for MODE = 1 or 2	01	1
current measurement 4 A for MODE = 1 or 2	02	2

The operating mode should not be changed during operation.

15672

NOTE

Is the measuring range for ACTUAL_CURRENT to be changed (to 4 A) at the function block OUTPUT_BRIDGE during operation?

► For both related outputs, during the init phase, call the function block SET_OUTPUT_MODE **before** calling the function block OUTPUT_BRIDGE!
CURRENT_RANGE = 2 (for 4 A)

© ifm electronic gmbh

Parameters of the inputs

12096

Parameter	Data type	Description
ENABLE	BOOL	FALSE ⇨ TRUE (edge): Initialise block (only 1 cycle) > Read block inputs TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
MODE	WORD	Operating mode of output channel CHANNEL: 1 = 0x0001 OUT_DIGITAL_H 2 = 0x0002 OUT_DIGITAL_L
CHANNEL	BYTE	Number of the output channel 0...15 for the outputs Q00...Q15 For the function block xxx_**E** (if available) applies: 0...31 for the outputs Q00_E...Q31_E
DIAGNOSTICS	BOOL	TRUE: Channel with diagnostic function only effective for OUT_DIGITAL_H • Wire break on outputs with current measurement: if output is switched on AND if current < 25 mA for ≥ 66 ms • Wire break on outputs without current measurement: if output is switched off AND if output voltage > 22 % VBBx for ≥ 100 ms • Overload with current measurement: if current > 112.5 % of the measuring range for ≥ 66 ms • Short circuit without current measurement: if VBBx > 7.3 V AND if output voltage < 88 % VBBx for ≥ 66 ms For current measurement: the filter settings Qxx_FILTER influence the diagnosis time FALSE: channel without diagnostic function
PROTECTION	BOOL	TRUE: Protection against overload only for OUT_DIGITAL_H AND output with current measurement If an overload or a short circuit is detected, the output switches off for 1 s, then on again. FALSE: function element is not executed
CURRENT_RANGE	BYTE	Current measuring in the output channel CHANNEL: 0 = 0x00 OUT_CURRENT_RANGE_NONE (off) only for output ... without current measurement or for OUT_DIGITAL_L 1 = 0x01 OUT_CURRENT_RANGE1 2 A / 3 A only for OUT_DIGITAL_H 2 = 0x02 OUT_CURRENT_RANGE2 4 A only for OUT_DIGITAL_H

© ifm electronic gmbh

Parameters of the outputs

12102

Parameter	Data type	Description
ERROR	DWORD	Error code from this function block call → **Error codes** (→ p. 314) (possible messages → following table)

Possible results for ERROR (n=any desired value):

The 32-bit error code consists of four 8-bit values (DWORD).

4th byte	3rd byte	2nd byte	1st byte
Error class	Application-specific error code	Error source	Error cause

Value [hex]	Description
00 00 00 00	no error
01 00 00 F8	Wrong parameter ⇨ general error

© ifm electronic gmbh

5.2.12 Function elements: PWM functions

Contents

13758

Here, you will find **ifm** function blocks that allow you to operate the outputs with Pulse-Width Modulation (PWM).

© ifm electronic gmbh

OUTPUT_BRIDGE

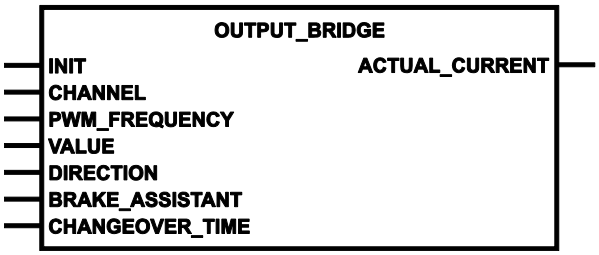
2198

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

For the extended side of the ExtendedControllers the FB name ends with '_E'. (not for CR0133)

Symbol in CODESYS:



Description

19298

OUTPUT_BRIDGE controls the H-bridges on the PWM channels.

By means of this FB the outputs can easily be used as H-bridge. To do so, two successive output channels are combined to one bridge by means of a minus switching driver. If DIRECTION = FALSE, for the first output the plus switching driver is triggered via a PWM signal and the minus switching driver of the second output is switched.

NOTE

When using the H-bridge current control is not supported.

Outputs which are operated in the PWM mode do not support any diagnostic functions and no ERROR flags are set. This is due to the structure of the outputs.

Calling this FB with an output configured as B(L) is not permitted.

An output defined as PWM output can no longer be used as binary output afterwards.

The function overload protection is not active in this mode!
The function overload protection will be resetted by function OUTPUT_BRIDGE.

► Change the parameters only during operation when INIT=FALSE. The new parameters will not be adopted before the current PWM period has elapsed.

Changes of the PWM frequencies during operation:
only permissible in the range 40...2 000 Hz.

If VALUE = 0, output will not be fully deactivated. In principle the output will be active during a timer tick of the PWM timer (typically approx. 50 µs).

► FB must be called in each cycle.

Assignment of the output channels usable as an H-bridge → data sheet.

15672

NOTE

Is the measuring range for ACTUAL_CURRENT to be changed (to 4 A) at the function block OUTPUT_BRIDGE during operation?

► For both related outputs, during the init phase, call the function block SET_OUTPUT_MODE **before** calling the function block OUTPUT_BRIDGE!
CURRENT_RANGE = 2 (for 4 A)

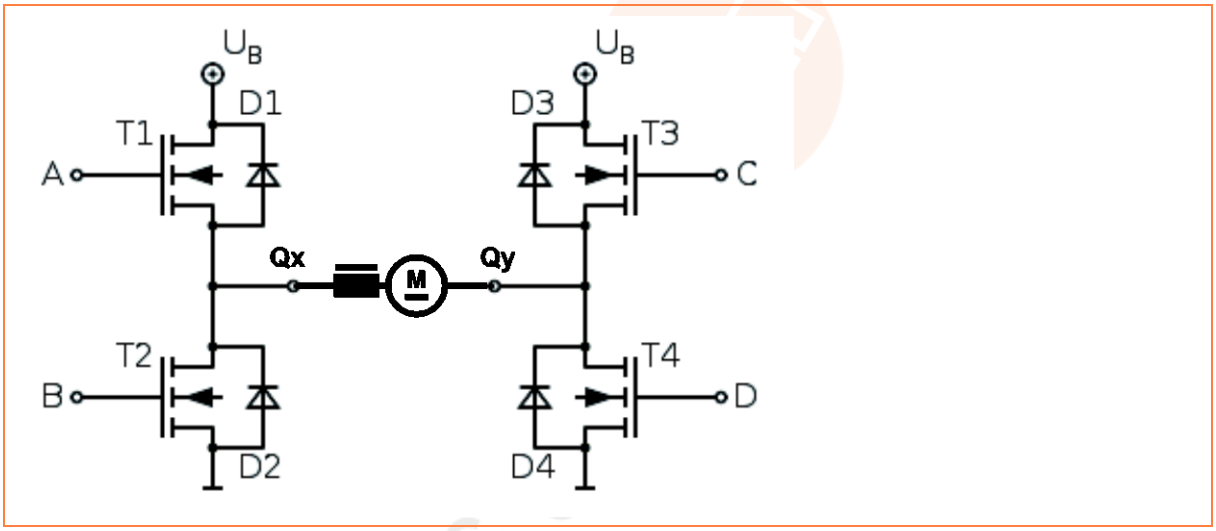
ACTUAL_CURRENT shows the current of the H bridge in [mA].
Measurement method:
• PWM frequency < 100 Hz: averaging over one PWM period
• PWM frequency ≥ 100 Hz: averaging over (PWM frequency/50) PWM periods (rounded down)

Principle of the H-bridge

9990
16411

Here you can see how a h-bridge can be run via PWM outputs at a ifm controller.

Basic circuit of a H-bridge with PWM control:



T1 and T2 together are e.g. output Qx.
Also T3 and T4 belongs together, e.g. for Qy.
Therfore you only need two pins for connecting the DC motor.

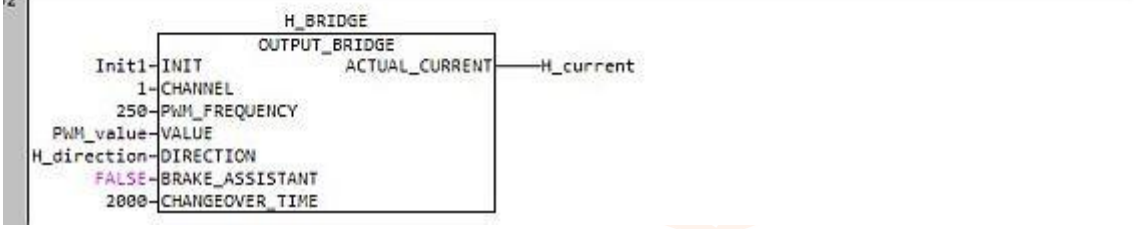
© ifm electronic

Program example:

```
VAR
    Init1: BOOL:=TRUE;
    CycleTime:DWORD;
    MaxCycleTime:DWORD;
    ResetMax:BOOL;

    DownloadID: CAN1_DOWNLOADID;

(*****************************************************************************
CHANNEL = 1:    Motor between OUT01 (Pin17) and OUT03(Pin15)
CHANNEL = 2:    Motor between OUT09 (Pin03) and OUT11(Pin05)
*****************************************************************************)
    H_BRIDGE: OUTPUT_BRIDGE;
    PWM_value: WORD := 100;          (* current PWM value - VALUE = 0...1000 *)
    H_direction: BOOL;               (* TRUE = counter clockwise; FALSE = clockwise *)
    H_current: WORD;                 (* output current in mA *)
    changeover_time: WORD := 500;    (* Space time [ms] during which the motor is not triggered
                                        (> 10 ms) in the case of a change of the rotational direction. *)
END_VAR
```



© ifm electronic gmbh

Parameters of the inputs

2204

Parameter	Data type	Description
INIT	BOOL	TRUE (only for 1 cycle): Function block is initialised FALSE: during further processing of the program
CHANNEL	BYTE	Name of output pair: 1 = bridge 1 at Q01 + Q03 2 = bridge 2 at Q09 + Q11 For the function block xxx_**E** (if available) applies: 1 = bridge 1 at Q01_E + Q03_E 2 = bridge 2 at Q09_E + Q11_E
PWM_FREQUENCY	WORD	PWM frequency [Hz] for load on output > function block limited to value of 20...2 000 = 0x0014...0x07D0 Changes of the PWM frequencies during operation: only permissible in the range 40...2 000 Hz.
VALUE	WORD	PWM value (mark-to-space ratio) in [‰] allowed = 0...1 000 = 0x0000...0x03E8 Values > 1 000 are regarded as = 1 000
DIRECTION	BOOL	Direction of rotation of the motor: TRUE: Counter-clockwise (ccw): Bridge 1: current flow Q01(_E) ⇦ Q03(_E) Bridge 2: current flow Q09(_E) ⇦ Q11(_E) FALSE: Clockwise (cw): Bridge 1: current flow Q01(_E) ⇨ Q03(_E) Bridge 2: current flow Q09(_E) ⇨ Q11(_E)
BRAKE_ASSISTANT	BOOL	TRUE: When changing the rotational direction: the function block switches both outputs to ground to brake the motor as long as CHANGEOVER_TIME is running. FALSE: function element is not executed
CHANGEOVER_TIME	WORD	Space time in [ms] during which the motor is not triggered in the case of a change of the rotational direction (≥ cycle time, at least 10 ms) values < 10 ms work as = 10 ms

Parameters of the outputs

2205

Parameter	Data type	Description
ACTUAL_CURRENT	WORD	Output current in [mA]

© ifm electronic gmbh www.ifm.com

OUTPUT_CURRENT

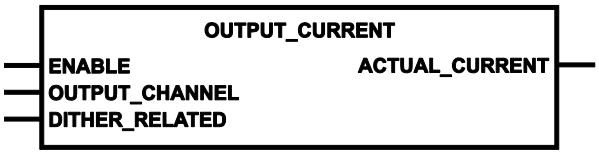
382

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

For the extended side of the ExtendedControllers the FB name ends with '_E'. (not for CR0133)

Symbol in CODESYS:



Description

385

OUTPUT_CURRENT handles the current measurement in conjunction with an active PWM channel.

The FB provides the current output current if the outputs are used as PWM outputs or as plus switching. The current measurement is carried out in the device, i.e. no external measuring resistors are required.

Parameters of the inputs

17894

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
OUTPUT_CHANNEL	BYTE	Number of the current-controlled output channel (0...15) 0...15 for the outputs Q00...Q15 For the function block xxx_**E** (if available) applies: 0...15 for the outputs Q00_E...Q15_E
DITHER_RELATED	BOOL	Current is determined as an average value via... TRUE: one dither period FALSE: one PWM period

Parameters of the outputs

387

Parameter	Data type	Description
ACTUAL_CURRENT	WORD	Output current in [mA]

© ifm electronic gmbh www.ifm.com

OUTPUT_CURRENT_CONTROL

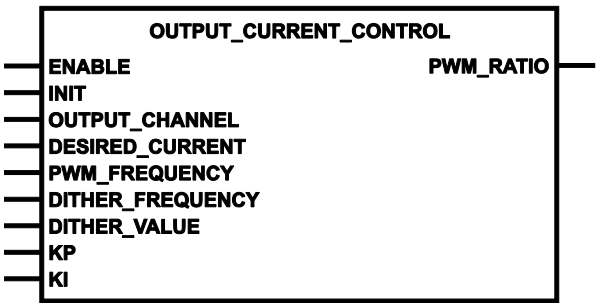
2196

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

For the extended side of the ExtendedControllers the FB name ends with '_E'. (not for CR0133)

Symbol in CODESYS:



Description

2200

OUTPUT_CURRENT_CONTROL operates as current controller for the PWM outputs.

The setting parameters KI and KP represent the I-component and the P-component of the controller. It is recommended to set KI=50 and KP=50 as start values so as to determine the best setting of the controller. Depending on the requested controller behaviour the values can gradually be incremented (controller is stronger / faster) or decremented (controller is weaker / slower).

At the preset value DESIRED_CURRENT=0 the output is controlled down to 0 mA within about 100 ms with the setting parameters being ignored.

Depending on the controller hardware used, a different teach performance has to be noted.

If parameters are changed during running, than the following can occur:
- the control possibly can skip completely or
- the control can need a longer time to tune the given value.

▶ Therefore validate the measured current and restart the control if necessary.

© ifm electronic gmbh www.ifm.com

NOTE

► When defining the parameter DITHER_VALUE make sure that the resulting PWM ratio in the operating range of the loop control remains between 0...1000 %:
  • PWM ratio + DITHER_VALUE < 1000 ‰ and
  • PWM ratio - DITHER_VALUE > 0 ‰.
  Outside this permissible area, DITHER_VALUE is internally reduced to the maximum possible value temporarily, so that the average value of the PWM ratio corresponds to the required value.

> When the dither is activated, changes to PWM_FREQUENCY, DITHER_VALUE and DITHER_FREQUENCY become effective only when the current dither period has been completed.

► Change the parameters only during operation when INIT=FALSE. The new parameters will not be adopted before the current PWM period has elapsed.

► Changes of the PWM frequencies during operation:
  only permissible in the range 40...2 000 Hz.

> If the current indicated in the parameter DESIRED_CURRENT cannot be reached because the PWM ratio is already at 100 %, this is indicated by the system variable ERROR_CONTROL_Qx.

> If KI = 0, there is no loop control.

> If during the loop control a PWM_RATIO = 0 results, the output is not deactivated completely. In principle the output will be active during a timer tick of the PWM timer (typically approx. 50 µs).

► The initialisation of the FB (INIT = TRUE) may only be carried out once per PLC cycle.

► Calling this FB with an output configured as B(L) is not permitted.

► An output defined as PWM output can no longer be used as binary output afterwards.

> If the flowing current in the switched-on condition exceeds the measuring range, control will no longer be possible because the AD converter is at the end of the measuring range and therefore provides wrong values (the max. value).

© ifm electronic gmbh www.ifm.com

Parameters of the inputs

2201

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
INIT	BOOL	TRUE (only for 1 cycle): Function block is initialised FALSE: during further processing of the program
OUTPUT_CHANNEL	BYTE	Number of the current-controlled output channel (0...15) 0...15 for the outputs Q00...Q15 For the function block xxx_**E** (if available) applies: 0...15 for the outputs Q00_E...Q15_E
DESIRED_CURRENT	WORD	Desired current value of the output in [mA] allowed = 0...2 000 / 0...4 000 (dependent on output and configuration)
PWM_FREQUENCY	WORD	PWM frequency [Hz] for load on output > function block limited to value of 20...2 000 = 0x0014...0x07D0 Changes of the PWM frequencies during operation: only permissible in the range 40...2 000 Hz.
DITHER_FREQUENCY	WORD	dither frequency in [Hz] value range = 0...FREQUENCY / 2 FREQUENCY / DITHER_FREQUENCY must be even-numbered! The FB increases all other values to the next matching value.
DITHER_VALUE	WORD	peak-to-peak value of the dither in [‰] permissible values = 0...1 000 = 0000...03E8
KP	BYTE	proportional component of the output signal
KI	BYTE	Integral component of the output signal if KI = 0 no rule

Parameters of the outputs

2202

Parameter	Data type	Description
PWM_RATIO	WORD	For monitoring purposes: display PWM pulse ratio 0...999 %

© ifm electronic gmbh www.ifm.com

PWM1000

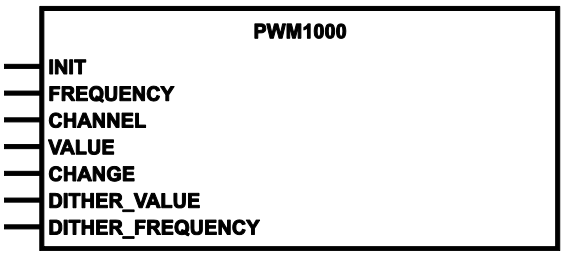
326

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

For the extended side of the ExtendedControllers the FB name ends with '_E'. (not for CR0133)

Symbol in CODESYS:



Description

2311

PWM1000 handles the initialisation and parameter setting of the PWM outputs.

The FB enables a simple use of the PWM function in the device. For each channel an own PWM frequency, the mark-to-period ratio and the dither can be set.

The PWM frequency FREQUENCY can be directly indicated in [Hz] and the mark-to-period ratio VALUE in steps of 1 ‰.

If VALUE = 0, output will not be fully deactivated. In principle the output will be active during a timer tick of the PWM timer (typically approx. 50 µs).

- ► When defining the parameter DITHER_VALUE make sure that the resulting PWM ratio in the operating range of the loop control remains between 0...1000 %:
  - PWM ratio + DITHER_VALUE < 1000 ‰ and
  - PWM ratio - DITHER_VALUE > 0 ‰.

  Outside this permissible area, DITHER_VALUE is internally reduced to the maximum possible value temporarily, so that the average value of the PWM ratio corresponds to the required value.
- ► Activate the function block permanently!
- ► Calling this FB with an output configured as B(L) is not permitted.

> **NOTE**
>
> The function change of a channel defined as PWM function during operation is not possible. The PWM function remains set until a hardware reset is carried out on the controller ⇨ power off and on again.
>
> For high PWM frequencies differences can occur between the set ratio and the ratio on the output due to the system.
>
> - ► Change the parameters only during operation when INIT=FALSE. The new parameters will not be adopted before the current PWM period has elapsed.
> - ► Changes of the PWM frequencies during operation: only permissible in the range 40...2 000 Hz.

Changes during the runtime:

Always when the input CHANGE is set to TRUE, the FB adopts the value ...
- FREQUENCY after the current PWM period
- VALUE after the current PWM period
- DITHER_VALUE after the current dither period
- DITHER_FREQUENCY after the current dither period

© ifm electronic gmbh www.ifm.com

Parameters of the inputs

2312

Parameter	Data type	Description
INIT	BOOL	TRUE (only 1 cycle): Function block is initialised Adopting new value from FREQUENCY FALSE: during further processing of the program
FREQUENCY	WORD	PWM frequency in [Hz] > function block limited to value of 20...2 000 = 0x0014...0x07D0 Changes of the PWM frequencies during operation: only permissible in the range 40...2 000 Hz.
CHANNEL	BYTE	Number of the PWM output channel 0...15 for the outputs Q00...Q15 For the function block xxx_**E** (if available) applies: 0...15 for the outputs Q00_E...Q15_E
VALUE	WORD	PWM value (mark-to-space ratio) in [‰] allowed = 0...1 000 = 0x0000...0x03E8 Values > 1 000 are regarded as = 1 000
CHANGE	BOOL	TRUE: adoption of the new value of ... • FREQUENCY: after the current PWM period • VALUE: after the current PWM period • DITHER_VALUE: after the current dither period • DITHER_FREQUENCY: after the current dither period FALSE: the changed PWM value has no influence on the output
DITHER_VALUE	WORD	peak-to-peak value of the dither in [‰] permissible values = 0...1 000 = 0000...03E8
DITHER_FREQUENCY	WORD	dither frequency in [Hz] value range = 0...FREQUENCY / 2 FREQUENCY / DITHER_FREQUENCY must be even-numbered! The FB increases all other values to the next matching value.

© ifm electronic gmbh www.ifm.com

5.2.13 Function elements: hydraulic control

Contents

13760

The library `ifm_HYDRAULIC_32bit_Vxxyyzz.Lib` contains the following function blocks:

CONTROL_OCC (→ p. 174)	OCC = **O**utput **C**urrent **C**ontrol Scales the input value [WORD] to an indicated current range
JOYSTICK_0 (→ p. 176)	Scales signals [INT] from a joystick to clearly defined characteristic curves, standardised to 0... 1000
JOYSTICK_1 (→ p. 179)	Scales signals [INT] from a joystick D standardised to 0... 1000
JOYSTICK_2 (→ p. 183)	Scales signals [INT] from a joystick to a configurable characteristic curve; free selection of the standardisation
NORM_HYDRAULIC (→ p. 186)	Normalises a value [DINT] within defined limits to a value with new limits

The following function blocks are needed from the library `UTIL.Lib` (in the CODESYS package):
- RAMP_INT
- CHARCURVE

These function blocks are automatically called and configured by the function blocks of the hydraulics library.

The following function blocks are needed from the library:`ifm_CR0033_Vxxyyzz.LIB`

OUTPUT_CURRENT (→ p. 167)	Measures the current (average via dither period) on an output channel
OUTPUT_CURRENT_CONTROL (→ p. 168)	Current controller for a PWMi output channel

These function blocks are automatically called and configured by the function blocks of the hydraulics library.

© ifm electronic gmbh

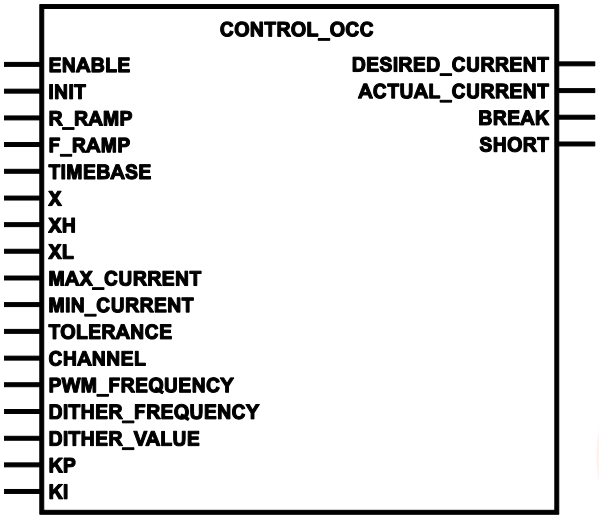
CONTROL_OCC

2735

Unit type = function block (FB)

Unit is contained in the library `ifm_HYDRAULIC_32bit_Vxxyyzz.Lib`

Symbol in CODESYS:



Description

2737

CONTROL_OCC scales the input value X to a specified current range.

Each instance of the FB is called once in each PLC cycle.

This function block uses the following function blocks from the library:`ifm_CR0033_Vxxyyzz.LIB`

- **OUTPUT_CURRENT_CONTROL** (→ p. 168)
- **OUTPUT_CURRENT** (→ p. 167)

The controller controls on the basis of the cycle period of the PWM signal.

The setting parameters KI and KP represent the **i**ntegral and the **p**roportional component of the controller. It is recommended to set KI=50 and KP=50 as start values so as to determine the best setting of the controller.

► Increasing the values for KI and / or KP: ⇨ controller becomes more sensitive / faster
Decreasing the values for KI and / or KP: ⇨ controller becomes less sensitive / slower

> At the output DESIRED_CURRENT=0 the output is **immediately** switched to 0 mA and is **not** adjusted downward to 0 mA in accordance with the set parameters.

The controller has a fast compensation mechanism for voltage drops of the supply voltage. In addition to the controller behaviour of the controller and on the basis of the voltage drop, the ratio of the PWM is increased such that the controller reaches as quickly as possible the desired value.

ⓘ The input X of CONTROL_OCC should be supplied by the output of the JOYSTICK FBs.

© ifm electronic gmbh

Parameters of the inputs

2739

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
INIT	BOOL	FALSE ⇨ TRUE (edge): unit is initialised FALSE: during further processing of the program
R_RAMP	INT	Rising edge of the ramp in [increments/PLC cycle] 0 = no ramp
F_RAMP	INT	Falling edge of the ramp in [increments/PLC cycle] 0 = no ramp
TIMEBASE	TIME	Reference for rising and falling edge of the ramp: t#0s = rising/falling edge in [increments/PLC cycle] Fast controllers have very short cycle times! otherwise = rising/falling edge in [increments/TIMEBASE]
X	WORD	input value
XH	WORD	Upper limit of input value range [increments]
XL	WORD	Lower limit of input value range [increments]
MAX_CURRENT	WORD	Max. valve current in [mA]
MIN_CURRENT	WORD	Min. valve current in [mA]
TOLERANCE	BYTE	Tolerance for min. valve current in [increments] When the tolerance is exceeded, jump to MIN_CURRENT is effected
CHANNEL	BYTE	Number of the current-controlled output channel 0...15 for the outputs Q00...Q15 For the function block xxx_**E** (if available) applies: 0...15 for the outputs Q00_E...Q15_E
PWM_FREQUENCY	WORD	PWM frequency [Hz] for load on input
DITHER_FREQUENCY	WORD	dither frequency in [Hz] value range = 0...FREQUENCY / 2 FREQUENCY / DITHER_FREQUENCY must be even-numbered! The FB increases all other values to the next matching value.
DITHER_VALUE	BYTE	peak-to-peak value of the dither in [%] permissible values = 0...100 = 0x00...0x64
KP	BYTE	proportional component of the output signal
KI	BYTE	integral component of the output signal

For KP, KI applies: recommended start value = 50

Parameters of the outputs

602

Parameter	Data type	Description
DESIRED_CURRENT	WORD	Desired current value in [mA] for OCC (for monitoring purposes)
ACTUAL_CURRENT	WORD	Output current in [mA]
BREAK	BOOL	Error: cable interrupted at output
SHORT	BOOL	Error: short circuit in cable at output

© ifm electronic gmbh

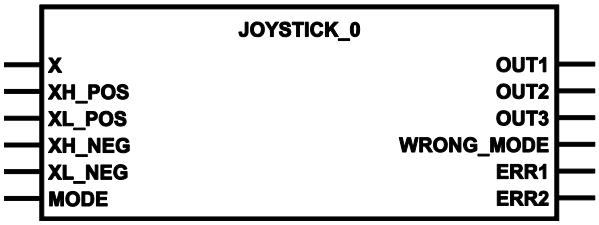
JOYSTICK_0

6250

Unit type = function block (FB)

Unit is contained in the library `ifm_hydraulic_32bit_Vxxyyzz.Lib`

Symbol in CODESYS:



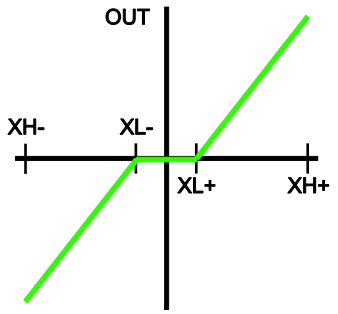
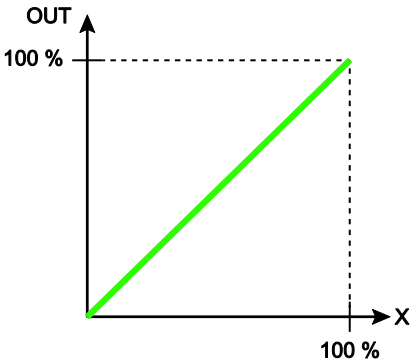
Description

432

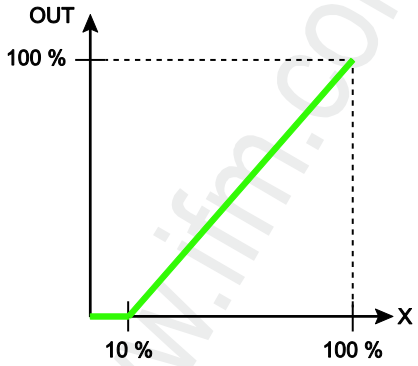
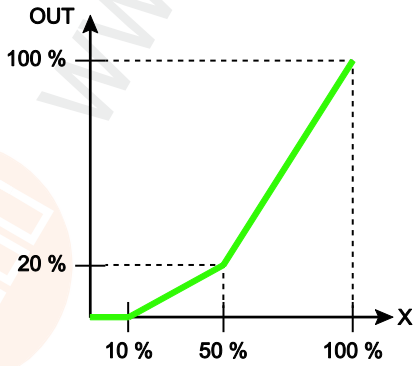
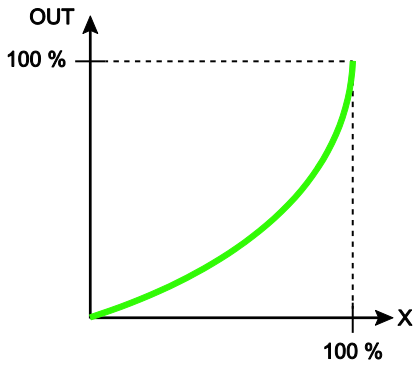
JOYSTICK_0 scales signals from a joystick to clearly defined characteristic curves, standardised to 0...1000.

For this FB the characteristic curve values are specified (→ figures):

- Rising edge of the ramp = 5 increments/PLC cycle
  Fast Controllers have a very short cycle time!
- Falling edge of the ramp = no ramp

The parameters XL_POS (XL+), XH_POS (XH+), XL_NEG (XL-) and XH_NEG (XH-) are used to evaluate the joystick movements only in the requested area. The values for the positive and negative area may be different. The values for XL_NEG and XH_NEG are negative here.	
Mode 0: characteristic curve linear for the range XL to XH	

© ifm electronic gmbh www.ifm.com

Mode 1: Characteristic curve linear with dead band Values fixed to: Dead band: 0…10% of 1000 increments	
Mode 2: 2-step linear characteristic curve with dead band Values fixed to: Dead band: 0…10% of 1000 increments Step: X = 50 % of 1000 increments Y = 20 % of 1000 increments	
Characteristic curve mode 3: Curve rising (line is fixed)	

© ifm electronic gmbh

Parameters of the inputs

433

Parameter	Data type	Description
X	INT	Input value [increments]
XH_POS	INT	Max. preset value positive direction [increments] (negative values also permissible)
XL_POS	INT	Min. preset value positive direction [increments] (negative values also permissible)
XH_NEG	INT	Max. preset value negative direction [increments] (negative values also permissible)
XL_NEG	INT	Min. preset value negative direction [increments] (negative values also permissible)
MODE	BYTE	Mode selection characteristic curve: 0 = linear (X|OUT = 0|0 ... 1000|1000) 1 = linear with dead band (X|OUT = 0|0 ... 100|0 ... 1000|1000) 2 = 2-step linear with dead band (X|OUT = 0|0 ... 100|0 ... 500|200 ... 1000|1000) 3 = curve rising (line is fixed)

Parameters of the outputs

6252

Parameter	Data type	Description
OUT1	WORD	Standardised output value: 0…1000 increments e.g. for valve left
OUT2	WORD	Standardised output value: 0…1000 increments e.g. for valve right
OUT3	INT	Standardised output value -1000…0…1000 increments e.g. for valve on output module (e.g. CR2011 or CR2031)
WRONG_MODE	BOOL	Error: invalid mode
ERR1	BYTE	Error code for rising edge (referred to the internally used function blocks CHARCURVE and RAMP_INT from `util.lib`) (possible messages → following table)
ERR2	BYTE	Error code for falling edge (referred to the internally used function blocks CHARCURVE and RAMP_INT from `util.lib`) (possible messages → following table)

Possible results for ERR1 and ERR2:

Value dec	Value hex	Description
0	00	no error
1	01	Error in array: wrong sequence
2	02	Error: Input value IN is not contained in value range of array
4	04	Error: invalid number N for array

© ifm electronic gmbh www.ifm.com

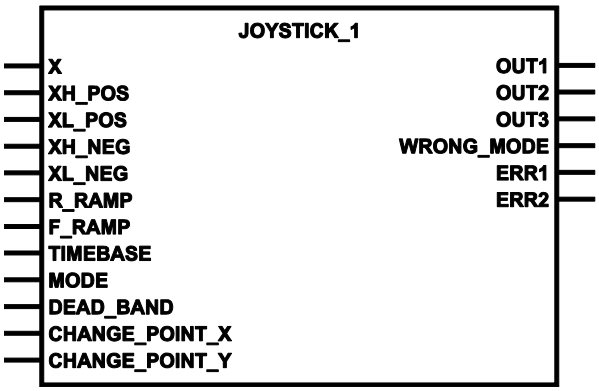
JOYSTICK_1

6255

Unit type = function block (FB)

Unit is contained in the library `ifm_hydraulic_32bit_Vxxyyzz.Lib`

Symbol in CODESYS:

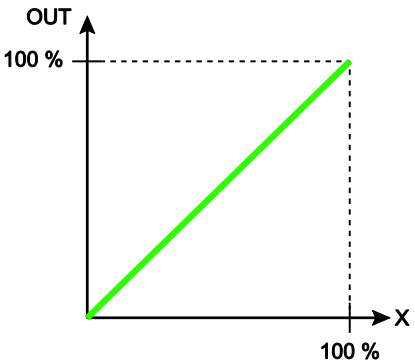
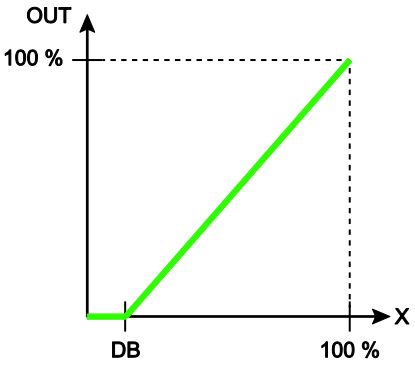


Description

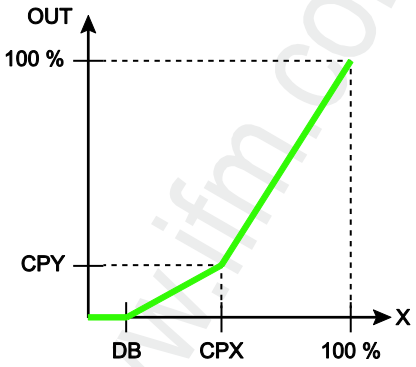
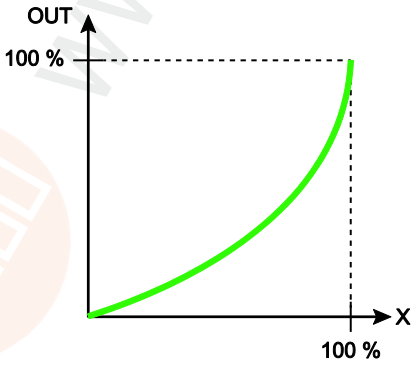
425

JOYSTICK_1 scales signals from a joystick to configurable characteristic curves, standardised to 0...1000.

