ifm electronic

Original Programming Manual
SmartController

ecomat100

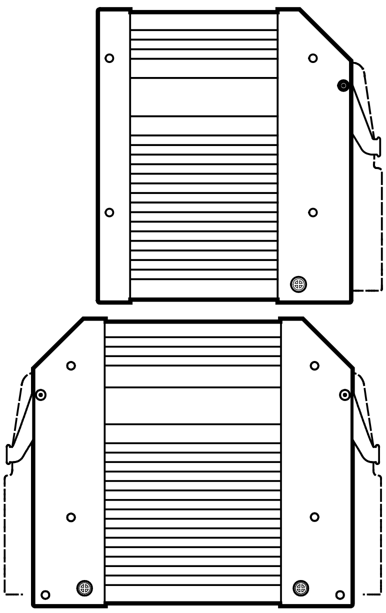
CR2530

with integrated I/O modul: **CR2532**

Runtime system v03.03.04

CODESYS® ≥ v2.3.9.33 (< v3.0)

English



7391003_06_UK 2016-09-23

Contents

1 About this manual

Contents

202

1.1 Copyright

6088

1.2 Overview: documentation modules for ecomatmobile devices

17405

The documentation for **ecomat*mobile*** devices consists of the following modules:

1.	**Data sheet**
Contents	Technical data in a table
Source	→ www.ifm.com > select your country > [Data sheet search] > CR2530 > [Technical data in PDF format]
2.	**Installation instructions / operating instructions**
Contents	Instructions for installation, electrical installation, (commissioning*), technical data
Source	The instructions are supplied with the device They are also found on **ifm**'s homepage: → www.ifm.com > select your country > [Data sheet search] > CR2530 > [Operating instructions]
3.	**Programming manual + online help**
Contents	Description of the configuration and the functions of the device software
Source	→ www.ifm.com > select your country > [Data sheet search] > CR2530 > [Operating instructions]
4.	**System manual "Know-how ecomatmobile"**
Contents	Know-how about the following topics: • Overview Templates and demo programs • CAN, CANopen • Control outputs • User flash memory • Visualisations • Overview of the files and libraries used
Source	→ www.ifm.com > select your country > [Data sheet search] > CR2530 > [Operating instructions]

*) The descriptions in brackets are only included in the instructions of certain devices.

1.3 CODESYS programming manual

17542

In the additional "Programming Manual for CODESYS V2.3" you obtain more details about the use of the programming system.
This manual can be downloaded free of charge from **ifm's** website:
→ www.ifm.com > Select your country > [Service] > [Download] > [Systems for mobile machines]
You also find manuals and online help for **ecomat*mobile*** at:
→ **ecomat*mobile*** DVD "Software, tools and documentation"

1.4 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 notes concerning malfunctions or disturbances
i	Other remarks
► ...	Request for action
> ...	Reaction, result
→ ...	"see"
abc	Cross-reference
123 0x123 0b010	Decimal number Hexadecimal number Binary number
[...]	Designation of pushbuttons, buttons or indications

1.5 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:
→ www.ifm.com > Select your country > [Contact].

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 at:
→ www.ifm.com > Select your country > [Service] > [Download]
⇒ Our online help is mostly updated without delay.
⇒ The pdf manuals are only updated at long intervals.

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 CR2530!

You can find more information about the integrated I/O module in:
→ chapter **Integrated I/O module: Description** (→ page 215) in the appendix of this documentation.

1.6 History of the instructions (CR253n)

15326

What has been changed in this manual? An overview:

Date	Theme	Change
2014-02-03	integrated I/O module	CR2532 description added
2014-04-28	Various function blocks	More precise description of the function block input CHANNEL
2014-04-29	FB CAN_REMOTE_RESPONSE	More precise description of the function block ENABLE
2014-05-12	Limitations CAN	Limitations added for CAN, CANopen and CAN J1939
2014-06-30	Name of the documentation	"System manual" renamed as "Programming manual"
2014-08-08	Chapter "Inputs of the integrated I/Omodule"	completed by sections "Analogue inputs" and "binary inputs"
2014-08-08	Chapter "Object directory of the integrated I/O module"	in the headlines "SDOs" replaced by "Object directory"
2014-08-08	Chapter "Input group I1 (IN04...IN05)"	replaced by section "Resistance measurement"
2014-08-08	FB PERIOD	completed by operating mode "phase measurement" (LZS V03.02.zz or higher)
2014-08-26	Description of inputs, outputs	highside / lowside replaced by positive / negative switching
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
2015-06-10	Various function blocks	Description of the FB input CHANNEL corrected
2015-07-20	Inputs IN12...IN15	now without operating mode 19
2015-08-04	Outputs OUT00...01	Diagnosis via current and voltage measurement
2015-10-22	Operating modes for inputs IN12...IN15	binary input = mode 01 (instead of mode 10)
2016-03-02	FBs OUTPUT, PWM1000	Description FB input CHANNEL corrected

2 Safety instructions

Contents

213

2.1 Please note

6091
11212

No characteristics are warranted with the information, notes and examples provided in this manual. With the drawings, representations and examples given no responsibility for the system is assumed and no application-specific particularities are taken into account.

► 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 personal 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 at:
→ www.ifm.com > Select your country > [Data sheet search] > (article number.) > [Technical data in PDF format]

► Note the installation and wiring information as well as the functions and features of the devices!
→ supplied installation instructions or on **ifm's** homepage:
→ www.ifm.com > Select your country > [Data sheet search] > (article number.) > [Operating instructions]

► Please note the corrections and notes in the release notes for the existing documentation, available on the **ifm** website:
→ www.ifm.com > Select your country > [Data sheet search] > (article number.) > [Operating instructions]

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 watchdog response because of too long a cycle time
- error elimination after an E-stop

To ensure a safe behaviour of the controller:

► Monitor the voltage supply in the application program.

► In case of an error switch off all relevant outputs in the application program.

► 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.

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.

3 System description

Contents

975

3.1 Information about the device

15329

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

- SmartController: CR2530

3.2 Hardware description

Contents

14081

3.2.1 Hardware structure

15332

Conditions

1376

The device does not start until sufficient voltage is applied to the supply connection VBBs.
A voltage > 8 V is deemed sufficient.
Permissible operating voltage → data sheet

Principle block diagram

1377

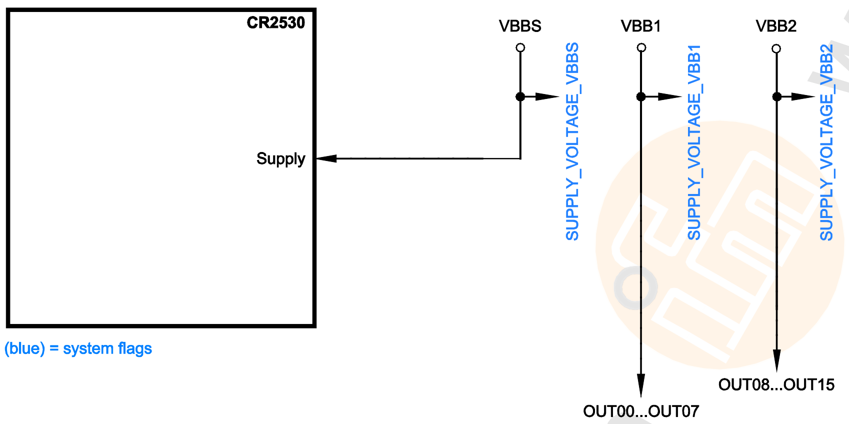


Figure: Block diagram of the supply

Available memory

13736

FLASH-Speicher

13053

FLASH memory (non-volatile, slow memory) overall existing in the device	1 536 kByte

Thereof the following memory areas are reserved for ...

maximum size of the application program	512 kByte
data other than the application program read data with FB **FLASH_READ** (→ page 190) (files: 128 bytes less for header)	64 kByte

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

SRAM

14027

SRAM (volatile, fast memory) overall existing in the device SRAM indicates here all kinds of volatile and fast memories.	592 kByte

Thereof the following memory areas are reserved for ...

data reserved by the application program	128 kByte

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

FRAM

2262

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

Thereof the following memory areas are reserved for ...

variables in the application program, declared as VAR_RETAIN	128 Byte
fixed as remanent defined flags (%MB0...127)	128 Byte

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

3.2.2 Inputs (technology)

Contents

14090

Analogue inputs

15444

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

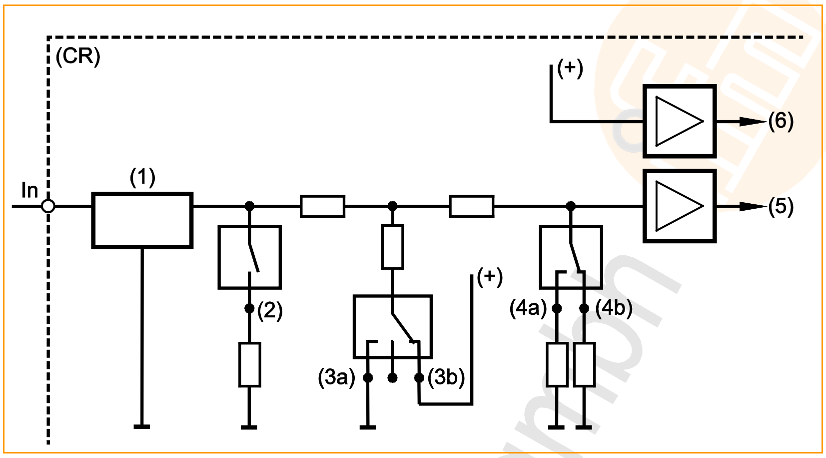
- current input 0...20 mA
- voltage input 0...10 V
- voltage input 0...32 V
- resistance measurement 16...30 000 Ω (measurement to GND)

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

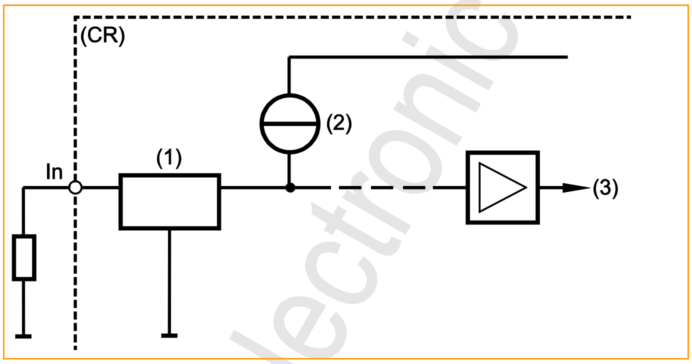


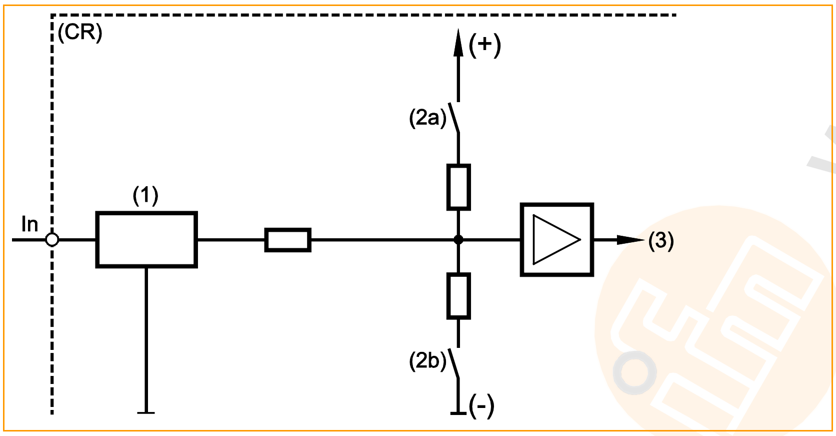
Figure: block diagram of the resistor survey input

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

Binary inputs 1015 7345

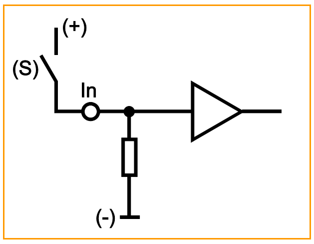
The binary input can be operated in following modes:
- binary input plus switching (BL) for positive sensor signal
- binary input minus switching (BH) for negative sensor signal

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



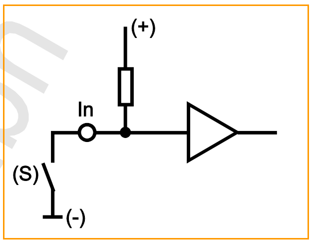
In = pin binary-input n
(CR) = device
(1) = input filter
(2a) = input minus switching
(2b) = input plus switching
(3) = voltage

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.

Input group IN00...IN03

15339

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

→ chapter **Possible operating modes inputs/outputs** (→ page 212)

All inputs show the same behaviour concerning function and diagnosis.

► Configuration of each input is made via the application program:
  • FB **INPUT** (→ page 176) > input MODE

> If the analogue inputs are configured for current measurement, the device switches to the safe voltage measurement range (0...32 V DC) and the output RESULT is set accordingly in the function block INPUT when the final value (23 mA for ≥ 40 ms) is exceeded. After about one second the input automatically switches back to the current measuring range.

Input group IN04...IN05

15341

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) (with/without diagnosis)

→ chapter **Possible operating modes inputs/outputs** (→ page 212)

► Configuration of each input is made via the application program:
  • FB **INPUT** (→ page 176) > input MODE

Resistance measurement

9773

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

8972

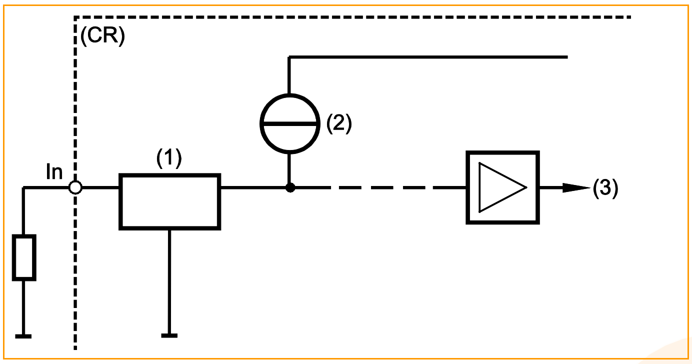


Figure: block diagram of the resistor survey input

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

8970

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

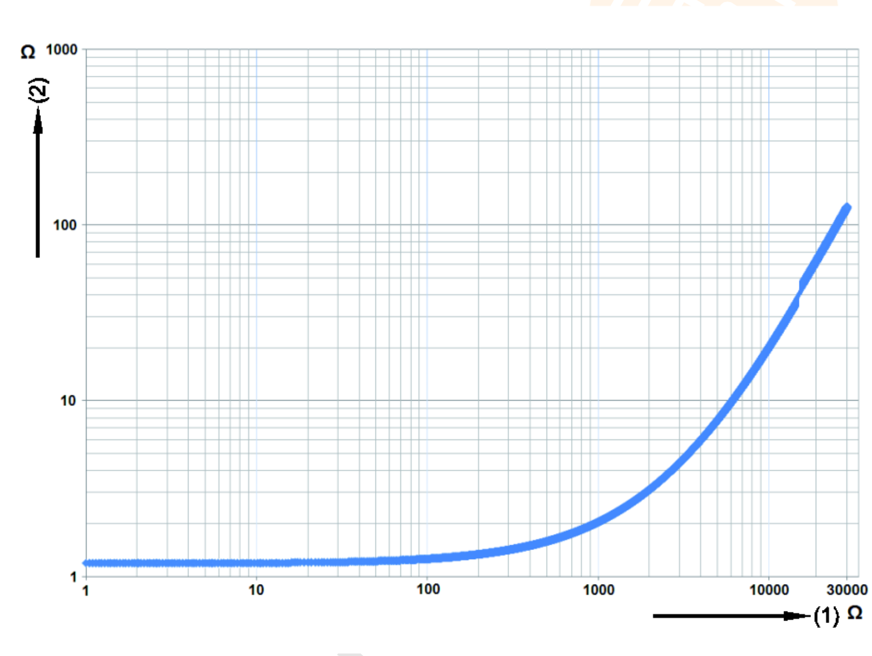


Figure: resolution dependent on the resistance value

(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.

Input group IN06...IN11

15344

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 inputs/outputs** (→ page 212)

Sensors with diagnostic capabilities to NAMUR can be evaluated.

► Configuration of each input is made via the application program:
  - FB **INPUT** (→ page 176) > input MODE

Input group IN12...IN15

15346

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 inputs/outputs** (→ page 212)

All inputs show the same behaviour concerning function and diagnosis.

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

► Configuration of each input is made via the application program:
  - FB **INPUT** (→ page 176) > input MODE

3.2.3 Outputs (technology)

Contents

14093

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

15644

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 93,5 % (± 2,0 %) of the corresponding supply voltage."

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

Reaction of the outputs to overload or short circuit

15251

Self-protection of the output

15253

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:
• FB **OUTPUT** (→ page 184) > input MODE = 16
This is also the default setting if only the flags in the control configuration are used.

Reaction according to the operating mode of the output

15252

In case of an overload or a short circuit, the behaviour of the output depends on its operating mode (→ FB **OUTPUT** (→ page 184) > input MODE):

- MODE=2: binary output plus switched: no diagnosis and no protection
  > the output continues to be operated.
- MODE=15: binary output plus switched with diagnosis
  > error is detected and signalled on the output RESULT by the FB OUTPUT:
  e.g.: RESULT = 128, 141, 142 or 145.
  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.
- MODE=16: binary output plus switched with diagnosis and protection
  > error is detected and signalled on the output RESULT by the FB OUTPUT.
  > The respective output is switched off.
  > ! The logic state of the output remains unaffected by this!

Reaction when using PWM or CURRENT_CONTROL

15254

The situation is different when the FBs PWM or CURRENT_CONTROL are used:
There is no diagnosis. The **Self-protection of the output** (→ page 22) becomes active.

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

Output group OUT0, OUT1

15351

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 inputs/outputs** (→ page 212)

► Configuration of each output is made via the application program:
→ FB **OUTPUT** (→ page 184)> input MODE
PWM output: → FB **PWM1000** (→ page 186)
Current control and load current indication → FB **CURRENT_CONTROL** (→ page 182)

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

Diagnosis: binary outputs (via current and voltage measurement)

19433

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

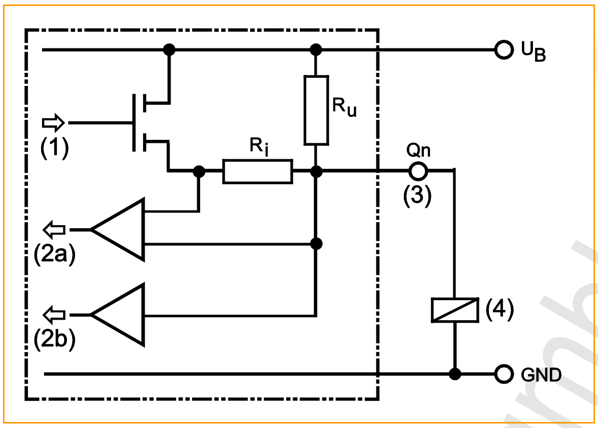


Figure: principle block diagram
(1) Output channel
(2a) Read back channel for diagnosis via current measuring
(2b) Read back channel for diagnosis via voltage measuring
(3) Pin output n
(4) Load

Diagnosis: overload (via current measurement)

19437

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

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)

Output group OUT02...OUT07

15353

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 inputs/outputs** (→ page 212)

► Configuration of each output is made via the application program:
→ FB **OUTPUT** (→ page 184)> input MODE
PWM output: → FB **PWM1000** (→ page 186)

► ⓘ 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:

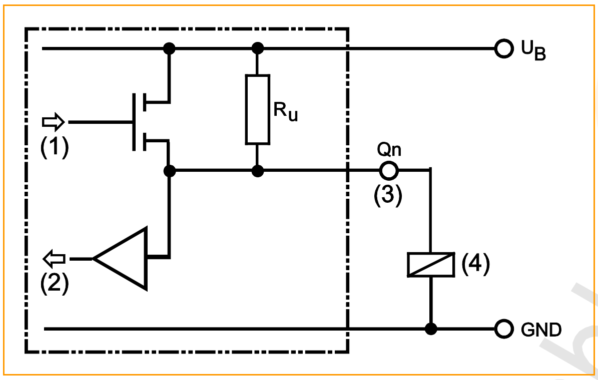


Figure: principle block diagram
(1) Output channel
(2) Read back channel for diagnosis
(3) Pin output n
(4) Load

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)

Output group OUT08...OUT09

15355

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
- These outputs are not able for diagnosis.

→ chapter **Possible operating modes inputs/outputs** (→ page 212)

► Configuration of each output is made via the application program:
→ FB **OUTPUT** (→ page 184)> input MODE
PWM output: → FB **PWM1000** (→ page 186)

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

Output group OUT10...OUT11

15362

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)
- These outputs are not able for diagnosis.

→ chapter **Possible operating modes inputs/outputs** (→ page 212)

► Configuration of each output is made via the application program:
→ FB **OUTPUT** (→ page 184)> input MODE
PWM output: → FB **PWM1000** (→ page 186)

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

Output group OUT12...OUT15

15364

These outputs are a group of channels with a single specified function.

These outputs have the following fixed setting:
- binary output, plus switching (BH)
- These outputs are not able for diagnosis.

→ chapter **Possible operating modes inputs/outputs** (→ page 212)

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

3.2.4 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

3.2.5 Safety instructions about Reed relays

7348

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

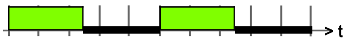
> ! Contacts of Reed relays may be clogged (reversibly) if connected to the device inputs without series resistor.

- ► **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

3.2.6 Status LED

20646

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

LED colour	Display	Description
Off	Permanently off	no operating voltage
		
Orange	Briefly on	initialisation or reset checks
		(time frame = 200 ms)
Green	Flashing with 5 Hz	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 5 Hz	application = stopped because of undervoltage
		(time frame = 200 ms)
Red	Permanently on	system error (FATAL ERROR): application = STOP
		

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

Control the LED in the application program

15481

Via SET_LED frequency and color of the status LED can be changed in the application program.

> ! The use of the LED function block in the application program replaces the system setting of the status LED in the RUN state.

3.3 Interface description

Contents

14098

3.3.1 CAN interfaces

Contents

14101

Connections and data → data sheet

CAN: interfaces and protocols

14589
15238

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:

- RAW-CAN (Layer 2): CAN on level 2 (→ chapter **Function elements: RAW-CAN (Layer 2)** (→ page 68))
- CANopen master / CANopen slave (→ chapter **Function elements: CANopen** (→ page 94))
- CANopen network variables (via CODESYS) (→ chapter **Network variables** (→ page 60))
- SAE J1939 (for drive management, → chapter **Function elements: SAE J1939** (→ page 139))
- Bus load detection
- Error frame counter
- Download interface
- 100 % bus load without package loss

14591

The following CAN interfaces and CAN protocols are available in this **ecomatmobile** 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 CANopen SAE J1939	CAN Layer 2 CANopen SAE J1939	Interface does not exist	Interface does not exist

Standard baud rate = 250 Kbits/s

All CAN interfaces can operate with all CAN protocols at the same time. The IDs used must not impair each other!

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** (→ page 45)

Below we describe this software module:

Bootloader

14111

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.

Runtime system

14112

Basic program in the device, establishes the connection between the hardware of the device and the application program.

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**:
→ www.ifm.com > Select your country > [Service] > [Download]

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.

Libraries

15409

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

Library	Use
`ifm_CR2530_Vxxyyzz.LIB`	Device-specific library Must always be contained in the application program!
`ifm_RawCAN_NT_Vxxyyzz.LIB`	(optional) when a CAN interface of the device is to be operated with CAN Layer 2
`ifm_CANopen_NT_Vxxyyzz.LIB`	(optional) when a CAN interface of the device is to be operated as CANopen master or CANopen slave
`ifm_J1939_NT_Vxxyyzz.LIB`	(optional) when a CAN interface of the device is to communicate with a motor control

Details: → chapter **ifm libraries for the device CR2530** (→ page 61)

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
→ www.ifm.com > select your country > [Data sheet search] > CR2530 > [Operating instructions]
→ **ecomat*mobile*** DVD "Software, tools and documentation".

FB, FUN, PRG in CODESYS

15410

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 may contain maximum 8000 function elements; in this device maximum 512 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 function block call in a function:
- every time there is an additional initialisation call and
- the data of the last call gets lost.

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.

Important note to program the device

20763

Applies to the following devices:
• BasicController relay CR0431

► For the time of programming interconnect the connections B:1 (VBB15) and B:8 (VBBs). Otherwise programming is not possible.

Background:

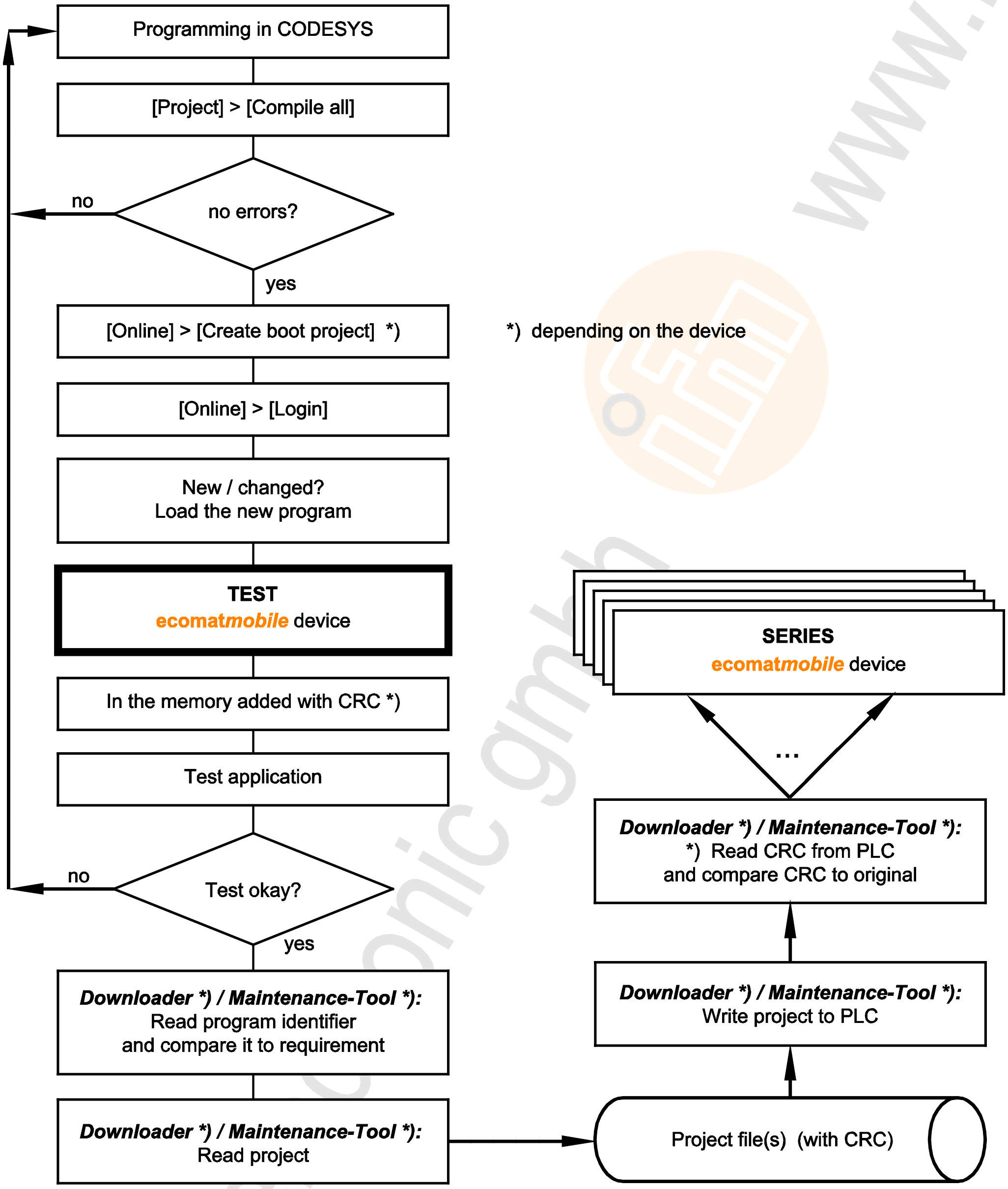
- The controller resets all outputs when programming begins, also SUPPLY_SWITCH.
- Without VBB15 the controller would be separated from the voltage supply and is switched off.
- When the controller is switched on again, the device is in bootloader mode.
  The programmer has to load the Basic System to the device again.
  Then reload the application program to the device.

Creating application program

8007

The application program is generated by the CODESYS 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 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

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:
→ www.ifm.com > Select your country > [Service] > [Download] > [Systems for mobile machines]
→ **ecomat*mobile*** DVD "Software, tools and documentation" under the tab 'R360 tools [D/E]'

Distribution of the application program

8493

We recommend the following sequence, if the application software is to be copied to the series machine and used:

- Saving the software
  After completion of program development the latest version of the application program loaded in the controller using the **ifm** Maintenance Tool has to be read from the controller and saved on a data carrier using the name `project_file.RESX`. Only this process ensures that the application software and its checksums are stored.
- Download of the software.
  To equip all machines of a series production with an identical software only this file may be loaded in the controllers using the **ifm** Maintenance Tool.
- An error in the data of this file is automatically recognised by the integrated checksum when loaded again using the **ifm** Maintenance Tool.

3.4.3 Operating states

1075

After power on the **ecomat*mobile*** device can be in one of five possible operating states:
• BOOTLOADER
• INIT
• STOP
• RUN
• SYSTEM STOP

INIT state (Reset)

20647

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 orange.

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

STOP state

8288

A transition into this state is possible in the following cases:
• from the INIT state if no application program is loaded.
• From the RUN state if the following condition is met:
  • The STOP command is sent via the CODESYS interface.

In the STOP state:

> The outputs of the device are switched off.

> Processing of the application program is stopped.

> The LED lights green.

A transition from this state into one of the following states is possible:
• RUN
• ERROR
• FATAL ERROR
• INIT (after power-on-reset)

RUN state

8287

A transition into this state is possible in the following cases:

- from the INIT state (autostart) if the following conditions are met:
  • The operating voltage has reached a minimum value. AND:
  • The application program exists.
- From the STOP state:
  • via the CODESYS command RUN.
  • The operating voltage has reached or exceeded a minimum value.

In the RUN state:

> The runtime system is running.

> The application program is running.

> The LED flashes green with 2 Hz.
The LED can be controlled differently by the application program → FB **SET_LED** (→ page 202).

A transition from this state into one of the following states is possible:
• INIT (after power-on-reset)
• STOP
• ERROR
• FATAL ERROR

ERROR state

8290

A transition into this state is possible in the following cases:
• if the supply voltage is too low.

In the ERROR state:

> The outputs of the device are switched off.

> Processing of the application program is stopped.

> System parameters are saved.

> The LED flashed red with 5 Hz.

A transition from this state into one of the following states is possible:
• INIT (after power-on-reset)
• RUN
• STOP
• FATAL ERROR

FATAL ERROR state

8289

A transition into this state is possible in the following cases:
• memory error (RAM / Flash)
• exception error
• runtime system error

In the FATAL ERROR state:

> The outputs of the device are switched off.

> The application program is terminated.

> The runtime system is terminated.

> The LED lights red.

A transition from this state into one of the following states is possible:
• INIT (after power-on-reset)

3.4.4 Performance limits of the device

7358

Note the limits of the device! → Data sheet

Watchdog behaviour

15365

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

If the maximum watchdog time (100 ms) is exceeded:
> the device changes to the "Timeout Error" state
> all processes are stopped (reset)
> all outputs are switched off
> the status LED flashes red at 10 Hz

Eliminate the fault:
- ▶ Delete application program!
- ▶ PowerOn reset
- ▶ Reload the application program into the device

If the watchdog in question fails:
> a second watchdog leads the device to the state "Fatal Error"
> the status LED lights red

Eliminate the fault:
- ▶ PowerOn reset

If without success:
- ▶ Goto Bootloader
- ▶ PowerOn reset
- ▶ Reload the runtime system into the device
- ▶ Reload the application program into the device

If without success:
- ▶ Hardware error: send device to **ifm**!

Limitations for CAN in this device

17975

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).

Some Raw-CAN function elements enable transmitting and receiving of several messages in one PLC cycle as the messages are temporarily stored in a FiFo:
- CAN_TX..., → Function elements: transmit RAW-CAN data
- **CAN_RX_ENH_FIFO** (→ page 78)
- **CAN_RX_RANGE_FIFO** (→ page 82)

The number of FIFo messages is limited. The following limitations of the devices are valid:

Device / **Criterion**	BasicController: CR040n, CR041n, CR043n BasicDisplay: CR045n SmartController: CR253n	PDM360 NG: CR108n, CR120n
max. FiFo transmit - with FB CAN_TX... - with FB CAN_TX_ENH...	 4 messages 16 messages	 4 messages 16 messages
max. FiFo receive - with FB CAN_RX_..._FIFO	32 messages	32 messages

Limitations for CANopen in this device

17976

The following limitations of the devices are valid:

Device / **Criterion**	BasicController: CR040n, CR041n, CR043n BasicDisplay: CR045n SmartController: CR253n	PDM360 NG: CR108n, CR120n
max. guarding error	32 messages	128 messages
max. SDO data	2 048 bytes	2 048 bytes

Limitations for CAN J1939 in this device

17977

The following limitations of the devices are valid:

Device / **Criterion**	BasicController: CR040n, CR041n, CR043n BasicDisplay: CR045n SmartController: CR253n	PDM360 NG: CR108n, CR120n
max. FiFo transmit - with FB J1939_TX - with FB J1939_TX_ENH	 4 messages 16 messages	 4 messages 16 messages
max. FiFo receive - with FB J1939_RX_FIFO	32 messages	32 messages
max. DTCs	64 messages	64 messages
max. data J1939	1 785 bytes	1 785 bytes

4 Configurations

Contents

18065
1016

The device configurations described in the corresponding installation instructions or in the **Appendix** (→ page 208) 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 CR2530!

You can find more information about the integrated I/O module in:
→ chapter **Integrated I/O module: Description** (→ page 215) in the appendix of this documentation.

4.1 Set up the runtime system

Contents

14091

4.1.1 Reinstall the runtime system

14635
8486

On delivery of the **ecomat*mobile*** controller no runtime system is normally loaded (LED flashes green at 5 Hz). Only the boot loader is active in this operating mode. It provides the minimum functions for loading the operating 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** at:
→ www.ifm.com > Select your country > [Service] > [Download] > [Systems for mobile machines]

NOTICE

Risk of data loss!

In case of power failure during the data transmission data can be lost so that the device is no longer functionable. Repair is only possible by **ifm electronic**.

► Ensure an uninterrupted power supply during the data transmission!

NOTE

The software versions suitable for the selected target must always be used:
• runtime system (`ifm_CR2530_Vxxyyzz.RESX`),
• PLC configuration (`ifm_CR2530_Vxx.CFG`),
• device library (`ifm_CR2530_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. "`CR2530`") and the software version number "xx" (e.g. "`01`") 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**.

The runtime system is transferred to the device using the separate program "Maintenance Tool". (The downloader is on the **ecomat*mobile*** DVD "Software, tools and documentation" or can be downloaded from **ifm's** website, if necessary):
→ www.ifm.com > Select your country > [Service] > [Download] > [Systems for mobile machines].

Normally the application program is loaded to the device via the programming system. But it can also be loaded using the "Maintenance Tool" if it was first read from the device.

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

14637

► After loading of the runtime system into the controller:
  • Check whether the runtime system was transmitted correctly!
  • Check whether the correct runtime system is loaded in the controller!

► 1st test:
  Test with the **ifm** maintenance tool if the correct runtime system version was loaded:
  • Read name and version of the runtime system in the device!
  • Manually compare this information with the target data!

► 2nd test (optional):
  Check in the application program if the correct runtime system version was loaded:
  • read 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_SW_INFO (→ page 194)	Delivers information about the system software of the device: • software name, • software version, • build number, • build date

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

4.2 Set up the programming system

Contents

14461

4.2.1 Set up the programming system manually

Contents

3963

Set up the target

13136
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_CR2530_Vxxyyzz.LIB` manually!