For this FB the characteristic curve values can be configured (→ figures):

Mode 0: Linear characteristic curve 100 % = 1000 increments	
Mode 1: Characteristic curve linear with dead band Value for the dead band (DB) can be set in % of 1000 increments 100 % = 1000 increments DB = Dead_Band	

© ifm electronic gmbh

Mode 2: 2-step linear characteristic curve with dead band Values can be configured to: Dead band: 0…DB in % of 1000 increments Step: X = CPX in % of 1000 increments Y= CPY in % of 1000 increments 100 % = 1000 increments DB = Dead_Band CPX = Change_Point_X CPY = Change_Point_Y	
Characteristic curve mode 3: Curve rising (line is fixed)	

© ifm electronic gmbh

Parameters of the inputs

6256

Parameter	Data type	Description
X	INT	Input value [increments]
XH_POS	INT	Max. preset value positive direction [increments] (negative values also permissible)
XL_POS	INT	Min. preset value positive direction [increments] (negative values also permissible)
XH_NEG	INT	Max. preset value negative direction [increments] (negative values also permissible)
XL_NEG	INT	Min. preset value negative direction [increments] (negative values also permissible)
R_RAMP	INT	Rising edge of the ramp in [increments/PLC cycle] 0 = no ramp
F_RAMP	INT	Falling edge of the ramp in [increments/PLC cycle] 0 = no ramp
TIMEBASE	TIME	Reference for rising and falling edge of the ramp: t#0s = rising/falling edge in [increments/PLC cycle] Fast controllers have very short cycle times! otherwise = rising/falling edge in [increments/TIMEBASE]
MODE	BYTE	Mode selection characteristic curve: 0 = linear (X|OUT = 0|0 ... 1000|1000) 1 = linear with dead band (X|OUT = 0|0 ... DB|0 ... 1000|1000) 2 = 2-step linear with dead band (X|OUT = 0|0 ... DB|0 ... CPX|CPY ... 1000|1000) 3 = curve rising (line is fixed)
DEAD_BAND	BYTE	Adjustable dead band in [% of 1000 increments]
CHANGE_POINT_X	BYTE	For mode 2: ramp step, value for X in [% of 1000 increments]
CHANGE_POINT_Y	BYTE	For mode 2: ramp step, value for Y in [% of 1000 increments]

© ifm electronic gmbh www.ifm.com

Parameters of the outputs

6252

Parameter	Data type	Description
OUT1	WORD	Standardised output value: 0…1000 increments e.g. for valve left
OUT2	WORD	Standardised output value: 0…1000 increments e.g. for valve right
OUT3	INT	Standardised output value -1000…0…1000 increments e.g. for valve on output module (e.g. CR2011 or CR2031)
WRONG_MODE	BOOL	Error: invalid mode
ERR1	BYTE	Error code for rising edge (referred to the internally used function blocks CHARCURVE and RAMP_INT from `util.lib`) (possible messages → following table)
ERR2	BYTE	Error code for falling edge (referred to the internally used function blocks CHARCURVE and RAMP_INT from `util.lib`) (possible messages → following table)

Possible results for ERR1 and ERR2:

Value dec	Value hex	Description
0	00	no error
1	01	Error in array: wrong sequence
2	02	Error: Input value IN is not contained in value range of array
4	04	Error: invalid number N for array

© ifm electronic gmbh www.ifm.com

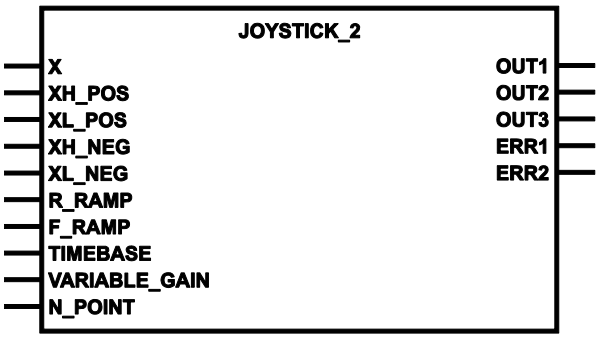
JOYSTICK_2

6258

Unit type = function block (FB)

Unit is contained in the library `ifm_hydraulic_32bit_Vxxyyzz.Lib`

Symbol in CODESYS:

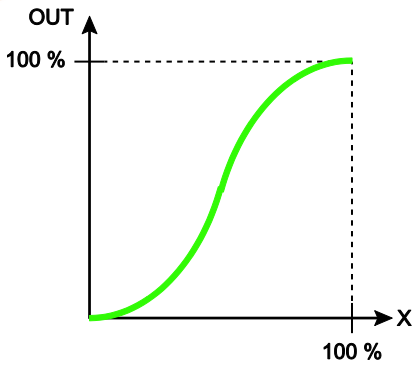


Description

418

JOYSTICK_2 scales the signals from a joystick to a configurable characteristic curve. Free selection of the standardisation.

For this FB, the characteristic curve is freely configurable (→ figure):

Characteristic curve freely configurable	

© ifm electronic gmbh

Parameters of the inputs

6261

Parameter	Data type	Description
X	INT	Input value [increments]
XH_POS	INT	Max. preset value positive direction [increments] (negative values also permissible)
XL_POS	INT	Min. preset value positive direction [increments] (negative values also permissible)
XH_NEG	INT	Max. preset value negative direction [increments] (negative values also permissible)
XL_NEG	INT	Min. preset value negative direction [increments] (negative values also permissible)
R_RAMP	INT	Rising edge of the ramp in [increments/PLC cycle] 0 = no ramp
F_RAMP	INT	Falling edge of the ramp in [increments/PLC cycle] 0 = no ramp
TIMEBASE	TIME	Reference for rising and falling edge of the ramp: t#0s = rising/falling edge in [increments/PLC cycle] Fast controllers have very short cycle times! otherwise = rising/falling edge in [increments/TIMEBASE]
VARIABLE_GAIN	ARRAY [0..10] OF POINT	Pairs of values describing the curve The first pairs of values indicated in N_POINT are used. n = 2…11 **Example:** 9 pairs of values declared as variable VALUES: `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);` There may be blanks between the values.
N_POINT	BYTE	Number of points (pairs of values in VARIABLE_GAIN) by which the curve characteristic is defined: n = 2…11

© ifm electronic gmbh www.ifm.com

Parameters of the outputs

420

Parameter	Data type	Description
OUT1	WORD	Standardised output value: 0…1000 increments e.g. for valve left
OUT2	WORD	Standardised output value: 0…1000 increments e.g. for valve right
OUT3	INT	Standardised output value -1000…0…1000 increments e.g. for valve on output module (e.g. CR2011 or CR2031)
ERR1	BYTE	Error code for rising edge (referred to the internally used function blocks CHARCURVE and RAMP_INT from `util.lib`) (possible messages → following table)
ERR2	BYTE	Error code for falling edge (referred to the internally used function blocks CHARCURVE and RAMP_INT from `util.lib`) (possible messages → following table)

Possible results for ERR1 and ERR2:

Value dec	Value hex	Description
0	00	no error
1	01	Error in array: wrong sequence
2	02	Error: Input value IN is not contained in value range of array
4	04	Error: invalid number N for array

© ifm electronic gmbh

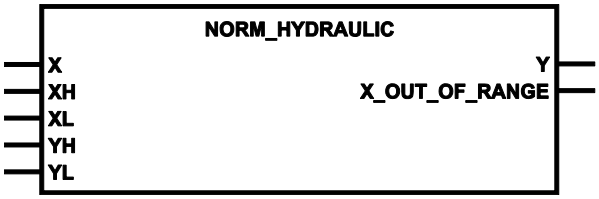
NORM_HYDRAULIC

394

Unit type = function block (FB)

Unit is contained in the library `ifm_hydraulic_32bit_Vxxyyzz.Lib`

Symbol in CODESYS:



Description

397

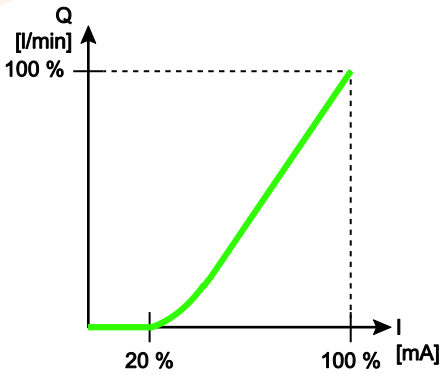
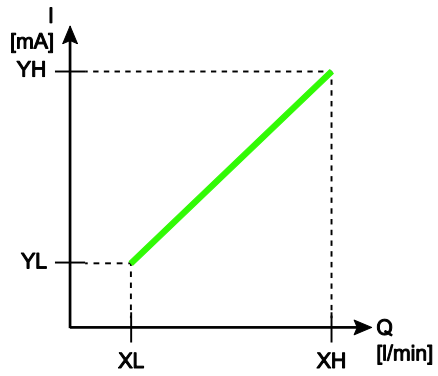
NORM_HYDRAULIC standardises input values with fixed limits to values with new limits.

This function block corresponds to NORM_DINT from the CODESYS library `UTIL.Lib`.

The function block standardises a value of type DINT, which is within the limits of XH and XL, to an output value within the limits of YH and YL.

Due to rounding errors deviations from the standardised value of 1 may occur. If the limits (XH/XL or YH/YL) are indicated in inversed form, standardisation is also inverted.

If X is outside the limits of XL…XH, the error message will be X_OUT_OF_RANGE = TRUE.

Typical characteristic curve of a hydraulic valve: The oil flow will not start before 20% of the coil current has been reached. At first the oil flow is not linear.	
Characteristics of the function block	

© ifm electronic gmbh www.ifm.com

Parameters of the inputs

398

Parameter	Data type	Description
X	DINT	current input value
XH	DINT	Max. input value [increments]
XL	DINT	Min. input value [increments]
YH	DINT	Max. output value [increments], e.g.: valve current [mA] / flow [l/min]
YL	DINT	Min. output value [increments], e.g.: valve current [mA], flow [l/min]

Parameters of the outputs

399

Parameter	Data type	Description
Y	DINT	output value
X_OUT_OF_RANGE	BOOL	Error: X is beyond the limits of XH and XL

Example: NORM_HYDRAULIC

400

Parameter	Case 1	Case 2	Case 3
Upper limit value input XH	100	100	2000
Lower limit value input XL	0	0	0
Upper limit value output YH	2000	0	100
Lower limit value output YL	0	2000	0
Non standardised value X	20	20	20
Standardised value Y	400	1600	1

- Case 1:
  Input with relatively coarse resolution.
  Output with high resolution.
  1 X increment results in 20 Y increments.
- Case 2:
  Input with relatively coarse resolution.
  Output with high resolution.
  1 X increment results in 20 Y increments.
  Output signal is inverted as compared to the input signal.
- Case 3:
  Input with high resolution.
  Output with relatively coarse resolution.
  20 X increments result in 1 Y increment.

© ifm electronic gmbh www.ifm.com

5.2.14 Function elements: controllers

Contents

1634

The section below describes in detail the units that are provided for set-up by software controllers in the ecomatmobile device. The units can also be used as basis for the development of your own control functions.

Setting rule for a controller

1627

For controlled systems, whose time constants are unknown the setting procedure to Ziegler and Nickols in a closed control loop is of advantage.

Setting control

1628

At the beginning the controlling system is operated as a purely P-controlling system. In this respect the derivative time T_V is set to 0 and the reset time T_N to a very high value (ideally to ∞) for a slow system. For a fast controlled system a small T_N should be selected.

Afterwards the gain KP is increased until the control deviation and the adjustment deviation perform steady oscillation at a constant amplitude at KP = $KP_{critical}$. Then the stability limit has been reached.

Then the time period $T_{critical}$ of the steady oscillation has to be determined.

Add a differential component only if necessary.

T_V should be approx. 2...10 times smaller than T_N.

KP should be equal to KD.

Idealised setting of the controlled system:

Control unit	KP = KD	TN	TV
P	2.0 • $KP_{critical}$	—	—
PI	2.2 • $KP_{critical}$	0.83 • $T_{critical}$	—
PID	1.7 • $KP_{critical}$	0.50 • $T_{critical}$	0.125 • $T_{critical}$

For this setting process it has to be noted that the controlled system is not harmed by the oscillation generated. For sensitive controlled systems KP must only be increased to a value at which no oscillation occurs.

Damping of overshoot

1629

To dampen overshoot **PT1** (→ p. 194) (low pass) can be used. In this respect the preset value XS is damped by the PT1 link before it is supplied to the controller function.

The setting variable T1 should be approx. 4...5 times greater than TN of the controller.

© ifm electronic gmbh

DELAY

585

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

588

DELAY delays the output of the input value by the time T (dead-time element).

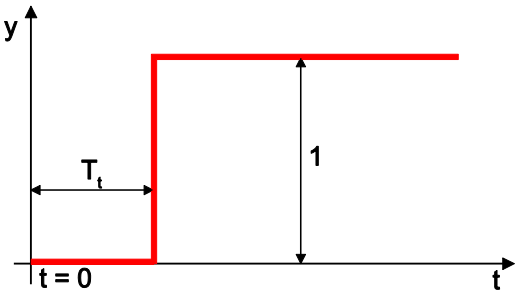


Figure: Time characteristics of DELAY

The dead time is influenced by the duration of the PLC cycle.
The dead time my not exceed 100 • PLC cycle time (memory limit!).
In case a longer delay is set, the resolution of the values at the output of the FB will be poorer, which may cause that short value changes will be lost.

To ensure that the FB works correctly: FB must be called in each cycle.

Parameters of the inputs

2615

Parameter	Data type	Description
X	REAL	Input value
T	TIME	Delay time (dead time) allowed: 0...100 • cycle time

Parameters of the outputs

2616

Parameter	Data type	Description
Y	REAL	Input value, delayed by the time T

© ifm electronic gmbh

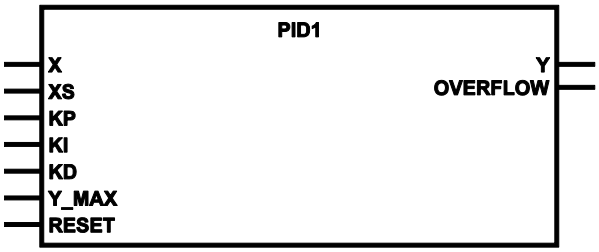
PID1

19235

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

19237

PID1 handles a PID controller.

The change of the manipulated variable of a PID controller has a **p**roportional, **i**ntegral and **d**ifferential component.

OVERFLOW = TRUE is signalled when the 'I' part reaches an internal limitation because a control deviation could not be corrected.
OVERFLOW remains TRUE as long as the limitation is active.

Parameters of the inputs

19238

Parameter	Data type	Description
X	REAL	Input value
XS	REAL	preset value
KP	REAL	Proportional component of the output signal (only positive values permissible)
KI	REAL	Integral component of the output signal (only positive values permissible)
KD	REAL	Differential component of the output signal (only positive values permissible)
Y_MAX	REAL	maximum control value
RESET	BOOL	TRUE: reset the function element FALSE: function element is not executed

Parameters of the outputs

19241

Parameter	Data type	Description
Y	REAL	Output value
OVERFLOW	BOOL	TRUE: Overflow of the data buffer ⇨ loss of data! FALSE: Data buffer is without data loss

© ifm electronic gmbh
www.ifm.com

Recommended settings

19242

- ► Start values:
  KP = 0
  KD = 0
- ► Adapt KI to the process.
- ► Then modify KP and KI gradually.

© ifm electronic gmbh www.ifm.com

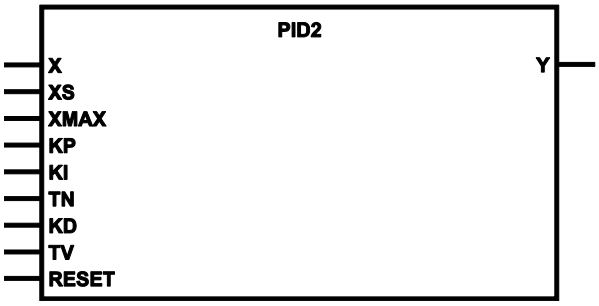
PID2

344

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

6262

PID2 handles a PID controller.

The change of the manipulated variable of a PID controller has a **p**roportional, **i**ntegral and **d**ifferential component. The manipulated variable changes first by an amount which depends on the rate of change of the input value (D component). After the end of the derivative action time TV the manipulated variable returns to the value corresponding to the proportional component and changes in accordance with the reset time TN.

The manipulated variable Y is already standardised to **PWM1000** (→ p. 171).

Rules:

- Negative values for KP, KI and KD are not permitted.
- In case of TN = 0, the I value is not calculated
- In case of XS > XMAX, XS is limited to XMAX.
- In case of X > XMAX, Y is set to 0.
- If X > XS, the manipulated variable is increased.
- If X < XS, the manipulated variable is reduced.

A reference variable is internally added to the manipulated variable.
Y = Y + 65 536 – (XS / XMAX • 65 536).

The manipulated variable Y has the following time characteristics.

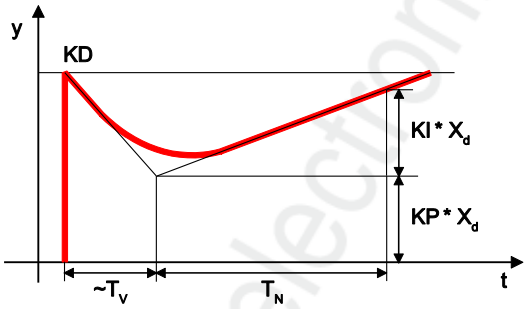


Figure: Typical step response of a PID controller

© ifm electronic gmbh
www.ifm.com

Parameters of the inputs

12963

Parameter	Data type	Description
X	WORD	input value
XS	WORD	preset value
XMAX	WORD	maximum preset value
KP	REAL	Proportional component of the output signal (only positive values permissible)
KI	REAL	Integral component of the output signal (only positive values permissible)
TN	TIME	integral action time (integral component)
KD	REAL	Differential component of the output signal (only positive values permissible)
TV	TIME	derivative action time (differential component)
RESET	BOOL	TRUE: reset the function element FALSE: function element is not executed

Parameters of the outputs

349

Parameter	Data type	Description
Y	WORD	Manipulated variable (0...1000 ‰)

Recommended setting

350

- ► Select TN according to the time characteristics of the system:
  fast system = small TN
  slow system = large TN
- ► Slowly increment KP gradually, up to a value at which still definitely no fluctuation will occur.
- ► Readjust TN if necessary.
- ► Add **d**ifferential component only if necessary:
  Select a TV value approx. 2...10 times smaller than TN.
  Select a KD value more or less similar to KP.

Note that the maximum control deviation is + 127. For good control characteristics this range should not be exceeded, but it should be exploited to the best possible extent.

© ifm electronic gmbh www.ifm.com

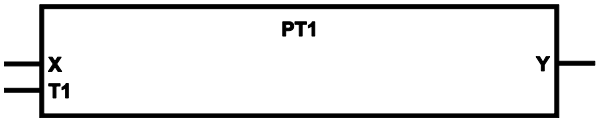
PT1

338

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

341

PT1 handles a controlled system with a first-order time delay.

This FB is a proportional controlled system with a time delay. It is for example used for generating ramps when using the PWM FBs.

> The output of the FB can become instable if T1 is shorter than the SPS cycle time.

The output variable Y of the low-pass filter has the following time characteristics (unit step):

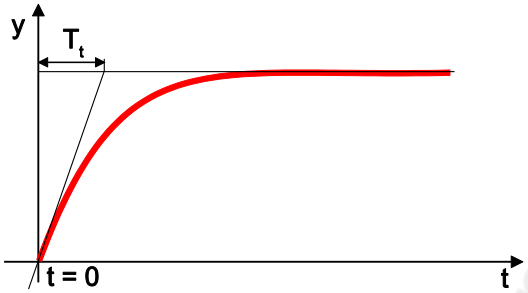


Figure: Time characteristics of PT1

Parameters of the inputs

2618

Parameter	Data type	Description
X	DINT	current input value
T1	TIME	Delay time (time constant)

Parameters of the outputs

2619

Parameter	Data type	Description
Y	DINT	output value

© ifm electronic gmbh

5.2.15 Function elements: software reset

Contents

1594

Using this FB the control can be restarted via an order in the application program.

© ifm electronic gmbh

SOFTRESET

260

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

263

SOFTRESET leads to a complete reboot of the device.

The FB can for example be used in conjunction with CANopen if a node reset is to be carried out. FB SOFTRESET executes an immediate reboot of the controller. The current cycle is not completed.

Before reboot, the retain variables are stored.
The reboot is logged in the error memory.

In case of active communication: the long reset period must be taken into account because otherwise guarding errors will be signalled.

Parameters of the inputs

264

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified

© ifm electronic gmbh

5.2.16 Function elements: measuring / setting of time

Contents

1601

Using the following function blocks of ifm electronic you can...
- measure time and evaluate it in the application program,
- change time values, if required.

© ifm electronic gmbh

TIMER_READ

236

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

239

TIMER_READ reads the current system time.

When the supply voltage is applied, the device generates a clock pulse which is counted upwards in a register. This register can be read using the FB call and can for example be used for time measurement.

The system timer goes up to 0xFFFF FFFF at the maximum (corresponds to 49d 17h 2min 47s 295ms) and then starts again from 0.

Parameters of the outputs

241

Parameter	Data type	Description
T	TIME	Current system time [ms]

© ifm electronic gmbh www.ifm.com

TIMER_READ_US

657

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

660

TIMER_READ_US reads the current system time in [µs].

When the supply voltage is applied, the device generates a clock pulse which is counted upwards in a register. This register can be read by means of the FB call and can for example be used for time measurement.

Info

The system timer runs up to the counter value 4 294 967 295 µs at the maximum and then starts again from 0.

4 294 967 295 µs = 1h 11min 34s 967ms 295µs

Parameters of the outputs

662

Parameter	Data type	Description
TIME_US	DWORD	current system time [µs]

© ifm electronic gmbh

5.2.17 Function elements: device temperature

Contents

2364

© ifm electronic gmbh
www.ifm.com

TEMPERATURE

2216

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

2365

TEMPERATURE reads the current temperature in the device.

The FB can be called cyclically and indicates the current device temperature (-40...125 °C) on its output.

Parameters of the inputs

2366

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified

Parameters of the outputs

2367

Parameter	Data type	Description
TEMPERATURE	INT	Current internal temperature of the device [°C]

© ifm electronic gmbh

5.2.18 Function elements: saving, reading and converting data in the memory

Contents

13795

Storage types for data backup

13805

The device provides the following memory types:

Flash memory

13803

Properties:
- non-volatile memory
- writing is relatively slow and only block by block
- before re-writing, memory content must be deleted
- fast reading
- limited writing and reading frequency
- really useful only for storing large data quantities
- saving data with FLASHWRITE
- reading data with FLASHREAD

FRAM memory

13802

FRAM indicates here all kinds of non-volatile and fast memories.

Properties:
- fast writing and reading
- unlimited writing and reading frequency
- any memory area can be selected
- saving data with FRAMWRITE
- reading data with FRAMREAD

© ifm electronic gmbh

File system

2690

The file system coordinates the storage of the information in the memory. The size of the file system is 128 kbytes.

The file names of the data system are limited:
max. length for Controller: CR0n3n, CR7n3n: 15 characters
max. for all other units: 11 characters

Behaviour of the file system in the Controller: CR0n3n, CR7n3n**:**

- The controller always tries to write the file, even if the same file name already exists. The file might be saved several times. Only the current file is used. Via the download (see below) this multiple filing can be prevented.
- Individual files cannot be overwritten or deleted.
- The file system is completely deleted during each download (boot project download or RAM download). Then e.g. a symbol file or a project file (FBs in CODESYS) can be written.
- The file system is also deleted during a [Reset (Original)] (CODESYS function in the menu [Online]).

© ifm electronic gmbh

Automatic data backup

Contents

14168
2347

The **ecomatmobile** controllers allow to save data (BOOL, BYTE, WORD, DWORD) non-volatilely (= saved in case of voltage failure) in the memory. If the supply voltage drops, the backup operation is automatically started. Therefore it is necessary that the data is defined as RETAIN variables (→ CODESYS).

A distinction is made between variables declared as RETAIN and variables in the flag area which can be configured as a remanent block with **MEMORY_RETAIN_PARAM** (→ p. 205).
Details → chapter **Variables** (→ p. 69)

The advantage of the automatic backup is that also in case of a sudden voltage drop or an interruption of the supply voltage, the storage operation is triggered and thus the current values of the data are saved (e.g. counter values).

If supply voltage < 8 V, retain data is no longer backed up!
In this case, flag RETAIN_WARNING = TRUE.

© ifm electronic gmbh

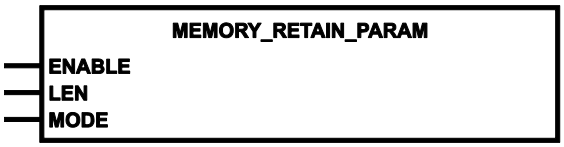
MEMORY_RETAIN_PARAM

2372

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2374

MEMORY_RETAIN_PARAM determines the remanent data behaviour for various events. Variables declared as VAR_RETAIN in CODESYS have a remanent behaviour from the outset.

Remanent data keep their value (as the variables declared as VAR_RETAIN) after an uncontrolled termination as well as after normal switch off and on of the controller. After a restart the program continues to work with the stored values.

For groups of events that can be selected (with MODE), this function block determines how many (LEN) data bytes (from flag byte %MB0) shall have retain behaviour even if they have not been explicitly declared as VAR_RETAIN.

Event	MODE = 0	MODE = 1	MODE = 2	MODE = 3
Power OFF ⇨ ON	Data is newly initialised	Data is remanent	Data is remanent	Data is remanent
Soft reset	Data is newly initialised	Data is remanent	Data is remanent	Data is remanent
Cold reset	Data is newly initialised	Data is newly initialised	Data is remanent	Data is remanent
Reset default	Data is newly initialised	Data is newly initialised	Data is remanent	Data is remanent
Load application program	Data is newly initialised	Data is newly initialised	Data is remanent	Data is remanent
Load runtime system	Data is newly initialised	Data is newly initialised	Data is newly initialised	Data is remanent

If MODE = 0, only those data have retain behaviour as with MODE=1 which have been explicitly declared as VAR_RETAIN.

If the FB is never called, the flag bytes act according to MODE = 0. The flag bytes which are above the configured area act according to MODE = 0, too.

Once a configuration has been made, it remains on the device even if the application or the runtime system is reloaded.

Parameters of the inputs

2375

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
LEN	WORD	Number of data bytes from flag address %MB0 onwards to show remanent behaviour allowed = 0...4 096 = 0x0...0x1000 LEN > 4 096 will be corrected automatically to LEN = 4 096
MODE	BYTE	Events for which these variables shall have retain behaviour (0...3; → table above) For MODE > 3 the last valid setting will remain

© ifm electronic gmbh

Manual data storage

Contents

13801

Besides the possibility to store data automatically, user data can be stored manually, via function block calls, in integrated memories from where they can also be read.

By means of the storage partitioning (→ chapter **Available memory** (→ p. 15)) the programmer can find out which memory area is available.

© ifm electronic gmbh

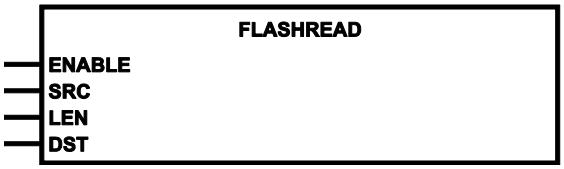
FLASHREAD

561

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

564

FLASHREAD enables reading of different types of data directly from the flash memory.

> The FB reads the contents as from the address of SRC from the flash memory. In doing so, as many bytes as indicated under LEN are transmitted.

> The contents are read completely during the cycle in which the FB is called up.

► Please make sure that the target memory area in the RAM is sufficient.

► To the destination address DST applies:
(!) Determine the address by means of the operator ADR and assigne it to the POU!

Parameters of the inputs

2318

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
SRC	DWORD	relative start address in memory allowed = 0...65 535 = 0x0...0x0000 FFFF (!) If start address is outside the permissible range: > no data transfer
LEN	DWORD	number of data bytes (max. 65 536 = 0x0001 0000) (!) If the indicated number of bytes exceeded the flash memory space, the data would only be transmitted to the end of the flash memory space.
DST	DWORD	destination address (!) Determine the address by means of the operator ADR and assigne it to the POU!

© ifm electronic gmbh www.ifm.com

FLASHWRITE

555

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

19245

► Activate the TEST input to use the function block! Otherwise, a watchdog error occurs.

Test input is active:
• Programming mode is enabled
• Software download is possible
• Status of the application program can be queried
• Protection of stored software is not possible

558

⚠ WARNING

Danger due to uncontrollable process operations!

The status of the inputs/outputs is "frozen" during execution of FLASHWRITE.

► Do not execute this FB when the machine is running!

FLASHWRITE enables writing of different data types directly into the flash memory.

Using this FB, large data volumes are to be stored during set-up, to which there is only read access in the process.

► If a page has already been written (even if only partly), the entire flash memory area needs to be deleted before new write access to this page. This is done by write access to the address 0.

► Never write to a page several times! Always delete everything first!
Otherwise, traps or watchdog errors occur.

► [!] Do not delete the flash memory area more often than 100 times. Otherwise, the data consistency in other flash memory areas is no longer guaranteed.

► During each SPS cycle, FLASHWRITE may only be started once!

► To the source start address SRC applies:
[!] Determine the address by means of the operator ADR and assigne it to the POU!

> The FB writes the contents of the address SRC into the flash memory. In doing so, as many bytes as indicated under LEN are transmitted.

[!] If destination start address DST is outside the permissible range: no data transfer!

© ifm electronic gmbh

Parameters of the inputs

2603

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
DST	DWORD	Relative start address in memory allowed = 0...65 535 = 0x0...0x0000 FFFF Determine the address by means of the operator ADR and assigne it to the POU!
LEN	DWORD	number of data bytes (max. 65 536 = 0x0001 0000) If the indicated number of bytes exceeded the flash memory space, the data would only be transmitted to the end of the flash memory space.
SRC	DWORD	source address

© ifm electronic gmbh

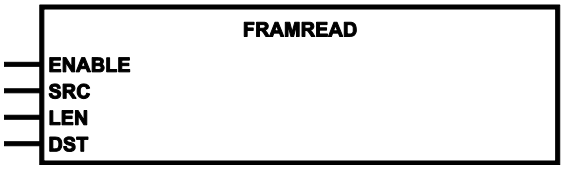
FRAMREAD

549

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

552

FRAMREAD enables quick reading of different data types directly from the FRAM memory [1]).

The FB reads the contents as from the address of SRC from the FRAM memory. In doing so, as many bytes as indicated under LEN are transmitted.
If the FRAM memory area were to be exceeded by the indicated number of bytes, only the data up to the end of the FRAM memory area will be read.

► To the destination address DST applies:
(!) Determine the address by means of the operator ADR and assigne it to the POU!

[1]) FRAM indicates here all kinds of non-volatile and fast memories.

Parameters of the inputs

2606

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
SRC	DWORD	relative start address in memory allowed = 0... 16 383 = 0x0000 0000...0x0000 3FFF
LEN	DWORD	number of data bytes allowed = 0...16 384 = 0x0000 0000...0x0000 4000
DST	DWORD	destination address (!) Determine the address by means of the operator ADR and assigne it to the POU!

© ifm electronic gmbh www.ifm.com

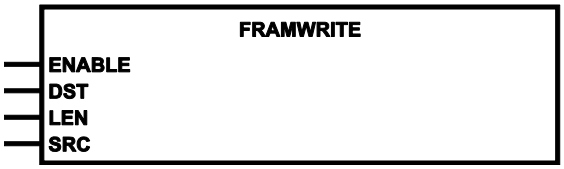
FRAMWRITE

543

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

546

FRAMWRITE enables the quick writing of different data types directly into the FRAM memory [1]).

The FB writes the contents of the address SRC to the non-volatile FRAM memory. In doing so, as many bytes as indicated under LEN are transmitted.
If the FRAM memory area were to be exceeded by the indicated number of bytes, only the data up to the end of the FRAM memory area will be written.

► To the source address SRC applies:
  Determine the address by means of the operator ADR and assigne it to the POU!

If the target address DST is outside the permissible range: no data transfer!

[1]) FRAM indicates here all kinds of non-volatile and fast memories.

Parameters of the inputs

2605

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
DST	DWORD	Relative start address in memory allowed = 0... 16 383 = 0x0...0x0000 3FFF
LEN	DWORD	number of data bytes allowed = 0...16 384 = 0x0000 0000...0x0000 4000
SRC	DWORD	start address in source memory Determine the address by means of the operator ADR and assigne it to the POU!

© ifm electronic gmbh www.ifm.com

MEMCPY

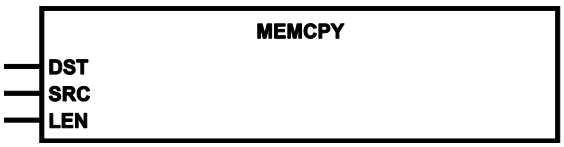
409

= memory copy

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

15944
412

MEMCPY enables writing and reading different types of data directly in the memory.

The FB writes the contents of the address of SRC to the address DST.

► To the addresses SRC and DST apply:
(!) Determine the address by means of the operator ADR and assigne it to the POU!