NOTE

The software versions suitable for the selected target must always be used:
- runtime system (`ifm_CR2530_Vxxyyzz.RESX`),
- PLC configuration (`ifm_CR2530_Vxx.CFG`),
- device library (`ifm_CR2530_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. "`CR2530`") and the software version number "xx" (e.g. "`01`") 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**.

Activate the PLC configuration

10079

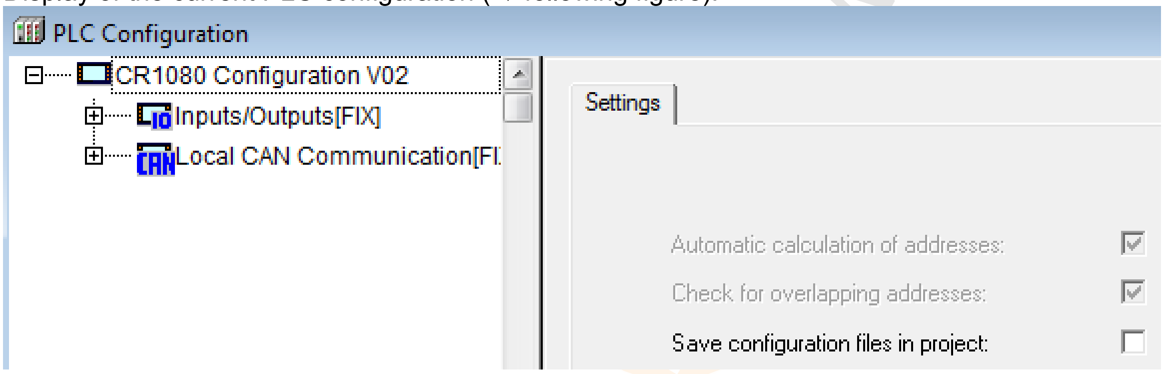
The PLC configuration is automatically loaded with the target system. The PLC configuration maps the contents of the file CR2530.cfg in CODESYS. Like this, the programmer has easy access to predefined system and error flags, inputs and outputs as well as to the CAN interfaces of the device.

To access the PLC configuration (e.g. CR1080):

► Click on the tab [Resources] in CoDeSys:

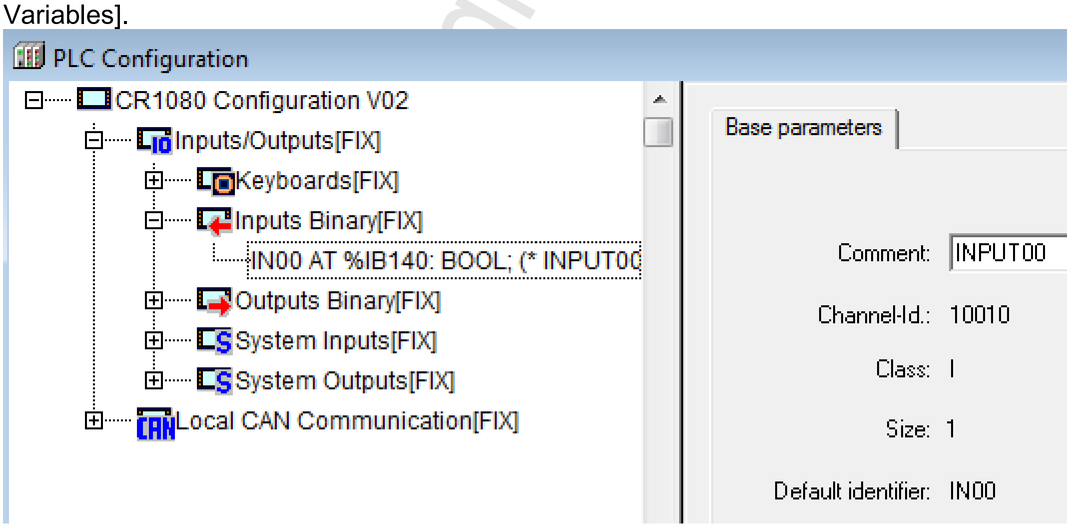
► Double-click on [PLC Configuration] in the left column.

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



> Based on the configuration the following is available in the program environment for the user:

- 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.
- Structure of the inputs and outputs
  These can be directly symbolically designated (highly recommended!) in the window [PLC Configuration] (example → figure below) and are available in the whole project as [Global Variables].

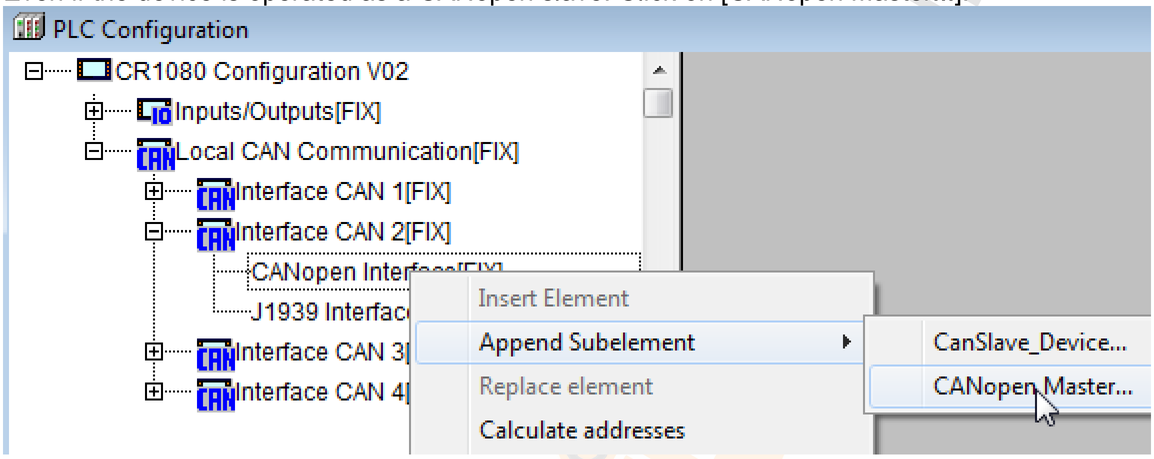


CAN declaration (e.g. CR1080)

10080

In the CODESYS PLC configuration you now have to declare the CAN interface(s).

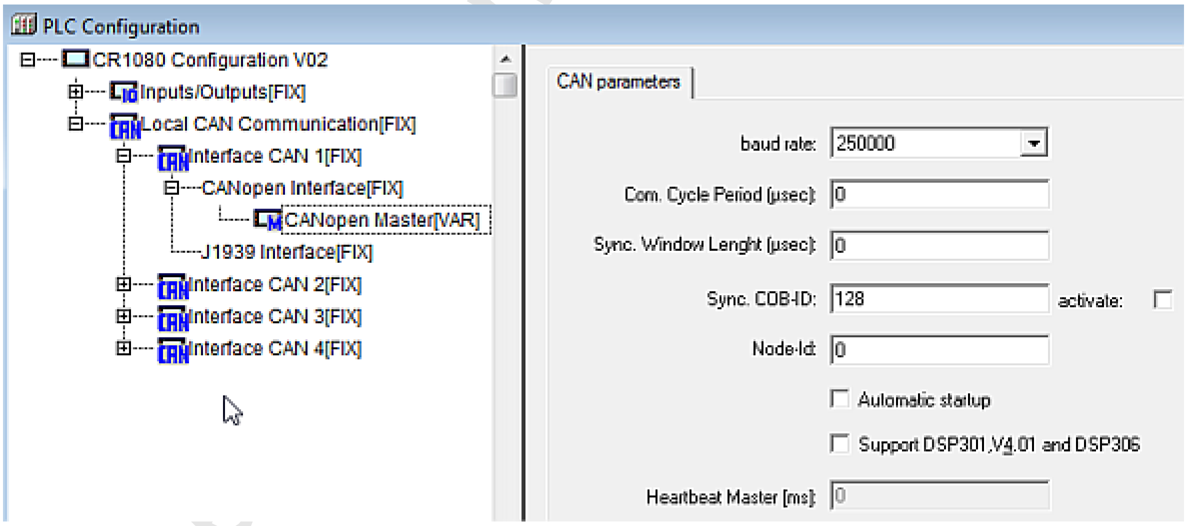
► Right-click on the name of the PLC configuration. [CANopen Interface [FIX]] of the desired CAN interface.
► Click on [Append Subelement].
► Even if the device is operated as a CANopen slave: Click on [CANopen Master...]:



Info

If the device is operated as a slave, the selection [CanSlave_Device] would also be possible.
For the simpler configuration as a master, all CAN Layer 2 and network variable functions can also be used.

> The CAN parameters of the PLC configuration are displayed. Some CAN parameters are already set as default:



- ► If the device is operated on CAN Layer 2 or as a slave via network variables or CAN_RX / CAN_TX:
  Check whether the correct baud rate is set for the device (baud rate must be identical for all participants).
- ► If the device is operated as a CANopen master:
  Check all parameter settings.
- ► Close the window [PLC Configuration].
- ► In the menu [File] > [Save as ...] give a sensible name to the project and save it in the requested directory.
- ► In the application program always call an own instance of the FB **CANOPEN_ENABLE** (→ page 95) for every CAN interface!

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 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.

4.3 Function configuration in general

3971

4.3.1 System variables

15576

All system variables (→ chapter **System flags** (→ page 208)) have defined addresses which cannot be shifted.

4.4 Function configuration of the inputs and outputs

Contents

7995
1394

For some devices of the **ecomat*mobile*** 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_CR2530_Vxxyyzz.LIB`) for the configuration of the inputs and outputs.
  For details → **Possible operating modes inputs/outputs** (→ page 212).

4.4.1 Configuration of the inputs and outputs (default setting)

2249

- All inputs and outputs are in the binary mode (plus switching!) when delivered.
- The diagnostic function is not active.
- The overload protection is active.

4.4.2 Configure inputs

Contents

3973

Valid operating modes → chapter **Possible operating modes inputs/outputs** (→ page 212)

Safety instructions about Reed relays

7348

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

> 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

Configure the software filters of the inputs

15418

Via the input FILTER in the FB **INPUT** (→ page 176) a software filter can be configured which filters the measured input voltage at the analogue inputs.

The filter behaves like a low-pass filter; the filter frequency is set with the value entered in FILTER. For FILTER, values from 0...8 are permitted.

Table: limit frequency software low-pass filter at the analogue input

FILTER	Filter frequency [Hz]	Step response [ms] for ...			Remarks
		0…70 %	0…90 %	0…99 %	
0	Filter deactivated				
1	120	2	4	7	
2	47	5	9	17	
3	22	10	18	35	
4	10	19	36	72	recommended
5	5	38	73	146	
6	2.5	77	147	293	
7	1.2	154	294	588	
8	0.7	308	589	1177	

The following statements of the step response are relevant:
- Input analogue: 0...90 % and 0...99 %
- Input binary: 0...70 %

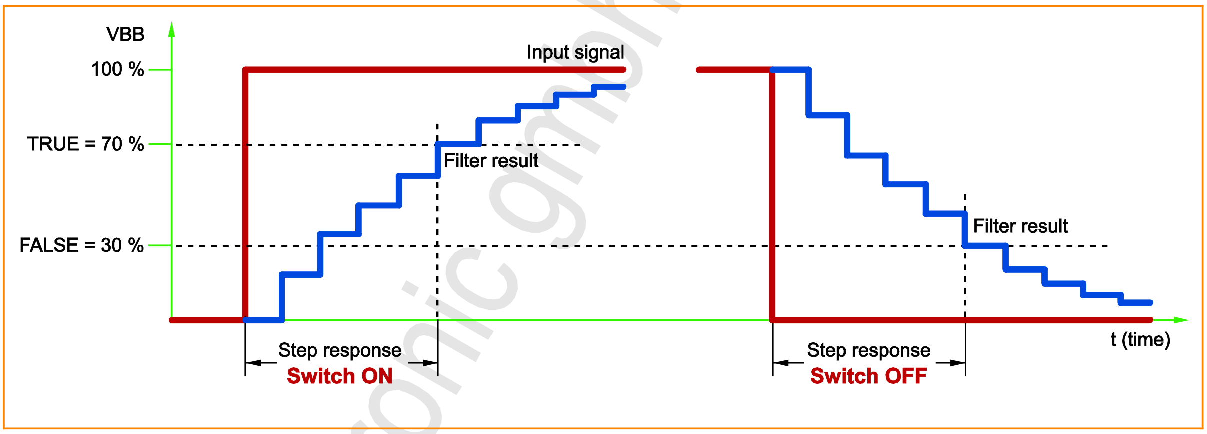


Figure: course of time binary signal at the input upon switch-on / switch-off

Analogue inputs: configuration and diagnosis

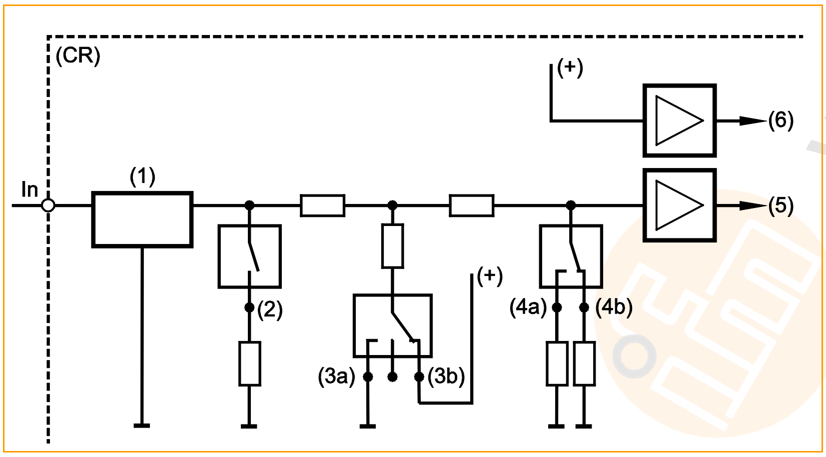
14656

Configuration of each input is made via the application program:
• FB **INPUT** (→ page 176) > input MODE

► If the analogue inputs are configured for current measurement, the device switches to the safe voltage measurement range (0...32 V DC) and the output RESULT is set accordingly in the function block INPUT when the final value (23 mA for ≥ 40 ms) is exceeded. After about one second the input automatically switches back to the current measuring range.

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

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

Binary inputs: configuration and diagnosis

14672

Configuration of each input is made via the application program:
• FB **INPUT** (→ page 176) > input MODE

MODE	BYTE	operating mode of the input channel:		
		0 = 0x00	off	
		3 = 0x03	voltage input	0…10 000 mV
		6 = 0x06	voltage input, ratiometric	0…1 000 ‰
		7 = 0x07	current input	0…20 000 µA
		9 = 0x09	voltage input	0…32 000 mV
		10 = 0x0A	(only for analogue evaluated inputs) binary input, plus switching (BL)	
		11 = 0x0B	(only for analogue evaluated inputs) binary input, plus switching (BL) with diagnosis (Namur)	
		12 = 0x0C	binary input, minus switching (BH)	
		18 = 0x12	resistance input	16…3 600 Ω from HW state AD: 16...30 000 Ω

Activation of the input diagnosis

7352

If the diagnosis is to be used, it needs to be activated additionally.

► Set the mode of the input via input MODE of the function block **INPUT** (→ page 176).

> The FB **INPUT** (→ page 176) provides the diagnostic messages of the inputs on its RESULT output.

NAMUR diagnosis for binary signals of non-electronic switches:

► Equip the switch with an additional resistor connection!

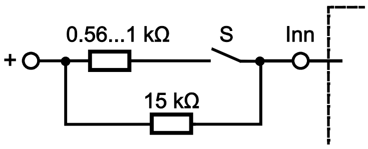


Figure: non-electronic switch S at input Inn

Sensors with diagnostic capabilities to NAMUR can be used on these inputs.
In this case, no additional resistor connection is required.

Fast inputs

8292

The devices dispose of fast counting/pulse inputs for an input frequency up to 30 kHz (→ data sheet).

! If, for example, mechanical switches are connected to these inputs, there may be faulty signals in the controller due to contact bouncing.

Appropriate function blocks are e.g.:

FASTCOUNT (→ page 172)	Counter block for fast input pulses
INC_ENCODER (→ page 174)	Up/down counter function for the evaluation of encoders
PERIOD (→ page 179)	Measures the frequency and the cycle period (cycle time) in [µs] at the indicated channel

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

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!

4.4.3 Configure outputs

Contents

3976

Valid operating modes → chapter **Possible operating modes inputs/outputs** (→ page 212)

Configure the software filters of the outputs

15421

Via the input FILTER in the FB **OUTPUT** (→ page 184) a software filter can be configured which filters the measured output current at the PWM outputs.

The FILTER byte is only valid for outputs with current measurement.
For outputs without current measurement: set FILTER = 0!

The current at the output is averaged over a PWM period.
If dithering is set, the current is averaged over the dither period.

The filter behaves like a low-pass filter; the limit frequency is set by the value entered in FILTER. For FILTER, values from 0...8 are permitted.

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 %	
0	Filter deactivated				outputs without current measurement
1	600	0.8	1.0	1.4	
2	233	1.8	2.2	3.4	
3	109	3.6	4.6	7.0	
4	52	7.2	9.4	14.4	recommended
5	26	14.6	19.0	29.2	
6	13	29.4	38.2	58.6	
7	6	58.8	76.4	117.6	
8	4	117.8	153.2	235.4	

The following statements of the step response are relevant:
• Output current: 0...90 % and 0...99 %

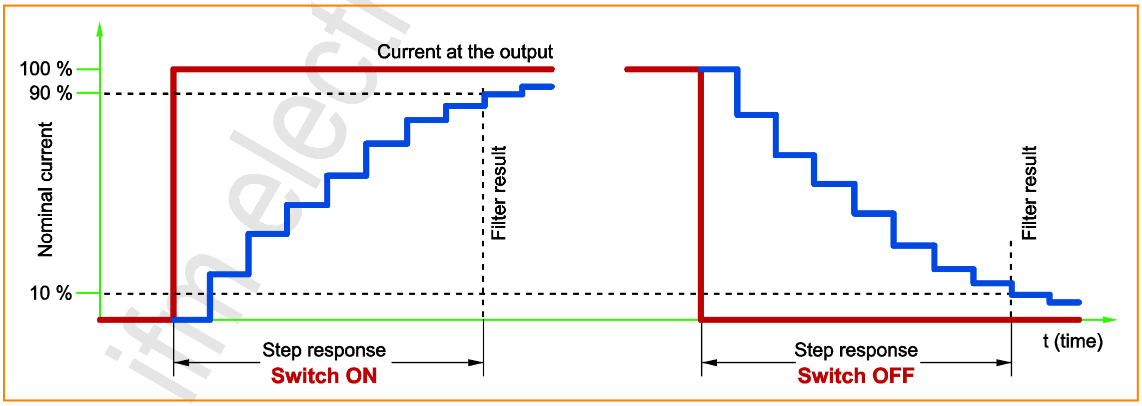


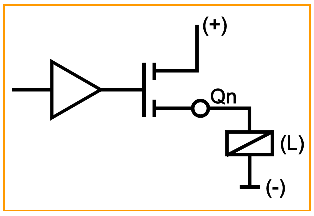
Figure: time sequence binary current signal on output upon switch-on / switch-off

Binary outputs: configuration and diagnosis

14689

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

15451



Qn = pin output n
(L) = load

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

► Configuration of each output is made via the application program:
→ FB **OUTPUT** (→ page 184)> input MODE.