> In doing so, as many bytes as indicated under LEN are transmitted. So it is also possible to transmit exactly one byte of a word variable.

> If the memory area into which the data are to be copied is not entirely in a permissible memory area, the data will not be copied and a parameter error will be signalled.

DST memory area	Device	Memory size
Application data	(all)	192 Kbytes

Tables "Available memory" → chapter **Available memory** (→ p. 15)

Parameters of the inputs

413

Parameter	Data type	Description
DST	DWORD	destination address (!) Determine the address by means of the operator ADR and assigne it to the POU!
SRC	DWORD	start address in source memory (!) Determine the address by means of the operator ADR and assigne it to the POU!
LEN	WORD	number (≥ 1) of the data bytes to be transmitted

© ifm electronic gmbh www.ifm.com

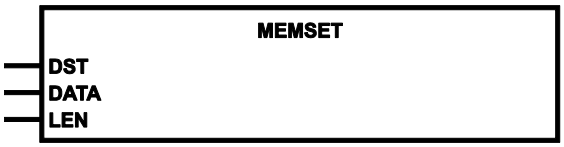
MEMSET

2348

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

2350

MEMSET enables writing to a defined data area.

The FB writes the content of DATA into the memory as from the address of DST as many bytes as indicated under LEN.

► For the destination address DST applies:
[!] Determine the address by means of the operator ADR and assigne it to the POU!

> If the memory area into which the data are to be copied is not entirely in a permissible memory area, the data will not be copied and a parameter error will be signalled.

DST memory area	Device	Memory size
Application data	(all)	192 Kbytes

Parameters of the inputs

2351

Parameter	Data type	Description
DST	DWORD	destination address [!] Determine the address by means of the operator ADR and assigne it to the POU!
DATA	BYTE	Value to be written
LEN	WORD	number of data bytes to be overwritten with DATA

© ifm electronic gmbh www.ifm.com

5.2.19 Function elements: data access and data check

Contents

1598

The FBs described in this chapter control the data access and enable a data check.

© ifm electronic gmbh

CHECK_DATA

603

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

606

CHECK_DATA generates a checksum (CRC) for a configurable memory area and checks the data of the memory area for undesired changes.

- ► Create a separate instance of the function block for each memory area to be monitored.
- ► Determine the address by means of the operator ADR and assigne it to the POU!
- ► In addition, indicate the number of data bytes LENGTH (length from the STARTADR).

Undesired change: Error!
If input UPDATE = FALSE and data in the memory is changed inadvertently, then RESULT = FALSE. The result can then be used for further actions (e.g. deactivation of the outputs).

Desired change:
Data changes in the memory (e.g. of the application program or ecomatmobile device) are only permitted if the output UPDATE is set to TRUE. The value of the checksum is then recalculated. The output RESULT is permanently TRUE again.

Parameters of the inputs

2612

Parameter	Data type	Description
STARTADR	DWORD	Start address of the monitored data memory (WORD address as from %MW0) Determine the address by means of the operator ADR and assigne it to the POU!
LENGTH	DWORD	length of the monitored data memory in [byte]
UPDATE	BOOL	TRUE: Data was changed > function block calculates new checksum FALSE: Data was not changed > function block checks memory area

Parameters of the outputs

2613

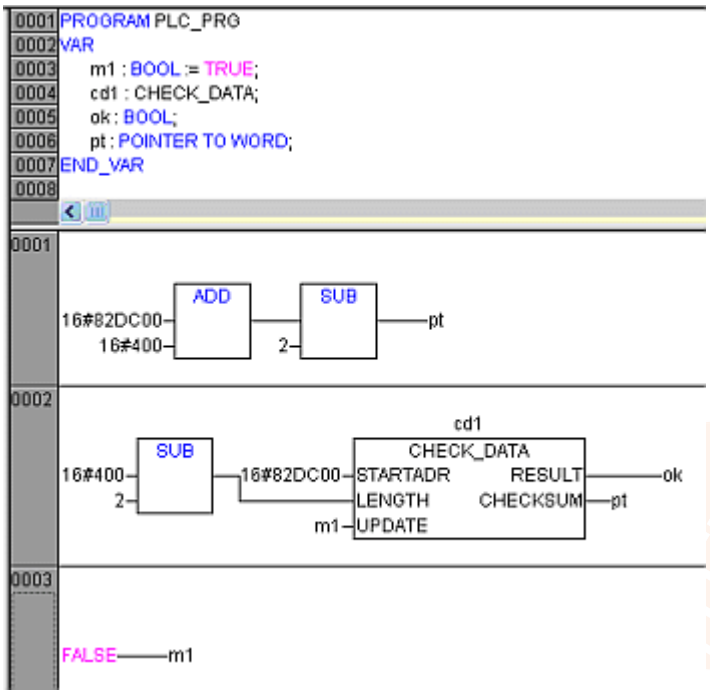
Parameter	Data type	Description
RESULT	BOOL	TRUE: CRC checksum OK: intentional data change or no change FALSE: CRC checksum faulty: data was changed inadvertently
CHECKSUM	DWORD	Current CRC checksum

© ifm electronic gmbh www.ifm.com

Example: CHECK_DATA

4168

In the following example the program determines the checksum and stores it in the RAM via pointer pt:



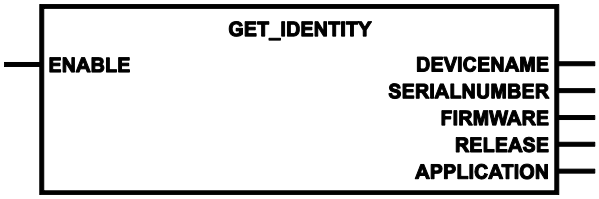
© ifm electronic gmbh
www.ifm.com

GET_IDENTITY

14505

Unit type = function block (FB)
Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

14507

GET_IDENTITY reads the specific identifications stored in the device:
- hardware name and hardware version of the device
- serial number of the device
- name of the runtime system in the device
- version and revision no. of the runtime system in the device
- name of the application (has previously been saved by means of **SET_IDENTITY** (→ p. 219))

Parameters of the inputs

2609

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified

Parameters of the outputs

14508

Parameter	Data type	Description
DEVICENAME	STRING(31)	hardware name as a string of max. 31 characters, e.g.: "CR0403"
SERIALNUMBER	STRING(31)	Serial number of the device as character string of max. 31 characters e.g.: "12345678"
FIRMWARE	STRING(31)	Name of the runtime system in the device as character string of max. 31 characters e.g.: "CR0403"
RELEASE	STRING(31)	software version as a character string of max. 31 characters
APPLICATION	STRING(79)	Name of the application as a string of max. 79 characters e.g.: "Crane1704"

© ifm electronic gmbh www.ifm.com

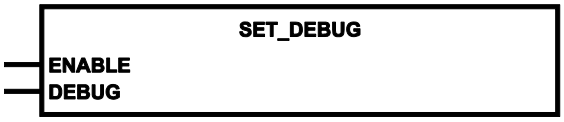
SET_DEBUG

290

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

293

SET_DEBUG handles the DEBUG mode without active test input (→ chapter **TEST mode** (→ p. 51)).

If the input DEBUG of the FB is set to TRUE, the programming system or the downloader, for example, can communicate with the device and execute some special system commands (e.g. for service functions via the GSM modem CANremote).

> In this operating mode a software download is not possible because the test input is not connected to supply voltage. Only read access is possible.

Parameters of the inputs

294

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
DEBUG	BOOL	TRUE: debugging via the interfaces possible FALSE: debugging via the interfaces not possible

© ifm electronic gmbh www.ifm.com

SET_IDENTITY

11927

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

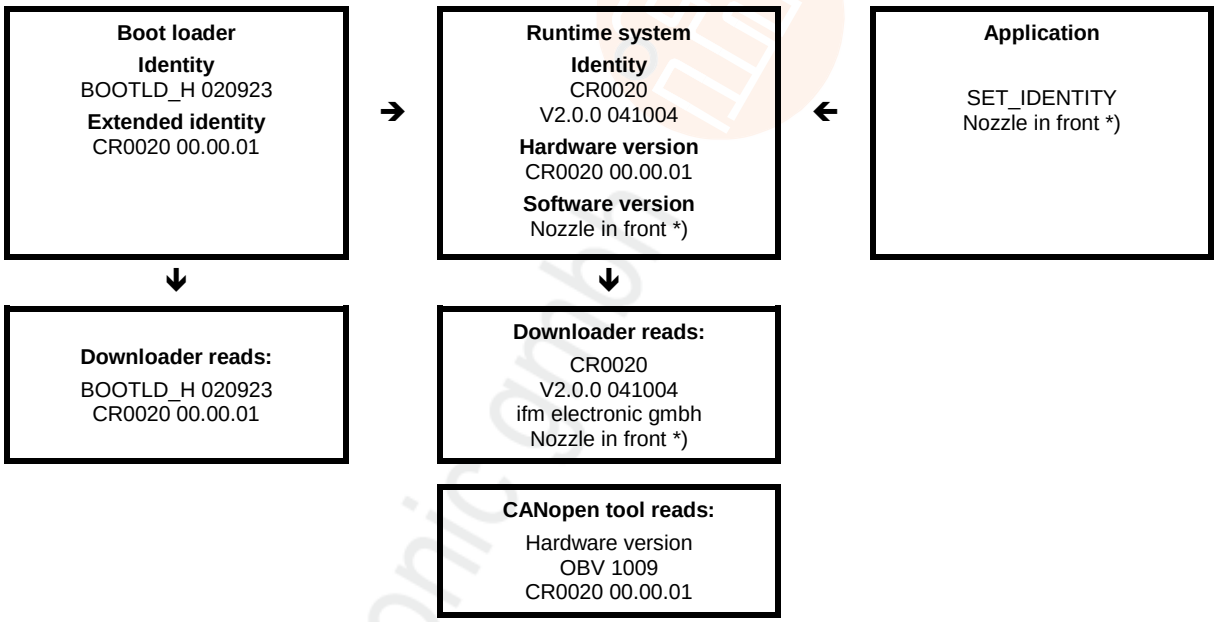
Description

287

SET_IDENTITY sets an application-specific program identification.

Using this FB, a program identification can be created by the application program. This identification (i.e. the software version) can be read via the software tool DOWNLOADER.EXE in order to identify the loaded program.

The following figure shows the correlations of the different identifications as indicated by the different software tools. (Example: ClassicController CR0020):



*) 'Nozzle in front' is substitutionally here for a customised text.

Parameters of the inputs

11928

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
ID	STRING(79)	Any desired text with a maximum length of 79 characters

© ifm electronic gmbh www.ifm.com

SET_PASSWORD

266

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

269

SET_PASSWORD sets a user password for the program and memory upload with the DOWNLOADER.

If the password is activated, reading of the application program or the data memory with the software tool DOWNLOADER is only possible if the correct password has been entered.

If an empty string (default condition) is assigned to the input PASSWORD, an upload of the application software or of the data memory is possible at any time.

A new password can be set only after resetting the previous password.

The password is reset when loading a new application program as boot project.

Parameters of the inputs

2353

Parameter	Data type	Description
ENABLE	BOOL	FALSE ⇨ TRUE (edge): Initialise block (only 1 cycle) > Read block inputs TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
PASSWORD	STRING(16)	password If PASSWORD = "", than access is possible without enter of a password

© ifm electronic gmbh www.ifm.com

5.2.20 Function elements: administer error messages

Contents

19229

Here we show you functions that enable you to
- generate application-specific error codes
- list or delete error codes

© ifm electronic gmbh

ERROR_REPORT

12357

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

12364

With ERROR_REPORT the application program signals an application-specific error to the system.

► Program the result of the error condition to the input ENABLE.
If ENABLE=TRUE, the function block enters the error code in the error list.
- view current error list with **SHOW_ERROR_LIST** (→ p. 226)
- delete an error entry from the error list using **ERROR_RESET** (→ p. 223)

► Program the corresponding error code to the input ERRORCODE:
Structure: yy xx 00 00 → chapter **Error codes** (→ p. 314)
xx = application-specific error code
yy = error class
Here, the function block **PACK_ERRORCODE** (→ p. 225)

> The function block output signals whether this function block call has been configured correctly.
The function block does not analyse whether the error codes are meaningful.

Parameters of the inputs

12363

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element an error is active signal the corresponding ERRORCODE FALSE: unit is not executed the fault is not active (anymore)
ERRORCODE	DWORD	When this error code occurs, the configured behaviour is to be applied. → chapter **Error codes** (→ p. 314) The function block does not analyse whether the error codes are meaningful.

Parameters of the outputs

19255

Parameter	Data type	Description
ERROR	DWORD	Error code from this function block call → **Error codes** (→ p. 314) (possible messages → following table)

Possible results for ERROR (n=any desired value):

The 32-bit error code consists of four 8-bit values (DWORD).

4th byte	3rd byte	2nd byte	1st byte
Error class	Application-specific error code	Error source	Error cause

Value [hex]	Description
00 00 00 00	no error
01 00 00 F8	Wrong parameter ⇨ general error

ERROR_RESET

12376

Unit type = function block (FB)
Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

12378

With ERROR_RESET the application program can reset upcoming error messages:

- a single error message
- a group of error messages of the same type (same source or same cause)
- application-specific error:
  - All errors containing the indicated USER error code are deleted.
  - The error class is ignored.
  - Error cause and error source must have the value '0'.
- If an error of class 3 or 4 is reset, then...
  - the flag ERROR=FALSE is set
  - all other errors of classes 3 and 4 are also deleted.
- all errors

Depending on the entered ERRORCODE the following happens with ENABLE=TRUE:

ERRORCODE	Note	Description
0xCL 00 SO **00**	Eror cause = 0	Reset all errors with the same error source
0xCL 00 **00** CA	Error source = 0	Reset all errors with the same error cause
0xXX **AS** SO CA	Application-specific error	Reset all errors with the same number (irrespective of the error class)
0x**00 00 00 00**	Error code = 0	Reset all errors

Legend:
CL = code for error class
AS = code for application-specific error
SO = code for error source
CA = code for error cause
XX = code value without any effect

> As long as an error is active, the following happens with a reset:
  - The fault is reset briefly.
  - When the diagnostic time has elapsed plus 1 program cycle: The fault is signalled again.

► Between two reset actions of the FB:
set ENABLE=FALSE (at least for one PLC cycle)!

© ifm electronic gmbh

Parameters of the inputs

12379

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
ERRORCODE	DWORD	When this error code occurs, the configured behaviour is to be applied. → chapter **Error codes** (→ p. 314) The function block does not analyse whether the error codes are meaningful.

Parameters of the outputs

19257

Parameter	Data type	Description
ERROR	DWORD	Error code from this function block call → **Error codes** (→ p. 314) (possible messages → following table)

Possible results for ERROR (n=any desired value):

The 32-bit error code consists of four 8-bit values (DWORD).

4th byte	3rd byte	2nd byte	1st byte
Error class	Application-specific error code	Error source	Error cause

Value [hex]	Description
00 00 00 00	no error
01 00 00 F8	Wrong parameter ⇨ general error

Example: ERROR_RESET

13054

If all errors "Overload" are to be reset with the error class "General errors", ERRORCODE=0x01000004 needs to be indicated:

4th byte	3rd byte	2nd byte	1st byte
Error class	Application-specific error code	Error source	Error cause
0x01	0x00	0x00	0x04
General error	No application-specific error	All error sources	Overload

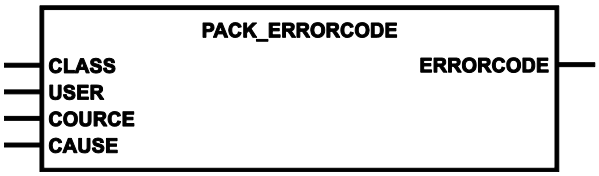
© ifm electronic gmbh www.ifm.com

PACK_ERRORCODE

12382

Unit type = function block (FB)
Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

12384

PACK_ERRORCODE helps to build an ERRORCODE from the components:
- error class
- application-specific error
- error source
- error cause

(structure → chapter **Error codes** (→ p. 314)).

The function block does not analyse whether the error codes are meaningful.

Parameters of the inputs

12385

Parameter	Data type	Description
CAUSE	BYTE	Code for error cause
SOURCE	BYTE	Code for error source
USERCODE	BYTE	Code for application-specific error
CLASS	BYTE	Code for error class

Parameters of the outputs

12390

Parameter	Data type	Description
ERRORCODE	DWORD	created error code → chapter **Error codes** (→ p. 314) The function block does not analyse whether the error codes are meaningful.

© ifm electronic gmbh www.ifm.com

SHOW_ERROR_LIST

12360

Unit type = function block (FB)
Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

12367

The function block SHOW_ERROR_LIST serves to read the currently present error codes.
With ENABLE=TRUE the function block creates a list of up to 64 currently existing error codes.
ENABLE=FALSE preserves an unchanged version of the most recent list.
If the list is full, no more error codes are accepted.

Parameters of the inputs

12368

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified

Parameters of the outputs

12369

Parameter	Data type	Description
ERRORS	ARRAY [0..63] OF DWORD	List of currently present error codes → chapter **Error codes** (→ p. 314)

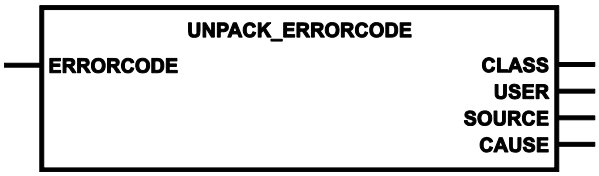
© ifm electronic gmbh www.ifm.com

UNPACK_ERRORCODE

13650

Unit type = function block (FB)
Unit is contained in the library `ifm_CR0033_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

13653

UNPACK_ERRORCODE divides an ERRORCODE into its components:
- error class
- application-specific error
- error source
- error cause

(structure → chapter **Error codes** (→ p. 314)).

The function block does not analyse whether the error codes are meaningful.

Parameters of the inputs

13654

Parameter	Data type	Description
ERRORCODE	DWORD	Error code →chapter **Error codes** (→ p. 314) The function block does not analyse whether the error codes are meaningful.

Parameters of the outputs

13675

Parameter	Data type	Description
CLASS	BYTE	Code for error class
USERCODE	BYTE	Code for application-specific error
SOURCE	BYTE	Code for error source
CAUSE	BYTE	Code for error cause

© ifm electronic gmbh www.ifm.com

6 Diagnosis and error handling

Contents

19598

The runtime-system (RTS) checks the device by internal error checks:
• during the boot phase (reset phase)
• during executing the application program
→ chapter ***Operating states*** (→ p. 47)

In so doing a high operating reliability is provided, as much as possible.

6.1 Diagnosis

19601

During the diagnosis, the "state of health" of the device is checked. It is to be found out if and what →faults are given in the device.

Depending on the device, the inputs and outputs can also be monitored for their correct function.
- wire break,
- short circuit,
- value outside range.

For diagnosis, configuration and log data can be used, created during the "normal" operation of the device.
The correct start of the system components is monitored during the initialisation and start phase.
Errors are recorded in the log file.
For further diagnosis, self-tests can also be carried out.

6.2 Fault

19602

A fault is the state of an item characterized by the inability to perform the requested function, excluding the inability during preventive maintenance or other planned actions, or due to lack of external resources.
A fault is often the result of a failure of the item itself, but may exist without prior failure.
In →ISO 13849-1 "fault" means "random fault".

© ifm electronic gmbh

6.3 Reaction in case of an error

19603
12217

When errors are detected the system flag ERROR can also be set in the application program. Thus, in case of a fault, the controller reacts as follows:

> the operation LED lights red,
> the output relays switch off,
> the outputs protected by the relays are disconnected from power,
> the logic signal states of the outputs remain unchanged.

NOTE

If the outputs are switched off by the relays, the logic signal states remain unchanged.

► The programmer must evaluate the ERROR bit and thus also reset the output logic in case of a fault.

Complete list of the device-specific error codes and diagnostic messages
→ chapter **System flags** (→ p. 231).

6.4 Relay: important notes!

14034

NOTICE

Premature wear of the relay contacts possible.

► In normal operation, only switch the relays without load!
For this purpose, set all relevant outputs to FALSE via the application program!

© ifm electronic gmbh

6.5 Response to system errors

14033
4320

The programmer has the sole responsibility for the safe processing of data in the application software.

► Process the specific error flags and/or error codes in the application program!
An error description is provided via the error flag / error code.
This error flag / error code can be further processed if necessary.

After analysis and elimination of the error cause:

► As a general rule, reset all error flags via the application program.
Without explicit reset of the error flags the flags remain set with the corresponding effect on the application program.

6.6 CAN / CANopen: errors and error handling

19604

→ System manual "Know-How ecomat*mobile*"
→ chapter **CAN / CANopen: errors and error handling**

© ifm electronic gmbh www.ifm.com

7 Appendix

Contents

1664

Additionally to the indications in the data sheets you find summary tables in the appendix.

7.1 System flags

Contents

12167

The addresses of the system flags can change if the PLC configuration is extended.
► While programming only use the symbol names of the system flags!

→ System manual "Know-How ecomat*mobile*"
→ chapter **Error codes and diagnostic information**

© ifm electronic gmbh

7.1.1 System flags: CAN

12820

System flags (symbol name)	Type	Description
CANx_BAUDRATE	WORD	CAN interface x: set baud rate in [kBaud]
CANx_BUSOFF	BOOL	CAN interface x: Error "CAN-Bus off" Reset of the error code alse resets the flag
CANx_DOWNLOADID	BYTE	CAN interface x: set download identifier
CANx_ERRORCOUNTER_RX	BYTE	CAN interface x: Error counter receiver Reset of the flag is possible via write access
CANx_ERRORCOUNTER_TX	BYTE	CAN interface x: error counter transmission A reset of the flag is possible via write access
CANx_LASTERROR	BYTE	CAN interface x: Error number of the last CAN transmission: 0 = no error — Initial value 1 = stuff error — more than 5 identical bits in series on the bus 2 = form error — received message had wrong format 3 = ack error — sent message was not confirmed 4 = bit1 error — a recessive bit was sent outside the arbitration area, but a dominant bit was read on the bus 5 = bit0 error — it was tried to send a dominant bit, but a recessive level was read OR: a sequence of 11 recessive bits was read during bus-off recovery 6 = CRC error — checksum of the received message was wrong
CANx_WARNING	BOOL	CAN interface x: warning threshold reached (≥ 96) A reset of the flag is possible via write access

CANx stands for x = 1...4 = number of the CAN interface

© ifm electronic gmbh www.ifm.com

7.1.2 System flags: SAE-J1939

12815

System flags (symbol name)	Type	Description
J1939_RECEIVE_OVERWRITE	BOOL	Setting only applies to J1939 data that has not been transmitted via a J1939 transport protocol. TRUE: The old data is overwritten by the new data if the old data has not yet been read from the function block instance FALSE: New data is rejected as long as the old data has not been read from the function block instance New data can arrive before the old data has been read out if the IEC cycle is longer than the refresh rate of the J1939 data
J1939_TASK	BOOL	Using J1939_TASK, the time requirement for sending J1939 messages is met. If J1939 messages are to be sent with a repetition time ≤ 50 ms, the runtime system automatically sets J1939_TASK=TRUE. For applications for which the time requirement is ≥ PLC cycle time: ▶ Reduce system load with J1939_TASK=FALSE! TRUE: J1939 task is active (= initial value) The task is called every 2 ms. The J1939 stack sends its messages in the required time frame FALSE: J1939 task is not active

© ifm electronic gmbh

7.1.3 System flags: error flags (standard side)

23784

System flags (symbol name)	Type	Description
ERROR	BOOL	TRUE: safe state assumed all outputs = OFF output relays = OFF (e.g. fatal error / Error-Stop) FALSE: no serious error occurred
ERROR_BREAK_Ix (0...x, value depends on the device, → data sheet)	DWORD	input double word x: wire break error or (resistance input): short to supply [Bit 0 for input 0] ... [bit z for input z] of this group Bit = TRUE: error Bit = FALSE: no error
ERROR_BREAK_Qx (0...x, value depends on the device, → data sheet)	DWORD	output double word x: wire break error [Bit 0 for output 0] ... [bit z for output z] of this group Bit = TRUE: error Bit = FALSE: no error
ERROR_CONTROL_Qx (0...x, value depends on the device, → data sheet)	DWORD	output double word x:current control error final value cannot be reached [Bit 0 for output 0] ... [bit z for output z] of this group Bit = TRUE: error Bit = FALSE: no error
ERROR_CURRENT_Ix (0...x, value depends on the device, → data sheet)	DWORD	input double word x: over-current error only if Ixx_MODE = IN_CURRENT [Bit 0 for input 0] ... [bit z for input z] of this group Bit = TRUE: error Bit = FALSE: no error
ERROR_POWER	BOOL	Overvoltage error for VBBs / clamp 15: TRUE: Value out of range or: difference (VBB15 - VBBs) > 1 V > general error > application STOP > outputs = inactive > no communication > message "Overvoltage clamp 15" FALSE: Value OK
ERROR_SHORT_Ix (0...x, value depends on the device, → data sheet)	DWORD	input double word x: short circuit error only if input mode = IN_DIGITAL_H [Bit 0 for input 0] ... [bit z for input z] of this group Bit = TRUE: error Bit = FALSE: no error
ERROR_SHORT_Qx (0...x, value depends on the device, → data sheet)	DWORD	output double word x: short circuit error or overload error [Bit 0 for output 0] ... [bit z for output z] of this group Bit = TRUE: error Bit = FALSE: no error
ERROR_TEMPERATURE	BOOL	Temperature error TRUE: Value out of range > general error FALSE: Value OK
ERROR_VBBx	BOOL	Supply voltage error on VBBx (x = o | r): TRUE: Value out of range > general error FALSE: Value OK
ERRORCODE	DWORD	Last error written to internal error list The list contains all error codes that occurred.
LAST_RESET	BYTE	Cause for the last reset: 00 = reset of the application

© ifm electronic gmbh www.ifm.com

7.1.4 System flags: status LED (standard side)

12817

System flags (symbol name)	Type	Description
LED	WORD	LED color for "LED switched on": 0x0000 = LED_GREEN (preset) 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 color for "LED switched off": 0x0000 = LED_GREEN 0x0001 = LED_BLUE 0x0002 = LED_RED 0x0003 = LED_WHITE 0x0004 = LED_BLACK (preset) 0x0005 = LED_MAGENTA 0x0006 = LED_CYAN 0x0007 = LED_YELLOW
LED_MODE	WORD	LED flashing frequency: 0x0000 = LED_2HZ (flashes at 2 Hz; preset) 0x0001 = LED_1HZ (flashes at 1 Hz) 0x0002 = LED_05HZ (flashes at 0.5 Hz) 0x0003 = LED_0HZ (lights permanently with value in LED)

© ifm electronic gmbh www.ifm.com

7.1.5 System flags: voltages (standard side)

12135

System flags (symbol name)	Type	Description
CLAMP_15_VOLTAGE	WORD	voltage applied to clamp 15 in [mV]
REF_VOLTAGE	WORD	Voltage on reference voltage output in [mV]
REFERENCE_VOLTAGE_5	BOOL	Reference voltage output with 5 V activated
REFERENCE_VOLTAGE_10	BOOL	Reference voltage output with 10 V activated
RELAIS_VBBy y = o | r	BOOL	TRUE: relay for VBBy activated voltage is applied to output group x (x = 1 | 2) FALSE: relay for VBBy deactivated no voltage is applied to output group x
SERIAL_MODE	BOOL	Activate serial interface (RS232) for use in the application TRUE: The RS232 interface can be used in the application, but no longer for programming, debugging or monitoring of the device. FALSE: The RS232 interface cannot be used in the application. Programming, debugging or monitoring of the device is possible.
SUPPLY_SWITCH	BOOL	Bit for switching off the supply latching VBBs. Resetting the flag is only accepted by the runtime system if the voltage at clamp 15 < 4 V, otherwise the flag is activated again. Separation of VBBs is done before the next PLC cycle starts. Depending on the charging status of the internal capacitors it may take some time until the device switches off. TRUE: Supply of the device via VBBs is active FALSE: Supply of the device via VBBs is deactivated
SUPPLY_VOLTAGE	WORD	supply voltage at VBBs in [mV]
TEST	BOOL	TRUE: Test input is active: • Programming mode is enabled • Software download is possible • Status of the application program can be queried • Protection of stored software is not possible FALSE: application is in operation
VBBx_RELAIS_VOLTAGE x = o | r	WORD	Supply voltage on VBBx to relay contact in [mV]
VBBx_VOLTAGE x = o | r	WORD	Supply voltage on VBBx in [mV]

© ifm electronic gmbh www.ifm.com

7.1.6 System flags: 16 inputs and 16 outputs (standard side)

13121

System flags (symbol name)	Type	Description
ANALOGxx xx = 00...15	WORD	Analogue input xx: filtered A/D converter raw value (12 bits) without calibration or standardisation
ANALOG_IRQxx xx = 00...07	WORD	Analogue input xx: unfiltered A/D converter raw value (12 bits) without calibration or standardisation Use in FB **SET_INTERRUPT_I** (→ p. 126) or **SET_INTERRUPT_XMS** (→ p. 128)
CURRENTxx xx = 00...15	WORD	PWM output xx: filtered A/D converter raw values (12 bits) of the current measurement without calibration or standardisation
Ixx xx = 00...15	BOOL	Status on binary input xx Condition: input is configured as binary input (MODE = IN_DIGITAL_H or IN_DIGITAL_L) TRUE: Voltage on binary input > 70 % of VBBS FALSE: Voltage on binary input < 30 % of VBBS or: not configured as binary input or: wrong configuration
Ixx_DFILTER xx = 00...11	DWORD	Pulse input xx: pulse duration in [µs] which is to be ignored as a glitch. Acquisition of the input signal is delayed by the set time. allowed = 0...100 000 µs preset = 0 µs = no filter
Ixx_FILTER xx = 00...15	BYTE:=4	Binary and analogue input xx: limit frequency (or signal rise time) of the first-order software low-pass filter 0 = 0x00 = no filter 1 = 0x01 = 390 Hz (1 ms) 2 = 0x02 = 145 Hz (2.5 ms) 3 = 0x03 = 68 Hz (5 ms) 4 = 0x04 = 34 Hz (10 ms) (preset) 5 = 0x05 = 17 Hz (21 ms) 6 = 0x06 = 8 Hz (42 ms) 7 = 0x07 = 4 Hz (84 ms) 8 = 0x08 = 2 Hz (169 ms) higher = → preset value
Qxx xx = 00...15	BOOL	Status on binary output xx: Condition: output is configured as binary output TRUE: output activated FALSE: output deactivated (= initial value) or: not configured as binary output
Qxx_FILTER xx = 00...15	BYTE	Output xx: limit frequency of the first-order software low-pass filter for the current measurement only if Qxx_MODE = OUT_DIGITAL_H not if PWM mode 0 = 0x00 = no filter 1 = 0x01 = 580 Hz (0.6 ms) 2 = 0x02 = 220 Hz (1.6 ms) 3 = 0x03 = 102 Hz (3.5 ms) 4 = 0x04 = 51 Hz (7 ms) (preset) 5 = 0x05 = 25 Hz (14 ms) 6 = 0x06 = 12 Hz (28 ms) 7 = 0x07 = 6 Hz (56 ms) 8 = 0x08 = 3 Hz (112 ms) higher = → preset value

© ifm electronic gmbh www.ifm.com

7.2 Address assignment and I/O operating modes

Contents

1656

→ also data sheet

7.2.1 Addresses / variables of the I/Os

Contents

2376

© ifm electronic gmbh

Inputs: addresses and variables (standard side) (16 inputs)

10432

Abbreviations →chapter Note on wiring (→ p. 33)
Operating modes of the inputs/outputs →chapter Possible operating modes inputs/outputs (→ p. 243)

IEC address	I/O variable	Remark
%IX0.0	I00	Binary input channel 0
%IX0.1	I01	Binary input channel 1
%IX0.2	I02	Binary input channel 2
%IX0.3	I03	Binary input channel 3
%IX0.4	I04	Binary input channel 4
%IX0.5	I05	Binary input channel 5
%IX0.6	I06	Binary input channel 6
%IX0.7	I07	Binary input channel 7
%IX0.8	I08	Binary input channel 8
%IX0.9	I09	Binary input channel 9
%IX0.10	I10	Binary input channel 10
%IX0.11	I11	Binary input channel 11
%IX0.12	I12	Binary input channel 12
%IX0.13	I13	Binary input channel 13
%IX0.14	I14	Binary input channel 14
%IX0.15	I15	Binary input channel 15
%IW2	ANALOG00	Analogue input channel 0
%IW3	ANALOG01	Analogue input channel 1
%IW4	ANALOG02	Analogue input channel 2
%IW5	ANALOG03	Analogue input channel 3
%IW6	ANALOG04	Analogue input channel 4
%IW7	ANALOG05	Analogue input channel 5
%IW8	ANALOG06	Analogue input channel 6
%IW9	ANALOG07	Analogue input channel 7
%IW10	ANALOG08	Analogue input channel 8
%IW11	ANALOG09	Analogue input channel 9
%IW12	ANALOG10	Analogue input channel 10
%IW13	ANALOG11	Analogue input channel 11
%IW14	ANALOG12	Analogue input channel 12
%IW15	ANALOG13	Analogue input channel 13
%IW16	ANALOG14	Analogue input channel 14
%IW17	ANALOG15	Analogue input channel 15
%IW18	CURRENT00	Output current (raw value) on Q00
%IW19	CURRENT01	Output current (raw value) on Q01
%IW20	CURRENT02	Output current (raw value) on Q02
%IW21	CURRENT03	Output current (raw value) on Q03
%IW22	CURRENT04	Output current (raw value) on Q04
%IW23	CURRENT05	Output current (raw value) on Q05

© ifm electronic gmbH

IEC address	I/O variable	Remark
%IW24	CURRENT06	Output current (raw value) on Q06
%IW25	CURRENT07	Output current (raw value) on Q07
%IW26	CURRENT08	Output current (raw value) on Q08
%IW27	CURRENT09	Output current (raw value) on Q09
%IW28	CURRENT10	Output current (raw value) on Q10
%IW29	CURRENT11	Output current (raw value) on Q11
%IW30	CURRENT12	Output current (raw value) on Q12
%IW31	CURRENT13	Output current (raw value) on Q13
%IW32	CURRENT14	Output current (raw value) on Q14
%IW33	CURRENT15	Output current (raw value) on Q15
%IW34	SUPPLY_VOLTAGE	Supply voltage on VBBs in [mV]
%IW35	CLAMP_15_VOLTAGE	Voltage clamp 15
%IW36	VBBO_VOLTAGE	Supply voltage on VBBo in [mV]
%IW37	VBBR_VOLTAGE	Supply voltage on VBBr in [mV]
%IW38	VBBO_RELAIS_VOLTAGE	Supply voltage VBBo to relay contact in [mV]
%IW39	VBBR_RELAIS_VOLTAGE	Supply voltage VBBr to relay contact in [mV]
%IW40	REF_VOLTAGE	Voltage on the reference output pin 51
%IW41	ANALOG_IRQ00	Interrupt to analogue input channel 0
%IW42	ANALOG_IRQ01	Interrupt to analogue input channel 1
%IW43	ANALOG_IRQ02	Interrupt to analogue input channel 2
%IW44	ANALOG_IRQ03	Interrupt to analogue input channel 3
%IW45	ANALOG_IRQ04	Interrupt to analogue input channel 4
%IW46	ANALOG_IRQ05	Interrupt to analogue input channel 5
%IW47	ANALOG_IRQ06	Interrupt to analogue input channel 6
%IW48	ANALOG_IRQ07	Interrupt to analogue input channel 7
%MB7960	ERROR_CURRENT_I0	Error DWORD overcurrent
%MB7964	ERROR_SHORT_I0	Error DWORD short circuit
%MB7968	ERROR_BREAK_I0	Error DWORD wire break

© ifm electronic gmbh

Outputs: addresses and variables (standard side) (16 outputs)

10433

Abbreviations →chapter Note on wiring (→ p. 33)
Operating modes of the inputs/outputs →chapter Possible operating modes inputs/outputs (→ p. 243)

IEC address	I/O variable	Remark
%QX0.0	Q00	Binary output / PWM output channel 0
%QX0.1	Q01	Binary output / PWM output channel 1
%QX0.2	Q02	Binary output / PWM output channel 2
%QX0.3	Q03	Binary output / PWM output channel 3
%QX0.4	Q04	Binary output / PWM output channel 4
%QX0.5	Q05	Binary output / PWM output channel 5
%QX0.6	Q06	Binary output / PWM output channel 6
%QX0.7	Q07	Binary output / PWM output channel 7
%QX0.8	Q08	Binary output / PWM output channel 8
%QX0.9	Q09	Binary output / PWM output channel 9
%QX0.10	Q10	Binary output / PWM output channel 10
%QX0.11	Q11	Binary output / PWM output channel 11
%QX0.12	Q12	Binary output / PWM output channel 12
%QX0.13	Q13	Binary output / PWM output channel 13
%QX0.14	Q14	Binary output / PWM output channel 14
%QX0.15	Q15	Binary output / PWM output channel 15
%QB2	REFERENCE_VOLTAGE_5	Activating the reference voltage output with 5 V
%QB3	REFERENCE_VOLTAGE_10	Activating the reference voltage output with 10 V
%QB68	I00_FILTER	Filter byte for %IX0.0 / %IW2
%QB69	I01_FILTER	Filter byte for %IX0.1 / %IW3
%QB70	I02_FILTER	Filter byte for %IX0.2 / %IW4
%QB71	I03_FILTER	Filter byte for %IX0.3 / %IW5
%QB72	I04_FILTER	Filter byte for %IX0.4 / %IW6
%QB73	I05_FILTER	Filter byte for %IX0.5 / %IW7
%QB74	I06_FILTER	Filter byte for %IX0.6 / %IW8
%QB75	I07_FILTER	Filter byte for %IX0.7 / %IW9
%QB76	I08_FILTER	Filter byte for %IX0.8 / %IW2
%QB77	I09_FILTER	Filter byte for %IX0.9 / %IW3
%QB78	I10_FILTER	Filter byte for %IX0.10 / %IW4
%QB79	I11_FILTER	Filter byte for %IX0.11 / %IW5
%QB80	I12_FILTER	Filter byte for %IX0.12 / %IW6
%QB81	I13_FILTER	Filter byte for %IX0.13 / %IW7
%QB82	I14_FILTER	Filter byte for %IX0.14 / %IW8
%QB83	I15_FILTER	Filter byte for %IX0.15 / %IW9
%QB84	Q00_FILTER	Filter byte for %QX0.0
%QB85	Q01_FILTER	Filter byte for %QX0.1
%QB86	Q02_FILTER	Filter byte for %QX0.2

© ifm electronic gmbh

IEC address	I/O variable	Remark
%QB87	Q03_FILTER	Filter byte for %QX0.3
%QB88	Q04_FILTER	Filter byte for %QX0.4
%QB89	Q05_FILTER	Filter byte for %QX0.5
%QB90	Q06_FILTER	Filter byte for %QX0.6
%QB91	Q07_FILTER	Filter byte for %QX0.7
%QB92	Q08_FILTER	Filter byte for %QX0.8
%QB93	Q09_FILTER	Filter byte for %QX0.9
%QB94	Q10_FILTER	Filter byte for %QX0.10
%QB95	Q11_FILTER	Filter byte for %QX0.11
%QB96	Q12_FILTER	Filter byte for %QX0.12
%QB97	Q13_FILTER	Filter byte for %QX0.13
%QB98	Q14_FILTER	Filter byte for %QX0.14
%QB99	Q15_FILTER	Filter byte for %QX0.15
%QD25	I00_DFILTER	Filter value counting/pulse input 0
%QD26	I01_DFILTER	Filter value counting/pulse input 1
%QD27	I02_DFILTER	Filter value counting/pulse input 2
%QD28	I03_DFILTER	Filter value counting/pulse input 3
%QD29	I04_DFILTER	Filter value counting/pulse input 4
%QD30	I05_DFILTER	Filter value counting/pulse input 5
%QD31	I06_DFILTER	Filter value counting/pulse input 6
%QD32	I07_DFILTER	Filter value counting/pulse input 7
%QD33	I08_DFILTER	Filter value counting/pulse input 8
%QD34	I09_DFILTER	Filter value counting/pulse input 9
%QD35	I10_DFILTER	Filter value counting/pulse input 10
%QD36	I11_DFILTER	Filter value counting/pulse input 11
%MB7948	ERROR_SHORT_Q0	Error DWORD short circuit
%MB7952	ERROR_BREAK_Q0	Error DWORD wire break
%MB7956	ERROR_CONTROL_Q0	Error DWORD current control

© ifm electronic gmbh

7.2.2 Possible operating modes inputs/outputs

Contents

2386

© ifm electronic gmbh
www.ifm.com

Inputs: operating modes (standard side) (16 inputs)

19368

= this configuration value is default

Inputs	Possible operating mode		Set with function block	Function block input	Value	
					dec	hex
I00…I11	IN_NOMODE	off	INPUT_ANALOG SET_INPUT_ANALOG	MODE	0	0000
	IN_DIGITAL_H	plus	INPUT_ANALOG SET_INPUT_ANALOG	MODE	1	0001
	IN_DIGITAL_L	minus	INPUT_ANALOG SET_INPUT_ANALOG	MODE	2	0002
	IN_CURRENT	0…20 000 µA	INPUT_ANALOG SET_INPUT_ANALOG	MODE	4	0004
	IN_VOLTAGE10	0…10 000 mV	INPUT_ANALOG SET_INPUT_ANALOG	MODE	8	0008
	IN_VOLTAGE30	0…32 000 mV	INPUT_ANALOG SET_INPUT_ANALOG	MODE	16	0010
	IN_RATIO	0…1 000 ‰	INPUT_ANALOG SET_INPUT_ANALOG	MODE	32	0020
	Diagnosis	for IN_DIGITAL_H	SET_INPUT_MODE	DIAGNOSTICS	TRUE	
	Frequency measurement Interval measurement Phase measurement	0…30 000 Hz	FREQUENCY FREQUENCY_PERIOD PHASE			
	Interval measurement	0,1...5 000 Hz	PERIOD			
	Period and ratio measurement	0,1...5 000 Hz	PERIOD_RATIO			
	Counter	0…30 000 Hz	FAST_COUNT			
I00...I07	Detect encoder	0…30 000 Hz 0...5 000 Hz	INC_ENCODER INC_ENCODER_HR			
I12...I14	IN_NOMODE	off	INPUT_ANALOG SET_INPUT_ANALOG	MODE	0	0000
	IN_DIGITAL_H	plus	INPUT_ANALOG SET_INPUT_ANALOG	MODE	1	0001
	Diagnosis	for IN_DIGITAL_H	SET_INPUT_MODE	DIAGNOSTICS	TRUE	
	IN_RESISTANCE	16…30 000 Ω	INPUT_ANALOG SET_INPUT_ANALOG	MODE	512	0200
I15	IN_NOMODE	off	INPUT_ANALOG SET_INPUT_ANALOG	MODE	0	0000
	IN_DIGITAL_H	plus	INPUT_ANALOG SET_INPUT_ANALOG	MODE	1	0001
	Diagnosis	for IN_DIGITAL_H	SET_INPUT_MODE	DIAGNOSTICS	TRUE	
	IN_RESISTANCE	3…680 Ω	INPUT_ANALOG SET_INPUT_ANALOG	MODE	512	0200

Set operating modes with the following function block:

FAST_COUNT (→ p. 142)	Counter block for fast input pulses
FREQUENCY (→ p. 144)	Measures the frequency of the signal arriving at the selected channel
FREQUENCY_PERIOD (→ p. 146)	Measures the frequency and the cycle period (cycle time) in [µs] at the indicated channel
INC_ENCODER (→ p. 148)	Up/down counter function for the evaluation of encoders
INC_ENCODER_HR (→ p. 150)	Up/down counter function for the high resolution evaluation of encoders

INPUT_ANALOG (→ p. 131)	analogue input channel: alternatively measurement of... • current • voltage • resistance
PERIOD (→ p. 152)	Measures the frequency and the cycle period (cycle time) in [µs] at the indicated channel
PERIOD_RATIO (→ p. 154)	Measures the frequency and the cycle period (cycle time) in [µs] during the indicated periods at the indicated channel. In addition, the mark-to-space ratio is indicated in [‰].
SET_INPUT_MODE (→ p. 134)	Assigns an operating mode to an input channel

© ifm electronic gmbh www.ifm.com

Outputs: operating modes (standard side) (16 outputs)

15523

= this configuration value is default

Outputs	Possible operating mode		Set with function block	Function block input	Value	
					dec	hex
Q00…Q15	OUT_DIGITAL_H	plus	SET_OUTPUT_MODE	MODE	1	0001
	OUT_DIGITAL_L	minus	SET_OUTPUT_MODE	MODE	2	0002
	Diagnosis	for OUT_DIGITAL_H via current measurement	SET_OUTPUT_MODE	DIAGNOSTICS	TRUE	
	Overload protection	for OUT_DIGITAL_H with current measurement	SET_OUTPUT_MODE	PROTECTION	TRUE	
	Current measuring range	no current measurement	SET_OUTPUT_MODE	CURRENT_RANGE	0	00
		2 A / 3 A	SET_OUTPUT_MODE	CURRENT_RANGE	1	01
		4 A	SET_OUTPUT_MODE	CURRENT_RANGE	2	02

Details → chapter **Outputs Q00...Q15: permitted operating modes** (→ p. 247)

Set operating modes with the following function block:

OUTPUT_BRIDGE (→ p. 163)	H-bridge on a PWM channel pair
OUTPUT_CURRENT_CONTROL (→ p. 168)	Current controller for a PWMi output channel
PWM1000 (→ p. 171)	Initialises and configures a PWM-capable output channel the mark-to-space ratio can be indicated in steps of 1 ‰
SET_OUTPUT_MODE (→ p. 159)	Sets the operating mode of the selected output channel

© ifm electronic gmbh

Outputs Q00...Q15: permitted operating modes

15525

Operating mode		Q00	Q01	Q02	Q03	Q04	Q05	Q06	Q07
OUT_DIGITAL_H	plus	X	X	X	X	X	X	X	X
OUT_DIGITAL_L	minus	--	X	--	X	--	--	--	--
Diagnosis	for OUT_DIGITAL_H via current measurement	X	X	X	X	X	X	X	X
Overload protection	for OUT_DIGITAL_H with current measurement	X	X	X	X	X	X	X	X
Current measuring range	2 A	X	X	X	X	--	--	--	--
	3 A	--	--	--	--	X	X	X	X
	4 A	X	X	X	X	--	--	--	--
PWM		X	X	X	X	X	X	X	X
PWMi		X	X	X	X	X	X	X	X
H-bridge		--	X	--	X	--	--	--	--

Operating mode		Q08	Q09	Q10	Q11	Q12	Q13	Q14	Q15
OUT_DIGITAL_H	plus	X	X	X	X	X	X	X	X
OUT_DIGITAL_L	minus	--	X	--	X	--	--	--	--
Diagnosis	for OUT_DIGITAL_H via current measurement	X	X	X	X	X	X	X	X
Overload protection	for OUT_DIGITAL_H with current measurement	X	X	X	X	X	X	X	X
Current measuring range	2 A	X	X	X	X	--	--	--	--
	3 A	--	--	--	--	X	X	X	X
	4 A	X	X	X	X	--	--	--	--
PWM		X	X	X	X	X	X	X	X
PWMi		X	X	X	X	X	X	X	X
H-bridge		--	X	--	X	--	--	--	--

© ifm electronic gmbh www.ifm.com

7.3 Integrated I/O module: Description

Contents

16418

7.3.1 System description I/O module ExB01

Contents

16422

Hardware description I/O module

Contents

16423

Hardware structure I/O module

16425

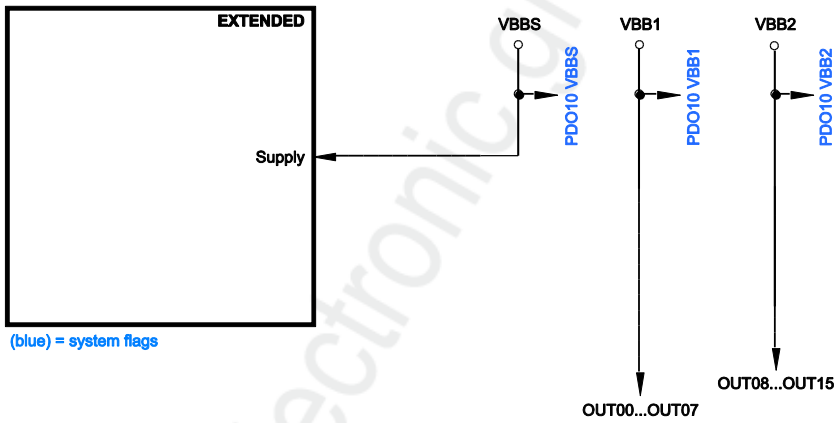


Figure: Block diagram of the supply

© ifm electronic gmbh

Status LED I/O module

16414

The operating states are indicated by the integrated status LED (default setting).

LED colour	Display	Description
Off	Permanently off	No operating voltage
		
Yellow	Briefly on	Status = INIT
	 (time frame = 200 ms)	
Green	Permanently on	Status = PRE-OPERATIONAL
		
Green	Flashing with 2 Hz	Status = OPERATIONAL
	 (time frame = 200 ms)	
Green	Flashing as 1 pulse	Status = STOP
	 (time frame = 200 ms)	
Red	Permanently on	Bug : CAN bus-off
		
Red	Flashing as 1 pulse	EMCY: CAN error warning
	 (time frame = 200 ms)	
Red	Flashing as 2 pulses	EMCY: guarding / heartbeat
	 (time frame = 200 ms)	
Red	Flashing as 3 pulses	EMCY: synch error
	 (time frame = 200 ms)	

© ifm electronic gmbh

Inputs of the integrated I/O module ExB01

Contents

16229

Analogue inputs

15444

The analogue inputs can be configured via the application program. The measuring range can be set as follows:

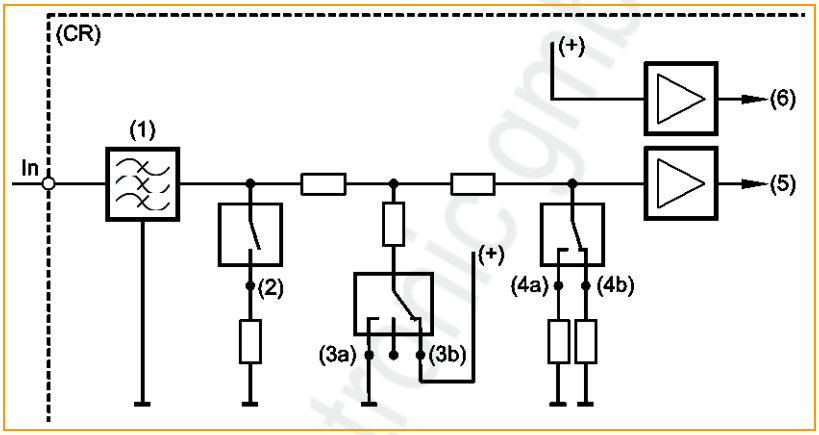
- current input 0...20 mA
- voltage input 0...10 V
- voltage input 0...32 V
- resistance measurement 16...30 000 Ω (measurement to GND)

The voltage measurement can also be carried out ratiometrically (0...1000 ‰, adjustable via function blocks). This means potentiometers or joysticks can be evaluated without additional reference voltage. A fluctuation of the supply voltage has no influence on this measured value.

As an alternative, an analogue channel can also be evaluated binarily.

> In case of ratiometric measurement the connected sensors should be supplied with VBBs of the device. So, faulty measurements caused by offset voltage are avoided.

8971



In = pin multifunction input n
(CR) = device
(1) = input filter
(2) = analogue current measuring
(3a) = binary-input plus switching
(3b) = binary-input minus switching
(4a) = analogue voltage measuring 0...10 V
(4b) = analogue voltage measuring 0...32 V
(5) = voltage
(6) = reference voltage

Figure: principle block diagram multifunction input

8972

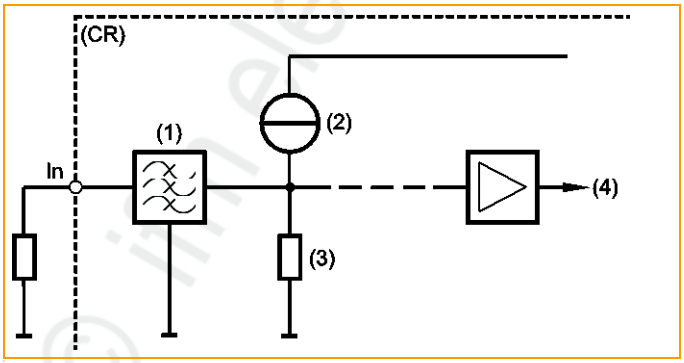


Figure: block diagram of the resistor survey input

In = pin resistor survey input n
(CR) = device
(1) = input filter
(2) = constant-current source
(3) = internal resistance
(4) = voltage

Binary inputs

1015
7345

The binary input can be operated in following modes:
- binary input plus switching (BL) for positive sensor signal
- binary input minus switching (BH) for negative sensor signal

Depending on the device the binary inputs can configured differently. In addition to the protective mechanisms against interference, the binary inputs are internally evaluated via an analogue stage. This enables diagnosis of the input signals. But in the application software the switching signal is directly available as bit information

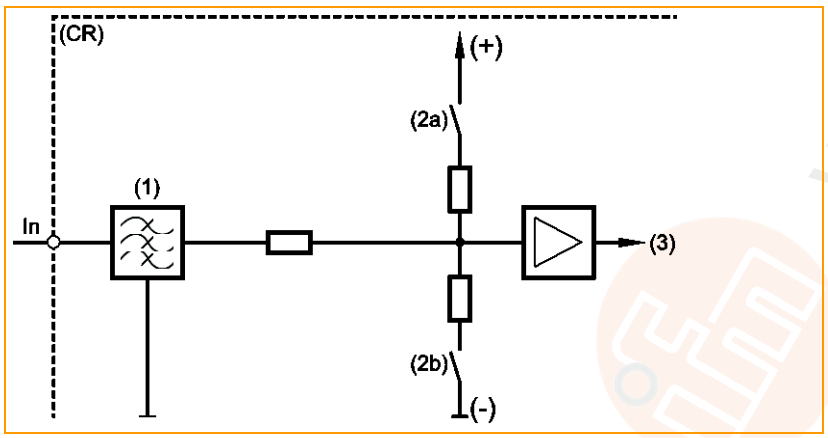
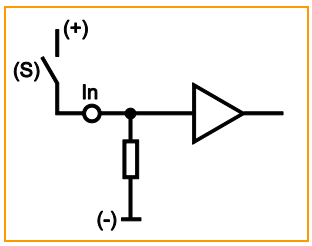
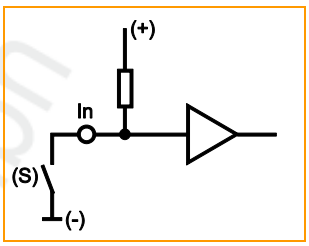


Figure: basic circuit of binary input minus switching / plus switching for negative and positive sensor signals



Basic circuit of binary input plus switching (BL)
for positive sensor signal:
Input = open ⇨ signal = low (GND)



Basic circuit of binary input minus switching (BH)
for negative sensor signal:
Input = open ⇨ signal = high (supply)

For some of these inputs (→ data sheet) the potential can be selected to which it will be switched.

© ifm electronic gmbh

I/O module input group IN00...IN03

15801

These inputs are a group of multifunction channels.

These inputs can be used as follows (each input separately configurable):
- analogue input 0...20 mA
- analogue input 0...10 V
- analogue input 0...32 V
- binary input plus switching (BL) for positive sensor signal (with/without diagnosis)
- binary input minus switching (BH) for negative sensor signal

→ chapter **Possible operating modes I/O module** (→ p. 270)

All inputs show the same behaviour concerning function and diagnosis.

► Configuration of each input is made via the PLC configuration:
→ chapter **Configure inputs of the integrated I/O module** (→ p. 264)

> If the analogue inputs are configured for current measurement and if the final value (23 mA for ≥ 40 ms) is exceeded, the device switches to the safe voltage measuring range (0...32 V DC). In this case PDO1 sends the message "overcurrent". After about one second the input automatically switches back to the current measuring range.

I/O module input group IN04...IN05

15803

These inputs are a group of multifunction channels.

These inputs can be used as follows (each input separately configurable):
- binary input plus switching (BL) for positive sensor signal (with/without diagnosis)
- input for resistance measurement (e.g. temperature sensors or fuel sensors)

→ chapter **Possible operating modes I/O module** (→ p. 270)

► Configuration of each input is made via the PLC configuration:
→ chapter **Configure inputs of the integrated I/O module** (→ p. 264)

© ifm electronic gmbh www.ifm.com

Resistance measurement

9773

Typical sensors on these inputs:
- tank level
- temperature (PT1000, NTC)

8972

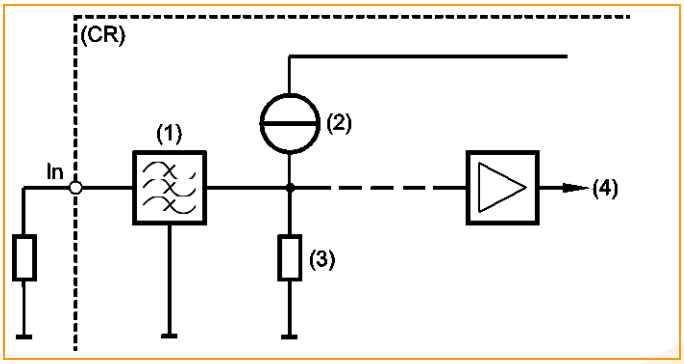


Figure: block diagram of the resistor survey input

In = pin resistor survey input n
(CR) = device
(1) = input filter
(2) = constant-current source
(3) = internal resistance
(4) = voltage

8970

The resistance for this device is not linearly dependent on the resistance value, → figure:

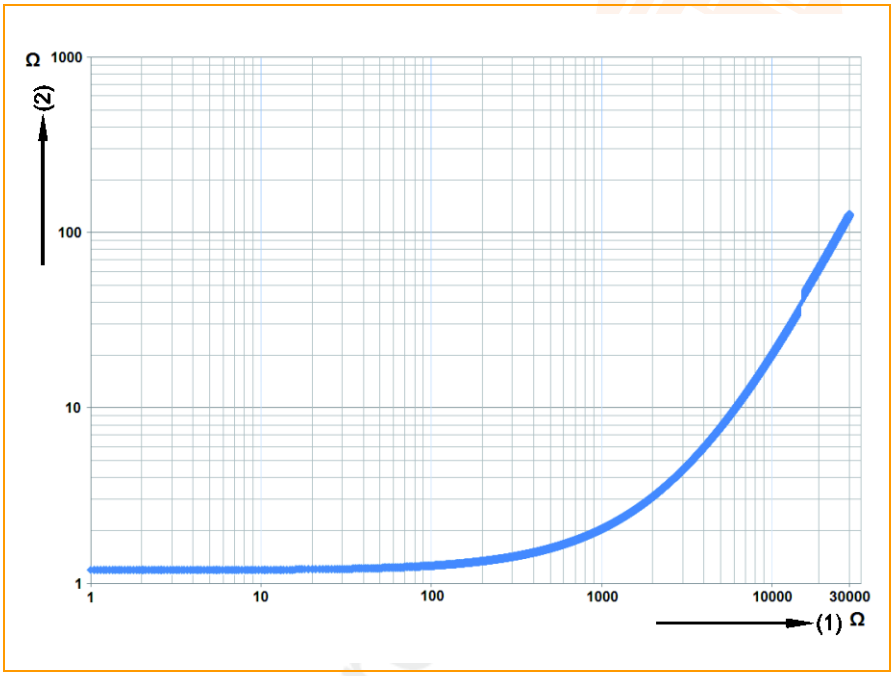


Figure: resolution dependent on the resistance value

(1) = resistance value at the input
(2) = resolution

By how many ohms does the measured value change when the signal of the A/D converter on the input changes by 1?
Examples:
- In the range of 1...100 Ω the resolution is 1.2 Ω.
- In the range of 1 kΩ the resolution is approx. 2 Ω.
- In the range of 2 kΩ the resolution is approx. 3 Ω.
- In the range of 3 kΩ the resolution is approx. 6 Ω.
- In the range of 6 kΩ the resolution is approx. 10 Ω.
- In the range of 10 kΩ the resolution is approx. 11 Ω
- In the range of 20 kΩ the resolution is approx. 60 Ω.

© ifm electronic

I/O module input group IN06...IN11

15804

These inputs are a group of multifunction channels.

These inputs can be used as follows (each input separately configurable):
• binary input plus switching (BL) for positive sensor signal (with/without diagnosis)
→ chapter **Possible operating modes I/O module** (→ p. 270)

Sensors with diagnostic capabilities to NAMUR can be evaluated.

► Configuration of each input is made via the PLC configuration:
→ chapter **Configure inputs of the integrated I/O module** (→ p. 264)

I/O module input group IN12...IN15

15805

These inputs are a group of multifunction channels.

These inputs can be used as follows (each input separately configurable):
• binary input plus switching (BL) for positive sensor signal
• fast input for e.g. incremental encoders and frequency or interval measurement
→ chapter **Possible operating modes I/O module** (→ p. 270)

► Configuration of each input is made via the PLC configuration:
→ chapter **Configure inputs of the integrated I/O module** (→ p. 264)

© ifm electronic gmbh www.ifm.com

Outputs of the integrated I/O module ExB01

Contents

16234

© ifm electronic gmbh

I/O module output group OUT0, OUT1

15806

These outputs are a group of multifunction channels.

These outputs provide several function options (each output separately configurable):
- binary output, plus switching (BH) with diagnostic function and protection
- analogue current-controlled output (PWMi)
- analogue output with Pulse Width Modulation (PWM)

→ chapter **Possible operating modes I/O module** (→ p. 270)

► Configuration of each output is made via the PLC configuration:
→ chapter **Configure outputs of the integrated I/O module** (→ p. 267)

► For the limit values please make sure to adhere to the data sheet!

Diagnosis: binary outputs (via current and voltage measurement)

19433
19434

The diagnostics of these outputs is made via internal current and voltage measurement in the output:

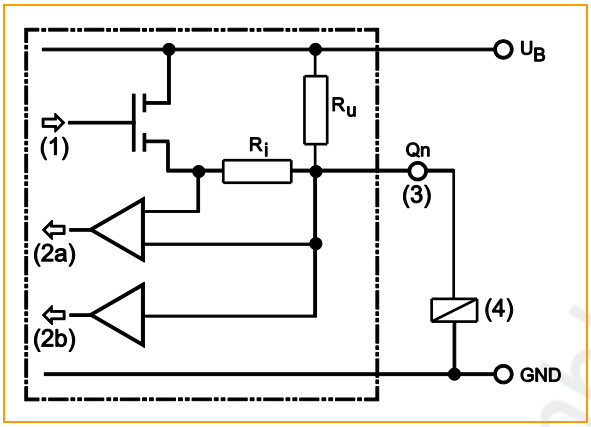


Figure: principle block diagram
(1) Output channel
(2a) Read back channel for diagnosis via current measuring
(2b) Read back channel for diagnosis via voltage measuring
(3) Pin output n
(4) Load

Diagnosis: overload (via current measurement)

19437
15249

Overload can only be detected on an output with current measurement.

Overload is defined as ...
"a nominal maximum current of 12.5 %".

Diagnosis: wire break (via voltage measurement)

19436
19404

Wire-break detection is done via the read back channel inside the output.

Prerequisite for diagnosis:	output = FALSE
Diagnosis = wire break:	the resistor Ru pulls the read back channel to HIGH potential (power supply) Without the wire break the low-resistance load (RL < 10 kΩ) would force a LOW (logical 0).

© ifm electronic gmbh www.ifm.com

Diagnosis: short circuit (via voltage measurement)

19405

Wire-break detection is done via the read back channel inside the output.

Prerequisite for diagnosis:	output = TRUE
Diagnosis = short circuit against GND:	the read back channel is pulled to LOW potential (GND)

© ifm electronic gmbh www.ifm.com

I/O module output group OUT02...OUT07

15808

These outputs are a group of multifunction channels.

These outputs provide several function options (each output separately configurable):
- binary output, plus switching (BH) with/without diagnostic function
- analogue output with Pulse Width Modulation (PWM)

→ chapter **Possible operating modes I/O module** (→ p. 270)

► Configuration of each output is made via the PLC configuration:
→ chapter **Configure outputs of the integrated I/O module** (→ p. 267)

► For the limit values please make sure to adhere to the data sheet!

Diagnosis: binary outputs (via voltage measurement)

19403
19397

The diagnostics of these outputs is made via internal voltage measurement in the output:

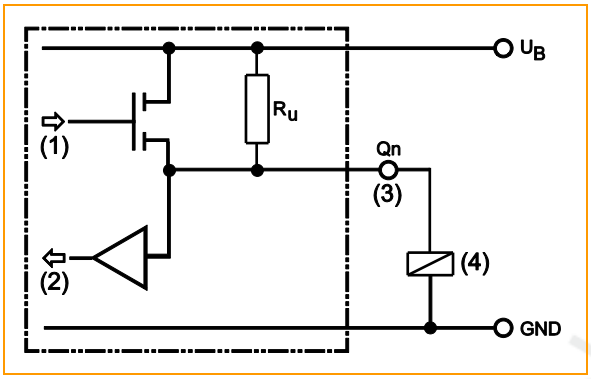


Figure: principle block diagram
(1) Output channel
(2) Read back channel for diagnosis
(3) Pin output n
(4) Load

Diagnosis: overload

19448

The outputs have no current measuring, no overload detection.

Diagnosis: wire break (via voltage measurement)

19404

Wire-break detection is done via the read back channel inside the output.

Prerequisite for diagnosis:	output = FALSE
Diagnosis = wire break:	the resistor Ru pulls the read back channel to HIGH potential (power supply) Without the wire break the low-resistance load (RL < 10 kΩ) would force a LOW (logical 0).

Diagnosis: short circuit (via voltage measurement)

19405

Wire-break detection is done via the read back channel inside the output.

Prerequisite for diagnosis:	output = TRUE
Diagnosis = short circuit against GND:	the read back channel is pulled to LOW potential (GND)

© ifm electronic gmbh www.ifm.com

I/O module output group OUT08...OUT09

15809

These outputs are a group of multifunction channels.

These outputs provide several function options (each output separately configurable):
- binary output, plus switching (BH)
- analogue output with Pulse Width Modulation (PWM)
- analogue output with Pulse Width Modulation (PWM), voltage-controlled

→ chapter **Possible operating modes I/O module** (→ p. 270)

► Configuration of each output is made via the PLC configuration:
→ chapter **Configure outputs of the integrated I/O module** (→ p. 267)

► For the limit values please make sure to adhere to the data sheet!

I/O module output group OUT10...OUT11

15810

These outputs are a group of multifunction channels.

These outputs provide several function options (each output separately configurable):
- binary output, plus switching (BH)
- analogue output with Pulse Width Modulation (PWM)

→ chapter **Possible operating modes I/O module** (→ p. 270)

► Configuration of each output is made via the PLC configuration:
→ chapter **Configure outputs of the integrated I/O module** (→ p. 267)

► For the limit values please make sure to adhere to the data sheet!

I/O module output group OUT12...OUT15

15811

These outputs are a group of channels with a single specified function.

These outputs have the following fixed setting:
- binary output, plus switching (BH)

→ chapter **Possible operating modes I/O module** (→ p. 270)

► For the limit values please make sure to adhere to the data sheet!

© ifm electronic gmbh www.ifm.com

Interface description I/O module

Contents

16426

CAN interfaces I/O module

Contents

16608

Connections and data → data sheet

CAN: Interfaces and protocols: I/O module in CR0133

15833
15835

The following CAN interfaces and CAN protocols are available in the integrated I/O module of the device:

CAN interface	CAN 1	CAN 2	CAN 3	CAN 4
Default download ID	ID 123	ID 122	---	---
CAN protocols	---	CANopen slave	---	---

Standard baud rate = 125 Kbits/s

CAN: Interfaces and protocols: I/O module in CR2532

16429
16435

The following CAN interfaces and CAN protocols are available in the integrated I/O module of the device:

CAN interface	CAN 1	CAN 2	CAN 3	CAN 4
Default download ID	ID 125	ID 124	---	---
CAN protocols	---	CANopen slave	---	---

Standard baud rate = 250 Kbits/s

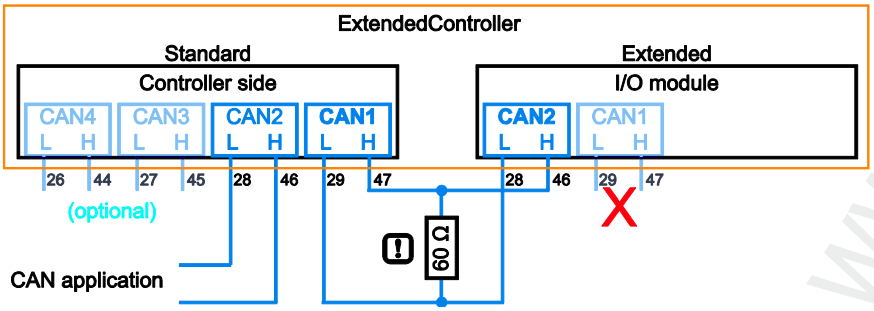
© ifm electronic gmbh

Connect integrated I/O module ExB01 as CANopen slave

15829

The integrated I/O module of the device is based on the Smart Controller CR2530:
- This side is preset as CANopen slave ExB01
- Use this side as input/output module.