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.

Configuration of the output diagnosis

8301

If the diagnosis is to be used, it needs to be activated additionally.

► If using the output as binary output with diagnosis (→ data sheet):
→ FB OUTPUT > input MODE = 15 or 18

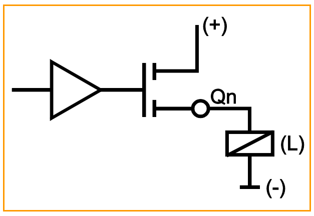
> The FB **OUTPUT** (→ page 184) provides the diagnostic messages of the outputs on its RESULT output.

PWM outputs

14705

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

15414

⚠ WARNING

Property damage or bodily injury possible due to malfunctions!

For outputs in PWM mode:
- there are no diagnostic functions
- the overload protection OUT_OVERLOAD_PROTECTION is NOT active

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.

Availability of PWM

15371

Device	Number of available PWM outputs	of which current-controlled (PWMi)	PWM frequency [Hz]
SmartController: CR2530	12	2	20...250

FBs for PWM functions

14718

The following function blocks are available for the PWM function of the outputs:

CURRENT_CONTROL (→ page 182)	Current controller for a PWMi output channel
PWM1000 (→ page 186)	Initialises and configures a PWM-capable output channel the mark-to-space ratio can be indicated in steps of 1 ‰

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.

4.5 Variables

Contents

3130

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

4.5.1 Retain variables

8672

Retain variables can be saved automatically in a protected memory area and be reloaded automatically during a reboot.

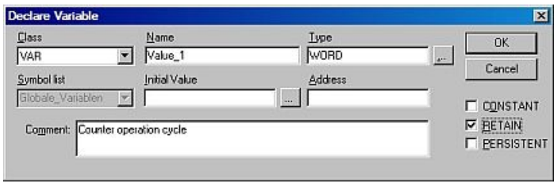
14166

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).



Save retain variables

9853

In the device the data type RETAIN is only stored in the volatile memory (RAM) during the runtime. To save the data permanently, at the end of each cycle they are automatically be saved in the FRAM memory [1]).

[1]) FRAM indicates here all kinds of non-volatile and fast memories.

> **NOTE**
>
> In this device, do NOT use the following functions from the 3S library `SysLibPlcCtrl.lib`:
> - FUN SysSaveRetains
> - FUN SysRestoreRetains

Read back retain variables

9854

After power on and before the first program cycle the device automatically writes the saved data back to the working memory once. To do so, no additional FBs must be integrated into the application program.

> **NOTE**
>
> In this device, do NOT use the following functions from the 3S library `SysLibPlcCtrl.lib`:
> - FUN SysSaveRetains
> - FUN SysRestoreRetains

4.5.2 Network variables

15242
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_NetVarLib_NT_Vxxyyzz.lib`

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 CR2530

Contents

14235

5.1.1 Bibliothek ifm_CR2530_V03yyzz.LIB

14736

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

Function element	Short description
FASTCOUNT (→ page 172)	Counter block for fast input pulses
FLASH_READ (→ page 190)	transfers different data types directly from the flash memory to the RAM
GET_APP_INFO (→ page 191)	Delivers information about the application program stored in the device: • name of the application, • version of the application, • unique CODESYS build number, • CODESYS build date
GET_HW_INFO (→ page 192)	Delivers information about the device hardware: • ifm article number (e.g. CR0403), • article designation, • unambiguous serial number, • hardware revision, • production date
GET_IDENTITY (→ page 193)	Reads the identification of the application stored in the device (has previously been saved by means of **SET_IDENTITY** (→ page 201))
GET_SW_INFO (→ page 194)	Delivers information about the system software of the device: • software name, • software version, • build number, • build date
GET_SW_VERSION (→ page 195)	Delivers information about the software versions stored in the device: • BasicSystem version, • bootloader version, • SIS version, • application program version, • user flash version
INC_ENCODER (→ page 174)	Up/down counter function for the evaluation of encoders
INPUT (→ page 176)	Assigns an operating mode to an input channel Provides the current state of the selected channel
MEM_ERROR (→ page 196)	Signals errors in some parameters or in the memory (Re-)initialisation of system resources
MEMCPY (→ page 197)	Writes and reads different data types directly in the memory
OHC (→ page 199)	Adjustable operating hours counter (0...3)
OUTPUT (→ page 184)	Assigns an operating mode to an output channel Provides the current state of the selected channel
PERIOD (→ page 179)	Measures the frequency and the cycle period (cycle time) in [µs] at the indicated channel
PWM1000 (→ page 186)	Initialises and configures a PWM-capable output channel the mark-to-space ratio can be indicated in steps of 1 ‰
SET_IDENTITY (→ page 201)	Sets an application-specific program identification
SET_LED (→ page 202)	Change the frequency and color of the status LED in the application program
SET_PASSWORD (→ page 204)	Sets a user password for access control to program and memory upload
TIMER_READ_US (→ page 205)	Reads out the current system time in [µs] Max. value = 1h 11min 34s 967ms 295µs

5.1.2 Library ifm_RAWCan_NT_Vxxyyzz.LIB

14715

This **ifm** library contains the following function blocks:

Function element	Short description
CAN_ENABLE (→ page 69)	Initialises the indicated CAN interface Configures the CAN baud rate
CAN_RECOVER (→ page 70)	Activate / deactivate the automatic bus off handling Restart the CAN interface in case of bus off
CAN_REMOTE_REQUEST (→ page 91)	Send a corresponding request and return the response of the other device as a result
CAN_REMOTE_RESPONSE (→ page 92)	Provides data to the CAN controller in the device which is automatically sent as a response to the request of a remote message
CAN_RX (→ page 75)	Configures a data receive object and reads out the receive buffer of the data object
CAN_RX_ENH (→ page 76)	• Configures a data receive object and reads out the receive buffer of the data object • Frame type and mask can be selected
CAN_RX_ENH_FIFO (→ page 78)	• Configures a data receive object and reads out the receive buffer of the data object • Frame type and mask can be selected • Several CAN messages per cycle possible
CAN_RX_RANGE (→ page 80)	• Configures a range of data receive objects and reads out the receive buffer of the data objects • Frame type and mask can be selected
CAN_RX_RANGE_FIFO (→ page 82)	• Configures a range of data receive objects and reads out the receive buffer of the data objects • Frame type and mask can be selected • Several CAN messages per cycle possible
CAN_SETDOWNLOADID (→ page 71)	= Set CAN download ID Sets the download identifier for the CAN interface
CAN_STATUS (→ page 72)	Get status information on the CAN bus selected: BAUDRATE, DOWNLOAD_ID, BUSOFF, WARNING_RX, WARNING_TX, VERSION, BUSLOAD and reset if required: BUSOFF, WARNING_RX, WARNING_TX
CAN_TX (→ page 85)	Transfers a CAN data object (message) to the configured CAN interface for transmission at each call
CAN_TX_ENH (→ page 86)	Transfers a CAN data object (message) to the configured CAN interface for transmission at each call CAN-specific characteristics can be set
CAN_TX_ENH_CYCLIC (→ page 88)	Cyclically transfers a CAN data object (message) to the configured CAN interface for transmission CAN-specific characteristics can be set

5.1.3 Library ifm_CANopen_NT_Vxxyyzz.LIB

14914

This **ifm** library contains the following function blocks:

Function element	Short description
CANOPEN_ENABLE (→ page 95)	Initialises the indicated CANopen master interface Configures the CAN baud rate
CANOPEN_GETBUFFERFLAGS (→ page 97)	= CANopen get buffer flags Provides information on the buffer flags The flags can be reset via the optional inputs.
CANOPEN_GETEMCYMESSAGES (→ page 134)	= Get CANopen emergency messages Lists all emergency messages that have been received by the controller from other nodes in the network since the last deletion of messages The list can be reset by setting the according input.
CANOPEN_GETERRORREGISTER (→ page 136)	= Get CANopen error register Reads the error registers 0x1001 and 0x1003 from the controller The registers can be reset by setting the respective inputs.
CANOPEN_GETGUARDHBERRLIST (→ page 130)	= get CANopen guard and heartbeat error list Lists all nodes in an array for which the master has detected an error: guarding error, heartbeat error The list can be reset by setting the according input.
CANOPEN_GETGUARDHBSTATSLV (→ page 131)	= CANopen slave get guard and heartbeat state Signals the following states to the controller in slave operation: node guarding monitoring, heartbeat monitoring The signalled errors can be reset by setting the respective input.
CANOPEN_GETNMTSTATESLAVE (→ page 104)	= CANopen slave get network management state Signals the network operating status of the node
CANOPEN_GETODCHANGEDFLAG (→ page 108)	= Get object directory changed flag Reports any change of value for a particular object directory entry
CANOPEN_GETSTATE (→ page 99)	= CANopen set state Request the parameters of the master, a slave device or a specific node in the network
CANOPEN_GETSYNCSTATE (→ page 126)	= CANopen get SYNC state • Reads the setting of the SYNC functionality (active / not active), • reads the error state of the SYNC functionality (SyncError)
CANOPEN_NMTSERVICES (→ page 105)	= CANopen network management services Updates the internal node status and, depending on the NMT command entries: • triggers an NMT command or • triggers the initialisation of a node
CANOPEN_READOBJECTDICT (→ page 109)	= CANopen read object directory Reads configuration data from the object directory of the device
CANOPEN_SDOREAD (→ page 113)	= CANopen read SDO Reads an "Expedited SDO" = Expedited Service Data Object
CANOPEN_SDOREADBLOCK (→ page 115)	= CANopen read SDO block Reads the indicated entry in the object directory of a node in the network via SDO block transfer
CANOPEN_SDOREADMULTI (→ page 117)	= CANopen read SDO multi Reads the indicated entry in the object directory of a node in the network
CANOPEN_SDOWRITE (→ page 119)	= SDO write Writes an "Expedited SDO" = Expedited Service Data Object
CANOPEN_SDOWRITEBLOCK (→ page 121)	= CANopen write SDO block Writes in the indicated entry in the object directory of a node in the network via SDO block transfer
CANOPEN_SDOWRITEMULTI (→ page 123)	= CANopen write SDO multi Writes in the indicated entry in the object directory of a node in the network
CANOPEN_SENDEMCYMESSAGE (→ page 137)	= CANopen send emergency message Sends an EMCY message. The message is assembled from the according parameters and entered in register 0x1003

Function element	Short description
CANOPEN_SETSTATE (→ page 101)	= CANopen set state Set the parameters of the master, a slave device or a specific node in the network
CANOPEN_SETSYNCSTATE (→ page 128)	= CANopen set SYNC state Switch the SYNC functionality on and off
CANOPEN_WRITEOBJECTDICT (→ page 110)	= CANopen write object directory Writes configuration data into the object directory of the device

5.1.4 Library ifm_J1939_NT_Vxxyyzz.LIB

14912

This **ifm** library contains the following function blocks:

Function element	Short description
J1939_DM1RX (→ page 164)	J1939 Diagnostic Message 1 RX Receives diagnostic messages DM1 or DM2 from other ECUs
J1939_DM1TX (→ page 166)	J1939 Diagnostic Message 1 TX Transmit an active error message to the CAN stack
J1939_DM1TX_CFG (→ page 169)	J1939 Diagnostic Message 1 TX configurable CAN stack does not send cyclic DM1 "zero active faults" messages
J1939_DM3TX (→ page 170)	J1939 Diagnostic Message 3 TX Deletes inactive DTCs (DM2) on a device
J1939_ENABLE (→ page 140)	Initialises the J1939 stack
J1939_GETDABYNAME (→ page 142)	= Get destination arbitrary name Determine the target address of one or several participants by means of the name information
J1939_NAME (→ page 144)	Give the device a name for identification in the network
J1939_RX (→ page 151)	Receives a single frame message Shows the message last read on the CAN bus
J1939_RX_FIFO (→ page 152)	= J1939 RX with FIFO Receives all specific messages and successively reads them from a FiFo
J1939_RX_MULTI (→ page 154)	= J1939 RX multiframe message Receives multiframe messages
J1939_SPEC_REQ (→ page 148)	= J1939 specific request Requests and receives a specific message from another controller
J1939_SPEC_REQ_MULTI (→ page 149)	= J1939 specific request multiframe message Requests and receives a specific multiframe message from another controller
J1939_STATUS (→ page 146)	Shows relevant information on the J1939 stack
J1939_TX (→ page 156)	Sends individual single frame messages
J1939_TX_ENH (→ page 157)	= J1939 TX enhanced Sends individual single frame messages Can also be set: transmission priority, data length
J1939_TX_ENH_CYCLIC (→ page 159)	= J1939 TX enhanced cyclic Cyclically sends single frame messages Can also be set: transmission priority, data length, period
J1939_TX_ENH_MULTI (→ page 161)	= J1939 TX enhanced Multiframe Message Sends individual multiframe messages

5.2 ifm function elements for the device CR2530

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 element outputs 83547556

Some function elements return a RESULT message.

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1...31		Global return values; examples:
1	01	FB execution completed without error – data is valid
4	04	FB is being processed – data is cyclically processed
5	05	FB is being processed – still receiving
6	06	FB is being processed – still sending
7	07	FB is being processed – remote for ID active
8	08	function block is active
14	0E	FB is active CANopen manager configures devices and sends SDOs
15	0F	FB is active CANopen manager is started
32_{10}...63		FB specific return values
64_{10}...127		FB specific error messages
128_{10}...255		Global error messages; examples:
238	EE	Error: CANopen configuration is too large and cannot be started
239	EF	Error: CANopen manager could not be started
240	F0	Error: several modal inputs are active e.g. CANopen NTM services
241	F1	Error: CANopen state transition is not permitted
242	F2	Error: setting is not possible
247	F7	Error: memory exceeded (length larger than array)
250	FA	Error: FiFo is full – data was lost
252	FC	Error: CAN multiframe transmission failed
253	FD	Error: CAN transmission failed. Data cannot be sent.
255	FF	Error: not enough memory available for the consuming multiframe

5.2.2 Function elements: RAW-CAN (Layer 2)

Contents

15051

Here we describe the RAW-CAN function blocks (CAN Layer 2) of **ifm electronic** to be used in the application program.

Function elements: RAW-CAN status

Contents

15049

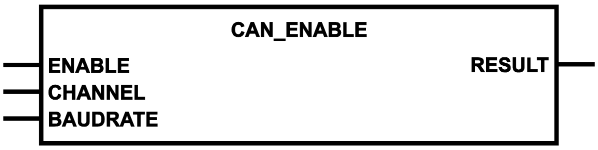
CAN_ENABLE

7492

Unit type = function block (FB)

Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7494

With CAN_ENABLE the CAN hardware is initialised. Without this call no other calls are possible in RAW-CAN or they return an error.

In order to change the baud rate the following procedure is required:

► Maintain the function block on ENABLE=FALSE for the duration of one cycle.
> All protocols are reset.
> Re-initialisation of the CAN interface and the CAN protocols running on it. Any information available for cyclical transmission is lost as well and must be newly created.
> At renewed ENABLE=TRUE, the new baud rate is adopted.

Parameters of the inputs

7495

Parameter	Data type	Description
ENABLE	BOOL := FALSE	TRUE: enable CAN interface FALSE: disable CAN interface
CHANNEL	BYTE	CAN interface (1...n) depending on the device
BAUDRATE	WORD := 250	Baudrate [kbits/s] Permissible = 20, 50, 100, 125, 250, 500, 1000

Parameters of the outputs

8530

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
8	08	function block is active
9	09	CAN is not active
242	F2	Error: setting is not possible

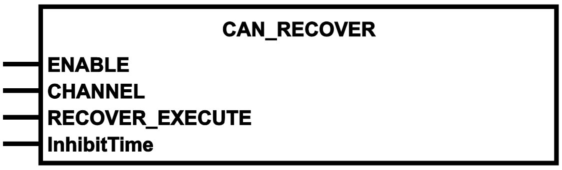
CAN_RECOVER

7512

Unit type = function block (FB)

Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7513

CAN_RECOVER has the following tasks:

- to activate / deactivate the automatic bus off handling
- to restart the CAN interface in case of bus off

> In case of bus off: CAN Controller deletes all buffers (including the buffers of the other protocols).

If CAN_RECOVER is not used (ENABLE=FALSE):

> in case of a bus off a recovery attempt is automatically made after 1 s.

> after 4 failed recovery attempts in a row the affected CAN interface is deactivated.

Parameters of the inputs

7514

Parameter	Data type	Description
ENABLE	BOOL := FALSE	TRUE: No automatic recovery after CAN bus off FALSE: Automatic recovery after CAN bus off
CHANNEL	BYTE	CAN interface (1...n) depending on the device
RECOVER_EXECUTE	BOOL	TRUE (only for 1 cycle): restart of CAN interface remedy bus off condition FALSE: function element is not executed
InhibitTime (optional use of the parameter)	TIME := T#1s	Waiting time between bus off and restart of the CAN interface

CAN_SETDOWNLOADID

7516

= Set download ID

Unit type = function block (FB)

Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

7517

The download ID is required for data exchange when connecting the runtime system and the CODESYS development environment. When the device is started the download ID is set with the default value from the hardware configuration.

With CAN_SETDOWNLOADID this value can be set in the PLC program (e.g. using certain inputs). The changed ID is also written into the hardware configuration.

Parameters of the inputs

7519

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
DOWNLOAD_ID	BYTE	1...127 = set download ID 0 = read download ID

Parameters of the outputs

7520

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	function block is active
242	F2	Error: setting is not possible

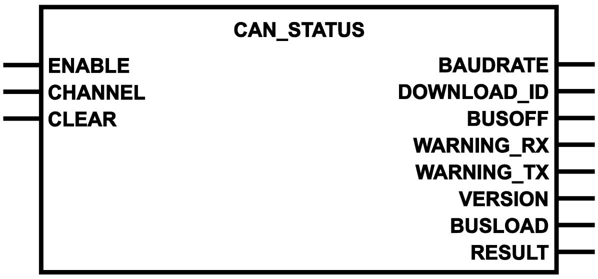
CAN_STATUS

7499

Unit type = function block (FB)

Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7501

CAN_STATUS provides information on the chosen CAN bus.

Without hardware initialisation the following flags can be reset to FALSE:
- BUSOFF
- WARNING_RX
- WARNING_TX

Parameters of the inputs

7502

Parameter	Data type	Description
ENABLE	BOOL := FALSE	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
CHANNEL	BYTE	CAN interface (1...n) depending on the device
CLEAR	BOOL := FALSE	TRUE: Reset the following flags: • WARNING_RX • WARNING_TX • BUSOFF FALSE: function element is not executed

Parameters of the outputs

7504

Parameter	Data type	Description
BAUDRATE	WORD	current baudrate of the CANopen node in [kBaud]
DOWNLOAD_ID	BYTE	current download ID
BUSOFF	BOOL	Error CAN BUS OFF at the interface
WARNING_RX	BOOL	Warning threshold for receiving is exceeded at the interface
WARNING_TX	BOOL	Warning threshold for transmitting is exceeded at the interface
VERSION	DWORD	Version of the ifm CAN stack library
BUSLOAD	BYTE	Current bus load in [%]
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	function block is active
9	09	CAN is not active
242	F2	Error: setting is not possible

Function elements: receive RAW-CAN data

Contents

15050

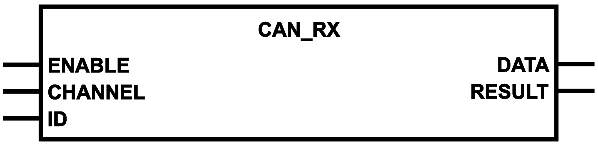
CAN_RX

7586

Unit type = function block (FB)

Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7588

CAN_RX is used for receiving a message.
The FB limits itself to a few functions and the required memory space is low.

CAN_RX filters for the set identifier. If several CAN messages with the same identifier are received in one cycle, only the last / latest message is available.

Parameters of the inputs

7589

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
ID	DWORD	Number of the data object identifier: normal frame (2 048 IDs): 0...2 047 = 0x0000 0000...0x0000 07FF extended Frame (536 868 864 IDs): 2 048...536 870 911 = 0x0000 0800...0x1FFF FFFF

Parameters of the outputs

7590

Parameter	Data type	Description
DATA	ARRAY [0..7] OF BYTE	received data, (1...8 bytes)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

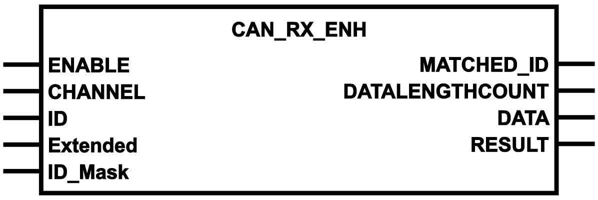
Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
5	05	FB is being processed – still receiving
9	09	CAN is not active
242	F2	Error: setting is not possible

CAN_RX_ENH

7606

= CAN RX enhanced
Unit type = function block (FB)
Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7608

In addition, CAN_RX_ENH provides the following possibilities (as opposed to **CAN_RX** (→ page 75)):
• select the frame type (11 or 29 bits),
• define a mask for the evaluation of the CAN ID.

Bit comparison of ID and mask:	If ID_MASK-Bit = 0, then CAN-ID-Bit may be = 0 or 1. If ID_MASK-Bit = 1, then CAN-ID-Bit must be = ID-Bit.

With the mask several identifiers can be defined as filters.

Example:

ID =	0x100 = 0b0001 0000 0000
ID_MASK =	0x1F1 = 0b0001 1111 0001
Result	The CAN IDs with the following bit pattern are evaluated: 0bxxx1 0000 xxx0 (x = any), i.e. for this example (all in [hex]): 100, 102, 104, 106, 108, 10A, 10C, 10E, 300, 302, 304, 306, 308, 30A, 30C, 30E, 500, 502, 504, 506, 508, 50A, 50C, 50E, 700, 702, 704, 706, 708, 70A, 70C, 70E

Parameters of the inputs

7609

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
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
Extended (optional use of the parameter)	BOOL := FALSE	TRUE: extended Frame (ID = 0...2^{29}-1) FALSE: normal Frame (ID = 0...2^{11}-1)
ID_Mask (optional use of the parameter)	DWORD := 0	filter mask for the identifier: if ID_MASK bit = 0, CAN ID bit may be = 0 or 1 if ID_MASK bit = 1, CAN ID bit must be = ID bit

Parameters of the outputs

7613

Parameter	Data type	Description
MATCHED_ID	DWORD	number of the data object identifier
DATALENGTHCOUNT	BYTE	= Data Length Count number of the data bytes received
DATA	ARRAY [0..7] OF BYTE	received data, (1...8 bytes)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
5	05	FB is being processed – still receiving
9	09	CAN is not active
242	F2	Error: setting is not possible

CAN_RX_ENH_FIFO

7615

= CAN RX enhanced with FiFo

Unit type = function block (FB)

Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:

```
CAN_RX_ENH_FIFO
ENABLE                MATCHED_ID
CHANNEL          DATALENGTHCOUNT
ID                          DATA
Extended     MORE_DATA_AVAILABLE
ID_Mask                   RESULT
```

Description

7616

In addition, CAN_RX_ENH_FIFO provides a FiFo for the received data (as opposed to **CAN_RX_ENH** (→ page 76)). Thus several CAN messages can be received in one cycle.

> No overwriting takes place when the FiFo is full. Inbound messages will be lost.

In this event:

► Deactivate and reactive the FB via ENABLE.

> The FiFo is deleted and can be newly filled.

Description to the filter mask: → **CAN_RX_ENH** (→ page 76) > chapter **Description**

Parameters of the inputs

7609

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
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
Extended (optional use of the parameter)	BOOL := FALSE	TRUE: extended Frame (ID = $0...2^{29}$-1) FALSE: normal Frame (ID = $0...2^{11}$-1)
ID_Mask (optional use of the parameter)	DWORD := 0	filter mask for the identifier: if ID_MASK bit = 0, CAN ID bit may be = 0 or 1 if ID_MASK bit = 1, CAN ID bit must be = ID bit

Parameters of the outputs

7617

Parameter	Data type	Description
MATCHED_ID	DWORD	number of the data object identifier
DATALENGTHCOUNT	BYTE	= Data Length Count number of the data bytes received
DATA	ARRAY [0..7] OF BYTE	received data, (1...8 bytes)
MORE_DATA_AVAILABLE	BOOL	TRUE: further received data available in the FiFo FALSE: no further data available in the FiFo
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
5	05	FB is being processed – still receiving
9	09	CAN is not active
242	F2	Error: setting is not possible
250	FA	Error: FiFo is full – data was lost

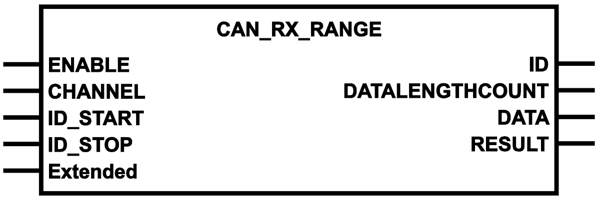
CAN_RX_RANGE

7592

Unit type = function block (FB)

Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7594

CAN_RX_RANGE provides the following settings:

- select the message type (11 or 29 bits),
- define an identifier range.

CAN_RX filters for the set identifier. If several CAN messages with the same identifier are received in one cycle, only the last / latest message is available.

Parameters of the inputs

7595

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
ID_START	DWORD	start number of the data object identifier range: normal frame (2^{11}): 0...2 047 = 0x0000 0000...0x0000 07FF extended frame (2^{29}): 0...536 870 911 = 0x0000 0000...0x1FFF FFFF
ID_STOP	DWORD	end number of the data object identifier range: normal frame (2^{11}): 0...2 047 = 0x0000 0000...0x0000 07FF extended frame (2^{29}): 0...536 870 911 = 0x0000 0000...0x1FFF FFFF
Extended (optional use of the parameter)	BOOL := FALSE	TRUE: extended Frame (ID = 0...2^{29}-1) FALSE: normal Frame (ID = 0...2^{11}-1)

Parameters of the outputs

7598

Parameter	Data type	Description
ID	DWORD	Number of the data object identifier: normal frame (2 048 IDs): 0...2 047 = 0x0000 0000...0x0000 07FF extended Frame (536 868 864 IDs): 2 048...536 870 911 = 0x0000 0800...0x1FFF FFFF
DATALENGTHCOUNT	BYTE	= Data Length Count number of the data bytes received
DATA	ARRAY [0..7] OF BYTE	received data, (1...8 bytes)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
5	05	FB is being processed – still receiving
9	09	CAN is not active
242	F2	Error: setting is not possible

CAN_RX_RANGE_FIFO

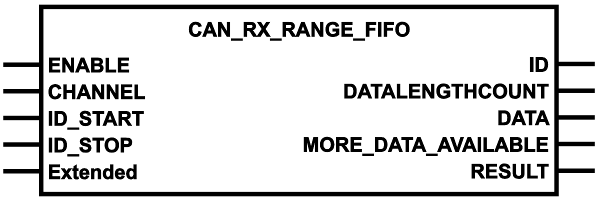
7601

= CAN RX range with FiFo

Unit type = function block (FB)

Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7603

CAN_RX_RANGE_FIFO basically works like **CAN_RX_RANGE** (→ page 80).

In addition, CAN_RX_RANGE_FIFO provides a FiFo for the received data. Thus several CAN messages can be received in one cycle.

(!) No overwriting takes place when the FiFo is full. Inbound messages will be lost.

In this event:

► Use ENABLE to deactivate and reactivate the function.

\> The FiFo is deleted and can be newly filled.

Parameters of the inputs

7595

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
ID_START	DWORD	start number of the data object identifier range: normal frame (2^{11}): 0...2 047 = 0x0000 0000...0x0000 07FF extended frame (2^{29}): 0...536 870 911 = 0x0000 0000...0x1FFF FFFF
ID_STOP	DWORD	end number of the data object identifier range: normal frame (2^{11}): 0...2 047 = 0x0000 0000...0x0000 07FF extended frame (2^{29}): 0...536 870 911 = 0x0000 0000...0x1FFF FFFF
Extended (optional use of the parameter)	BOOL := FALSE	TRUE: extended Frame (ID = 0...2^{29}-1) FALSE: normal Frame (ID = 0...2^{11}-1)

Parameters of the outputs

7604

Parameter	Data type	Description
ID	DWORD	Number of the data object identifier: normal frame (2 048 IDs): 0...2 047 = 0x0000 0000...0x0000 07FF extended Frame (536 868 864 IDs): 2 048...536 870 911 = 0x0000 0800...0x1FFF FFFF
DATALENGTHCOUNT	BYTE	= Data Length Count number of the data bytes received
DATA	ARRAY [0..7] OF BYTE	received data, (1...8 bytes)
MORE_DATA_AVAILABLE	BOOL	TRUE: further received data available in the FiFo FALSE: no further data available in the FiFo
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
5	05	FB is being processed – still receiving
9	09	CAN is not active
242	F2	Error: setting is not possible
250	FA	Error: FiFo is full – data was lost

Function elements: transmit RAW-CAN data

Contents

15055

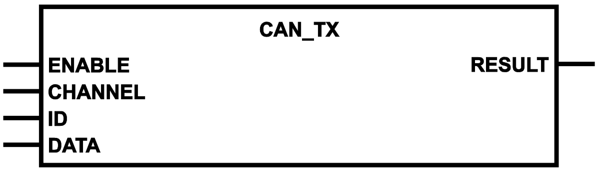
CAN_TX

7522

Unit type = function block (FB)

Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7523

CAN_TX sends a standard message per cycle.
The FB limits itself to a few functions and the required memory space is low.

> If an instance of this FB is called several times during a cycle, the data is also sent several times.

In case of the simple functions CAN_TX and CAN_RX, it is determined by means of the ID whether a standard or an extended frame is to be sent. With the enhanced versions this is set via the input EXTENDED. Therefore, extended frames in the ID area 0...2047 cannot be sent via the easy functions.

Parameters of the inputs

7524

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
ID	DWORD	Number of the data object identifier: normal frame (2 048 IDs): 0...2 047 = 0x0000 0000...0x0000 07FF extended Frame (536 868 864 IDs): 2 048...536 870 911 = 0x0000 0800...0x1FFF FFFF
DATA	ARRAY [0..7] OF BYTE	data to be sent (1...8 bytes)

Parameters of the outputs

7527

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
242	F2	Error: setting is not possible
250	FA	Error: FiFo is full – data was lost

CAN_TX_ENH

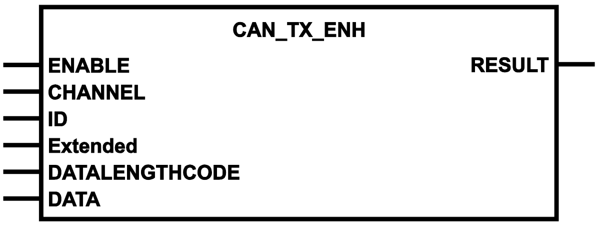
7558

= CAN TX enhanced

Unit type = function block (FB)

Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7559

Additional setting options are offered through CAN_TX_ENH (for: enhanced). Here, all CAN specific characteristics can be set individually, e.g.:

- Is it an 11 or a 29 bit identifier?
- The additional inputs can be preset so that **CAN_TX** (→ page 85) is not required.

\> If an instance of this FB is called several times during a cycle, the data is also sent several times.

Parameters of the inputs

7634

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
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
Extended (optional use of the parameter)	BOOL := FALSE	TRUE: extended Frame (ID = 0...2^{29}-1) FALSE: normal Frame (ID = 0...2^{11}-1)
DATALENGTHCODE	BYTE	= Data Length Code number of the data bytes to be sent (0...8)
DATA	ARRAY [0..7] OF BYTE	data to be sent (1...8 bytes)

Parameters of the outputs

7527

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

Possible results for RESULT:

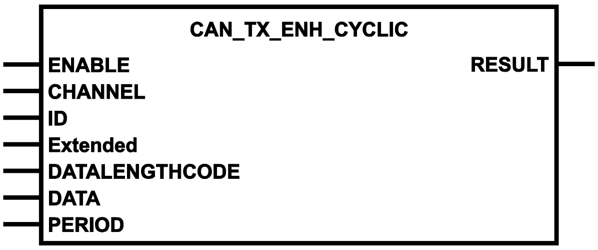
Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
242	F2	Error: setting is not possible
250	FA	Error: FiFo is full – data was lost

CAN_TX_ENH_CYCLIC

7568

= CAN TX enhanced cyclic
Unit type = function block (FB)
Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7569

CAN_TX_ENH_CYCLIC serves for cyclical transmitting of CAN messages.
Otherwise, the FB corresponds to **CAN_TX_ENH** (→ page 86).

► Set the period duration via the parameter PERIOD.

> ❗ If a period is too short, this could lead to a high bus load which could affect the performance of the complete system.

Parameters of the inputs

7582

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
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
Extended (optional use of the parameter)	BOOL := FALSE	TRUE: extended Frame (ID = 0...2^{29}-1) FALSE: normal Frame (ID = 0...2^{11}-1)
DataLengthCode (optional use of the parameter)	BYTE := 8	length of the data to be sent (0...8 bytes)
DATA	ARRAY [0..7] OF BYTE	data to be sent (1...8 bytes)
PERIOD	TIME	period duration

Parameters of the outputs

7510

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
8	08	function block is active
9	09	CAN is not active
250	FA	Error: FiFo is full – data was lost

Function elements: RAW-CAN remote

Contents

15057

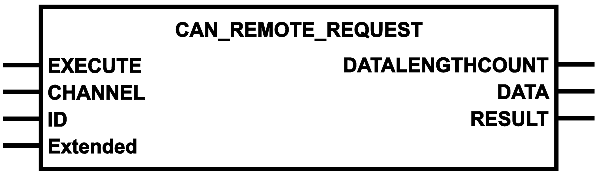
CAN_REMOTE_REQUEST

7625

Unit type = function block (FB)

Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7627

In order to request a remote message, an according requirement is dispatched via CAN_REMOTE_REQUEST and the response of the other device is sent back as result.

Parameters of the inputs

7628

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
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
Extended (optional use of the parameter)	BOOL := FALSE	TRUE: extended Frame (ID = 0...2^{29}-1) FALSE: normal Frame (ID = 0...2^{11}-1)

Parameters of the outputs

7629

Parameter	Data type	Description
DATALENGTHCOUNT	BYTE	= Data Length Count number of the data bytes received
DATA	ARRAY [0..7] OF BYTE	received data, (1...8 bytes)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
5	05	FB is being processed – still receiving
9	09	CAN is not active
242	F2	Error: setting is not possible

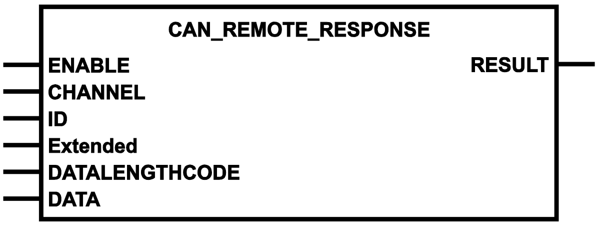
CAN_REMOTE_RESPONSE

7631

Unit type = function block (FB)

Unit is contained in the library `ifm_RawCAN_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7633

CAN_REMOTE_RESPONSE provides data to the CAN controller in the device which is automatically sent upon the request of a remote message.