We recommend the following connection method:



- CAN1 of the I/O module exclusively serves as service or maintenance interface!

► Only use the shown connection for the standard side of the controller with the integrated I/O module!
Do NOT use these connections for other purposes!

► For the CAN network in the application only use the interfaces ≥ CAN2 of the standard side!

© ifm electronic gmbh
www.ifm.com

7.3.2 Configuration of the I/O module

Contents

16427

Set up the programming system (I/O module)

Contents

16609

Set up the programming system manually (I/O module)

Contents

16610

© ifm electronic gmbh

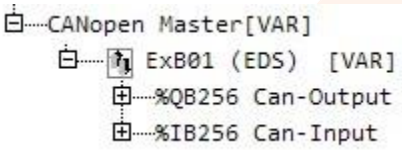
Integrate the internal I/O module ExB01

15828

Integrate the internal I/O module of the device as CANopen slave via the CODESYS control configuration!
For this, the same method is used as for integrating an external I/O module:

- ► Left clicking on the first line (`CR0033 Configuration Vxx`) in the CODESYS control configuration for highlighting.
- ► A click on the right mouse button opens the context menu.
- ► Select [`Append Subelement`] there.
- ► Select [`CANopen master...`] in the context menu.
  It always makes sense to configure the first CANopen master for CAN1.
- ► Open the context menu again by right clicking on [`CANopen master`].
- ► Select [`Append Subelement`] there.
- ► Select the EDS file for the integrated I/O module of the device in the context menu: [`ExB01_Vxxyyzz.EDS`].

> Result:



The IEC addresses for CAN-Input and CAN-Output result from the following details:
- type of the device used as CANopen master,
- position of the I/O module behind the CANopen master,
- assigned node ID.

The I/O module uses three consecutive node IDs. Rule:
⇨ [node ID of the following CAN slave] ≥ [node ID of the I/O module] + 3

- ► Set the CAN parameter:
  - node ID
  - nodeguarding
  - heartbeat settings
- ► Parameter setting of the inputs and outputs in the I/O module:
  → chapter **Object directory of the integrated I/O module** (→ p. 274)

Set up programming system via template (I/O module)

16611
13745

ifm offers ready-to-use templates (program templates), by means of which the programming system can be set up quickly, easily and completely.

970

When installing the **ecomatmobile** DVD "Software, tools and documentation", projects with templates have been stored in the program directory of your PC:
`…\ifm electronic\CoDeSys V…\Projects\Template_DVD_V…`

- ► Open the requested template in CODESYS via:
  [File] > [New from template…]

> CODESYS creates a new project which shows the basic program structure. It is strongly recommended to follow the shown procedure.

Function configuration of the inputs and outputs in the I/O module

Contents

16430

Configure inputs of the integrated I/O module

Contents

16244

Configure software filter of the inputs (I/O module)

15898

The software filter is preset and cannot be changed:

Table: limit frequency software low-pass filter on the analogue input

FILTER	Filter frequency [Hz]	Step response [ms] for ...			Remarks
		0…70 %	0…90 %	0…99 %	
fixed	10	19	36	72	

© ifm electronic gmbh

Analogue inputs: configuration and diagnosis (I/O module ExB01)

15894

Configuration of each input is made via the PLC configuration:
- Click on line [ExB01 (EDS)] below [CANopen Master]
- Click on the [Service Data Objects] tab
- Select index / sub-index of the requested parameter
- Click on the existing value in the [Value] column
- Change the value and confirm with [ENTER]

Permissible values → chapter **Inputs: operating modes (I/O module)** (→ p. 272)

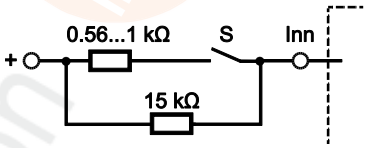
> If the analogue inputs are configured for current measurement and if the final value (23 mA for ≥ 40 ms) is exceeded, the device switches to the safe voltage measuring range (0...32 V DC). In this case PDO1 sends the message "overcurrent". After about one second the input automatically switches back to the current measuring range.

Binary inputs: Configuration and diagnosis (I/O module ExB01)

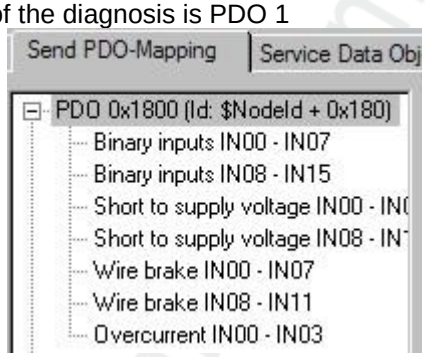
15896

► Configuration of each input is made via the PLC configuration:
- Click on line [ExB01 (EDS)] below [CANopen Master]
- Click on the [Service Data Objects] tab
- Select index / sub-index of the requested parameter
- Click on the existing value in the [Value] column
- Change the value and confirm with [ENTER]

Permissible values → chapter **Inputs: operating modes (I/O module)** (→ p. 272)

NAMUR diagnosis for binary signals of non-electronic switches: ► Equip the switch with an additional resistor connection!	 Figure: non-electronic switch S at input Inn

> The result of the diagnosis is PDO 1



© ifm electronic gmbh

Fast inputs: I/O module ExB01

15869

The devices dispose of fast counting/pulse inputs for an input frequency up to 30 kHz (→ data sheet).

14677

If, for example, mechanical switches are connected to these inputs, there may be faulty signals in the controller due to contact bouncing.

Configuration of each input is made via the PLC configuration:
Permissible values → chapter **Inputs: operating modes (I/O module)** (→ p. 272)

3804

The permissible high input frequencies also ensure the detection of faulty signals, e.g. bouncing contacts of mechanical switches.

► If required, suppress the faulty signals in the application program!

© ifm electronic gmbh
www.ifm.com

Configure outputs of the integrated I/O module

Contents

16248

Configure software filter of the outputs (I/O module)

15900

For the I/O module the following applies:
The software filter is preset and cannot be changed.

Table: limit frequency software low-pass filter on PWM output

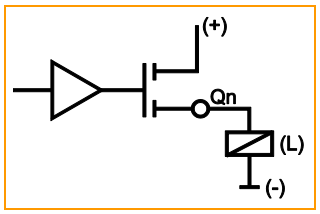
FILTER	Filter frequency [Hz]	Step response [ms] for ...			Remarks
		0…90 %	0…95 %	0…99 %	
fixed	52	7.2	9.4	14.4	

Binary outputs: configuration and diagnosis (I/O module ExB01)

15882

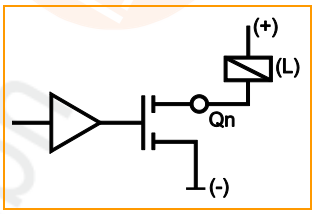
The following operating modes are possible for the device outputs (→ data sheet):
- binary output, plus switching (BH) with/without diagnostic function

15450



Qn = pin output n
(L) = load

Basic circuit of output plus switching (BH)
for positive output signal



Qn = pin output n
(L) = load

Basic circuit of output minus switching (BL)
for negative output signal

13975

⚠ WARNING

Dangerous restart possible!
Risk of personal injury! Risk of material damage to the machine/plant!

If in case of a fault an output is switched off via the hardware, the logic state generated by the application program is not changed.

► Remedy:
- Reset the output logic in the application program!
- Remove the fault!
- Reset the outputs depending on the situation.

© ifm electronic gmbh

Binary outputs: Configuration (I/O module ExB01)

15887

► Configuration of each output is made via the PLC configuration:
- Click on line [ExB01 (EDS)] below [CANopen Master]
- Click on the [Service Data Objects] tab
- Select index / sub-index of the requested parameter
- Click on the existing value in the [Value] column
- Change the value and confirm with [ENTER]

Permissible values → chapter **Outputs: Operating modes (I/O module)** (→ p. 273)

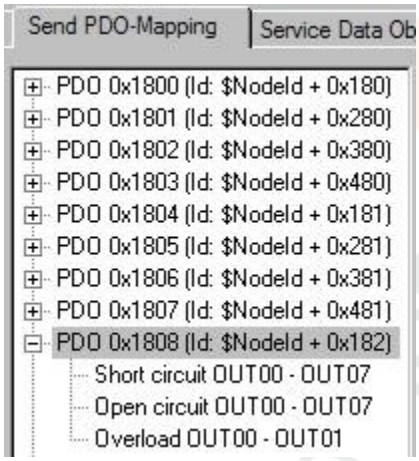
Binary outputs: Diagnosis (I/O module ExB01)

15889

► Configuration of each output is made via the PLC configuration:
Activate diagnosis with...
- Mode = 15 (OUT_BINARY_HIGH_DIAG) or
- Mode = 16 (OUT_BINARY_HIGH_DIAG_PROT)

Permissible values → chapter **Outputs: Operating modes (I/O module)** (→ p. 273)

> The result shows PDO 9:



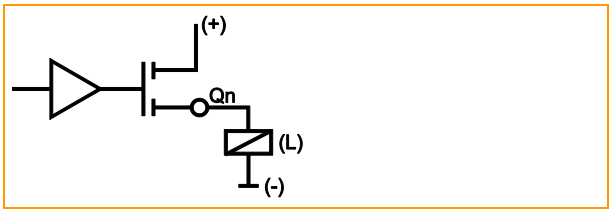
© ifm electronic gmbh www.ifm.com

PWM outputs: I/O module ExB01

16415

The following operating modes are possible for the device outputs (→ data sheet):
- PWM output, plus switching (BH) without diagnostic function

15451



Qn = pin output n
(L) = load

Basic circuit of output plus switching (BH)
for positive output signal

16253

⚠ WARNING

Property damage or bodily injury possible due to malfunctions!

For outputs in PWM mode:
- There are no diagnostic functions

9980

NOTE

PWM outputs must NOT be operated in parallel, e.g. in order to increase the max. output current. The outputs do not operate synchronously.

Otherwise the entire load current could flow through only one output. The current measurement would no longer function.

- PWM outputs can be operated with and without current control function.
  ℹ Current-controlled PWM outputs are mainly used for triggering proportional hydraulic functions.
  ! The medium current across a PWM signal can only be correctly determined if the current flowing in the switched-on state is within the measuring range.

Availability of PWM

16364

Device	Number of available PWM outputs	of which current-controlled (PWMi)	PWM frequency [Hz]
Integrated I/O module ExB01	12	2	20...250

Configure outputs for PWM functions

15888

The following settings are available for the PWM function of the outputs:
- Mode = 4 (OUT_PWM) or
- Mode = 5 (OUT_CURRENT)

Permissible values → chapter **Outputs: Operating modes (I/O module)** (→ p. 273)

Current control with PWM (= PWMi)

14722

Current measurement of the coil current can be carried out via the current measurement channels integrated in the controller. This way, the current can for example be re-adjusted if the coil heats up. The hydraulic conditions in the system are maintained.

In principle, the current-controlled outputs are protected against short circuit.

Possible operating modes I/O module

Contents

16440

© ifm electronic gmbh

Overview

15859

For the inputs and outputs the following operating modes are possible (detailed description: → the following pages):

SDO value dec	hex	Mode	Input Output	Description
0	0x00	OFF	Input Output	switched off, without function
1	0x01	IN_BINARY_LOW_DIGITAL	Input	binary plus switching, digital detection (if possible use mode 10 instead!)
2	0x02	OUT_BINARY_HIGH	Output	binary plus switching: output = FALSE ⇨ 0 V output = TRUE ⇨supply voltage of the outputs
3	0x03	IN_VOLTAGE_10V	Input	analogue voltage measurement in the 10 V measuring range
4	0x04	OUT_PWM	Output	PWM mode
5	0x05	OUT_CURRENT	Output	current-controlled
6	0x06	IN_VOLTAGE_RATIO	Input	analogue voltage measurement, ratiometric with regard to the supply voltage VBBs
7	0x07	IN_CURRENT	Input	analogue current measurement (up to 23 mA)
8	0x08	---	---	reserved
9	0x09	IN_VOLTAGE_32	Input	analogue voltage measurement in the 32 V measuring range
10	0x0A	IN_BINARY_LOW	Input	binary plus switching (depending on the supply voltage VBBs) (analogue or digital detection)
11	0x0B	IN_BINARY_LOW_DIAG	Input	binary plus switching with diagnosis (analogue detection) depending on the supply voltage VBBs diagnosis of short circuit with VBBs or GND
12	0x0C	IN_BINARY_HIGH	Input	binary minus switching (analogue detection) depending on the supply voltage VBBs
13	0x0D	---	---	reserved
14	0x0E	IN_FREQUENCY	Input	frequency measurement (digital detection)
15	0x0F	OUT_BINARY_HIGH_DIAG	Output	binary plus switching: output = FALSE ⇨ 0 V output = TRUE ⇨ supply voltage of the outputs diagnosis of wire break and short circuit
16	0x10	OUT_BINARY_HIGH_DIAG_PROT	Output	binary plus switching: output = FALSE ⇨ 0 V output = TRUE ⇨ supply voltage of the outputs diagnosis of wire break and short circuit output is switched off in case of short circuit
17	0x11	---	---	reserved
18	0x12	IN_RESISTOR	Input	resistance measurement (analogue detection)
19	0x13	---	---	reserved
20	0x14	IN_PERIOD_RATIO	Input	period duration measurement as ratio (digital detection)
21	0x15	---	---	reserved
22	0x16	---	---	reserved
23	0x17	---	---	reserved
24	0x18	---	---	reserved

© ifm electronic gmbh www.ifm.com

Inputs: operating modes (I/O module)

15965

► Configuration of each input is made via the PLC configuration:
- Click on line [ExB01 (EDS)] below [CANopen Master]
- Click on the [Service Data Objects] tab
- Select index / sub-index of the requested parameter
- Click on the existing value in the [Value] column
- Change the value and confirm with [ENTER]

= this configuration value is default

Inputs	Possible operating mode		In the object directory Index	Sub-Index	Value	
					dec	hex
IN00…IN03	off		0x2000	0x01...0x04	0	0x00
	voltage input	0…10 000 mV	0x2000	0x01...0x04	3	0x03
	voltage input ratiometric	0…1 000 ‰	0x2000	0x01...0x04	6	0x06
	current input	0…20 000 µA	0x2000	0x01...0x04	7	0x07
	voltage input	0…32 000 mV	0x2000	0x01...0x04	9	0x09
	binary input	plus switching	0x2000	0x01...0x04	10	0x0A
	binary input with diagnosis (Namur)	plus switching	0x2000	0x01...0x04	11	0x0B
	binary input	minus switching	0x2000	0x01...0x04	12	0x0C
IN04…IN05	off		0x2000	0x05...0x06	0	0x00
	binary input	plus switching	0x2000	0x05...0x06	10	0x0A
	binary input with diagnosis (Namur)	plus switching	0x2000	0x05...0x06	11	0x0B
	resistance input	16…30 000 Ohm	0x2000	0x05...0x06	18	0x12
IN06…IN11	off		0x2000	0x07...0x0C	0	0x00
	binary input	plus switching	0x2000	0x07...0x0C	10	0x0A
	binary input with diagnosis (Namur)	plus switching	0x2000	0x07...0x0C	11	0x0B
IN12…IN15	off		0x2000	0x0D...0x10	0	0x00
	binary input, digital evaluation	plus switching	0x2000	0x0D...0x10	1	0x01
	frequency measurement	0…30 000 Hz	0x2000	0x0D...0x10	14	0x0E
	period duration and ratio measurement	0.1…3 000 Hz	0x2000	0x0D...0x10	20	0x14

© ifm electronic gmbh www.ifm.com

Outputs: Operating modes (I/O module)

15966

► Configuration of each output is made via the PLC configuration:
- Click on line [ExB01 (EDS)] below [CANopen Master]
- Click on the [Service Data Objects] tab
- Select index / sub-index of the requested parameter
- Click on the existing value in the [Value] column
- Change the value and confirm with [ENTER]

= this configuration value is default

Outputs	Possible operating mode		In the object directory Index	Sub-Index	Value	
					dec	hex
OUT00 ...OUT01	off		0x2000	0x11...0x12	0	0x00
	binary output highside	plus	0x2000	0x11...0x12	2	0x02
	analogue output with pulse-width modulation		0x2000	0x11...0x12	4	0x04
	analogue current-controlled output		0x2000	0x11...0x12	5	0x05
	binary output highside with diagnosis	plus	0x2000	0x11...0x12	15	0x0F
	binary output highside with diagnosis and protection	plus	0x2000	0x11...0x12	16	0x10
OUT02 ...OUT07	off		0x2000	0x13...0x18	0	0x00
	binary output highside	plus	0x2000	0x13...0x18	2	0x02
	analogue output with pulse-width modulation		0x2000	0x13...0x18	4	0x04
	binary output highside with diagnosis	plus	0x2000	0x13...0x18	15	0x0F
	binary output highside with diagnosis and protection	plus	0x2000	0x13...0x18	16	0x10
OUT08 ...OUT09	off		0x2000	0x19...0x1A	0	0x00
	binary output highside	plus	0x2000	0x19...0x1A	2	0x02
	analogue output with pulse-width modulation		0x2000	0x19...0x1A	4	0x04
	analogue output with pulse-width modulation, voltage-controlled	(at pins 25 + 43)	0x2000	0x19...0x1A	4	0x04
OUT10 ...OUT11	off		0x2000	0x1B...0x1C	0	0x00
	binary output highside	plus	0x2000	0x1B...0x1C	2	0x02
	analogue output with pulse-width modulation		0x2000	0x1B...0x1C	4	0x04
OUT12 ...OUT15	off		0x2000	0x1D...0x20	0	0x00
	binary output highside	plus	0x2000	0x1D...0x20	2	0x02

© ifm electronic gmbh www.ifm.com

7.3.3 Object directory of the integrated I/O module

Contents

15837

© ifm electronic gmbh
www.ifm.com

Object directory parameter tables, overview

Contents

15977

General

15967

► Activate or deactivate the automatic saving of the communication and device parameters via the entry [Save Parameter] (→ object directory, index 0x1010):

- If SubIndex 0x1 = 0x02:
  save all parameters automatically when changed
- If SubIndex 0x1 = 0x00:
  do not save the parameters automatically.
  Changed parameters are only valid ...
  • until the device is switched off
  • until the next reset.

► Restore the preset values of the parameters via the function [Restore] (→ object directory, index 0x1011) (does not apply to baud rate and node ID). These values become valid with the next power on.

Data types in the EDS file

16409

EDS data type	IEC data type	min. value	max. value	Size in the memory
	BOOL	FALSE	TRUE	8 bits = 1 byte
	BYTE	0	255	8 bits = 1 byte
	WORD	0	65 535	16 bits = 2 bytes
	DWORD	0	4 294 967 295	32 bits = 4 bytes
	SINT	-128	127	8 bits = 1 byte
0x0005	USINT	0	255	8 bits = 1 byte
0x0003	INT	-32 768	32 767	16 bits = 2 bytes
0x0006	UINT	0	65 535	16 bits = 2 bytes
	DINT	-2 147 483 648	2 147 483 647	32 bits = 4 bytes
0x0007	UDINT	0	4 294 967 295	32 bits = 4 bytes
0x0008	REAL	-3.402823466 • 1038	3.402823466 • 1038	32 bits = 4 bytes
	ULINT	0	18 446 744 073 709 551 615	64 bits = 8 bytes
0x0009	STRING			number of char. + 1

© ifm electronic gmbh

Object directory obligatory objects (index 0x1000…0x1FFF), overview

15979

Object directory		Parameter description	Parameter for	Preset parameter value	Change saved automatically?	When does the change become effective?
Index	Sub-Idx					
0x1000		Device type	device	0xF0191	yes	at once (via CAN stack)
0x1001		Error register	device	---	yes	at once (via CAN stack)
0x1018		Device identification	device	---	--	--
	0x1	Vendor ID	device	6907501	yes	once when manufactured
	0x2	Product code	device	0	yes	once when manufactured
	0x3	Revision number	device	0	yes	once when manufactured
	0x4	Serial number	device	0	yes	once when manufactured

© ifm electronic gmbh

Object directory optional objects (index 0x1000…0x1FFF), overview

15980

Object directory		Parameter description	Parameter for	Preset parameter value	Change saved automatically?	When does the change become effective?
Index	Sub-Idx					
0x1003	0x1…0x5	Predefined error field	CANopen Basic configuration	0	yes	at once (via CAN stack)
0x1005		COB ID synch message	CANopen Basic configuration	0x80	yes	at once (via CAN stack)
0x1006		Communication cycle period	CANopen Basic configuration	0	yes	at once
0x1008		Manufacturer device name	CANopen Basic configuration	ExB01	yes	at once
0x1009		Manufacturer hardware version	CANopen Basic configuration	V00.00.00	yes	at once
0x100A		Manufacturer software version	CANopen Basic configuration	V00.00.00	yes	at once
0x100C		Guard time	CANopen Basic configuration	0	yes	at once
0x100D		Lifetime factor	CANopen Basic configuration	0	yes	at once
0x1010		Store parameters	CANopen Basic configuration		yes	at once
	0x1	Save all parameters	CANopen Basic configuration	1	yes	at once
0x1011		Restore default parameters	CANopen Basic configuration		no	after reset
	0x1	Restore all default parameters	CANopen Basic configuration	1	no	after reset
0x1014		COB ID emergency	CANopen Basic configuration	0x80 + node ID	yes	at once
0x1016		Consumer heartbeat times	CANopen Basic configuration		--	--
	0x1	Consumer heartbeat time	CANopen Basic configuration	0	yes	at once
0 1017		Producer heartbeat time	CANopen Basic configuration	0	yes	at once
0 1400		Receive PDO communication parameter	Configuration Receive PDO 1		--	--
	0x1	COB ID used by PDO	Configuration Receive PDO 1	0x0200 + node ID	yes	after PreOp
	0x2	transmission type	Configuration Receive PDO 1	1	yes	at once
0x1401		Receive PDO communication parameter	Configuration Receive PDO 2		--	--
	0x1	COB ID used by PDO	Configuration Receive PDO 2	0x0300 + node ID	yes	after PreOp
	0x2	transmission type	Configuration Receive PDO 2	1	yes	at once
0x1402		Receive PDO communication parameter	Configuration Receive PDO 3		--	--
	0x1	COB ID used by PDO	Configuration Receive PDO 3	0x0400 + node ID	yes	after PreOp

© ifm electronic gmbh www.ifm.com

Object directory		Parameter description	Parameter for	Preset parameter value	Change saved automatically?	When does the change become effective?
Index	Sub-Idx					
	0x2	transmission type	Configuration Receive PDO 3	1	yes	at once
0x1403		Receive PDO communication parameter	Configuration Receive PDO 4		--	--
	0x1	COB ID used by PDO	Configuration Receive PDO 4	0x0500 + node ID	yes	after PreOp
	0x2	transmission type	Configuration Receive PDO 4	1	yes	at once
0x1600		Receive PDO mapping	Mapping Receive PDO 1		yes	after PreOp
	0x1	PDO mapping	Mapping Receive PDO 1	0x6200 0108	yes	after PreOp
	0x2	PDO mapping	Mapping Receive PDO 1	0x6200 0208	yes	after PreOp
0x1601		Receive PDO mapping	Mapping Receive PDO 2		yes	after PreOp
	0x1	PDO mapping	Mapping Receive PDO 2	0x6414 0110	yes	after PreOp
	0x2	PDO mapping	Mapping Receive PDO 2	0x6414 0210	yes	after PreOp
	0x3	PDO mapping	Mapping Receive PDO 2	0x6414 0310	yes	after PreOp
	0x4	PDO mapping	Mapping Receive PDO 2	0x6414 0410	yes	after PreOp
0x1602		Receive PDO mapping	Mapping Receive PDO 3		yes	after PreOp
	0x1	PDO mapping	Mapping Receive PDO 3	0x6414 0510	yes	after PreOp
	0x2	PDO mapping	Mapping Receive PDO 3	0x6414 0610	yes	after PreOp
	0x3	PDO mapping	Mapping Receive PDO 3	0x6414 0710	yes	after PreOp
	0x4	PDO mapping	Mapping Receive PDO 3	0x6414 0810	yes	after PreOp
0x1603		Receive PDO mapping	Mapping Receive PDO 4		yes	after PreOp
	0x1	PDO mapping	Mapping Receive PDO 4	0x6414 0910	yes	after PreOp
	0x2	PDO mapping	Mapping Receive PDO 4	0x6414 0A10	yes	after PreOp
	0x3	PDO mapping	Mapping Receive PDO 4	0x6414 0B10	yes	after PreOp
	0x4	PDO mapping	Mapping Receive PDO 4	0x6414 0C10	yes	after PreOp
	0x5	PDO mapping	Mapping Receive PDO 4	0x00	yes	after PreOp
0x1800		Transmit PDO communication parameter	Configuration Transmit PDO 1		--	--
	0x1	COB ID used by PDO	Configuration Transmit PDO 1	0x180 + node ID	--	--

© ifm electronic gmbh

Object directory		Parameter description	Parameter for	Preset parameter value	Change saved automatically?	When does the change become effective?
Index	Sub-Idx					
	0x2	transmission type	Configuration Transmit PDO 1	1	yes	at once
	0x3	inhibit time	Configuration Transmit PDO 1	0	yes	at once
	0x4	reserved	Configuration Transmit PDO 1	0	no	--
	0x5	event time	Configuration Transmit PDO 1	0	yes	at ance
0x1801		Transmit PDO communication parameter	Configuration Transmit PDO 2		--	--
	0x1	COB ID used by PDO	Configuration Transmit PDO 2	0x280 + node ID	--	--
	0x2	transmission type	Configuration Transmit PDO 2	1	yes	at once
	0x3	inhibit time	Configuration Transmit PDO 2	0	yes	at once
	0x4	reserved	Configuration Transmit PDO 2	0	no	--
	0x5	event time	Configuration Transmit PDO 2	0	yes	at once
0x1802		Transmit PDO communication parameter	Configuration Transmit PDO 3		--	--
	0x1	COB ID used by PDO	Configuration Transmit PDO 3	0x380 + node ID	--	--
	0x2	transmission type	Configuration Transmit PDO 3	1	yes	at once
	0x3	inhibit time	Configuration Transmit PDO 3	0	yes	at once
	0x4	reserved	Configuration Transmit PDO 3	0	no	--
	0x5	event time	Configuration Transmit PDO 3	0	yes	at once
0x1803		Transmit PDO communication parameter	Configuration Transmit PDO 4		--	--
	0x1	COB ID used by PDO	Configuration Transmit PDO 4	0x480 + node ID	--	--
	0x2	transmission type	Configuration Transmit PDO 4	1	yes	at once
	0x3	inhibit time	Configuration Transmit PDO 4	0	yes	at once
	0x4	reserved	Configuration Transmit PDO 4	0	no	--
	0x5	event time	Configuration Transmit PDO 4	0	yes	at once
0x1804		Transmit PDO communication parameter	Configuration Transmit PDO 5		--	--
	0x1	COB ID used by PDO	Configuration Transmit PDO 5	0x181 + node ID	--	--
	0x2	transmission type	Configuration Transmit PDO 5	1	yes	at once
	0x3	inhibit time	Configuration Transmit PDO 5	0	yes	at once

© ifm electronic gmbH www.ifm.com

Object directory		Parameter description	Parameter for	Preset parameter value	Change saved automatically?	When does the change become effective?
Index	Sub-Idx					
	0x4	reserved	Configuration Transmit PDO 5	0	no	--
	0x5	event time	Configuration Transmit PDO 5	0	yes	at once
0x1805		Transmit PDO communication parameter	Configuration Transmit PDO 6		--	--
	0x1	COB ID used by PDO	Configuration Transmit PDO 6	0x281 + node ID	--	--
	0x2	transmission type	Configuration Transmit PDO 6	1	yes	at once
	0x3	inhibit time	Configuration Transmit PDO 6	0	yes	at once
	0x4	reserved	Configuration Transmit PDO 6	0	no	--
	0x5	event time	Configuration Transmit PDO 6	0	yes	at once
0x1806		Transmit PDO communication parameter	Configuration Transmit PDO 7		--	--
	0x1	COB ID used by PDO	Configuration Transmit PDO 7	0x381 + node ID	--	--
	0x2	transmission type	Configuration Transmit PDO 7	1	yes	at once
	0x3	inhibit time	Configuration Transmit PDO 7	0	yes	at once
	0x4	reserved	Configuration Transmit PDO 7	0	no	--
	0x5	event time	Configuration Transmit PDO 7	0	yes	at once
0x1807		Transmit PDO communication parameter	Configuration Transmit PDO 8		--	--
	0x1	COB ID used by PDO	Configuration Transmit PDO 8	0x481 + node ID	--	--
	0x2	transmission type	Configuration Transmit PDO 8	1	yes	at once
	0x3	inhibit time	Configuration Transmit PDO 8	0	yes	at once
	0x4	reserved	Configuration Transmit PDO 8	0	no	--
	0x5	event time	Configuration Transmit PDO 8	0	yes	at once
0x1808		Transmit PDO communication parameter	Configuration Transmit PDO 9		--	--
	0x1	COB ID used by PDO	Configuration Transmit PDO 9	0x182 + node ID	--	--
	0x2	transmission type	Configuration Transmit PDO 9	1	yes	at once
	0x3	inhibit time	Configuration Transmit PDO 9	0	yes	at once
	0x4	reserved	Configuration Transmit PDO 9	0	no	--
	0x5	event time	Configuration Transmit PDO 9	0	yes	at once

© ifm electronic gmbH www.ifm.com

Object directory		Parameter description	Parameter for	Preset parameter value	Change saved automatically?	When does the change become effective?
Index	Sub-Idx					
0x1809		Transmit PDO communication parameter	Configuration Transmit PDO 10		--	--
	0x1	COB ID used by PDO	Configuration Transmit PDO 10	0x282 + node ID	--	--
	0x2	transmission type	Configuration Transmit PDO 10	1	yes	at once
	0x3	inhibit time	Configuration Transmit PDO 10	0	yes	at once
	0x4	reserved	Configuration Transmit PDO 10	0	no	--
	0x5	event time	Configuration Transmit PDO 10	0	yes	at once
0x1A00		Transmit PDO mapping	Mapping Transmit PDO 1		yes	after PreOp
	0x1	PDO mapping	Mapping Transmit PDO 1	0x6000 0108	yes	after PreOp
	0x2	PDO mapping	Mapping Transmit PDO 1	0x6000 0208	yes	after PreOp
	0x3	PDO mapping	Mapping Transmit PDO 1	0x2020 0108	yes	after PreOp
	0x4	PDO mapping	Mapping Transmit PDO 1	0x2020 0208	yes	after PreOp
	0x5	PDO mapping	Mapping Transmit PDO 1	0x2021 0108	yes	after PreOp
	0x6	PDO mapping	Mapping Transmit PDO 1	0x2021 0208	yes	after PreOp
	0x7	PDO mapping	Mapping Transmit PDO 1	0x2025 0108	yes	after PreOp
0x1A01		Transmit PDO mapping	Mapping Transmit PDO 2		yes	after PreOp
	0x1	PDO mapping	Mapping Transmit PDO 2	0x6404 0110	yes	after PreOp
	0x2	PDO mapping	Mapping Transmit PDO 2	0x6404 0210	yes	after PreOp
	0x3	PDO mapping	Mapping Transmit PDO 2	0x6404 0310	yes	after PreOp
	0x4	PDO mapping	Mapping Transmit PDO 2	0x6404 0410	yes	after PreOp
0x1A02		Transmit PDO mapping	Mapping Transmit PDO 3		yes	after PreOp
	0x1	PDO mapping	Mapping Transmit PDO 3	0x2030 0110	yes	after PreOp
	0x2	PDO mapping	Mapping Transmit PDO 3	0x2030 0210	yes	after PreOp
	0x3	PDO mapping	Mapping Transmit PDO 3	0x2002 0110	yes	after PreOp
	0x4	PDO mapping	Mapping Transmit PDO 3	0x2002 0210	yes	after PreOp
	0x5	PDO mapping	Mapping Transmit PDO 3	0	yes	after PreOp

© ifm electronic gmbh

Object directory		Parameter description	Parameter for	Preset parameter value	Change saved automatically?	When does the change become effective?
Index	Sub-Idx					
0x1A03		Transmit PDO mapping	Mapping Transmit PDO 4		yes	after PreOp
	0x1	PDO mapping	Mapping Transmit PDO 4	0x2012 0120	yes	after PreOp
	0x2	PDO mapping	Mapping Transmit PDO 4	0x2012 0220	yes	after PreOp
	0x3	PDO mapping	Mapping Transmit PDO 4	0	yes	after PreOp
0x1A04		Transmit PDO mapping	Mapping Transmit PDO 5		yes	after PreOp
	0x1	PDO mapping	Mapping Transmit PDO 5	0x2012 0320	yes	after PreOp
	0x2	PDO mapping	Mapping Transmit PDO 5	0x2012 0420	yes	after PreOp
	0x3	PDO mapping	Mapping Transmit PDO 5	0	yes	after PreOp
0x1A05		Transmit PDO mapping	Mapping Transmit PDO 6		yes	after PreOp
	0x1	PDO mapping	Mapping Transmit PDO 6	0x2014 0110	yes	after PreOp
	0x2	PDO mapping	Mapping Transmit PDO 6	0x2014 0210	yes	after PreOp
	0x3	PDO mapping	Mapping Transmit PDO 6	0x2014 0310	yes	after PreOp
	0x4	PDO mapping	Mapping Transmit PDO 6	0x2014 0410	yes	after PreOp
	0x5	PDO mapping	Mapping Transmit PDO 6	0	yes	after PreOp
0x1A06		Transmit PDO mapping	Mapping Transmit PDO 7		yes	after PreOp
	0x1	PDO mapping	Mapping Transmit PDO 7	0x2015 0120	yes	after PreOp
	0x2	PDO mapping	Mapping Transmit PDO 7	0x2015 0220	yes	after PreOp
	0x3	PDO mapping	Mapping Transmit PDO 7	0	yes	after PreOp
0x1A07		Transmit PDO mapping	Mapping Transmit PDO 8		yes	after PreOp
	0x1	PDO mapping	Mapping Transmit PDO 8	0x2015 0320	yes	after PreOp
	0x2	PDO mapping	Mapping Transmit PDO 8	0x2015 0420	yes	after PreOp
	0x3	PDO mapping	Mapping Transmit PDO 8	0	yes	after PreOp
0x1A08		Transmit PDO mapping	Mapping Transmit PDO 9		yes	after PreOp
	0x1	PDO mapping	Mapping Transmit PDO 9	0x2022 0108	yes	after PreOp
	0x2	PDO mapping	Mapping Transmit PDO 9	0x2023 0108	yes	after PreOp
	0x3	PDO mapping	Mapping Transmit PDO 9	0x2024 0108	yes	after PreOp

© ifm electronic gmbH www.ifm.com

Object directory		Parameter description	Parameter for	Preset parameter value	Change saved automatically?	When does the change become effective?
Index	Sub-Idx					
	0x4	PDO mapping	Mapping Transmit PDO 9	0	yes	after PreOp
0x1A09		Transmit PDO mapping	Mapping Transmit PDO 10		yes	after PreOp
	0x1	PDO mapping	Mapping Transmit PDO 10	0x2040 0110	yes	after PreOp
	0x2	PDO mapping	Mapping Transmit PDO 10	0x2041 0110	yes	after PreOp
	0x3	PDO mapping	Mapping Transmit PDO 10	0x2041 0210	yes	after PreOp
	0x4	PDO mapping	Mapping Transmit PDO 10	0x2050 0010	yes	after PreOp
	0x5	PDO mapping	Mapping Transmit PDO 10	0	yes	after PreOp

The life time factor 0 is interpreted as 1.
The first guard protocol is interpreted as "start guarding" even if guarding is not yet active at that time (guard time =0).