This FB strongly depends on the device type. Only a limited number of remote messages can be set up:

BasicController: CR040n, CR041n, CR043n BasicDisplay: CR045n	max. 40 remote messages
PDM360 NG: CR108n, CR120n	max. 100 remote messages

Parameters of the inputs

7634

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
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
Extended (optional use of the parameter)	BOOL := FALSE	TRUE: extended Frame (ID = 0...2^{29}-1) FALSE: normal Frame (ID = 0...2^{11}-1)
DATALENGTHCODE	BYTE	= Data Length Code number of the data bytes to be sent (0...8)
DATA	ARRAY [0..7] OF BYTE	data to be sent (1...8 bytes)

Parameters of the outputs

7636

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
6	06	FB is being processed – remote for ID not active
7	07	FB is being processed – remote for ID active

5.2.3 Function elements: CANopen

Contents

15059

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

Function elements: CANopen status

Contents

15061

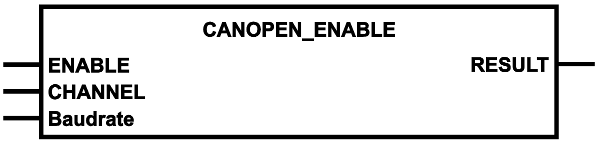
CANOPEN_ENABLE

7785

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7787

CANOPEN_ENABLE allows to switch the CANopen master on or off.

► ⓘ In the application program always call an own instance of the FB **CANOPEN_ENABLE** (→ page 95) for every CAN interface!

ⓘ To avoid guarding or heartbeat errors the nodes must be "shut down" via an appropriate sequence first.

If the master is restarted after a stop, all other connected nodes also have to be re-initialised.

Without CANOPEN_ENABLE, the CANopen master is started automatically, as far as this has been selected in the configuration.

The configured baud rate is only adopted if **CAN_ENABLE** (→ page 69) has not been activated before.

Parameters of the inputs

7788

Parameter	Data type	Description
ENABLE	BOOL := TRUE	TRUE: • Enable CANopen for the selected channel • Start CANopen manager or CANopen device according to the configuration settings FALSE: • Disable CANopen for the selected channel • Terminate CANopen manager or CANopen device
CHANNEL	BYTE	CAN interface (1...n) depending on the device
Baud rate (optional use of the parameter)	WORD := 0	Baud rate [kbits/s] permissible values = 20, 50, 100, 125, 250, 500, 800, 1 000 0 = use setting from the PLC configuration

Parameters of the outputs

7789

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
14	0E	FB is active CANopen manager configures devices and sends SDOs
15	0F	FB is active CANopen manager is started
238	EE	Error: CANopen configuration is too large and cannot be started
239	EF	Error: CANopen manager could not be started
242	F2	Error: setting is not possible

CANOPEN_GETBUFFERFLAGS

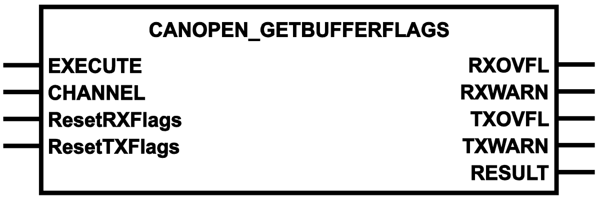
7890

= Get buffer flags

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7892

CANOPEN_GETBUFFERFLAGS supplies information on the buffer flags.
The flags can be reset via the optional inputs.
The function block returns the state of the overflow flags.

Parameters of the inputs

7893

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
ResetRXFlags (optional use of the parameter)	BOOL := FALSE	TRUE: Provide fag status at the output and then reset FALSE: function element is not executed
ResetTXFlags (optional use of the parameter)	BOOL := FALSE	TRUE: Provide flag status at the output and then reset FALSE: function element is not executed

Parameters of the outputs

7894

Parameter	Data type	Description
RXOVFL	BOOL	condition of the RX overflow flag TRUE: overflow in the receive buffer FALSE: no overflow in receive buffer
RXWARN	BOOL	condition of the RX overflow warning flag TRUE: level in the receive buffer is critical FALSE: level in the input buffer is uncritical
TXOVFL	BOOL	condition of the TX overflow flag TRUE: overflow in the transmit buffer FALSE: no overflow in transmit buffer
TXWARN	BOOL	Condition of the TX overflow warning flag TRUE: Level in the transmit buffer is critical FALSE: Level in the transmit buffer is uncritical
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	function block not yet executed
242	F2	Error: setting is not possible

CANOPEN_GETSTATE

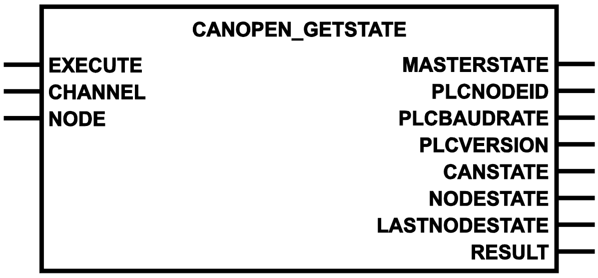
7865

= Get state

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7867

Via CANOPEN_GETSTATE, parameters of the master, a slave device or a specific node in the network can be set.

Parameters of the inputs

7868

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active. A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
NODE	BYTE	Node ID = ID of the node (0...127) **Device as CANopen master:** Value = 0: Only the status information of the device itself is returned at the outputs. The outputs with information on the nodes are invalid. Value not 0: Node ID of a node in the network. For this one as well as for the device the states are returned at the outputs. **Device as CANopen slave:** Value = 0 (preset): The status information of the slave is returned at the outputs. Value not 0: no action

Parameters of the outputs

7869

Parameter	Data type	Description
MASTERSTATE	BYTE	Master state = internal state of the master: 0 = 0x00 = master starts up 4 = 0x04 = node configuration running 5 = 0x05 = normal operating state of the master 255 = 0xFF = PLC running as slave
PLCNODEID	BYTE	PLC node ID = node ID of the PLC the program is running on Value = 0...127 = 0x00...0x7F
PLCBAUDRATE	DWORD	Baudrate of the PLC
PLCVERSION	DWORD	PLC version
CANSTATE	BYTE	Status of the CANopen network **Device operated as master:** **Node ID = 0 (device as such):** 0 = 0x00 = OK 128 = 0x80 = BUSOFF **Node ID ≠ 0 (node):** 0 = 0x00 = OK 1 = 0x01 = guard or heartbeat error on node 128 = 0x80 = BUSOFF **Device operated as slave:** 0 = 0x00 = OK 1 = 0x01 = guard or heartbeat error 128 = 0x80 = BUSOFF
NODESTATE	BYTE	Node state = internal node state of a slave seen from the master's perspective. The input NODEID identifies the node. -1 = 0xFF = reset after ResetNode 1 = 0x01 = waiting for BOOTUP 2 = 0x02 = after receipt of the BOOTUP message 3 = 0x03 = not yet configured: STOPPED 4 = 0x04 = after configuration with SDOs: PRE-OPERATIONAL 5 = 0x05 = after starting the node: OPERATIONAL 97 = 0x61 = optional node 98 = 0x62 = other device type configured than in 0x1000 99 = 0x63 = node guarding
LASTNODESTATE	BYTE	Last Node State Node state according to CANopen (with these values the status is also coded in the corresponding messages with regard to the node). 0 0x00 BOOTUP 4 0x04 STOPPED 5 0x05 OPERATIONAL 127 0x7F PRE-OPERATIONAL
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
8	08	FB is active – not yet processed
242	F2	Error: setting is not possible

CANOPEN_SETSTATE

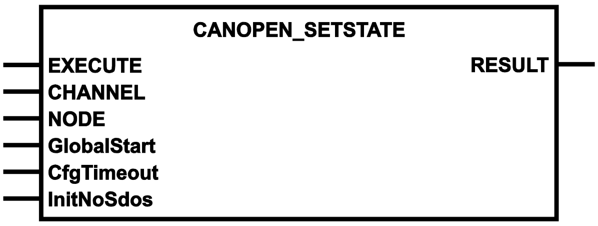
7858

= Set state

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7860

Via CANOPEN_SETSTATE, parameters of the master, a slave device or a node in the network can be set.

The treatment of the NMT state of master, node or device is carried out in the CAN stack or via the commands of the FB **CANOPEN_NMTSERVICES** (→ page 105). At the same time admissibility checks are carried out. For reasons of consistency no inputs are provided for this purpose.

Parameters of the inputs

7861

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
NODE	BYTE	Node ID = ID of the node (0...127) **Device as CANopen master:** Value = 0: The changes only refer to the device itself. Value not 0: Node ID of a node in the network the parameters of which are to be changed. The established settings are only adopted for this node (not for the device). **Device as CANopen slave:** In slave mode, the node ID of the slave can be set via this input. Value = 0: no action Value not 0: The function block adopts this value as the new node ID of the device.
GlobalStart (optional use of the parameter)	BOOL := TRUE	Requirement: FB must be called immediately after starting the IEC program. This setting overwrites the setting of the configuration. TRUE: Start all participants simultaneously FALSE: Start all participants one after the other
CfgTimeout (optional use of the parameter)	TIME := T#0ms	set configuration timeout for a node: Value = 0: no action – retain configuration data Value not 0: overwrite data from the configuration with the new value
InitNoSdos (optional use of the parameter)	BOOL := FALSE	To the node indicated in NODE, during initialisation,... TRUE: do not send configuration data FALSE: send configured SDOs

Parameters of the outputs

7862

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
8	08	FB is active – not yet processed
242	F2	Error: setting is not possible

Function elements: CANopen network management

Contents

15063

CANOPEN_GETNMTSTATESLAVE

7851

= Get network management state slave

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

7853

► Only use the FB if the device is operated as CANopen slave!

With CANOPEN_GETNMTSTATESLAVE, only the operating state according to CANopen and an error message are reported to the application if an invalid state transition has been requested.

Parameters of the inputs

7854

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device

Parameters of the outputs

7855

Parameter	Data type	Description
NMTSTATE	BYTE	Network operating status of the node 0 = INIT 1 = OPERATIONAL 2 = PRE-OPERATIONAL 3 = STOPPED
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	FB is active – not yet processed
242	F2	Error: setting is not possible

CANOPEN_NMTSERVICES

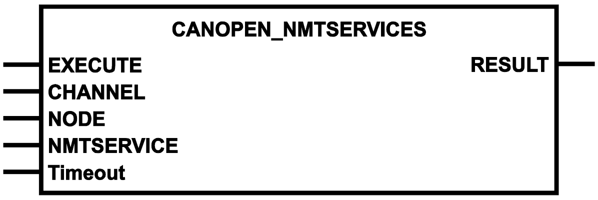
7843

= Network management services

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7844

Depending on its NMT command entries, CANOPEN_NMTSERVICES either triggers an NMT command or the initialisation of a node.

NMT = **N**etwork-**M**anagemen**T**

The function block updates the internal node status. If a state transition to CANopen (→ system manual "Know-How ecomat*mobile*" > **NMT state**) should not be permitted, the command is not executed.

A CANopen device can automatically change its CANopen state by means of the FB:
preoperational ⇦⇨ operational

Parameters of the inputs

7847

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
NODE	BYTE	CANopen ID of the node permissible = 0...127 = 0x00...0x7F NODE = 0: command applies to all nodes in the network NODE = Node ID of the device: command applies to the device as such
NMTSERVICE	BYTE	network command 0 = init node (except master) 1 = enter PRE-OPERATIONAL 2 = start node 3 = reset node 4 = reset communication 5 = stop node
Timeout (optional use of the parameter)	TIME := T#0ms	waiting time of the FB for the initialisation when the time has elapsed, the FB stops waiting. 0 = use value from the configuration

Parameters of the outputs

7848

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	function block execution completed without error
8	08	function block is active
35	23	at least 1 SDO of the configuration was not successful
36	24	node was already initialised
37	25	when initialisation was requested the node was not in the PRE-OPERATIONAL mode
43	2B	master / slave is not initialised
241	F1	Error: CANopen state transition is not permitted
242	F2	Error: setting is not possible

Function elements: CANopen object directory

Contents

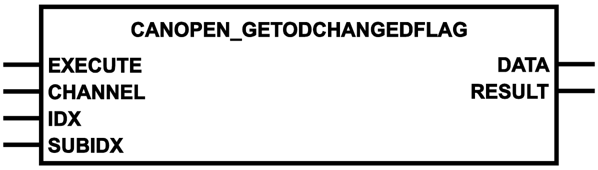
15065

CANOPEN_GETODCHANGEDFLAG

7927

= Get object directory changed flag
Unit type = function block (FB)
Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7928

CANOPEN_GETODCHANGEDFLAG reports any change of value for a particular object directory entry.

Parameters of the inputs

7930

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
IDX	WORD	index in object directory
SUBIDX	BYTE	sub-index referred to the index in the object directory

Parameters of the outputs

7931

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	FB is active – not yet processed
242	F2	Error: setting is not possible

CANOPEN_READOBJECTDICT

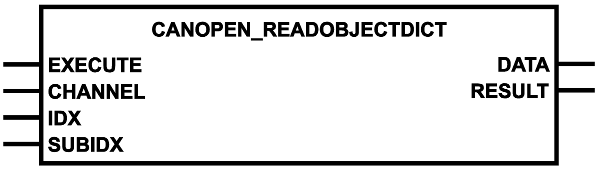
7933

= Read object directory

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7935

CANOPEN_READOBJECTDICT reads up to 4 bytes of configuration data from the object directory of the device for use in the application program.

Parameters of the inputs

7936

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
IDX	WORD	index in object directory
SUBIDX	BYTE	sub-index referred to the index in the object directory

Parameters of the outputs

7937

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	function block not yet executed
40	28	object directory entry is invalid
242	F2	Error: setting is not possible

CANOPEN_WRITEOBJECTDICT

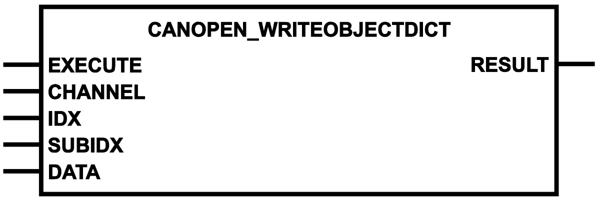
7940

= Write object directory

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7942

CANOPEN_WRITEOBJECTDICT writes configuration data to the object directory of the controller.

NOTICE

This could lead to falsification of important system settings, e.g.:
- guarding times
- heartbeat times

► Carefully verify input parameters!

Parameters of the inputs

7943

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
IDX	WORD	index in object directory
SUBIDX	BYTE	sub-index referred to the index in the object directory
DATA	DWORD	parameter value

Parameters of the outputs

7945

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	function block not yet executed
40	28	object directory entry is invalid
242	F2	Error: setting is not possible

Function elements: CANopen SDOs

Contents

2071

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

CANOPEN_SDOREAD

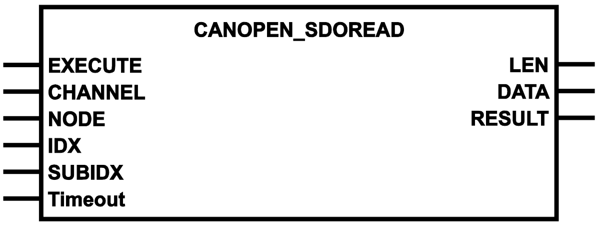
7791

= SDO read

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7793

CANOPEN_SDOREAD is an easy function block for editing "Expedited SDOs", i.e. SDOs with max. 4 bytes of user data. This type usually represents the bigger part of the SDO communication.

Expedited SDO = Expedited Service Data Object

A considerable amount of memory space can be saved due to the limitation of the data volume to max. 4 bytes of user data, as this FB only needs to reserve 4 bytes as buffer storage and does not create a large data array itself.

Parameters of the inputs

7794

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
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
Timeout (optional use of the parameter)	TIME := T#10ms	waiting time of the FB for the response when the time has elapsed, the FB stops waiting. value = 0: use value from the configuration

Parameters of the outputs

7795

Parameter	Data type	Description
LEN	BYTE	number of the bytes received (1...4)
DATA	DWORD	the received data value (up to 4 bytes)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
5	05	FB is active – no data received yet
32	20	SDO transmission aborted by client or server (SDO abort code 0x80)
33	21	TIMEOUT elapsed
242	F2	Error: setting is not possible
255	FF	buffer overflow – too many data bytes were received

CANOPEN_SDOREADBLOCK

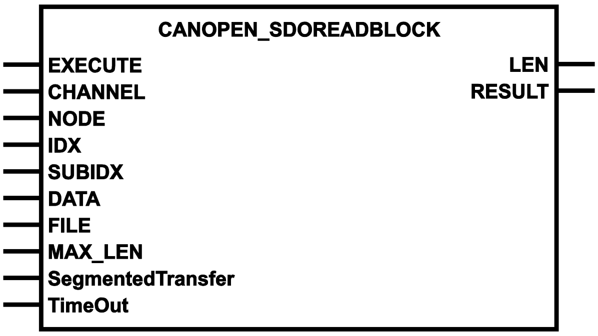
14942

= SDO Read Block

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

14943

CANOPEN_SDOREADBLOCK reads the indicated entry in the object directory of a node in the network via SDO block transfer.

> If the node doesn't support block transfer, the FB automatically changes to "segmented transfer". You can also directly change to "segmented transfer" via the input.

> The COB ID for the SDO is calculated from the transmitted node ID.

The length of multiframe SDOs is generally not limited.

For systems without a file system (e.g. BasicController CR04nn) the following applies:

► transmit an address to the FB which is accessed by the pointer for writing. The memory area determined by the start address DATA and the amount of data MAX_LEN must be available!

> If the amount of data is greater than indicated, the transfer is stopped and signalled via RESULT.

For systems with a file system (e.g. PDM360NG CR108n) the following applies:

► transmit the path and name of a file to the FB, in which the data is to be saved in binary format.

> The output RESULT provides information on the status of the SDO transmission.