© ifm electronic gmbh

Object directory manufacturer-specific objects (index 0x2000...0x6FFF), overview

15978

Object directory Index	Parameter description	Parameter for	Preset parameter value	Change saved automatically?	When does the change become effective?
0x2000	I/O configuration	IN00…IN11 IN12…IN15	10 01	yes	after PreOp
0x2001	PWM frequency	OUT00…OUT11	100	yes	after PreOp
0x2002	Current value	OUT00…OUT01	0	yes	after PreOp
0x2004	P-value	OUT00…OUT01	30	yes	after PreOp
0x2005	I-value	OUT00…OUT01	20	yes	after PreOp
0x2006	PWM dither frequency	OUT00…OUT11	0	yes	after PreOp
0x2007	PWM dither value	OUT00…OUT11	0	yes	after PreOp
0x2012	Input period duration	IN12…IN15	0	yes	after PreOp
0x2013	Number of periods	IN12…IN15	0	yes	after PreOp
0x2014	Period ratio value	IN12…IN15	0	yes	after PreOp
0x2015	Input frequency	IN12…IN15	0.0	yes	after PreOp
0x2016	Timebase frequency	IN12…IN15	50	yes	after PreOp
0x2020	Input short to VBBS	IN00…IN11	0	yes	after PreOp
0x2021	Input wire break	IN00…IN11	0	yes	after PreOp
0x2022	Output short circuit	OUT00…OUT07	0	yes	after PreOp
0x2023	Output open circuit	OUT00…OUT07	0	yes	after PreOp
0x2024	Output overload	OUT00…OUT01	0	yes	after PreOp
0x2025	Input overcurrent	IN00…IN03	0	yes	after PreOp
0x2030	Input resistance	IN04…IN05	0	yes	after PreOp
0x2040	Supply voltage	VBBS	0	yes	after PreOp
0x2041	Supply voltage	VBB1, VBB2	0	yes	after PreOp
0x2050	Device temperature	device	0	yes	after PreOp
0x20F0 != 0x20F1 *)	Node ID	device	124	if both are identical	after reset
0x20F2 != 0x20F3 *)	Baud rate	device	3	if both are identical	after reset
0x20F4	Autostart	device	0	yes	at once
0x6000	Binary inputs	IN00…IN07 IN08…IN15	0	yes	after PreOp
0x6200	Binary output	OUT00…OUT07 OUT08…OUT15	0	yes	after PreOp
0x6404	Analogue inputs	IN00…IN03	---	---	---
0x6414	Analogue outputs	OUT00…OUT11	---	---	---

*) values must be identical!

© ifm electronic gmbh

Object directory parameter tables, details

Contents

15982

Object directory obligatory objects (index 0x1000…0x1FFF), details

15985

Index	S-Idx	Parameter name	Data type		Default	Details
0x1000		Device type	ro	UDINT	0x000F 0191	Device type
0x1001		Error register	ro	USINT	0	Error register bitcoded to profile 301 permissible values = 0b0000 0000 = no error 0b0000 0001 = generic error 0b0001 0000 = communication error 0b1000 0000 = manufacturer specific
0x1018	0x0	Device identification Number of entries	ro	USINT	0x04	Device identification
	0x1	Vendor ID	ro	UDINT	0x0690 7501	Vendor ID of the device according to CiA specification
	0x2	Product code	ro	STRING	0	Product code of the device
	0x3	Revision number	ro	UDINT	0	Revision number of the device
	0x4	Serial number	ro	UDINT	0	Serial number of the device

Legend:
Data type: ro = read only / rw = read and write / wo = write only

© ifm electronic gmbh

Object directory optional objects (index 0x1000…0x10FFF), details

16603

Index	S-Idx	Parameter name	Data type		Default	Details
0x1003	0x0	Predefined error field Number of entries	rw	UDINT	0	An error list with 4 entries is supported
	0x1	Error history	ro	UDINT	0	Error occurred, coded according to EMCY list The last error is indicated in the sub-index 1
	0x2	Error history	ro	UDINT	0	Error occurred, coded according to EMCY list
	0x3	Error history	ro	UDINT	0	Error occurred, coded according to EMCY list
	0x4	Error history	ro	UDINT	0	Error occurred, coded according to EMCY list
	0x5	Error history	ro	UDINT	0	Error occurred, coded according to EMCY list
0x1005		COB-ID synch message	rw	UDINT	0x0000 0080	identifier of the synch message Bit 30 = 0 ⇨ device generates no synch message Bit 30 = 1 ⇨ device generates a synch message Bit 29 = 0 ⇨ 11-bit ID Bit 29 = 1 ⇨ ID = 0x80 + node ID
0x1006		Communication cycle period	rw	UDINT	0	Max. time between 2 synch. objects in [µs] Control resolution = 1 ms
0x1008		Manufacturer device name	ro	STRING	EXB01	Device designation
0x1009		Manufacturer hardware version	ro	STRING	V00.00.00	Hardware version
0x100A		Manufacturer software version	ro	STRING	V00.00.00	Software version
0x100C		Guard time	rw	UINT	0	Within this time in [ms] the device expects a "node guarding" of the master of the system. 0 = this function is not supported. The monitoring of the node with "node guarding" or "heartbeat" is only possible as an alternative.
0x100D		Lifetime factor	rw	USINT	0	If for "guard time" • "lifetime" no "node guarding" was received, the device switches off the outputs. The device changes the CANopen status to PREOP. Default: "guard time" • "lifetime" = 0...65535
0x1010	0x0	Store parameters Largest sub-index supported	ro	USINT	0x01	Number of "save options"
	0x1	Save all parameters	rw	UDINT	2	automatic saving of all parameters changed 0 = AutoSave OFF 2 = AutoSave ON
0x1011	0x0	Restore default parameters Largest sub-index supported	ro	USINT	0x01	Number of "restore options"
	0x1	Restore all default parameters	rw	UDINT	0x01	If the string "load" is entered here, the default parameters set at the factory are restored and become valid after the next reset.
0x1014		COBId Emergency	rw	UDINT	0x80 + node ID	Bit 31 = 0 ⇨ EMCY is valid Bit 31 = 1 ⇨EMCY is not valid Bit 29 = 0 ⇨ 11-bit ID Bit 29 = 1 ⇨ ID = 0x80 + node ID CAN identifier can be changed by the user.
0x1016	0x0	Consumer heartbeat times Nums consumer heartbeat time	ro	USINT	0x01	Heartbeat monitoring time for the node Number of devices monitored = 1

© ifm electronic gmbh

Index	S-Idx	Parameter name	Data type		Default	Details
	0x1	Consumer heartbeat time	rw	UDINT	0	Heartbeat monitoring time for the node Format: 0x0nnttttt tttt = monitoring time [ms] nn = node number if nn=0 or tttt=0 ⇨ no monitoring The monitoring of the node with "node guarding" or "heartbeat" is only possible as an alternative.
0x1017		producer heartbeat time	rw	UINT	0	Time interval [ms] during which the device generates a producer heartbeat

Legend:
Data type: ro = read only / rw = read and write / wo = write only

© ifm electronic gmbh

Object directory optional objects (index 0x1400…0x14FF), details

16604

Receive PDO communication parameters

Index	S-Idx	Parameter name	Data type		Default	Details
0x1400	0x0	Receive PDO Communication Parameter Number of entries	ro	USINT	0x02	Receive PDO 1: binary outputs number of entries = 2
	0x1	COBID used by PDO	rw	UDINT	0x200 + node ID	CAN-ID of the first read PDO Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; outputs are only updated after "n" synch objects. n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; outputs are updated immediately 0xFF = asynch device profile event; outputs are updated immediately
0x1401	0x0	Receive PDO Communication Parameter Number of entries	ro	USINT	0x02	Receive PDO 2: PWM outputs number of entries = 2
	0x1	COBID used by PDO	rw	UDINT	0x300 + node ID	CAN-ID of the first read PDO Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; outputs are only updated after "n" synch objects. n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; outputs are updated immediately 0xFF = asynch device profile event; outputs are updated immediately
0x1402	0x0	Receive PDO Communication Parameter Number of entries	ro	USINT	0x02	Receive PDO 3: PWM outputs number of entries = 2
	0x1	COBID used by PDO	rw	UDINT	0x400 + node ID	CAN-ID of the first read PDO Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; outputs are only updated after "n" synch objects. n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; outputs are updated immediately 0xFF = asynch device profile event; outputs are updated immediately
0x1403	0x0	Receive PDO Communication Parameter Number of entries	ro	USINT	0x02	Receive PDO 4: PWM outputs number of entries = 2
	0x1	COBID used by PDO	rw	UDINT	0x500 + node ID	CAN-ID of the first read PDO Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid

© ifm electronic gmbh www.ifm.com

Index	S-Idx	Parameter name	Data type		Default	Details
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; outputs are only updated after "n" synch objects. n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; outputs are updated immediately 0xFF = asynch device profile event; outputs are updated immediately

Legend:
Data type: ro = read only / rw = read and write / wo = write only

© ifm electronic gmbh

Object directory optional objects (index 0x1600…0x16FF), details

16605

Receive PDO mapping

Index	S-Idx	Parameter name	Data type		Default	Details
0x1600	0x0	Receive PDO mapping Number of mapped objects in PDO	rw	USINT	0x02	Mapping read PDO 1: binary outputs number of integrated application objects = 2

	0x1	PDO mapping	ro	UDINT	0x6200 0108	1 byte in index 0x6200, SubIndex 01
						Binary inputs IN00…IN07
						`0b---- ---X = IN00`
						`0b---- --X- = IN01`
						`0b---- -X-- = IN02`
						`0b---- X--- = IN03`
						`0b---X ---- = IN04`
						`0b--X- ---- = IN05`
						`0b-X-- ---- = IN06`
						`0bX--- ---- = IN07`

	0x2	PDO mapping	ro	UDINT	0x6200 0208	1 byte in index 0x6200, SubIndex 02
						Binary inputs IN08…IN15
						`0b---- ---X = IN08`
						`0b---- --X- = IN09`
						`0b---- -X-- = IN10`
						`0b---- X--- = IN11`
						`0b---X ---- = IN12`
						`0b--X- ---- = IN13`
						`0b-X-- ---- = IN14`
						`0bX--- ---- = IN15`

0x1601	0x0	Receive PDO mapping Number of mapped objects in PDO	rw	USINT	0x04	Mapping read PDO 2: PWM outputs OUT00…OUT03 number of integrated application objects = 4
	0x1	PDO mapping	rw	UDINT	0x6414 0110	PWM output OUT00 Index 0x6414, SubIndex 0x1 contains the preset value of the PWM output OUT00, the value is interpreted as duty cycle in ‰ or as target current value (depending on the configuration index 0x2000).
	0x2	PDO mapping	rw	UDINT	0x6414 0210	PWM output OUT01 Index 0x6414, SubIndex 0x2 contains the preset value of the PWM output OUT01, the value is interpreted as duty cycle in ‰ or as target current value (depending on the configuration index 0x2000).
	0x3	PDO mapping	rw	UDINT	0x6414 0310	PWM output OUT02 Index 0x6414, SubIndex 0x3 contains the preset value of the PWM output OUT02, the value is interpreted as duty cycle in ‰ or as target current value (depending on the configuration index 0x2000).
	0x4	PDO mapping	rw	UDINT	0x6414 0410	PWM output OUT03 Index 0x6414, SubIndex 0x4 contains the preset value of the PWM output OUT03, the value is interpreted as duty cycle in ‰ or as target current value (depending on the configuration index 0x2000).
0x1602	0x0	Receive PDO mapping Number of mapped objects in PDO	rw	USINT	0x04	Mapping read PDO 3: PWM outputs OUT04…OUT07 number of integrated application objects = 4
	0x1	PDO mapping	rw	UDINT	0x6414 0510	PWM output OUT04 Index 0x6414, SubIndex 0x5 contains the preset value of the PWM output OUT04, the value is interpreted as duty cycle in ‰ or as target current value (depending on the configuration index 0x2000).

© ifm electronic gmbh

Index	S-Idx	Parameter name	Data type		Default	Details
	0x2	PDO mapping	rw	UDINT	0x6414 0610	PWM output OUT05 Index 0x6414, SubIndex 0x6 contains the preset value of the PWM output OUT05, the value is interpreted as duty cycle in ‰ or as target current value (depending on the configuration index 0x2000).
	0x3	PDO mapping	rw	UDINT	0x6414 0710	PWM output OUT06 Index 0x6414, SubIndex 0x7 contains the preset value of the PWM output OUT06, the value is interpreted as duty cycle in ‰ or as target current value (depending on the configuration index 0x2000).
	0x4	PDO mapping	rw	UDINT	0x6414 0810	PWM output OUT07 Index 0x6414, SubIndex 0x8 contains the preset value of the PWM output OUT07, the value is interpreted as duty cycle in ‰ or as target current value (depending on the configuration index 0x2000).
	0x5	PDO mapping	rw	UDINT	0	reserve
0x1603	0x0	Receive PDO mapping Number of mapped objects in PDO	rw	USINT	0x04	Mapping read PDO 4: PWM outputs OUT08…OUT11 number of integrated application objects = 4
	0x1	PDO mapping	rw	UDINT	0x6414 0910	PWM output OUT08 Index 0x6414, SubIndex 0x9 contains the preset value of the PWM output OUT08, the value is interpreted as duty cycle in ‰ or as target current value (depending on the configuration index 0x2000).
	0x2	PDO mapping	rw	UDINT	0x6414 0A10	PWM output OUT09 Index 0x6414, SubIndex 0xA contains the preset value of the PWM output OUT09, the value is interpreted as duty cycle in ‰ or as target current value (depending on the configuration index 0x2000).
	0x3	PDO mapping	rw	UDINT	0x6414 0B10	PWM output OUT10 Index 0x6414, SubIndex 0xB contains the preset value of the PWM output OUT10, the value is interpreted as duty cycle in ‰ or as target current value (depending on the configuration index 0x2000).
	0x4	PDO mapping	rw	UDINT	0x6414 0C10	PWM output OUT11 Index 0x6414, SubIndex 0xC contains the preset value of the PWM output OUT11, the value is interpreted as duty cycle in ‰ or as target current value (depending on the configuration index 0x2000).
	0x5	PDO mapping	rw	UDINT	0	reserve

Legend:
Data type: ro = read only / rw = read and write / wo = write only

© ifm electronic gmbh

Object directory optional objects (index 0x1800…0x18FF), details

16606

Transmit PDO communication parameters

Index	S-Idx	Parameter name	Data type		Default	Details
0x1800	0x0	Transmit PDO Communication Parameter Number of entries	ro	USINT	0x05	configuration transmit PDO 1 number of entries = 5
	0x1	COBID used by PDO	rw	UDINT	0x180 + node ID	CAN ID of the transmit PDO 1 Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; values are only transmitted after "n" synch objects n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; Values are immediately transferred 0xFF = asynch device profile event; Values are immediately transferred
	0x3	inhibit time	rw	UINT	0	delay time in the transmission type "asynch" before the PDO is transmitted again at the earliest. (0...65535 • 100 µs)
	0x4	reserved	rw	USINT	0	reserve
	0x5	event time	rw	UINT	0	max. transfer break in the transmission type "asynch" (0...65535 ms) When this time has elapsed, the PDO is transferred even if the appl. event has not occurred.
0x1801	0x0	Transmit PDO Communication Parameter Number of entries	ro	USINT	0x05	configuration transmit PDO 2 number of entries = 5
	0x1	COBID used by PDO	rw	UDINT	0x280 + node ID	CAN ID of the transmit PDO 2 Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; values are only transmitted after "n" synch objects n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; values are immediately transferred 0xFF = asynch device profile event; values are immediately transferred
	0x3	inhibit time	rw	UINT	0	delay time in the transmission type "asynch" before the PDO is transmitted again at the earliest. (0...65535 • 100 µs)
	0x4	reserved	rw	USINT	0	reserve
	0x5	event time	rw	UINT	0	max. transfer break in the transmission type "asynch" (0...65535 ms) When this time has elapsed, the PDO is transferred even if the appl. event has not occurred.
0x1802	0x0	Transmit PDO Communication Parameter Number of entries	ro	USINT	0x05	configuration transmit PDO 3 number of entries = 5
	0x1	COBID used by PDO	rw	UDINT	0x380 + node ID	CAN ID of the transmit PDO 3 Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid

© ifm electronic gmbh

Index	S-Idx	Parameter name	Data type		Default	Details
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; values are only transmitted after "n" synch objects n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; values are immediately transferred 0xFF = asynch device profile event; values are immediately transferred
	0x3	inhibit time	rw	UINT	0	delay time in the transmission type "asynch" before the PDO is transmitted again at the earliest. (0...65535 • 100 µs)
	0x4	reserved	rw	USINT	0	reserve
	0x5	event time	rw	UINT	0	max. transfer break in the transmission type "asynch" (0...65535 ms) When this time has elapsed, the PDO is transferred even if the appl. event has not occurred.
0x1803	0x0	Transmit PDO Communication Parameter Number of entries	ro	USINT	0x05	configuration transmit PDO 4 number of entries = 5
	0x1	COBID used by PDO	rw	UDINT	0x480 + node ID	CAN ID of the transmit PDO 4 Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; values are only transmitted after "n" synch objects n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; values are immediately transferred 0xFF = asynch device profile event; values are immediately transferred
	0x3	inhibit time	rw	UINT	0	delay time in the transmission type "asynch" before the PDO is transmitted again at the earliest. (0...65535 • 100 µs)
	0x4	reserved	rw	USINT	0	reserve
	0x5	event time	rw	UINT	0	max. transfer break in the transmission type "asynch" (0...65535 ms) When this time has elapsed, the PDO is transferred even if the appl. event has not occurred.
0x1804	0x0	Transmit PDO Communication Parameter Number of entries	ro	USINT	0x05	configuration transmit PDO 5 number of entries = 5
	0x1	COBID used by PDO	rw	UDINT	0x181 + node ID	CAN ID of the transmit PDO 5 Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; values are only transmitted after "n" synch objects n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; values are immediately transferred 0xFF = asynch device profile event; values are immediately transferred

© ifm electronic gmbh

Index	S-Idx	Parameter name	Data type		Default	Details
	0x3	inhibit time	rw	UINT	0	delay time in the transmission type "asynch" before the PDO is transmitted again at the earliest. (0...65535 • 100 µs)
	0x4	reserved	rw	USINT	0	reserve
	0x5	event time	rw	UINT	0	max. transfer break in the transmission type "asynch" (0...65535 ms) When this time has elapsed, the PDO is transferred even if the appl. event has not occurred.
0x1805	0x0	Transmit PDO Communication Parameter Number of entries	ro	USINT	0x05	configuration transmit PDO 6 number of entries = 5
	0x1	COBID used by PDO	rw	UDINT	0x281 + node ID	CAN ID of the transmit PDO 6 Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; values are only transmitted after "n" synch objects n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; values are immediately transferred 0xFF = asynch device profile event; values are immediately transferred
	0x3	inhibit time	rw	UINT	0	delay time in the transmission type "asynch" before the PDO is transmitted again at the earliest. (0...65535 • 100 µs)
	0x4	reserved	rw	USINT	0	reserve
	0x5	event time	rw	UINT	0	max. transfer break in the transmission type "asynch" (0...65535 ms) When this time has elapsed, the PDO is transferred even if the appl. event has not occurred.
0x1806	0x0	Transmit PDO Communication Parameter Number of entries	ro	USINT	0x05	configuration transmit PDO 7 number of entries = 5
	0x1	COBID used by PDO	rw	UDINT	0x381 + node ID	CAN ID of the transmit PDO 7 Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; values are only transmitted after "n" synch objects n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; values are immediately transferred 0xFF = asynch device profile event; values are immediately transferred
	0x3	inhibit time	rw	UINT	0	delay time in the transmission type "asynch" before the PDO is transmitted again at the earliest. (0...65535 • 100 µs)
	0x4	reserved	rw	USINT	0	reserve
	0x5	event time	rw	UINT	0	max. transfer break in the transmission type "asynch" (0...65535 ms) When this time has elapsed, the PDO is transferred even if the appl. event has not occurred.
0x1807	0x0	Transmit PDO Communication Parameter Number of entries	ro	USINT	0x05	configuration transmit PDO 8 number of entries = 5

© ifm electronic gmbh

Index	S-Idx	Parameter name	Data type		Default	Details
	0x1	COBID used by PDO	rw	UDINT	0x481 + node ID	CAN ID of the transmit PDO 8 Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; values are only transmitted after "n" synch objects n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; values are immediately transferred 0xFF = asynch device profile event; values are immediately transferred
	0x3	inhibit time	rw	UINT	0	delay time in the transmission type "asynch" before the PDO is transmitted again at the earliest. (0...65535 • 100 µs)
	0x4	reserved	rw	USINT	0	reserve
	0x5	event time	rw	UINT	0	max. transfer break in the transmission type "asynch" (0...65535 ms) When this time has elapsed, the PDO is transferred even if the appl. event has not occurred.
0x1808	0x0	Transmit PDO Communication Parameter Number of entries	ro	USINT	0x05	configuration transmit PDO 9 number of entries = 5
	0x1	COBID used by PDO	rw	UDINT	0x181 + node ID	CAN ID of the transmit PDO 9 Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; values are only transmitted after "n" synch objects n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; values are immediately transferred 0xFF = asynch device profile event; values are immediately transferred
	0x3	inhibit time	rw	UINT	0	delay time in the transmission type "asynch" before the PDO is transmitted again at the earliest. (0...65535 • 100 µs)
	0x4	reserved	rw	USINT	0	reserve
	0x5	event time	rw	UINT	0	max. transfer break in the transmission type "asynch" (0...65535 ms) When this time has elapsed, the PDO is transferred even if the appl. event has not occurred.
0x1809	0x0	Transmit PDO Communication Parameter Number of entries	ro	USINT	0x05	configuration transmit PDO 10 number of entries = 5
	0x1	COBID used by PDO	rw	UDINT	0x281 + node ID	CAN ID of the transmit PDO 10 Bit 31 = 0 ⇨ PDO is valid Bit 31 = 1 ⇨ PDO is not valid

© ifm electronic gmbh

Index	S-Idx	Parameter name	Data type		Default	Details
	0x2	transmission type	rw	USINT	0x01	0x00 = synch acyclic 0x01...0xF0 = synch cyclic; values are only transmitted after "n" synch objects n = 1…240 = 0x01...0xF0 0xFC/0xFD not implemented 0xFE = asynch man. spec. event; values are immediately transferred 0xFF = asynch device profile event; values are immediately transferred
	0x3	inhibit time	rw	UINT	0	delay time in the transmission type "asynch" before the PDO is transmitted again at the earliest. (0...65535 • 100 µs)
	0x4	reserved	rw	USINT	0	reserve
	0x5	event time	rw	UINT	0	max. transfer break in the transmission type "asynch" (0...65535 ms) When this time has elapsed, the PDO is transferred even if the appl. event has not occurred.

Legend:
Data type: ro = read only / rw = read and write / wo = write only

© ifm electronic gmbh

Object directory optional objects (index 0x1A00…0x1AFF), details

16607

Transmit PDO mapping

Index	S-Idx	Parameter name	Data type		Default	Details
0x1A00	0x0	Transmit PDO mapping Number of mapped objects in PDO	rw	USINT	0x07	mapping transmit PDO 1 number of integrated application objects = 7
	0x1	PDO mapping	rw	UDINT	0x6000 0108	Index 0x6000, SubIndex 0x1 binary inputs 00...07: actual values (bit coded)
	0x2	PDO mapping	rw	UDINT	0x6000 0208	Index 0x6000, SubIndex 0x2 binary inputs 08...15: actual values (bit coded)
	0x3	PDO mapping	rw	UDINT	0x2020 0108	Index 0x2020, SubIndex 0x1 inputs 00...07: flag "short circuit" (bit coded)
	0x4	PDO mapping	rw	UDINT	0x2020 0208	Index 0x2020, SubIndex 0x2 inputs 08...11: flag "short circuit" (bit coded)
	0x5	PDO mapping	rw	UDINT	0x2021 0108	Index 0x2021, SubIndex 0x1 inputs 00...07: flag "wire break" (bit coded)
	0x6	PDO mapping	rw	UDINT	0x2021 0208	Index 0x2021, SubIndex 0x2 inputs 08...11: flag "wire break" (bit coded)
	0x7	PDO mapping	rw	UDINT	0x2025 0108	Index 0x2025, SubIndex 0x1 inputs 00...03: flag "overload" (bit coded)
0x1A01	0x0	Transmit PDO mapping Number of mapped objects in PDO	rw	USINT	0x04	mapping transmit PDO 2 (analogue inputs) number of integrated application objects = 4
	0x1	PDO mapping	rw	UDINT	0x6404 0110	Index 0x6404, SubIndex 0x1 analogue input 00: actual value (depending on the configuration 0x2000)
	0x2	PDO mapping	rw	UDINT	0x6404 0210	Index 0x6404, SubIndex 0x2 analogue input 01: actual value (depending on the configuration 0x2000)
	0x3	PDO mapping	rw	UDINT	0x6404 0310	Index 0x6404, SubIndex 0x3 analogue input 02: actual value (depending on the configuration 0x2000)
	0x4	PDO mapping	rw	UDINT	0x6404 0410	Index 0x6404, SubIndex 0x4 analogue input 03: actual value (depending on the configuration 0x2000)
0x1A02	0x0	Transmit PDO mapping Number of mapped objects in PDO	rw	USINT	0x04	mapping transmit PDO 3 number of integrated application objects = 4
	0x1	PDO mapping	rw	UDINT	0x2030 0110	Index 0x2030, SubIndex 0x1 input 04: actual resistor value
	0x2	PDO mapping	rw	UDINT	0x2030 0210	Index 0x2030, SubIndex 0x2 input 05: actual resistor value
	0x3	PDO mapping	rw	UDINT	0x2002 0110	Index 0x2002, SubIndex 0x1 output 00: actual current value
	0x4	PDO mapping	rw	UDINT	0x2002 0210	Index 0x2002, SubIndex 0x2 output 01: actual current value
	0x5	PDO mapping	rw	UDINT	0	reserve
0x1A03	0x0	Transmit PDO mapping Number of mapped objects in PDO	rw	USINT	0x02	mapping transmit PDO 4 (period time IN12...IN13) number of integrated application objects = 2
	0x1	PDO mapping	rw	UDINT	0x2012 0120	Index 0x2012, SubIndex 0x1 frequency input 12: period time of the signal
	0x2	PDO mapping	rw	UDINT	0x2012 0220	Index 0x2012, SubIndex 0x2 frequency input 13: period time of the signal

© ifm electronic gmbh

Index	S-Idx	Parameter name	Data type		Default	Details
	0x3	PDO mapping	rw	UDINT	0	reserve
0x1A04	0x0	Transmit PDO mapping Number of mapped objects in PDO	rw	USINT	0x02	mapping transmit PDO 5 (period time IN14...IN15) number of integrated application objects = 2
	0x1	PDO mapping	rw	UDINT	0x2012 0320	Index 0x2012, SubIndex 0x3 frequency input 14: period time of the signal
	0x2	PDO mapping	rw	UDINT	0x2012 0420	Index 0x2012, SubIndex 0x4 frequency input 15: period time of the signal
	0x3	PDO mapping	rw	UDINT	0	reserve
0x1A05	0x0	Transmit PDO mapping Number of mapped objects in PDO	rw	USINT	0x04	mapping transmit PDO 6 (duty cycle of the signal on the frequency input IN12...IN15) number of integrated application objects = 4
	0x1	PDO mapping	rw	UDINT	0x2014 0110	Index 0x2014, SubIndex 0x1 frequency input 12: duty cycle of the signal in ‰
	0x2	PDO mapping	rw	UDINT	0x2014 0210	Index 0x2014, SubIndex 0x2 frequency input 13: duty cycle of the signal in ‰
	0x3	PDO mapping	rw	UDINT	0x2014 0310	Index 0x2014, SubIndex 0x3 frequency input 14: duty cycle of the signal in ‰
	0x4	PDO mapping	rw	UDINT	0x2014 0410	Index 0x2014, SubIndex 0x4 frequency input 15: duty cycle of the signal in ‰
	0x5	PDO mapping	rw	UDINT	0	reserve
0x1A06	0x0	Transmit PDO mapping Number of mapped objects in PDO	rw	USINT	0x02	mapping transmit PDO 7 (frequency on IN12...IN13) number of integrated application objects = 2
	0x1	PDO mapping	rw	UDINT	0x2015 0120	Index 0x2015, SubIndex 0x1 frequency input 12: frequency value of the signal in Hz
	0x2	PDO mapping	rw	UDINT	0x2015 0220	Index 0x2015, SubIndex 0x2 frequency input 13: frequency value of the signal in Hz
	0x3	PDO mapping	rw	UDINT	0	reserve
0x1A07	0x0	Transmit PDO mapping Number of mapped objects in PDO	rw	USINT	0x02	mapping transmit PDO 8 (frequency on IN14...IN15) number of integrated application objects = 2
	0x1	PDO mapping	rw	UDINT	0x2015 0320	Index 0x2015, SubIndex 0x3 frequency input 14: frequency value of the signal in Hz
	0x2	PDO mapping	rw	UDINT	0x2015 0420	Index 0x2015, SubIndex 0x4 frequency input 15: frequency value of the signal in Hz
	0x3	PDO mapping	rw	UDINT	0	reserve
0x1A08	0x0	Transmit PDO mapping Number of mapped objects in PDO	rw	USINT	0x03	mapping transmit PDO 9 (error flag OUT00...OUT07) number of integrated application objects = 3
	0x1	PDO mapping	rw	UDINT	0x2022 0108	Index 0x2022, SubIndex 0x1 OUT00...OUT07: flag "short circuit" (bit coded)
	0x2	PDO mapping	rw	UDINT	0x2023 0108	Index 0x2023, SubIndex 0x1 OUT00...OUT07: flag "wire break" (bit coded)
	0x3	PDO mapping	rw	UDINT	0x2024 0108	Index 0x2024, SubIndex 0x1 OUT00...OUT01: flag "overload" (bit coded)
	0x4	PDO mapping	rw	UDINT	0	reserve
0x1A09	0x0	Transmit PDO mapping Number of mapped objects in PDO	rw	USINT	0x04	mapping transmit PDO 10 (system flag) number of integrated application objects = 4
	0x1	PDO mapping	rw	UDINT	0x2040 0110	Index 0x2040, SubIndex 0x1 supply voltage of the system VBBS

© ifm electronic gmbh www.ifm.com

Index	S-Idx	Parameter name	Data type		Default	Details
	0x2	PDO mapping	rw	UDINT	0x2041 0110	Index 0x2041, SubIndex 0x1 output supply voltage VBB1
	0x3	PDO mapping	rw	UDINT	0x2041 0210	Index 0x2041, SubIndex 0x2 output supply voltage VBB2
	0x4	PDO mapping	rw	UDINT	0x2050 0010	Index 0x2050, SubIndex 0x0 system temperature in °C
	0x5	PDO mapping	rw	UDINT	0	reserve

Legend:
Data type: ro = read only / rw = read and write / wo = write only

© ifm electronic gmbh

Object directory manufacturer-specific objects (index 0x2000...0x6FFF), details

15983

Index	S-Idx	Parameter name	Data type		Default	Details	
0x2000	0x0	IO configuration Largest sub-index supported	ro	USINT	32	configuration inputs/outputs largest supported sub-index = 32	
	0x1	Configuration IN00	rw	USINT	10	0 = 0x00 3 = 0x03 6 = 0x06 7 = 0x07 9 = 0x09 10 = 0x0A 11 = 0x0B 12 = 0x0C	off Input IN00 0...10 000 mV ratiometric 0...1000 ‰ 0...20 000 µA 0...32 000 mV binary plus switched binary plus switched with diagnosis binary minus switched
	0x2	Configuration IN01	rw	USINT	10	0 = 0x00 3 = 0x03 6 = 0x06 7 = 0x07 9 = 0x09 10 = 0x0A 11 = 0x0B 12 = 0x0C	off Input IN01 0...10 000 mV ratiometric 0...1000 ‰ 0...20 000 µA 0...32 000 mV binary plus switched binary plus switched with diagnosis binary minus switched
	0x3	Configuration IN02	rw	USINT	10	0 = 0x00 3 = 0x03 6 = 0x06 7 = 0x07 9 = 0x09 10 = 0x0A 11 = 0x0B 12 = 0x0C	off Input IN02 0...10 000 mV ratiometric 0...1000 ‰ 0...20 000 µA 0...32 000 mV binary plus switched binary plus switched with diagnosis binary minus switched
	0x4	Configuration IN03	rw	USINT	10	0 = 0x00 3 = 0x03 6 = 0x06 7 = 0x07 9 = 0x09 10 = 0x0A 11 = 0x0B 12 = 0x0C	off Input IN03 0...10 000 mV ratiometric 0...1000 ‰ 0...20 000 µA 0...32 000 mV binary plus switched binary plus switched with diagnosis binary minus switched
0x2000	0x5	Configuration IN04	rw	USINT	10	0 = 0x00 10 = 0x0A 11 = 0x0B 18 = 0x12	off Input IN04 binary plus switched binary plus switched with diagnosis 16…30 000 Ohm
	0x6	Configuration IN05	rw	USINT	10	0 = 0x00 10 = 0x0A 11 = 0x0B 18 = 0x12	off Input IN05 binary plus switched binary plus switched with diagnosis 16…30 000 Ohm
0x2000	0x7	Configuration IN06	rw	USINT	10	0 = 0x00 10 = 0x0A 11 = 0x0B	off Input IN06 binary plus switched binary plus switched with diagnosis
	0x8	Configuration IN07	rw	USINT	10	0 = 0x00 10 = 0x0A 11 = 0x0B	off Input IN07 binary plus switched binary plus switched with diagnosis
	0x9	Configuration IN08	rw	USINT	10	0 = 0x00 10 = 0x0A 11 = 0x0B	off Input IN08 binary plus switched binary plus switched with diagnosis
	0xA	Configuration IN09	rw	USINT	10	0 = 0x00 10 = 0x0A 11 = 0x0B	off Input IN09 binary plus switched binary plus switched with diagnosis
	0xB	Configuration IN10	rw	USINT	10	0 = 0x00 10 = 0x0A 11 = 0x0B	off Input IN10 binary plus switched binary plus switched with diagnosis

© ifm electronic gmbh

Index	S-Idx	Parameter name	Data type		Default	Details	
	0xC	Configuration IN11	rw	USINT	10	0 = 0x00 10 = 0x0A 11 = 0x0B	off Input IN11 binary plus switched binary plus switched with diagnosis
0x2000	0xD	Configuration IN12	rw	USINT	1	0 = 0x00 1 = 0x01 14 = 0x0E 19 = 0x13 20 = 0x14 21 = 0x15 22 = 0x16 23 = 0x17	off Input IN12 binary plus switched, digitally monitored frequency 0…30 000 Hz period duration period duration as ratio 0…1 000 ‰ counting up counting down incremental encoder
	0xE	Configuration IN13	rw	USINT	1	0 = 0x00 1 = 0x01 14 = 0x0E 19 = 0x13 20 = 0x14 21 = 0x15 22 = 0x16 23 = 0x17	off Input IN13 binary plus switched, digitally monitored frequency 0…30 000 Hz period duration period duration as ratio 0…1 000 ‰ counting up counting down incremental encoder
	0xF	Configuration IN14	rw	USINT	1	0 = 0x00 1 = 0x01 14 = 0x0E 19 = 0x13 20 = 0x14 21 = 0x15 22 = 0x16 23 = 0x17	off Input IN14 binary plus switched, digitally monitored frequency 0…30 000 Hz period duration period duration as ratio 0…1 000 ‰ counting up counting down incremental encoder
	0x10	Configuration IN15	rw	USINT	1	0 = 0x00 1 = 0x01 14 = 0x0E 19 = 0x13 20 = 0x14 21 = 0x15 22 = 0x16 23 = 0x17	off Input IN15 binary plus switched, digitally monitored frequency 0…30 000 Hz period duration period duration as ratio 0…1 000 ‰ counting up counting down incremental encoder
0x2000	0x11	Configuration OUT00	rw	USINT	2	0 = 0x00 2 = 0x02 4 = 0x04 5 = 0x05 15 = 0x0F 16 = 0x10	off Input OUT00 binary plus switched PWM output current control binary plus switched with diagnosis binary plus switched with diagnosis + protection
	0x12	Configuration OUT01	rw	USINT	2	0 = 0x00 2 = 0x02 4 = 0x04 5 = 0x05 15 = 0x0F 16 = 0x10	off Input OUT01 binary plus switched PWM output current control binary plus switched with diagnosis binary plus switched with diagnosis + protection
0x2000	0x13	Configuration OUT02	rw	USINT	2	0 = 0x00 2 = 0x02 4 = 0x04 15 = 0x0F 16 = 0x10	off Input OUT02 binary plus switched PWM output binary plus switched with diagnosis binary plus switched with diagnosis + protection
	0x14	Configuration OUT03	rw	USINT	2	0 = 0x00 2 = 0x02 4 = 0x04 15 = 0x0F 16 = 0x10	off Input OUT03 binary plus switched PWM output binary plus switched with diagnosis binary plus switched with diagnosis + protection
	0x15	Configuration OUT04	rw	USINT	2	0 = 0x00 2 = 0x02 4 = 0x04 15 = 0x0F 16 = 0x10	off Input OUT04 binary plus switched PWM output binary plus switched with diagnosis binary plus switched with diagnosis + protection

© ifm electronic gmbh

Index	S-Idx	Parameter name	Data type		Default	Details	
	0x16	Configuration OUT05	rw	USINT	2	0 = 0x00 2 = 0x02 4 = 0x04 15 = 0x0F 16 = 0x10	off Input OUT05 binary plus switched PWM output binary plus switched with diagnosis binary plus switched with diagnosis + protection
	0x17	Configuration OUT06	rw	USINT	2	0 = 0x00 2 = 0x02 4 = 0x04 15 = 0x0F 16 = 0x10	off Input OUT06 binary plus switched PWM output binary plus switched with diagnosis binary plus switched with diagnosis + protection
	0x18	Configuration OUT07	rw	USINT	2	0 = 0x00 2 = 0x02 4 = 0x04 15 = 0x0F 16 = 0x10	off Input OUT07 binary plus switched PWM output binary plus switched with diagnosis binary plus switched with diagnosis + protection
0x2000	0x19	Configuration OUT08	rw	USINT	2	0 = 0x00 2 = 0x02 4 = 0x04	off Input OUT08 binary plus switched PWM output + PWM output, voltage controlled
	0x1A	Configuration OUT09	rw	USINT	2	0 = 0x00 2 = 0x02 4 = 0x04	off Input OUT09 binary plus switched PWM output + PWM output, voltage controlled
0x2000	0x1B	Configuration OUT10	rw	USINT	2	0 = 0x00 2 = 0x02 4 = 0x04	off Input OUT10 binary plus switched PWM output
	0x1C	Configuration OUT11	rw	USINT	2	0 = 0x00 2 = 0x02 4 = 0x04	off Input OUT11 binary plus switched PWM output
0x2000	0x1D	Configuration OUT12	rw	USINT	2	0 = 0x00 2 = 0x02	off Input OUT12 binary plus switched
	0x1E	Configuration OUT13	rw	USINT	2	0 = 0x00 2 = 0x02	off Input OUT13 binary plus switched
	0x1F	Configuration OUT14	rw	USINT	2	0 = 0x00 2 = 0x02	off Input OUT14 binary plus switched
	0x20	Configuration OUT15	rw	USINT	2	0 = 0x00 2 = 0x02	off Input OUT15 binary plus switched
0x2001	0x0	PWM frequency	ro	USINT	12	Largest sub-index supported	
	0x1	PWM frequency OUT00	rw	UINT	100	20…250	OUT00 PWM frequency [Hz]
	0x2	PWM frequency OUT01	rw	UINT	100	20…250	OUT01 PWM frequency [Hz]
	0x3	PWM frequency OUT02	rw	UINT	100	20…250	OUT02 PWM frequency [Hz]
	0x4	PWM frequency OUT03	rw	UINT	100	20…250	OUT03 PWM frequency [Hz]
	0x5	PWM frequency OUT04	rw	UINT	100	20…250	OUT04 PWM frequency [Hz]
	0x6	PWM frequency OUT05	rw	UINT	100	20…250	OUT05 PWM frequency [Hz]
	0x7	PWM frequency OUT06	rw	UINT	100	20…250	OUT06 PWM frequency [Hz]
	0x8	PWM frequency OUT07	rw	UINT	100	20…250	OUT07 PWM frequency [Hz]
	0x9	PWM frequency OUT08	rw	UINT	100	20…250	OUT08 PWM frequency [Hz]
	0xA	PWM frequency OUT09	rw	UINT	100	20…250	OUT09 PWM frequency [Hz]
	0xB	PWM frequency OUT10	rw	UINT	100	20…250	OUT10 PWM frequency [Hz]
	0xC	PWM frequency OUT11	rw	UINT	100	20…250	OUT11 PWM frequency [Hz]
0x2002	0x0	Current value	ro	USINT	2	Largest sub-index supported	
	0x1	Current value OUT00	ro	UINT	0	0…2 000	OUT00 output current [mA]
	0x2	Current value OUT01	ro	UINT	0	0…2 000	OUT01 output current [mA]
0x2004	0x0	P-value	ro	USINT	2	Largest sub-index supported	

Index	S-Idx	Parameter name	Data type		Default	Details	
	0x1	P-value OUT00	rw	USINT	30	0…255	OUT00 P-value for current control
	0x2	P-value OUT01	rw	USINT	30	0…255	OUT01 P-value for current control
0x2005	0x0	I-value	ro	USINT	2	Largest sub-index supported	
	0x1	I-value OUT00	rw	USINT	20	0…255	OUT00 I-value for current control
	0x2	I-value OUT01	rw	USINT	20	0…255	OUT01 I-value for current control
0x2006	0x0	PWM dither frequency	ro	USINT	12	Largest sub-index supported	
	0x1	PWM dither frequency OUT00	rw	UINT	0	0…PWMfreq / 2	OUT00 PWM dither frequency [Hz]
	0x2	PWM dither frequency OUT01	rw	UINT	0	0…PWMfreq / 2	OUT01 PWM dither frequency [Hz]
	0x3	PWM dither frequency OUT02	rw	UINT	0	0…PWMfreq / 2	OUT02 PWM dither frequency [Hz]
	0x4	PWM dither frequency OUT03	rw	UINT	0	0…PWMfreq / 2	OUT03 PWM dither frequency [Hz]
	0x5	PWM dither frequency OUT04	rw	UINT	0	0…PWMfreq / 2	OUT04 PWM dither frequency [Hz]
	0x6	PWM dither frequency OUT05	rw	UINT	0	0…PWMfreq / 2	OUT05 PWM dither frequency [Hz]
	0x7	PWM dither frequency OUT06	rw	UINT	0	0…PWMfreq / 2	OUT06 PWM dither frequency [Hz]
	0x8	PWM dither frequency OUT07	rw	UINT	0	0…PWMfreq / 2	OUT07 PWM dither frequency [Hz]
	0x9	PWM dither frequency OUT08	rw	UINT	0	0…PWMfreq / 2	OUT08 PWM dither frequency [Hz]
	0xA	PWM dither frequency OUT09	rw	UINT	0	0…PWMfreq / 2	OUT09 PWM dither frequency [Hz]
	0xB	PWM dither frequency OUT10	rw	UINT	0	0…PWMfreq / 2	OUT10 PWM dither frequency [Hz]
	0xC	PWM dither frequency OUT11	rw	UINT	0	0…PWMfreq / 2	OUT11 PWM dither frequency [Hz]
0x2007	0x0	PWM dither value	ro	USINT	12	Largest sub-index supported	
	0x1	PWM dither value OUT00	rw	UINT	0	0…1 000	OUT00 PWM dither value [‰]
	0x2	PWM dither value OUT01	rw	UINT	0	0…1 000	OUT01 PWM dither value [‰]
	0x3	PWM dither value OUT02	rw	UINT	0	0…1 000	OUT02 PWM dither value [‰]
	0x4	PWM dither value OUT03	rw	UINT	0	0…1 000	OUT03 PWM dither value [‰]
	0x5	PWM dither value OUT04	rw	UINT	0	0…1 000	OUT04 PWM dither value [‰]
	0x6	PWM dither value OUT05	rw	UINT	0	0…1 000	OUT05 PWM dither value [‰]
	0x7	PWM dither value OUT06	rw	UINT	0	0…1 000	OUT06 PWM dither value [‰]
	0x8	PWM dither value OUT07	rw	UINT	0	0…1 000	OUT07 PWM dither value [‰]
	0x9	PWM dither value OUT08	rw	UINT	0	0…1 000	OUT08 PWM dither value [‰]
	0xA	PWM dither value OUT09	rw	UINT	0	0…1 000	OUT09 PWM dither value [‰]
	0xB	PWM dither value OUT10	rw	UINT	0	0…1 000	OUT10 PWM dither value [‰]
	0xC	PWM dither value OUT11	rw	UINT	0	0…1 000	OUT11 PWM dither value [‰]
0x2012	0x0	Period input	ro	USINT	4	Largest sub-index supported	
	0x1	Period duration IN12	ro	UDINT	0	IN12 period duration [µs]	
	0x2	Period duration IN13	ro	UDINT	0	IN13 period duration [µs]	
	0x3	Period duration IN14	ro	UDINT	0	IN14 period duration [µs]	
	0x4	Period duration IN15	ro	UDINT	0	IN15 period duration [µs]	
0x2013	0x0	Period input number of periods for average	ro	USINT	4	Largest sub-index supported	
	0x1	Number of periods IN12	rw	USINT	4	1…255	IN12 number of periods
	0x2	Number of periods IN13	rw	USINT	4	1…255	IN13 number of periods
	0x3	Number of periods IN14	rw	USINT	4	1…255	IN14 number of periods
	0x4	Number of periods IN15	rw	USINT	4	1…255	IN15 number of periods

© ifm electronic GmbH www.ifm.com

Index	S-Idx	Parameter name	Data type		Default	Details	
0x2014	0x0	Period input – ratio value	ro	USINT	4	Largest sub-index supported	
	0x1	Period ratio value IN12	ro	UINT	0	0…1 000	IN12 mark-to-space ratio [‰]
	0x2	Period ratio value IN13	ro	UINT	0	0…1 000	IN13 mark-to-space ratio [‰]
	0x3	Period ratio value IN14	ro	UINT	0	0…1 000	IN14 mark-to-space ratio [‰]
	0x4	Period ratio value IN15	ro	UINT	0	0…1 000	IN15 mark-to-space ratio [‰]
0x2015	0x0	Frequency input	ro	USINT	4	Largest sub-index supported	
	0x1	Frequency IN12	ro	REAL	1	0…30 000	IN12 frequency [Hz]
	0x2	Frequency IN13	ro	REAL	1	0…30 000	IN13 frequency [Hz]
	0x3	Frequency IN14	ro	REAL	1	0…30 000	IN14 frequency [Hz]
	0x4	Frequency IN15	ro	REAL	1	0…30 000	IN15 frequency [Hz]
0x2016	0x0	Timebase	ro	USINT	4	Largest sub-index supported	
	0x1	Timebase IN12	rw	UINT	50	0…2 000	IN12 timebase [ms]
	0x2	Timebase IN12	rw	UINT	50	0…2 000	IN13 timebase [ms]
	0x3	Timebase IN12	rw	UINT	50	0…2 000	IN14 timebase [ms]
	0x4	Timebase IN12	rw	UINT	50	0…2 000	IN15 timebase [ms]
0x2020	0x0	Input – short to supply voltage	ro	USINT	2	Largest sub-index supported	
	0x1	Short to supply voltage IN00...IN07	ro	USINT	0	0 = normal 1 = short circuit	channels (bit coded) 0b---- ---X = IN00 0b---- --X- = IN01 0b---- -X-- = IN02 0b---- X--- = IN03 0b---X ---- = IN04 0b--X- ---- = IN05 0b-X-- ---- = IN06 0bX--- ---- = IN07
	0x2	Short to supply voltage IN08…IN11	ro	USINT	0	0 = normal 1 = short circuit	channels (bit coded) 0b---- ---X = IN08 0b---- --X- = IN09 0b---- -X-- = IN10 0b---- X--- = IN11 0b---X ---- = IN12 0b--X- ---- = IN13 0b-X-- ---- = IN14 0bX--- ---- = IN15
0x2021	0x0	Input – wire break	ro	USINT	2	Largest sub-index supported	
	0x1	Wire break IN00...IN07	ro	USINT	0	0 = normal 1 = wire break	channels (bit coded) 0b---- ---X = IN00 0b---- --X- = IN01 0b---- -X-- = IN02 0b---- X--- = IN03 0b---X ---- = IN04 0b--X- ---- = IN05 0b-X-- ---- = IN06 0bX--- ---- = IN07
	0x2	Wire break IN08…IN11	ro	USINT	0	0 = normal 1 = wire break	channels (bit coded) 0b---- ---X = IN08 0b---- --X- = IN09 0b---- -X-- = IN10 0b---- X--- = IN11 0b---X ---- = IN12 0b--X- ---- = IN13 0b-X-- ---- = IN14 0bX--- ---- = IN15
0x2022	0x0	Output – short circuit	ro	USINT	1	Largest sub-index supported	

© ifm electronic gmbh

Index	S-Idx	Parameter name	Data type		Default	Details	
	0x1	Short circuit OUT00…OUT07	ro	USINT	0	0 = normal 1 = short circuit	channels (bit coded) 0b---- ---X = OUT00 0b---- --X- = OUT01 0b---- -X-- = OUT02 0b---- X--- = OUT03 0b---X ---- = OUT04 0b--X- ---- = OUT05 0b-X-- ---- = OUT06 0bX--- ---- = OUT07
0x2023	0x0	Output – open circuit	ro	USINT	1	Largest sub-index supported	
	0x1	Open circuit OUT00…OUT07	ro	USINT	0	0 = normal 1 = open circuit	channels (bit coded) 0b---- ---X = OUT00 0b---- --X- = OUT01 0b---- -X-- = OUT02 0b---- X--- = OUT03 0b---X ---- = OUT04 0b--X- ---- = OUT05 0b-X-- ---- = OUT06 0bX--- ---- = OUT07
0x2024	0x0	Output – overload	ro	USINT	1	Largest sub-index supported	
	0x1	Overload OUT00…OUT01	ro	USINT	0	0 = normal 1 = overload	channels (bit coded) 0b---- ---X = OUT00 0b---- --X- = OUT01
0x2025	0x0	Input analog – overcurrent	ro	USINT	1	Largest sub-index supported	
	0x1	Overcurrent IN00…IN03	ro	USINT	0	0 = normal 1 = overcurrent	channels (bit coded) 0b---- ---X = IN00 0b---- --X- = IN01 0b---- -X-- = IN02 0b---- X--- = IN03
0x2030	0x0	Input resistor	ro	USINT	2	Largest sub-index supported	
	0x1	Resistance IN04	ro	UINT	0	0…30 000	IN04 resistance [Ohms]
	0x2	Resistance IN05	ro	UINT	0	0…30 000	IN05 resistance [Ohms]
0x2040	0x0	System supply voltage VBBS	ro	USINT	1	Largest sub-index supported	
	0x1	VBBS	ro	USINT	0	VBBS voltage [mV]	
0x2041	0x0	Output supply voltage	ro	USINT	2	Largest sub-index supported	
	0x1	VBB1	ro	UINT	0	VBB1 voltage [mV]	
	0x2	VBB2	ro	UINT	0	VBB2 voltage [mV]	
0x2050		Device temperature	ro	INT	0	temperature [°C]	
0x20F0		Node ID	rw	USINT	124	1…125	node ID [!] value(0x20F0) != value(20F1)
0x20F1		Node ID	rw	USINT	124	1…125	node ID [!] value(0x20F0) != value(20F1)
0x20F2		Baud rate	rw	USINT	–	baud rate [!] value(0x20F2) != value(20F3)	
0x20F3		Baud rate	rw	USINT	–	baud rate [!] value(0x20F2) != value(20F3)	
0x20F4		Autostart	rw	UINT	0	not used	
0x6000	0x0	Binary input Largest sub-index supported	ro	USINT	0x02	binary inputs 00...07 largest supported sub-index = 2	

© ifm electronic gmbh

Index	S-Idx	Parameter name	Data type		Default	Details
	0x1	Binary inputs IN00 - IN07	ro	USINT	0	Binary inputs IN00…IN07 0b---- ---X = IN00 0b---- --X- = IN01 0b---- -X-- = IN02 0b---- X--- = IN03 0b---X ---- = IN04 0b--X- ---- = IN05 0b-X-- ---- = IN06 0bX--- ---- = IN07
	0x2	Binary inputs IN08 - IN15	ro	USINT	0	Binary inputs IN08…IN15 0b---- ---X = IN08 0b---- --X- = IN09 0b---- -X-- = IN10 0b---- X--- = IN11 0b---X ---- = IN12 0b--X- ---- = IN13 0b-X-- ---- = IN14 0bX--- ---- = IN15
0x6200	0x0	Binary output Largest sub-index supported	ro	USINT	0x02	binary outputs largest supported sub-index = 2
	0x1	Binary outputs OUT00 - OUT07	wo	USINT	0	binary outputs OUT00…OUT07 0b---- ---X = OUT00 0b---- --X- = OUT01 0b---- -X-- = OUT02 0b---- X--- = OUT03 0b---X ---- = OUT04 0b--X- ---- = OUT05 0b-X-- ---- = OUT06 0bX--- ---- = OUT07
	0x2	Binary outputs OUT08 - OUT15	wo	USINT	0	binary outputs OUT08…OUT15 0b---- ---X = OUT08 0b---- --X- = OUT09 0b---- -X-- = OUT10 0b---- X--- = OUT11 0b---X ---- = OUT12 0b--X- ---- = OUT13 0b-X-- ---- = OUT14 0bX--- ---- = OUT15
0x6404	0x0	Analogue input Largest sub-index supported	ro	USINT	0x04	analogue inputs largest supported sub-index = 4
	0x1	Analogue input IN00	ro	UINT	--	analogue value of input IN00
	0x2	Analogue input IN01	ro	UINT	--	analogue value of input IN01
	0x3	Analogue input IN02	ro	UINT	--	analogue value of input IN02
	0x4	Analogue input IN03	ro	UINT	--	analogue value of input IN03
0x6414	0x0	PWM output Largest sub-index supported	ro	USINT	0x12	PWM outputs largest supported sub-index = 12
	0x1	PWM output OUT00	wo	UINT	--	value for PWM output OUT00
	0x2	PWM output OUT01	wo	UINT	--	value for PWM output OUT01
	0x3	PWM output OUT02	wo	UINT	--	value for PWM output OUT02
	0x4	PWM output OUT03	wo	UINT	--	value for PWM output OUT03
	0x5	PWM output OUT04	wo	UINT	--	value for PWM output OUT04
	0x6	PWM output OUT05	wo	UINT	--	value for PWM output OUT05
	0x7	PWM output OUT06	wo	UINT	--	value for PWM output OUT06
	0x8	PWM output OUT07	wo	UINT	--	value for PWM output OUT07
	0x9	PWM output OUT08	wo	UINT	--	value for PWM output OUT08

© ifm electronic gmbh www.ifm.com

Index	S-Idx	Parameter name	Data type		Default	Details
	0xA	PWM output OUT09	wo	UINT	--	value for PWM output OUT09
	0xB	PWM output OUT10	wo	UINT	--	value for PWM output OUT10
	0xC	PWM output OUT11	wo	UINT	--	value for PWM output OUT11

Legend:
Data type: ro = read only / rw = read and write / wo = write only

© ifm electronic gmbh www.ifm.com

7.3.4 Operation of the I/O module

Contents

16433

Inputs: PDO mapping (I/O module)

15968

The following table contains the following entries from the control configuration:
- CAN input
- transmit PDO mapping

Bit coding:

```
0b---- ---X = IN00 (IN08)
...
0bX--- ---- = IN07 (IN15)
```

TX-PDO	Variable type	COB ID = node ID + ...	Comment
1	USINT	0x180	Input byte 0 (IN00…IN07)
1	USINT	0x180	Input byte 1 (IN08…IN15)
1	USINT	0x180	Short circuit between VBBS and the input (IN00...IN07)
1	USINT	0x180	Short circuit between VBBS and the input (IN08…IN15)
1	USINT	0x180	Wire break on input (IN00…IN07)
1	USINT	0x180	Wire break on input (IN08…IN15)
1	USINT	0x180	Overcurrent on input (IN00...IN03)
2	UINT	0x280	Analogue input IN00
2	UINT	0x280	Analogue input IN01
2	UINT	0x280	Analogue input IN02
2	UINT	0x280	Analogue input IN03
3	UINT	0x380	Resistance measurement input IN04
3	UINT	0x380	Resistance measurement input IN05
3	UINT	0x380	Output current on OUT00
3	UINT	0x380	Output current on OUT01
4	UDINT	0x480	Period in [µs] on IN12
4	UDINT	0x480	Period in [µs] on IN13
5	UDINT	0x181	Period in [µs] on IN14
5	UDINT	0x181	Period in [µs] on IN15
6	UINT	0x281	Mark-to-space ratio in [‰] on N12
6	UINT	0x281	Mark-to-space ratio in [‰] on N13
6	UINT	0x281	Mark-to-space ratio in [‰] on N14
6	UINT	0x281	Mark-to-space ratio in [‰] on N15
7	USINT	0x381	Frequency in [Hz] on IN12
7	REAL	0x381	Frequency in [Hz] on IN13

© ifm electronic gmbh

TX-PDO	Variable type	COB ID = node ID + ...	Comment
8	REAL	0x481	Frequency in [Hz] on IN14
8	REAL	0x481	Frequency in [Hz] on IN15
9	USINT	0x182	Short circuit on output (OUT00…OUT07)
9	USINT	0x182	Wire break on output (OUT00…OUT07)
9	USINT	0x182	Overcurrent on output (OUT00…OUT01)
10	UINT	0x282	Supply voltage on VBBS in [mV]
10	UINT	0x282	Supply voltage on VBB1 in [mV]
10	UINT	0x282	Supply voltage on VBB2 in [mV]
10	UINT	0x282	Temperature in the device

© ifm electronic gmbh
www.ifm.com

Outputs: PDO mapping (I/O module)

15969

The following table contains the following entries from the control configuration:
- CAN output
- Receive PDO mapping

Bit coding:

```
0b---- ---X = OUT00 (OUT08)
...
0bX--- ---- = OUT07 (OUT15)
```

RX-PDO	Variable type	COB ID = node ID + ...	Comment
1	USINT	0x200	Output byte 0 (OUT00…OUT07)
1	USINT	0x200	Output byte 1 (OUT08…OUT15)
2	UINT	0x300	PWM output OUT00
2	UINT	0x300	PWM output OUT01
2	UINT	0x300	PWM output OUT02
2	UINT	0x300	PWM output OUT03
3	UINT	0x400	PWM output OUT04
3	UINT	0x400	PWM output OUT05
3	UINT	0x400	PWM output OUT06
3	UINT	0x400	PWM output OUT07
4	UINT	0x500	PWM output OUT08
4	UINT	0x500	PWM output OUT09
4	UINT	0x500	PWM output OUT10
4	UINT	0x500	PWM output OUT11

© ifm electronic gmbh www.ifm.com

7.3.5 System flag for the integrated ExB01 I/O module

Contents

16270

System flag (ExB01 I/O module)

15957

There are no system flags for the integrated I/O module of the device.
Feedback is given for Process Data Objects (PDOs) via the EDS file.
→ Chapter **Inputs: PDO mapping (I/O module)** (→ p. 308)

7.3.6 Error messages for the I/O module

Contents

15891

EMCY objects

15981

The following error codes according to DSP-401 or DSP-301 are supported:

EMCY code	Error reg	Additional code	Description
0x6100	0x11	0x00	Internal software Overflow of an Rx queue e.g. frequency of the Rx PDOs is too high Reset only externally via entry in the index 0x1003 SubIdx 00
0x6101	0x11	0x00	Internal software Overflow of a Tx queue e.g. device does not communicate with the bus Reset only externally via entry in the index 0x1003 SubIdx 00
0x8100	0x11	0x00	Monitoring (Guarding Error) No guard object is received for "guard time" x " life time factor" Reset with the next communication.
0x8200	0x11	0x00	Monitoring (Synch Error) For "communication cycle" no sync object is received Only in OPERATIONAL Reset with the next sync OBJ or PREOP

CANopen does not provide for two identical EMCY objects to be sent consecutively.

© ifm electronic gmbh www.ifm.com

SDOs error messages

15951

The following messages are created in case of an error:

Index	SubIdx	Parameter name	Data type		Default	Details	
0x1001		Error register	ro	USINT	0	Error register bitcoded to profile 301 permissible values = `0b0000 0000` = no error `0b0000 0001` = generic error `0b0001 0000` = communication error `0b1000 0000` = manufacturer specific	
0x1003	0x0	Predefined error field Number of entries	rw	UDINT	0	An error list with 4 entries is supported	
	0x1	Error history	ro	UDINT	0	Error occurred; coded according to EMCY list The last error is indicated in the sub-index 1	
	0x2	Error history	ro	UDINT	0	Error occurred; coded according to EMCY list	
	0x3	Error history	ro	UDINT	0	Error occurred, coded according to EMCY list	
	0x4	Error history	ro	UDINT	0	Error occurred, coded according to EMCY list	
	0x5	Error history	ro	UDINT	0	Error occurred, coded according to EMCY list	
0x2020	0x0	Input – short to supply voltage	ro	USINT	2	Largest sub-index supported	

	0x1	Short to supply voltage
		IN00...IN07
			ro	USINT	0	0 = normal
						1 = short circuit
							channels (bit coded)
							`0b---- ---X = IN00`
							`0b---- --X- = IN01`
							`0b---- -X-- = IN02`
							`0b---- X--- = IN03`
							`0b---X ---- = IN04`
							`0b--X- ---- = IN05`
							`0b-X-- ---- = IN06`
							`0bX--- ---- = IN07`

	0x2	Short to supply voltage
		IN08…IN11
			ro	USINT	0	0 = normal
						1 = short circuit
							channels (bit coded)
							`0b---- ---X = IN08`
							`0b---- --X- = IN09`
							`0b---- -X-- = IN10`
							`0b---- X--- = IN11`
							`0b---X ---- = IN12`
							`0b--X- ---- = IN13`
							`0b-X-- ---- = IN14`
							`0bX--- ---- = IN15`

0x2021	0x0	Input – wire break	ro	USINT	2	Largest sub-index supported	

	0x1	Wire break IN00...IN07	ro	USINT	0	0 = normal
						1 = wire break
							channels (bit coded)
							`0b---- ---X = IN00`
							`0b---- --X- = IN01`
							`0b---- -X-- = IN02`
							`0b---- X--- = IN03`
							`0b---X ---- = IN04`
							`0b--X- ---- = IN05`
							`0b-X-- ---- = IN06`
							`0bX--- ---- = IN07`

	0x2	Wire break IN08…IN11	ro	USINT	0	0 = normal
						1 = wire break
							channels (bit coded)
							`0b---- ---X = IN08`
							`0b---- --X- = IN09`
							`0b---- -X-- = IN10`
							`0b---- X--- = IN11`
							`0b---X ---- = IN12`
							`0b--X- ---- = IN13`
							`0b-X-- ---- = IN14`
							`0bX--- ---- = IN15`

0x2022	0x0	Output – short circuit	ro	USINT	1	Largest sub-index supported	

© ifm electronic gmbh

Index	SubIdx	Parameter name	Data type		Default	Details	
	0x1	Short circuit OUT00…OUT07	ro	USINT	0	0 = normal 1 = short circuit	channels (bit coded) 0b---- ---X = OUT00 0b---- --X- = OUT01 0b---- -X-- = OUT02 0b---- X--- = OUT03 0b---X ---- = OUT04 0b--X- ---- = OUT05 0b-X-- ---- = OUT06 0bX--- ---- = OUT07
0x2023	0x0	Output – open circuit	ro	USINT	1	Largest sub-index supported	
	0x1	Open circuit OUT00…OUT07	ro	USINT	0	0 = normal 1 = open circuit	channels (bit coded) 0b---- ---X = OUT00 0b---- --X- = OUT01 0b---- -X-- = OUT02 0b---- X--- = OUT03 0b---X ---- = OUT04 0b--X- ---- = OUT05 0b-X-- ---- = OUT06 0bX--- ---- = OUT07
0x2024	0x0	Output – overload	ro	USINT	1	Largest sub-index supported	
	0x1	Overload OUT00…OUT01	ro	USINT	0	0 = normal 1 = overload	channels (bit coded) 0b---- ---X = OUT00 0b---- --X- = OUT01
0x2025	0x0	Input analog – overcurrent	ro	USINT	1	Largest sub-index supported	
	0x1	Overcurrent IN00…IN03	ro	USINT	0	0 = normal 1 = overcurrent	channels (bit coded) 0b---- ---X = IN00 0b---- --X- = IN01 0b---- -X-- = IN02 0b---- X--- = IN03

Legend
Data type: ro = read only / rw = read and write / wo = write only

© ifm electronic gmbh

7.4 Error tables

Contents

19606

7.4.1 Error codes

Contents

12334

Overview of the error codes output by some of the function blocks.

The 32-bit error code consists of four 8-bit values (DWORD).

4th byte	3rd byte	2nd byte	1st byte
Error class	Application-specific error code	Error source	Error cause

© ifm electronic gmbh

Error cause (1st byte)

19273

Value dec	hex	Description
0	00	No error cause or: application-specific error
1	01	Break
2	02	Short
4	04	Overload
5	05	Undervoltage
6	06	Overvoltage
7	07	Current control
24	18	Temperature
26	20	Memory test
27	21	Memory test
51	33	Integer overflow or: division by zero
56	38	FPU underflow
57	39	FPU overflow
58	3A	FPU division by zero
59	3B	FPU unspecific error
128	80	CRC
129	81	Corrupt data
130	82	Memory protection
131	83	No data
144	90	Watchdog
145	91	Trap
147	93	Assertion failed
194	C2	CAN bus-off
224	E0	Board link warning (ExtendedController)
225	E1	Board link error (ExtendedController)
240	F0	Serial number
241	F1	Runtime system expired
248	F8	Wrong parameter

© ifm electronic gmbh

Error source (2nd byte)

18660

Value dec	Value hex	Description
0	00	No error source or: application-specific error
1	01	CPU
2	02	Peripheral processor
3	03	Coprocessor
8	08	Floating point unit
16...31	10...1F	Input 0...15 (standard side)
32...63	20...3F	Input 0...31 (extended side)
64...79	40...4F	Output 0...15 (standard side)
80..111	50...6F	Output 0...31 (extended side)
128...131	80...83	CAN 1...4
144	90	Relay voltage VBBo (standard side)
145	91	Relay voltage VBBr (standard side)
146	92	VBBo (standard side)
147	93	VBBr (standard side)
148	94	VBBs (standard side)
149	95	clamp 15
150	96	Relay voltage VBB1 (extended side)
151	97	Relay voltage VBB2 (extended side)
152	98	Relay voltage VBB3 (extended side)
153	99	Relay voltage VBB4 (extended side)
154	9A	VBBrel (extended side)
155	9B	VBB1 (extended side)
156	9C	VBB2 (extended side)
157	9D	VBB3 (extended side)
158	9E	VBB4 (extended side)
160	A0	Analogue multiplexer
161	A1	Analogue reference
176	B0	Internal flash
177	B1	External flash
178	B2	Internal RAM
179	B3	External RAM
192	C0	Code Startupper
193	C1	Code Bootloader
194	C2	Code runtime system
196	C4	Boot project
197	C5	Code application program

© ifm electronic gmbh www.ifm.com

Value dec	Value hex	Description
198	C6	Scratchpad RAM
224	E0	System data
225	E1	System settings
226	E2	System information
227	E3	Calibration data
228	E4	FRAM / MRAM (user's area)

© ifm electronic gmbh
www.ifm.com

Application-specific error code (3rd byte)

12338

In case of an application-specific error, it is requested that:
ERRORCODE byte 1 = error cause = 0x00
ERRORCODE byte 2 = error source = 0x00

► Signal application-specific errors to the controller using **ERROR_REPORT** (→ p. 222).