Parameters of the inputs

14945

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
NODE	BYTE	(Node ID) ID of the node allowed = 1...127 = 0x01...0x7F The COB ID of the SDO is calculated from the node ID + 0x600
IDX	WORD	index in object directory
SUBIDX	BYTE	sub-index referred to the index in the object directory
DATA	DWORD	Address of the data zone for storage of the received data Input is without function for devices with file system (Linux).
FILE	STRING(80)	Path and file name for storage of the received data in binary format Input without function for device without file system (BasicSystem).
MAX_LEN	DWORD	Maximum permitted number of bytes which may be received
SegmentedTransfer (optional use of the parameter)	BOOL := FALSE	TRUE: Segmented SDO transfer FALSE: SDO block transfer
Timeout (optional use of the parameter)	TIME := T#10ms	waiting time of the FB for the response when the time has elapsed, the FB stops waiting. value = 0: use value from the configuration

Parameters of the outputs

14951

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
16	10	Transmission is active as a segmented download
17	11	Transmission is active as a block download
32	20	SDO transmission aborted by client or server (SDO abort code 0x80)
33	21	TIMEOUT elapsed
64	40	Error: Write pointer outside admissible data range
65	41	Error: File could not be opened
66	42	Error when writing to file
242	F2	Error: setting is not possible

CANOPEN_SDOREADMULTI

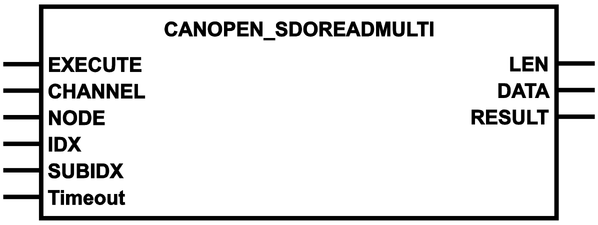
7806

= SDO read multi

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7808

CANOPEN_SDOREADMULTI reads the indicated entry in the object directory of a node in the network. The COB ID for the SDO is calculated from the transmitted node ID according to CANopen convention.

Parameters of the inputs

7809

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
NODE	BYTE	(Node ID) ID of the node allowed = 1...127 = 0x01...0x7F The COB ID of the SDO is calculated from the node ID + 0x600
IDX	WORD	index in object directory
SUBIDX	BYTE	sub-index referred to the index in the object directory
Timeout (optional use of the parameter)	TIME := T#10ms	waiting time of the FB for the response when the time has elapsed, the FB stops waiting. value = 0: use value from the configuration

Parameters of the outputs

7810

Parameter	Data type	Description
LEN	DWORD	number of the bytes received permissible values = 1...2 048 = 0x0000 0001...0x0000 0800
DATA	ARRAY [0..SDOMAXDATA] OF BYTE	buffer memory for user data of the SDO data transmission
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
5	05	FB is active – no data received yet
32	20	SDO transmission aborted by client or server (SDO abort code 0x80)
33	21	TIMEOUT elapsed
242	F2	Error: setting is not possible
255	FF	Error: not enough memory available for the consuming multiframe

CANOPEN_SDOWRITE

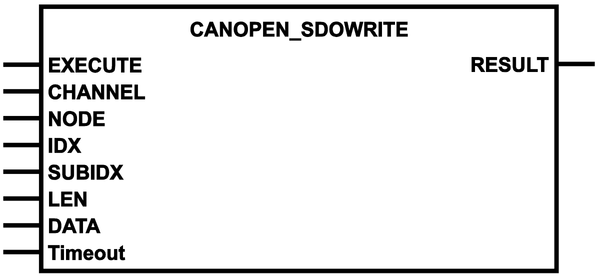
7825

= SDO write

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7826

CANOPEN_SDOWRITE is an easy function block for editing "Expedited SDOs", i.e. SDOs with max. 4 bytes user data. This type usually represents the bigger part of the SDO communication.

Expedited SDO = expedited service data object

A considerable amount of memory space can be saved due to the limitation of the data volume to max. 4 bytes of user data because this FB only needs to reserve 4 bytes as buffer storage and does not create a large data array itself.

Parameters of the inputs

7828

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
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	BYTE	number of the data bytes to be transmitted permissible values = 1...4 = 0x01...0x04
DATA	ARRAY [0..3] OF BYTE	data area (1...4 bytes)
Timeout (optional use of the parameter)	TIME := T#10ms	waiting time of the FB for the response when the time has elapsed, the FB stops waiting. value = 0: use value from the configuration

Parameters of the outputs

7829

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
8	08	function block is active
32	20	SDO transmission aborted by client or server (SDO abort code 0x80)
33	21	TIMEOUT elapsed
242	F2	Error: setting is not possible

CANOPEN_SDOWRITEBLOCK

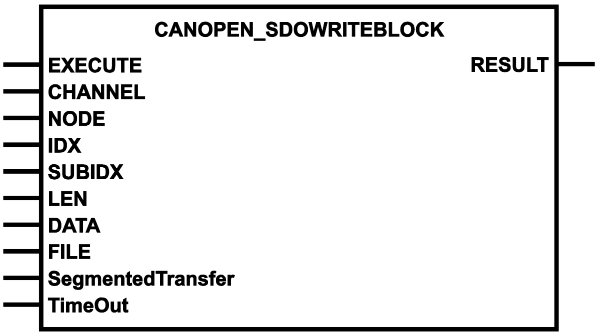
14961

= SDO Write Block

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

14963

CANOPEN _SDOWRITEBLOCK writes in the indicated entry in the object directory of a node in the network via SDO block transfer.

You can change to segmented transfer via the FB input if required.

> The COB ID for the SDO is calculated from the transmitted node ID.

> The output RESULT provides information on the status of the SDO transmission.

The length of multiframe SDOs is generally not limited.

For systems without a file system (e.g. BasicController CR04nn) the following applies:

► transmit an address to the FB which is accessed by the pointer for reading.

For systems with a file system (e.g. PDM360NG CR108n) the following applies:

► Transmit the path and name of a file to the FB, from which the data is to be read in binary format.

Parameters of the inputs

14964

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
NODE	BYTE	(Node ID) ID of the node allowed = 1...127 = 0x01...0x7F The COB ID of the SDO is calculated from the node ID + 0x600
IDX	WORD	index in object directory
SUBIDX	BYTE	sub-index referred to the index in the object directory
LEN	DWORD	number of data bytes to be transmitted in DATA allowed = 1...2 048 = 0x0000 0001...0x0000 0800
DATA	DWORD	Address of the data zone for reading of the data to be transmitted Input is without function for devices with file system (Linux).
FILE	STRING(80)	Path and file name for reading of the data to be transmitted in binary format Input without function for device without file system (BasicSystem).
SegmentedTransfer (optional use of the parameter)	BOOL := FALSE	TRUE: Segmented SDO transfer FALSE: SDO block transfer
Timeout (optional use of the parameter)	TIME := T#10ms	waiting time of the FB for the response when the time has elapsed, the FB stops waiting. value = 0: use value from the configuration

Parameters of the outputs

14968

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
16	10	Transmission is active as a segmented download
17	11	Transmission is active as a block download
32	20	SDO transmission aborted by client or server (SDO abort code 0x80)
33	21	TIMEOUT elapsed
65	41	Error: File could not be opened
242	F2	Error: setting is not possible

CANOPEN_SDOWRITEMULTI

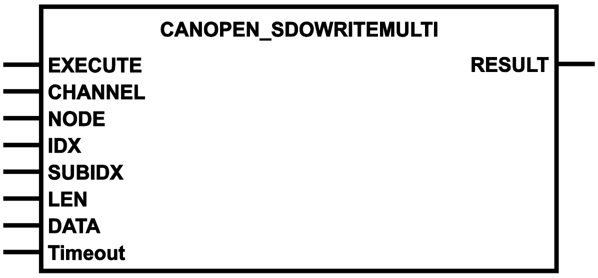
7832

= SDO write multi

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7834

CANOPEN_SDOWRITEMULTI writes the indicated entry in the object directory of a node in the network. The COB ID for the SDO is calculated from the transmitted node ID according to CANopen convention.

Parameters of the inputs

7835

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
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	DWORD	number of the data bytes to be transmitted permissible values = 1...2 048 = 0x0000 0001...0x0000 0800
DATA	ARRAY [0..SDOMAXDATA] OF BYTE	buffer memory for user data of the SDO data transmission
Timeout (optional use of the parameter)	TIME := T#10ms	waiting time of the FB for the response when the time has elapsed, the FB stops waiting. value = 0: use value from the configuration

Parameters of the outputs

7836

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
8	08	function block is active
32	20	SDO transmission aborted by client or server (SDO abort code 0x80)
33	21	TIMEOUT elapsed
242	F2	Error: setting is not possible

Function elements: CANopen SYNC

Contents

15069

CANOPEN_GETSYNCSTATE

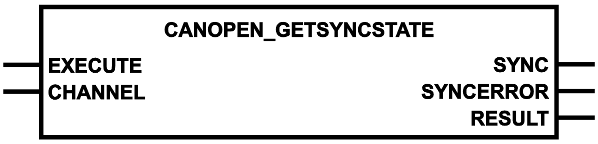
7871

= Get SYNC state

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7872

CANOPEN_GETSYNCSTATE reads...

- the setting of the SYNC functionality (active / not active),
- the error state of the SYNC functionality (SyncError).

If the PLC CAN runs as CANopen slave, it is signalled via this FB whether SYNC signals are absent or appear regularly.

Synchronous PDOS etc. are handled in the CAN stack. CANOPEN_GETSYNCSTATE, however, provides the error state so that the application program can react accordingly.

Parameters of the inputs

7874

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device

Parameters of the outputs

7875

Parameter	Data type	Description
SYNC	BOOL	status of the SYNC functionality TRUE: SYNC is activated: In the **master mode** SYNC telegrams are generated according to the settings in the configuration, and synchronous PDOs are transmitted and received. In the **slave mode** SYNC telegrams are received and accordingly processed. FALSE: SYNC is not active
SYNCERROR	BYTE	(sync error) SYNC error message 0 = no error >0 = SYNC error (slave mode)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	function block not yet executed
242	F2	Error: setting is not possible

CANOPEN_SETSYNCSTATE

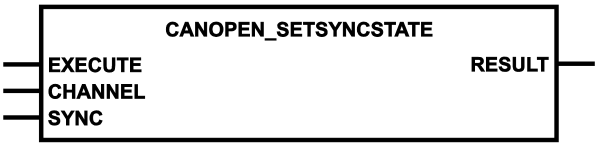
7883

= Set SYNC state

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7884

With CANOPEN_SETSYNCSTATE, the SYNC functionality is switched on and off.

Parameters of the inputs

7886

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
SYNC	BOOL	status of the SYNC functionality TRUE: SYNC is activated: In the **master mode** SYNC telegrams are generated according to the settings in the configuration, and synchronous PDOs are transmitted and received. In the **slave mode** SYNC telegrams are received and accordingly processed. FALSE: SYNC is not active

Parameters of the outputs

7887

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	function block not yet executed
38	26	SYNC could not be activated
242	F2	Error: setting is not possible

Function elements: CANopen guarding

Contents

15071

CANOPEN_GETGUARDHBERRLIST

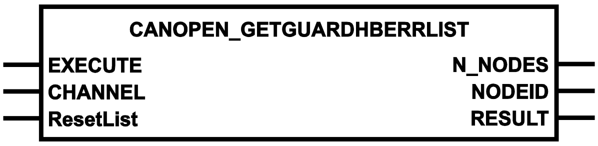
7896

= Get guard and heartbeat error list

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7898

CANOPEN_GETGUARDHBERRLIST lists all nodes in an array for which the master has detected an error:

- guarding error
- heartbeat error

Parameters of the inputs

7899

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
ResetList (optional use of the parameter)	BOOL := FALSE	Reset error list TRUE: Provide the error list as well as number of faulty nodes at the output and then reset. FALSE: function element is not executed

Parameters of the outputs

7900

Parameter	Data type	Description
N_NODES	WORD	Number of nodes with heartbeat or guarding error 0 = none of the nodes has a guarding or heartbeat error
NODEID	ARRAY [0..MAXGUARDERROR] OF BYTE	List of node IDs with heartbeat or guarding error. The most recent entry is in index 0. MAXGUARDERROR depends on device → chapter **Limitations for CANopen in this device** (→ page 40)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	FB is active – not yet processed
242	F2	Error: setting is not possible

CANOPEN_GETGUARDHBSTATSLV

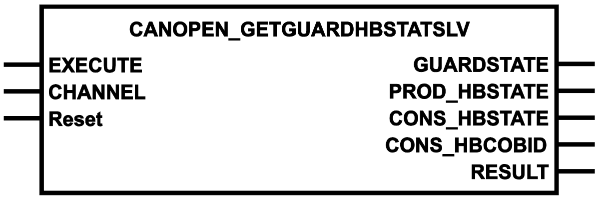
7902

= Get guard and heartbeat state slave

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7904

CANOPEN_GETGUARDANDHBSTATESLAVE reports the following states to the controller in slave operation:
- monitoring of node guarding
- monitoring of heartbeat

The controller can either be the heartbeat producer or the heartbeat consumer.

Parameters of the inputs

7905

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
Reset (optional use of the parameter)	BOOL := FALSE	TRUE: Provide the current states at the outputs and then reset to "No error" FALSE: function element is not executed

Parameters of the outputs

7906

Parameter	Data type	Description
GUARDSTATE	BYTE	Status of node guarding: 0 = 0x00 = no error (or: not active) 1 = 0x01 = timeout (configuration) 127 = 0x7F = no guarding message received
PROD_HBSTATE	BYTE	controller as heartbeat producer: 0 = 0x00 = inactive 1 = 0x01 = active
CONS_HBSTATE	BYTE	controller as heartbeat consumer: 0 = 0x00 = no fault 1 = 0x01 = timeout (configuration) 127 = 0x7F = no heartbeat message received yet
CONS_HBCOBID	WORD	COB-ID of the heartbeat message the consumer heartbeat of the controller is reacting to (configuration)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	FB is active – not yet processed
242	F2	Error: setting is not possible

Function elements: CANopen emergency

Contents

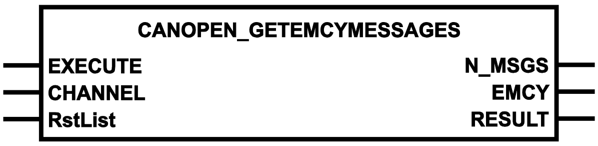
15073

CANOPEN_GETEMCYMESSAGES

7921

= Get emergency messages
Unit type = function block (FB)
Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7923

CANOPEN_GETEMCYMESSAGES returns all emergency messages that have been received by the controller from other nodes in the network since the last deletion of messages.

The list can be reset by setting the according input. A maximum of MAXEMCYMSGS messages is stored. Each message contains information from which the node it was sent. The most recent message is in index 0.

Parameters of the inputs

7924

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active. A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
RstList (optional use of the parameter)	BOOL := FALSE	TRUE: Provide list with accumulated CAN messages at the output and then delete FALSE: function element is not executed

Parameters of the outputs

7925

Parameter	Data type	Description
N_MSGS	DWORD	Number of accumulated messages
EMCY	ARRAY [0..MAXEMCYMSGS] OF T_EMCY	Emergency messages The most recent entry is in index 0. Structure of T_EMCY: .NODEID — ID of the node from which the message came .EEC — Emergency Error Code .ER — Error register .MSEF — Manufacturer Specific Error Code MAXEMCYMSG = 10
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	FB is active – not yet processed
242	F2	Error: setting is not possible

CANOPEN_GETERRORREGISTER

7915

= Get error register

Unit type = function block (FB)

Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

7917

CANOPEN_GETERRORREGISTER reads the error registers 0x1001 and 0x1003 from the controller.

Parameters of the inputs

7918

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
Reset_1001 (optional use of the parameter)	BOOL := FALSE	TRUE: Reset error register 0x1001 FALSE: function element is not executed
Reset_1003 (optional use of the parameter)	BOOL := FALSE	TRUE: Reset error register 0x1003 Set number of entries to 0 FALSE: function element is not executed The inputs remain unchanged.

Parameters of the outputs

7919

Parameter	Data type	Description
ER	BYTE	Content of the error register 0x1001
ERROR_FIELD	ARRAY [0..MAXERR] OF DWORD	Content of the error register 0x1003 Index 0 = number of the stored errors Index 1...MAXERR = stored errors The most recent error is in index 1 Preset: MAXERR = 5
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

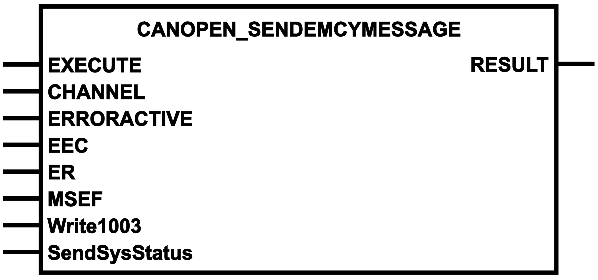
Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	FB is active – not yet processed
242	F2	Error: setting is not possible

CANOPEN_SENDEMCYMESSAGE

7908

= Send emergency message
Unit type = function block (FB)
Unit is contained in the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7910

CANOPEN_SENDEMCYMESSAGE sends an EMCY message. The message is assembled from the according parameters and entered in register 0x1003. The COB ID for the emergency message is determined from the configuration data.

Parameters of the inputs

7911

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
ERRORACTIVE	BOOL	FALSE ⇨ TRUE (edge): sends the next error code TRUE ⇨ FALSE (edge): If the error is no longer given, a message that there is no error is sent after a delay of 1 s.
EEC	WORD	EEC = Emergency Error Code
ER (optional use of the parameter)	BYTE := 0	0 = use value from error register 0x1001
MSEF	ARRAY [0..4] OF BYTE	MSEF = Manufacturer Specific Error Code = Additional error code which is defined by the manufacturer. Value comes from the application.
Write1003 (optional use of the parameter)	BOOL := FALSE	TRUE: Enter this EMCY message in object 0x1003 FALSE: function element is not executed
SendSysStatus (optional use of the parameter)	BOOL := FALSE	Send system status TRUE: The system status is checked and in case of an error state this is transmitted to the network. FALSE: function element is not executed

Parameters of the outputs

7912

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	FB is active – not yet processed
39	27	no object 1001_{16} in the configuration
242	F2	Error: setting is not possible

5.2.4 Function elements: SAE J1939

Contents

2273

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

Function elements: SAE J1939 status

Contents

15077

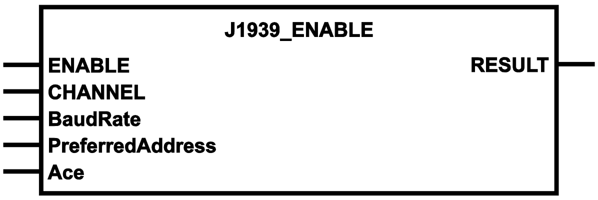
J1939_ENABLE

7641

Unit type = function block (FB)

Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7642

For initialisation of the J1939 stack, J1939_ENABLE is set to TRUE=1.