Value dec | hex		Description
0	00	No application-specific error
> 0	> 00	Application-specific error

Error class (4th byte)

19271

This information only applies of input TEST = FALSE.

Value dec | hex		Description
0	00	No error
1	01	General error, not relevant to safety > Error flag, error code > Error reset possible
3	03	ERROR STOP > Error code > Application program aborted > Safe state ► Power-off-on reset is required
4	04	Fatal error > Error code > Application program aborted > Safe state > Device restarts again (soft-reset)

14025

Fatal errors and ERROR STOP are only visible in the application if the TEST input was already active when the error occurred.

If TEST input = FALSE:
- an ERROR STOP causes the controller to STOP
- an fatal error causes the controller to reboot

© ifm electronic gmbh www.ifm.com

Error codes: Examples

19274

Byte 2	▼ Byte 1 ▼		
Error source [hex]	Error cause [hex]	Description	Function block
10...1F	01	Wire break Ixx	INPUT_ANALOG
20...3F (Ex)	01	Wire break Ixx_E	INPUT_ANALOG_E
40...4F	01	Wire break Qxx	
50...6F (Ex)	01	Wire break Qxx_E	
10...1F	02	Short circuit Ixx	INPUT_ANALOG
20...3F (Ex)	02	Short circuit Ixx_E	INPUT_ANALOG_E
40...4F	02	Short circuit Qxx	
50...6F (Ex)	02	Short circuit Qxx_E	
10...1F	04	Overcurrent Ixx	INPUT_ANALOG
20...3F (Ex)	04	Overcurrent Ixx_E	INPUT_ANALOG_E
40...4F	04	Overload Qxx	
50...6F (Ex)	04	Overload Qxx_E	
90	05	Undervoltage relay voltage VBBs	
91	05	Undervoltage relay voltage VBBr	
92	05	Undervoltage VBBo	
93	05	Undervoltage VBBr	
94	05	Undervoltage VBBs	
95	05	Undervoltage clamp 15	
96...99 (Ex)	05	Undervoltage relay voltage VBBx	
9A (Ex)	05	Undervoltage VBBrel	
9B...9E (Ex)	05	Undervoltage VBBx	
90	06	Overvoltage relay voltage VBBs	
91	06	Overvoltage relay voltage VBBr	
92	06	Overvoltage VBBo	
93	06	Overvoltage VBBr	
94	06	Overvoltage VBBs > 32 V	
94	06	Overvoltage VBBs > 34 V	
95	06	Overvoltage clamp 15	
96...99 (Ex)	06	Overvoltage relay voltage VBBx	
9A (Ex)	06	Overvoltage VBBrel	
9B...9E (Ex)	06	Overvoltage VBBx	
40...4F	07	Current control Qxx	
50...6F (Ex)	07	Current control Qxx_E	
10...1F	08 (safe)	Safety diagnostics on the current input	
10...1F	09 (safe)	Safety diagnostics on the voltage input	
40...4F	0A (safe)	Safety diagnostics on the activated output (stuck at 1, cross fault)	
10...17	0B (safe)	Safety diagnostics on the SafetySwitch	SAFETY_SWITCH
A0	0C (safe)	Analogue values monitoring/multiplexer	SAFETY_SWITCH
40...4F	0D (safe)	Safety diagnostics on the deactivated output (stuck at 1)	

Byte 2	▼ Byte 1 ▼		
Error source [hex]	Error cause [hex]	Description	Function block
00	18	Temperature error	
90...91	19 (safe)	Contact fault relays VBBo / VBBr	
B3	20	RAM memory test failed	
E4	20	FRAM/MRAM memory test failed	
B3	21 (safe)	RAM address test failed	
01	30 (safe)	Incorrect / missing interrupt	
01	31 (safe)	Error time base CPU	
08	39	Floating-Point Overflow	
08	3A	Floating-point division by 0	
08	3B	Unspecified floating-point error	
C2	80	Checksum error in the runtime system code (ifm code)	
C3 (safe)	80	Checksum error in the PCP data RAM	
C4	80	Checksum error in the boot project	
C5	80	Checksum error in the application program code	
C6	80	Checksum error in the SP-RAM	
C7 (safe)	80	Checksum error in the PCP code RAM	
E0	80	Checksum error in the system data	
E1	80	Checksum error in the system variables	
E2	80	Checksum error in the system parameters	
E3	80	Checksum error in the calibration data	
B3	81	Corrupt data in the RAM	
E3	83	Corrupt calibration data	
04	92 (safe)	Safety core stopped	
05	92 (safe)	Error safety code	
80...83	C1	CANx warning	
80...83	C2	CANx Busoff	
80...83	C3 (safe)	CANsafety reception error	
80...83	C4 (safe)	CANsafety transmission error	
80, 82	C5 (safe)	CANsafety configuration corrupt	
00 (Ex)	E1	Board link error	
00	F0	Serial number error	
00	F1	Runtime system expired	
00	F8	Parameter error	All FBs with ERROR output
10...1F	F8	Parameter error Ixx	INPUT_ANALOG SET_INPUT_MODE
20...3F (Ex)	F8	Parameter error Ixx_E	INPUT_ANALOG_E SET_INPUT_MODE_E
--	-	Parameter error Qxx	SET_OUTPUT_MODE
---	-	Parameter error Qxx_E	SET_OUTPUT_MODE_E
05 (safe)	ErrorCode	Safety code error	
06 (safe)	TrapID	Safety code trap error	

Legend:
(Ex) = applies to ExtendedController only
(safe) = applies to SafetyController only

The resulting error class (= byte 4) results from the context of the situation and the parameter setting. Here, byte 3 (application-specific error code) is always = 0.

7.4.2 Error flags

19608

→ chapter **System flags** (→ p. 231)

7.4.3 Errors: CAN / CANopen

19610
19604

→ System manual "Know-How ecomat*mobile*"
→ chapter **CAN / CANopen: errors and error handling**

EMCY codes: CANx

13094

ℹ The indications for CANx also apply to each of the CAN interfaces.

EMCY code object 0x1003		Object 0x1001	Manufactor specific information					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Description
00	80	11	--	--	--	--	--	CANx monitoring SYNC error (only slave)
00	81	11	--	--	--	--	--	CANx warning threshold (> 96)
10	81	11	--	--	--	--	--	CANx receive buffer overrun
11	81	11	--	--	--	--	--	CANx transmit buffer overrun
30	81	11	--	--	--	--	--	CANx guard/heartbeat error (only slave)

EMCY codes: I/Os, system (standard side)

2668

The following EMCY messages are sent automatically in the following cases:
- as CANopen master: if **CANx_MASTER_EMCY_HANDLER** (→ p. 86) is called cyclically
- as CANopen slave: if **CANx_SLAVE_EMCY_HANDLER** (→ p. 96) is called cyclically

EMCY code object 0x1003		Object 0x1001	Manufactor specific information					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Description
00	21	03	I07...I00	I15...I08				Inputs interruption
08	21	03	I07...I00	I15...I08				Inputs short circuit
10	21	03	I07...I00	I15...I08				Overcurrent 0…20 mA
00	23	03	Q07…Q00	Q15…Q08				Outputs interruption
08	23	03	Q07…Q00	Q15…Q08				Outputs short circuit
00	31	05						Terminal voltage VBBs
00	33	05						Terminal voltage VBBo
08	33	05						Terminal voltage VBBr
00	42	09						Excess temperature

8 Terms and abbreviations

A

Address

This is the "name" of the bus participant. All participants need a unique address so that the signals can be exchanged without problem.

Application software

Software specific to the application, implemented by the machine manufacturer, generally containing logic sequences, limits and expressions that control the appropriate inputs, outputs, calculations and decisions.

Architecture

Specific configuration of hardware and/or software elements in a system.

B

Baud

Baud, abbrev.: Bd = unit for the data transmission speed. Do not confuse baud with "bits per second" (bps, bits/s). Baud indicates the number of changes of state (steps, cycles) per second over a transmission length. But it is not defined how many bits per step are transmitted. The name baud can be traced back to the French inventor J. M. Baudot whose code was used for telex machines.
1 MBd = 1024 x 1024 Bd = 1 048 576 Bd

Boot loader

On delivery ecomat*mobile* controllers only contain the boot loader.
The boot loader is a start program that allows to reload the runtime system and the application program on the device.
The boot loader contains basic routines...
- for communication between hardware modules,
- for reloading the operating system.

The boot loader is the first software module to be saved on the device.

Bus

Serial data transmission of several participants on the same cable.

C

CAN

CAN = **C**ontroller **A**rea **N**etwork
CAN is a priority-controlled fieldbus system for large data volumes. There are several higher-level protocols that are based on CAN, e.g. 'CANopen' or 'J1939'.

CAN stack

CAN stack = software component that deals with processing CAN messages.

© ifm electronic gmbh www.ifm.com

CiA

CiA = CAN in Automation e.V.
User and manufacturer organisation in Germany / Erlangen. Definition and control body for CAN and CAN-based network protocols.
Homepage → www.can-cia.org

CiA DS 304

DS = **D**raft **S**tandard
CANopen device profile for safety communication

CiA DS 401

DS = **D**raft **S**tandard
CANopen device profile for binary and analogue I/O modules

CiA DS 402

DS = **D**raft **S**tandard
CANopen device profile for drives

CiA DS 403

DS = **D**raft **S**tandard
CANopen device profile for HMI

CiA DS 404

DS = **D**raft **S**tandard
CANopen device profile for measurement and control technology

CiA DS 405

DS = **D**raft **S**tandard
CANopen specification of the interface to programmable controllers (IEC 61131-3)

CiA DS 406

DS = **D**raft **S**tandard
CANopen device profile for encoders

CiA DS 407

DS = **D**raft **S**tandard
CANopen application profile for local public transport

Clamp 15

In vehicles clamp 15 is the plus cable switched by the ignition lock.

COB ID

COB = **C**ommunication **Ob**ject
ID = **Id**entifier
ID of a CANopen communication object
Corresponds to the identifier of the CAN message with which the communication project is sent via the CAN bus.

CODESYS

CODESYS® is a registered trademark of 3S – Smart Software Solutions GmbH, Germany.
'CODESYS for Automation Alliance' associates companies of the automation industry whose hardware devices are all programmed with the widely used IEC 61131-3 development tool CODESYS®.
Homepage → www.codesys.com

CSV file

CSV = **C**omma **S**eparated **V**alues (also: **C**haracter **S**eparated **V**alues)
A CSV file is a text file for storing or exchanging simply structured data.
The file extension is `.csv`.

Example: Source table with numerical values:

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

This results in the following CSV file:

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

Cycle time

This is the time for a cycle. The PLC program performs one complete run.
Depending on event-controlled branchings in the program this can take longer or shorter.

D

Data type

Depending on the data type, values of different sizes can be stored.

Data type	min. value	max. value	size in the memory
BOOL	FALSE	TRUE	8 bits = 1 byte
BYTE	0	255	8 bits = 1 byte
WORD	0	65 535	16 bits = 2 bytes
DWORD	0	4 294 967 295	32 bits = 4 bytes
SINT	-128	127	8 bits = 1 byte
USINT	0	255	8 bits = 1 byte
INT	-32 768	32 767	16 bits = 2 bytes
UINT	0	65 535	16 bits = 2 bytes
DINT	-2 147 483 648	2 147 483 647	32 bits = 4 bytes
UDINT	0	4 294 967 295	32 bits = 4 bytes
REAL	$-3.402823466 \cdot 10^{38}$	$3.402823466 \cdot 10^{38}$	32 bits = 4 bytes
ULINT	0	18 446 744 073 709 551 615	64 Bit = 8 Bytes
STRING			number of char. + 1

DC

Direct **C**urrent

© ifm electronic gmbh www.ifm.com

Diagnosis

During the diagnosis, the "state of health" of the device is checked. It is to be found out if and what →faults are given in the device.

Depending on the device, the inputs and outputs can also be monitored for their correct function.
- wire break,
- short circuit,
- value outside range.

For diagnosis, configuration and log data can be used, created during the "normal" operation of the device.
The correct start of the system components is monitored during the initialisation and start phase. Errors are recorded in the log file.
For further diagnosis, self-tests can also be carried out.

Dither

Dither is a component of the →PWM signals to control hydraulic valves. It has shown for electromagnetic drives of hydraulic valves that it is much easier for controlling the valves if the control signal (PWM pulse) is superimposed by a certain frequency of the PWM frequency. This dither frequency must be an integer part of the PWM frequency.

DLC

Data **L**ength **C**ode = in CANopen the number of the data bytes in a message.
For →SDO: DLC = 8

DRAM

DRAM = **D**ynamic **R**andom **A**ccess **M**emory.
Technology for an electronic memory module with random access (Random Access Memory, RAM). The memory element is a capacitor which is either charged or discharged. It becomes accessible via a switching transistor and is either read or overwritten with new contents. The memory contents are volatile: the stored information is lost in case of lacking operating voltage or too late restart.

DTC

DTC = **D**iagnostic **T**rouble **C**ode = error code
In the protocol J1939 faults and errors well be managed and reported via assigned numbers – the DTCs.

E

ECU

(1) **E**lectronic **C**ontrol **U**nit = control unit or microcontroller
(2) **E**ngine **C**ontrol **U**nit = control device of a engine

EDS-file

EDS = **E**lectronic **D**ata **S**heet, e.g. for:
- File for the object directory in the CANopen master,
- CANopen device descriptions.

Via EDS devices and programs can exchange their specifications and consider them in a simplified way.

© ifm electronic gmbh www.ifm.com

Embedded software

System software, basic program in the device, virtually the →runtime system.
The firmware establishes the connection between the hardware of the device and the application program. The firmware is provided by the manufacturer of the controller as a part of the system and cannot be changed by the user.

EMC

EMC = **E**lectro **M**agnetic **C**ompatibility.
According to the EC directive (2004/108/EEC) concerning electromagnetic compatibility (in short EMC directive) requirements are made for electrical and electronic apparatus, equipment, systems or components to operate satisfactorily in the existing electromagnetic environment. The devices must not interfere with their environment and must not be adversely influenced by external electromagnetic interference.

EMCY

Abbreviation for emergency
Message in the CANopen protocol with which errors are signalled.

Ethernet

Ethernet is a widely used, manufacturer-independent technology which enables data transmission in the network at a speed of 10...10 000 million bits per second (Mbps). Ethernet belongs to the family of so-called "optimum data transmission" on a non exclusive transmission medium. The concept was developed in 1972 and specified as IEEE 802.3 in 1985.

EUC

EUC = **E**quipment **U**nder **C**ontrol.
EUC is equipment, machinery, apparatus or plant used for manufacturing, process, transportation, medical or other activities (→ IEC 61508-4, section 3.2.3). Therefore, the EUC is the set of all equipment, machinery, apparatus or plant that gives rise to hazards for which the safety-related system is required.
If any reasonably foreseeable action or inaction leads to →hazards with an intolerable risk arising from the EUC, then safety functions are necessary to achieve or maintain a safe state for the EUC. These safety functions are performed by one or more safety-related systems.

F

FiFo

FIFO (**F**irst **I**n, **F**irst **O**ut) = Operating principle of the stack memory: The data packet that was written into the stack memory first, will also be read first. Each identifier has such a buffer (queue).

Flash memory

Flash ROM (or flash EPROM or flash memory) combines the advantages of semiconductor memory and hard disks. Similar to a hard disk, the data are however written and deleted blockwise in data blocks up to 64, 128, 256, 1024, ... bytes at the same time.

Advantages of flash memories

- The stored data are maintained even if there is no supply voltage.
- Due to the absence of moving parts, flash is noiseless and insensitive to shocks and magnetic fields.

© ifm electronic gmbh www.ifm.com

Disadvantages of flash memories

- A storage cell can tolerate a limited number of write and delete processes:
  - Multi-level cells: typ. 10 000 cycles
  - Single level cells: typ. 100 000 cycles
- Given that a write process writes memory blocks of between 16 and 128 Kbytes at the same time, memory cells which require no change are used as well.

FRAM

FRAM, or also FeRAM, means **Fe**rroelectric **R**andom **A**ccess **M**emory. The storage operation and erasing operation is carried out by a polarisation change in a ferroelectric layer.
Advantages of FRAM as compared to conventional read-only memories:
- non-volatile,
- compatible with common EEPROMs, but:
- access time approx. 100 ns,
- nearly unlimited access cycles possible.

H

Heartbeat

The participants regularly send short signals. In this way the other participants can verify if a participant has failed.

HMI

HMI = **H**uman **M**achine **I**nterface

I

ID

ID = **Id**entifier

Name to differentiate the devices / participants connected to a system or the message packets transmitted between the participants.

IEC 61131

Standard: Basics of programmable logic controllers
- Part 1: General information
- Part 2: Production equipment requirements and tests
- Part 3: Programming languages
- Part 5: Communication
- Part 7: Fuzzy Control Programming

IEC user cycle

IEC user cycle = PLC cycle in the CODESYS application program.

Instructions

Superordinate word for one of the following terms:
installation instructions, data sheet, user information, operating instructions, device manual, installation information, online help, system manual, programming manual, etc.

© ifm electronic gmbh www.ifm.com

Intended use

Use of a product in accordance with the information provided in the instructions for use.

IP address

IP = **I**nternet **P**rotocol.
The IP address is a number which is necessary to clearly identify an internet participant. For the sake of clarity the number is written in 4 decimal values, e.g. 127.215.205.156.

ISO 11898

Standard: Road vehicles – Controller area network
- Part 1: Data link layer and physical signalling
- Part 2: High-speed medium access unit
- Part 3: Low-speed, fault-tolerant, medium dependent interface
- Part 4: Time-triggered communication
- Part 5: High-speed medium access unit with low-power mode

ISO 11992

Standard: Interchange of digital information on electrical connections between towing and towed vehicles
- Part 1: Physical and data-link layers
- Part 2: Application layer for brakes and running gear
- Part 3: Application layer for equipment other than brakes and running gear
- Part 4: Diagnostics

ISO 16845

Standard: Road vehicles – Controller area network (CAN) – Conformance test plan

J

J1939

→ SAE J1939

L

LED

LED = **L**ight **E**mitting **D**iode.
Light emitting diode, also called luminescent diode, an electronic element of high coloured luminosity at small volume with negligible power loss.

Link

A link is a cross-reference to another part in the document or to an external document.

LSB

Least **S**ignificant **B**it/Byte

© ifm electronic gmbh www.ifm.com

M

MAC-ID

MAC = **M**anufacturer‘s **A**ddress **C**ode
= manufacturer's serial number.
→ID = **Id**entifier
Every network card has a MAC address, a clearly defined worldwide unique numerical code, more or less a kind of serial number. Such a MAC address is a sequence of 6 hexadecimal numbers, e.g. "00-0C-6E-D0-02-3F".

Master

Handles the complete organisation on the bus. The master decides on the bus access time and polls the →slaves cyclically.

Misuse

The use of a product in a way not intended by the designer.
The manufacturer of the product has to warn against readily predictable misuse in his user information.

MMI

→ **HMI** (→ p. 327)

MRAM

MRAM = **M**agnetoresistive **R**andom **A**ccess **M**emory
The information is stored by means of magnetic storage elements. The property of certain materials is used to change their electrical resistance when exposed to magnetic fields.
Advantages of MRAM as compared to conventional RAM memories:
- non volatile (like FRAM), but:
- access time only approx. 35 ns,
- unlimited number of access cycles possible.

MSB

Most **S**ignificant **B**it/Byte

N

NMT

NMT = **N**etwork **M**anagement = (here: in the CANopen protocol).
The NMT master controls the operating states of the NMT slaves.

Node

This means a participant in the network.

Node Guarding

Node = here: network participant
Configurable cyclic monitoring of each →slave configured accordingly. The →master verfies if the slaves reply in time. The slaves verify if the master regularly sends requests. In this way failed network participants can be quickly identified and reported.

O

Obj / object

Term for data / messages which can be exchanged in the CANopen network.

Object directory

Contains all CANopen communication parameters of a device as well as device-specific parameters and data.

OBV

Contains all CANopen communication parameters of a device as well as device-specific parameters and data.

OPC

OPC = **O**LE for **P**rocess **C**ontrol
Standardised software interface for manufacturer-independent communication in automation technology
OPC client (e.g. device for parameter setting or programming) automatically logs on to OPC server (e.g. automation device) when connected and communicates with it.

Operational

Operating state of a CANopen participant. In this mode →SDOs, →NMT commands and →PDOs can be transferred.

P

PC card

→PCMCIA card

PCMCIA card

PCMCIA = Personal Computer Memory Card International Association, a standard for expansion cards of mobile computers.
Since the introduction of the cardbus standard in 1995 PCMCIA cards have also been called PC card.

PDM

PDM = **P**rocess and **D**ialogue **M**odule.
Device for communication of the operator with the machine / plant.

PDO

PDO = **P**rocess **D**ata **O**bject.
The time-critical process data is transferred by means of the "process data objects" (PDOs). The PDOs can be freely exchanged between the individual nodes (PDO linking). In addition it is defined whether data exchange is to be event-controlled (asynchronous) or synchronised. Depending on the type of data to be transferred the correct selection of the type of transmission can lead to considerable relief for the →CAN bus.
According to the protocol, these services are unconfirmed data transmission: it is not checked whether the receiver receives the message. Exchange of network variables corresponds to a "1 to n connection" (1 transmitter to n receivers).

PDU

PDU = **P**rotocol **D**ata **U**nit = protocol data unit.
The PDU is a term from the →CAN protocol →SAE J1939. It refers to a component of the target address (PDU format 1, connection-oriented) or the group extension (PDU format 2, message-oriented).

PES

Programmable **E**lectronic **S**ystem ...
- for control, protection or monitoring,
- dependent for its operation on one or more programmable electronic devices,
- including all elements of the system such as input and output devices.

PGN

PGN = **P**arameter **G**roup **N**umber
PGN = 6 zero bits + 1 bit reserved + 1 bit data page + 8 bit PDU Format (PF) + 8 PDU Specific (PS)
The parameter group number is a term from the →CAN protocol →SAE J1939.

Pictogram

Pictograms are figurative symbols which convey information by a simplified graphic representation.
(→ chapter **What do the symbols and formats mean?** (→ p. 7))

PID controller

The PID controller (proportional–integral–derivative controller) consists of the following parts:
- P = proportional part
- I = integral part
- D = differential part (but not for the controller CR04nn, CR253n).

PLC configuration

Part of the CODESYS user interface.

► The programmer tells the programming system which hardware is to be programmed.
> CODESYS loads the corresponding libraries.
> Reading and writing the periphery states (inputs/outputs) is possible.

Pre-Op

Pre-Op = PRE-OPERATIONAL mode.
Operating status of a CANopen participant. After application of the supply voltage each participant automatically passes into this state. In the CANopen network only →SDOs and →NMT commands can be transferred in this mode but no process data.

Process image

Process image is the status of the inputs and outputs the PLC operates with within one →cycle.
- At the beginning of the cycle the PLC reads the conditions of all inputs into the process image. During the cycle the PLC cannot detect changes to the inputs.
- During the cycle the outputs are only changed virtually (in the process image).
- At the end of the cycle the PLC writes the virtual output states to the real outputs.

© ifm electronic gmbh www.ifm.com

PWM

PWM = pulse width modulation
The PWM output signal is a pulsed signal between GND and supply voltage.
Within a defined period (PWM frequency) the mark-to-space ratio is varied. Depending on the mark-to-space ratio, the connected load determines the corresponding RMS current.

R

ratiometric

Measurements can also be performed ratiometrically. If the output signal of a sensor is proportional to its suppy voltage then via ratiometric measurement (= measurement proportional to the supply) the influence of the supply's fluctuation can be reduced, in ideal case it can be eliminated.
→ analogue input

RAW-CAN

RAW-CAN means the pure CAN protocol which works without an additional communication protocol on the CAN bus (on ISO/OSI layer 2). The CAN protocol is international defined according to ISO 11898-1 and garantees in ISO 16845 the interchangeability of CAN chips in addition.

remanent

Remanent data is protected against data loss in case of power failure.
The →runtime system for example automatically copies the remanent data to a →flash memory as soon as the voltage supply falls below a critical value. If the voltage supply is available again, the runtime system loads the remanent data back to the RAM memory.
The data in the RAM memory of a controller, however, is volatile and normally lost in case of power failure.

ro

RO = read only for reading only
Unidirectional data transmission: Data can only be read and not changed.

RTC

RTC = **R**eal **T**ime **C**lock
Provides (batter-backed) the current date and time. Frequent use for the storage of error message protocols.

Runtime system

Basic program in the device, establishes the connection between the hardware of the device and the application program.
→ chapter **Software modules for the device** (→ p. 40)

rw

RW = read/ write
Bidirectional data transmission: Data can be read and also changed.

© ifm electronic gmbh www.ifm.com

S

SAE J1939

The network protocol SAE J1939 describes the communication on a →CAN bus in commercial vehicles for transmission of diagnosis data (e.g.engine speed, temperature) and control information.
Standard: Recommended Practice for a Serial Control and Communications Vehicle Network
- Part 2: Agricultural and Forestry Off-Road Machinery Control and Communication Network
- Part 3: On Board Diagnostics Implementation Guide
- Part 5: Marine Stern Drive and Inboard Spark-Ignition Engine On-Board Diagnostics Implementation Guide
- Part 11: Physical Layer – 250 kBits/s, Shielded Twisted Pair
- Part 13: Off-Board Diagnostic Connector
- Part 15: Reduced Physical Layer, 250 kBits/s, Un-Shielded Twisted Pair (UTP)
- Part 21: Data Link Layer
- Part 31: Network Layer
- Part 71: Vehicle Application Layer
- Part 73: Application Layer – Diagnostics
- Part 81: Network Management Protocol

SD card

An SD memory card (short for **S**ecure **D**igital Memory Card) is a digital storage medium that operates to the principle of →flash storage.

SDO

SDO = **S**ervice **D**ata **O**bject.
The SDO is used for access to objects in the CANopen object directory. 'Clients' ask for the requested data from 'servers'. The SDOs always consist of 8 bytes.

Examples:
- Automatic configuration of all slaves via →SDOs at the system start,
- reading error messages from the →object directory.

Every SDO is monitored for a response and repeated if the slave does not respond within the monitoring time.

Self-test

Test program that actively tests components or devices. The program is started by the user and takes a certain time. The result is a test protocol (log file) which shows what was tested and if the result is positive or negative.

Slave

Passive participant on the bus, only replies on request of the →master. Slaves have a clearly defined and unique →address in the bus.

stopped

Operating status of a CANopen participant. In this mode only →NMT commands are transferred.

Symbols

Pictograms are figurative symbols which convey information by a simplified graphic representation.
(→ chapter **What do the symbols and formats mean?** (→ p. 7))

© ifm electronic gmbh www.ifm.com

System variable

Variable to which access can be made via IEC address or symbol name from the PLC.

T

Target

The target contains the hardware description of the target device for CODESYS, e.g.: inputs and outputs, memory, file locations.
Corresponds to an electronic data sheet.

TCP

The **T**ransmission **C**ontrol **P**rotocol is part of the TCP/IP protocol family. Each TCP/IP data connection has a transmitter and a receiver. This principle is a connection-oriented data transmission. In the TCP/IP protocol family the TCP as the connection-oriented protocol assumes the task of data protection, data flow control and takes measures in the event of data loss. (compare: →UDP)

Template

A template can be filled with content.
Here: A structure of pre-configured software elements as basis for an application program.

U

UDP

UDP (**U**ser **D**atagram **P**rotocol) is a minimal connectionless network protocol which belongs to the transport layer of the internet protocol family. The task of UDP is to ensure that data which is transmitted via the internet is passed to the right application.
At present network variables based on →CAN and UDP are implemented. The values of the variables are automatically exchanged on the basis of broadcast messages. In UDP they are implemented as broadcast messages, in CAN as →PDOs.
According to the protocol, these services are unconfirmed data transmission: it is not checked whether the receiver receives the message. Exchange of network variables corresponds to a "1 to n connection" (1 transmitter to n receivers).

Use, intended

Use of a product in accordance with the information provided in the instructions for use.

W

Watchdog

In general the term watchdog is used for a component of a system which watches the function of other components. If a possible malfunction is detected, this is either signalled or suitable program branchings are activated. The signal or branchings serve as a trigger for other co-operating system components to solve the problem.

© ifm electronic gmbh www.ifm.com

9 Index

A

B

C

D

© ifm electronic gmbh www.ifm.com

E

F

G

H

I

© ifm electronic gmbh www.ifm.com

J

L

M

N

O

© ifm electronic gmbh
www.ifm.com

P

R

S

© ifm electronic gmbh www.ifm.com

T

U

V

W

© ifm electronic gmbh
www.ifm.com

10 Notizen • Notes • Notes

© ifm electronic gmbh www.ifm.com

© ifm electronic gmbh
www.ifm.com

© ifm electronic gmbh
www.ifm.com

11 ifm weltweit • ifm worldwide • ifm à l'échelle internationale

Version: 2017-12-18 8310

ifm electronic gmbh • Friedrichstraße 1 • 45128 Essen
www.ifm.com • Email: info@ifm.com
Service hotline: 0800 / 16 16 16 (only Germany, Mo-Fr 07.00...18.00 h)

ifm Niederlassungen • Sales offices • Agences

D	Niederlassung Nord • 31135 Hildesheim • Tel. 05121 7667-0
	Niederlassung West • 45128 Essen • Tel. 0201 36475 -0
	Niederlassung Mitte-West • 58511 Lüdenscheid • Tel. 02351 4301-0
	Niederlassung Süd-West • 64646 Heppenheim • Tel. 06252 7905-0
	Niederlassung Baden-Württemberg • 73230 Kirchheim • Tel. 07021 8086-0
	Niederlassung Bayern • 82178 Puchheim • Tel. 089 80091-0
	Niederlassung Ost • 07639 Tautenhain • Tel. 036601 771-0
AE	ifm electronic FZC • Saif Zone, Sharjah • phone +971- 6-5573601
AR	ifm electronic s.r.l. • 1107 Buenos Aires • phone +54 11 5353-3436
AT	ifm electronic gmbh • 1120 Wien • phone +43 / 1 / 617 45 00
AU	ifm efector pty ltd. • Mulgrave Vic 3170 • phone +61 1300 365 088
BE	ifm electronic n.v./s.a. • 1731 Zellik • phone +32 2 481 0220
BG	ifm electronic eood • 1202 Sofia • phone +359 2 807 59 69
BR	ifm electronic Ltda. • 03337-000 Sao Paulo / SP • phone +55-11-2672-1730
CA	ifm efector Canada inc. • Mississauga, ON L5N 2X7 • phone +1 855-436-2262
CH	ifm electronic ag • 4624 Härkingen • phone +41 / 800 88 80 33
CL	ifm electronic SpA • Oficina 5041 Comuna de Conchalí • phone +56-2-32239282
CN	ifm electronic (Shanghai) Co. Ltd. • 201203 Shanghai • phone +86 21 3813 4800
CZ	ifm electronic, spol. s.r.o. • 140 00 Praha 4 • phone +420 267 990 211
DK	ifm electronic a/s • 2605 Brøndby • phone +45 70 20 11 08
ES	ifm electronic s.a. • 08820 El Prat de Llobregat • phone +34 93 479 30 80
FI	ifm electronic oy • 00440 Helsinki • phone +358 75 329 5000
FR	ifm electronic s.a. • 93192 Noisy-le-Grand Cedex • phone +33 0820 22 30 01
GB	ifm electronic Ltd. • Hampton, Middlesex TW12 2HD • phone +44 / 20 / 8213 0000
GR	ifm electronic monoprosopi E.P.E. • 15125 Amaroussio • phone +30 210 61 800 90
HU	ifm electronic kft. • 9028 Györ • phone +36-96 / 518-397
IN	ifm electronic India Private Limited • Kolhapur, 416234 • phone +91 / 231 / 267 27 70
IE	ifm electronic (Ireland) Ltd. • Dublin 22 • phone +353 / 1 / 461 32 00
IT	ifm electronic s.r.l. • 20864 Agrate Brianza (MB) • phone +39 39-6899982
JP	efector co., ltd. • Chiba-shi, Chiba 261-7118 • phone +81 043-299-2070
KR	ifm electronic Ltd. • 04420 Seoul • phone +82 2-790-5610
MX	ifm efector S. de R.L. de C.V. • San Pedro Garza Garcia, N.L. 66269 • phone +52-81-8040-3535
MY	ifm electronic Pte. Ltd • 47100 Puchong, Selangor • phone +603 8066 9853
NA	ifm elctronic (pty) Ltd • 25 Dr. W. Kulz Street Windhoek • phone +264 61 300984
NL	ifm electronic b.v. • 3843 GA Harderwijk • phone +31 341-438 438
NZ	ifm efector pty ltd • 930 Great South Road Penrose, Auckland • phone +64 / 95 79 69 91
PL	ifm electronic sp. z o.o. • 40-106 Katowice • phone +48 32 70 56 400
PT	ifm electronic s.a. • 4410-137 São Félix da Marinha • phone +351 223 71 71 08
RO	ifm electronic s.r.l • Sibiu 557260 • phone +40 269 224 550
RU	ifm electronic • 105318 Moscow • phone +7 495 921-44-14
SG	ifm electronic Pte Ltd • 609 916 Singapore • phone +65 6562 8661
SK	ifm electronic s.r.o. • 831 06 Bratislava • phone +421 244 872 329
SE	ifm electronic ab • 412 50 Göteborg • phone +46 31-750 23 00
TR	ifm electronic Ltd. Sti. • 34381 Sisli, Istanbul • phone +90 212 210 50 80
TW	ifm electronic • Kaohsiung City, 806, Taiwan R.O.C. • phone +886 7 3357778
UA	TOV ifm electronic • 02660 Kiew • phone +380 44 501-85-43
US	ifm efector inc. • Malvern, PA 19355 • phone +1 800-441-8246
VN	ifm electronic Vietnam Co., Ltd. • 700000 Ho Chi Minh City • phone +84-28-2253.6715
ZA	ifm electronic (Pty) Ltd. • 0157 Pretoria • phone +27 12 450 0412

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.

© ifm electronic gmbh www.ifm.com