> This FB also causes booting of the soft I/Os of the CFG file.

> A different baud rate is only adopted if CAN_ENABLE has not been activated before.

ACE = **A**ddress **C**laiming **E**nable:

- If an ifm controller communicates with only one engine controller via J1939:
  set ACE = FALSE.
- If however several engine controllers are working on the same bus:
  set ACE = TRUE.
  In this case the engine controllers must support the address claiming!
  Otherwise you will risk an overlapping of addresses with subsequent system failure.

Parameters of the inputs

7643

Parameter	Data type	Description
ENABLE	BOOL := FALSE	TRUE: Enable J1939 channel Ace=TRUE: Address claiming effected FALSE: Block J1939 channel
CHANNEL	BYTE	CAN interface (1...n) depending on the device
Baud rate (optional use of the parameter)	WORD := 250	Baud rate [Kbits/s] permissible values: 20, 50, 100, 125, 250, 500, 800, 1 000
PreferredAddress (optional use of the parameter)	BYTE = 252	preferred source address
Ace (optional use of the parameter)	BOOL := TRUE	**A**ddress **C**laiming **E**nable TRUE: Address claiming enabled (control unit is self-configuring) FALSE: No address claiming

Parameters of the outputs

8542

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	function block is active
9	09	CAN is not active
242	F2	Error: setting is not possible

J1939_GETDABYNAME

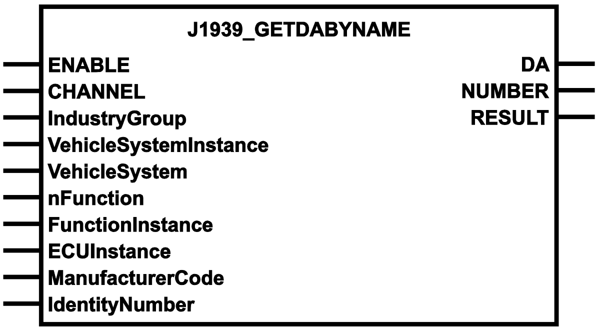
7664

= get destination arbitrary name

Unit type = function block (FB)

Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7665

Via J1939_GETDABYNAME, the target address of one or several participants can be determined by means of the name information.

- If a specific value is set on the optional inputs:
  ➩ the result list will only show the participants with this specific value.
- If no value or the default value is set on the optional inputs:
  ➩ this entry is not taken into account during filtration of the list.

Parameters of the inputs

7667

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
IndustryGroup (optional use of the parameter)	BYTE = 0xFF	industry group = industry group of the device permissible values = 0...7 255 = 0xFF = filter for all
VehicleSystemInstance (optional use of the parameter)	BYTE := 0xFF	instance of the vehicle system permissible values = 0...15 = 0x00...0x0F 255 = 0xFF = filter for all
VehicleSystem (optional use of the parameter)	BYTE := 0xFF	vehicle system permissible values = 0...127 = 0x00...0x7F 255 = 0xFF = filter for all
nFunction (optional use of the parameter)	WORD := 0xFFFF	function of the device permissible values = 0...255 = 0x0000...0x00FF 65 535 = 0xFFFF = filter for all
FunctionInstance (optional use of the parameter)	BYTE := 0xFF	instance of the function permissible values = 0...31 = 0x00...0x1F 255 = 0xFF = filter for all
ECUInstance (optional use of the parameter)	BYTE := 0xFF	instance of the control device permissible values = 0...7 255 = 0xFF = filter for all
ManufacturerCode (optional use of the parameter)	WORD := 0xFFFF	manufacturer code (must be requested from SAE) permissible values = 0...2047 (2^{11}-1) = 0x0000...0x07FF 65 535 = 0xFFFF = filter for all
IdentityNumber (optional use of the parameter)	DWORD := 0xFFFF FFFF	serial number of the device (should not be overwritten) permissible values = 0...2047 (2^{11}-1) = 0x0000 0000...0x0000 07FF 4 294 967 295 = 0xFFFF FFFF = filter for all

Parameters of the outputs

7668

Parameter	Data type	Description
DA	ARRAY [0..254] OF BYTE	List of found participants 255 = no participant found with this number
NUMBER	BYTE	Number of found bus participants.
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
8	08	function block is active
242	F2	Error: setting is not possible

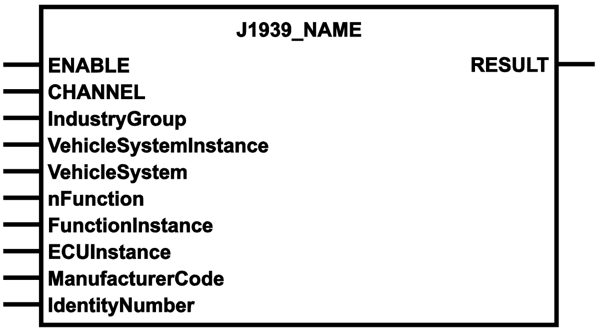
J1939_NAME

7646

Unit type = function block (FB)

Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7648

Via J1939_NAME, the device can be given a name for identification in the network.

By default the name of **ifm** is used.

The user has the following options to change the name of the device:

► use the information from the CFG file or

► overwrite the requested data via J1939_NAME.

> If no value or a default value is set at the optional inputs:
⇨ the preset value is not overwritten.

The following list shows the composition of the 64 bit NAME information according to SAE J1939-81:

Parameter	Data type	Description
arbitrary address capable	1 bit	any desired address available
industry group	3 bits	industry group of the device
vehicle system instance	4 bits	instance of the vehicle system
vehicle system	7 bits	vehicle system
reserved	1 bit	reserved
function	8 bits	function of the device
function instance	5 bits	instance of the function
ECU instance	3 bits	instance of the controller
manufacturer code	11 bits	manufacturer code (must be applied for at SAE)
identify number	21 bits	serial number of the device (should not be overwritten)

Table: Composition of the 64 bit NAME information according to SAE J1939-81

Parameters of the inputs

7652

Parameter	Data type	Description
ENABLE	BOOL := FALSE	TRUE: Any desired address available FALSE: Fixed address
CHANNEL	BYTE	CAN interface (1...n) depending on the device
IndustryGroup (optional use of the parameter)	BYTE = 0xFF	industry group = industry group of the device permissible values = 0...7 255 = 0xFF = filter for all
VehicleSystemInstance (optional use of the parameter)	BYTE := 0xFF	instance of the vehicle system permissible values = 0...15 = 0x00...0x0F 255 = 0xFF = filter for all
VehicleSystem (optional use of the parameter)	BYTE := 0xFF	vehicle system permissible values = 0...127 = 0x00...0x7F 255 = 0xFF = filter for all
nFunction (optional use of the parameter)	WORD := 0xFFFF	function of the device permissible values = 0...255 = 0x0000...0x00FF 65 535 = 0xFFFF = filter for all
FunctionInstance (optional use of the parameter)	BYTE := 0xFF	instance of the function permissible values = 0...31 = 0x00...0x1F 255 = 0xFF = filter for all
ECUInstance (optional use of the parameter)	BYTE := 0xFF	instance of the control device permissible values = 0...7 255 = 0xFF = filter for all
ManufacturerCode (optional use of the parameter)	WORD := 0xFFFF	manufacturer code (must be requested from SAE) permissible values = 0...2047 (2^{11}-1) = 0x0000...0x07FF 65 535 = 0xFFFF = filter for all
IdentityNumber (optional use of the parameter)	DWORD := 0xFFFF FFFF	serial number of the device (should not be overwritten) permissible values = 0...2047 (2^{11}-1) = 0x0000 0000...0x0000 07FF 4 294 967 295 = 0xFFFF FFFF = filter for all

Parameters of the outputs

7661

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	function block is active
242	F2	Error: setting is not possible

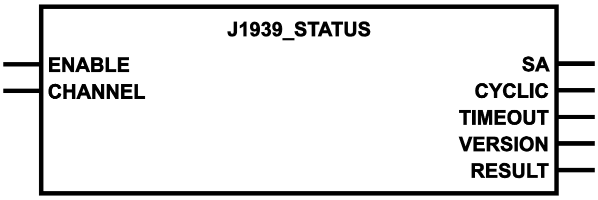
J1939_STATUS

7670

Unit type = function block (FB)

Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7672

Via J1939_STATUS, relevant information can be read back to the J1939 stack.

Parameters of the inputs

7673

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device

Parameters of the outputs

7674

Parameter	Data type	Description
SA	BYTE	current source address (e.g. after address claiming)
CYCLIC	WORD	number of the cyclic messages
TIMEOUT	BYTE	source address of the node which did not provided data for the process image in due time 255 = 0xFF = all nodes sent the data in due time
VERSION	DWORD	Version of the ifm CAN stack library
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	Protocol is active
2	02	Protocol is not active
3	03	Source address requested
4	04	Address lost
242	F2	Error: setting is not possible

Function elements: SAE J1939 request

Contents

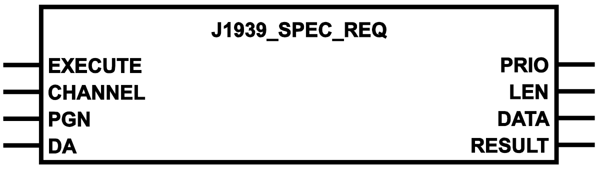
15079

J1939_SPEC_REQ

15023

= J1939 Specific Request
Unit type = function block (FB)
Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

15026

J1939_SPECIFIC_REQUEST requests and receives a specific message from another controller.

If a multiframe message is requested:
- the FB provides the first 8 bytes of the data
- RESULT indicates an error

Parameters of the inputs

15028

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
PGN	DWORD	PGN = **P**arameter **G**roup **N**umber Permissible = 0...262 143 = 0x00000000...0x0003FFFF
DA	BYTE	J1939 address of the requested device

Parameters of the outputs

15029

Parameter	Data type	Description
PRIO	BYTE	message priority (0…7)
LEN	WORD	number of the bytes received (0...8)
DATA	ARRAY [0..7] OF BYTE	received data, (1...8 bytes)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
5	05	FB is active – no data received yet
64	40	Error: receive multiframe
242	F2	Error: setting is not possible

J1939_SPEC_REQ_MULTI

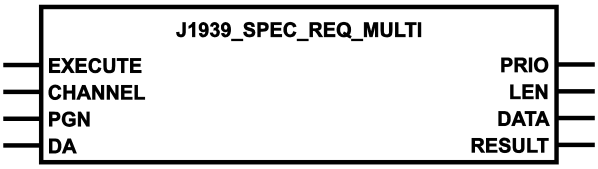
15033

= J1939 Specific Request Multiframe Message

Unit type = function block (FB)

Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

15036

J1939_SPECIFIC_REQUEST requests and receives a specific multiframe message from another controller.

Parameters of the inputs

15037

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
PGN	DWORD	PGN = **P**arameter **G**roup **N**umber Permissible = 0...262 143 = 0x00000000...0x0003FFFF
DA	BYTE	J1939 address of the requested device

Parameters of the outputs

15038

Parameter	Data type	Description
PRIO	BYTE	message priority (0…7)
LEN	WORD	number of data bytes to be transmitted allowed = 1...1 785 = 0x0001...0x06F9
DATA	ARRAY [0..1784] OF BYTE	Received data (1...1785 bytes)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
5	05	FB is active – no data received yet
242	F2	Error: setting is not possible

Function elements: receive SAE J1939

Contents

15081

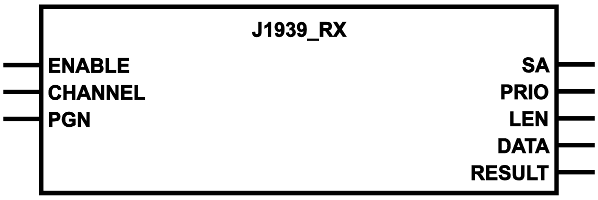
J1939_RX

7724

Unit type = function block (FB)

Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7725

J1939_RX is the easiest method for receiving single frame messages. The message read last on the CAN bus is returned.

Parameters of the inputs

7726

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
PGN	DWORD	PGN = **P**arameter **G**roup **N**umber Permissible = 0...262 143 = 0x00000000...0x0003FFFF

The PGN = 0 is not used.

Parameters of the outputs

7727

Parameter	Data type	Description
SA	BYTE	**S**ource **a**ddress of the transmitter
PRIO	BYTE	message priority (0…7)
LEN	WORD	number of the bytes received (0...8)
DATA	ARRAY [0..7] OF BYTE	received data, (1...8 bytes)
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	function block execution completed without error
5	05	FB is active – no data received yet
9	09	CAN is not active
242	F2	Error: setting is not possible

J1939_RX_FIFO

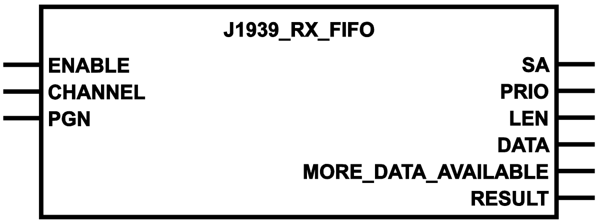
7732

= J1939 RX with FIFO

Unit type = function block (FB)

Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7733

J1939_RX_FIFO enables receipt of all specified messages and their successive reading from a FIFO.

Parameters of the inputs

7734

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
PGN	DWORD	PGN = **P**arameter **G**roup **N**umber Permissible = 0...262 143 = 0x00000000...0x0003FFFF

The PGN = 0 is not used.

Parameters of the outputs

7735

Parameter	Data type	Description
SA	BYTE	**S**ource **a**ddress of the transmitter
PRIO	BYTE	message priority (0…7)
LEN	BYTE	number of the bytes received (0...8)
DATA	ARRAY [0..7] OF BYTE	received data, (1...8 bytes)
MORE_DATA_AVAILABLE	BOOL	TRUE: further received data available in the FiFo FALSE: no further data available in the FiFo
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

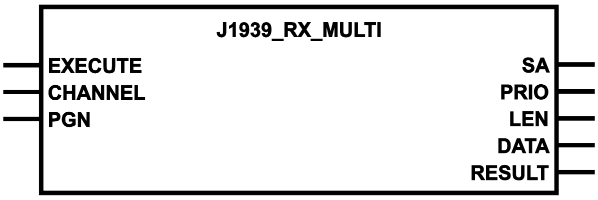
Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
5	05	FB is active – no data received yet
242	F2	Error: setting is not possible
250	FA	Error: FiFo is full – data was lost

J1939_RX_MULTI

7736

= J1939 RX multiframe message
Unit type = function block (FB)
Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7741

J1939_RX_MULTI enables receipt of multi-frame messages.

Parameters of the inputs

7743

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
PGN	DWORD	PGN = **P**arameter **G**roup **N**umber Permissible = 0...262 143 = 0x00000000...0x0003FFFF

The PGN = 0 is not used.

Parameters of the outputs

7744

Parameter	Data type	Description
SA	BYTE	**S**ource **a**ddress of the transmitter
PRIO	BYTE	message priority (0…7)
LEN	WORD	number of the bytes received permissible values = 0...1 785 = 0x0000 0000...0x0000 06F9
DATA	ARRAY [0..1784] OF BYTE	data to be sent (1...1785 bytes)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
5	05	FB is active – no data received yet
242	F2	Error: setting is not possible

Function elements: transmit SAE J1939

Contents

15083

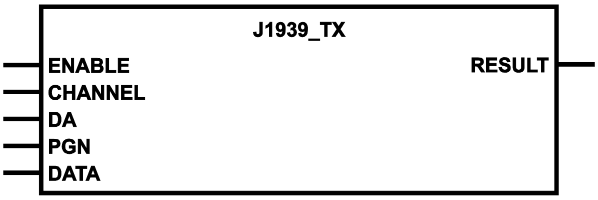
J1939_TX

7688

Unit type = function block (FB)

Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7689

J1939_TX is the easiest method for transmitting single frame messages.

Parameters of the inputs

7690

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
DA	BYTE := 249	DA = **D**estination **A**ddress of the ECU PGN > 61139: parameter DA is ignored
PGN	DWORD	PGN = **P**arameter **G**roup **N**umber Permissible = 0...262 143 = 0x00000000...0x0003FFFF
DATA	ARRAY [0..7] OF BYTE	data to be sent (1...8 bytes)

Parameters of the outputs

7693

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
242	F2	Error: setting is not possible
250	FA	Error: FiFo is full – data was lost

J1939_TX_ENH

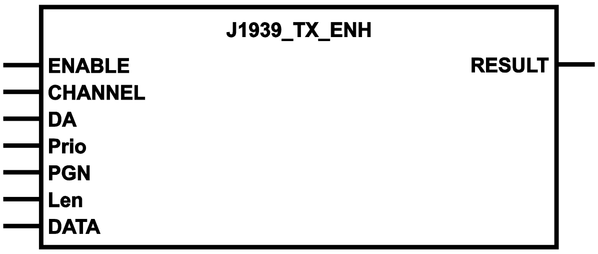
7696

= J1939 TX enhanced

Unit type = function block (FB)

Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7697

Additional setting options are provided by J1939_TX_ENH (for: enhanced) for single frame messages:

- transmitting priority
- data length

Multi frame messages → **J1939_TX_ENH_MULTI** (→ page 161).

Parameters of the inputs

7702

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
DA	BYTE := 249	DA = **D**estination **A**ddress of the ECU PGN > 61139: parameter DA is ignored
Prio (optional use of the parameter)	BYTE := 3	message priority permissible values = 0…7
PGN	DWORD	PGN = **P**arameter **G**roup **N**umber Permissible = 0...262 143 = 0x00000000...0x0003FFFF
Len (optional use of the parameter)	BYTE := 8	number of the bytes to be transmitted permissible values = 0...8
DATA	ARRAY [0..7] OF BYTE	data to be sent (1...8 bytes)

Parameters of the outputs

7969

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

Possible results for RESULT:

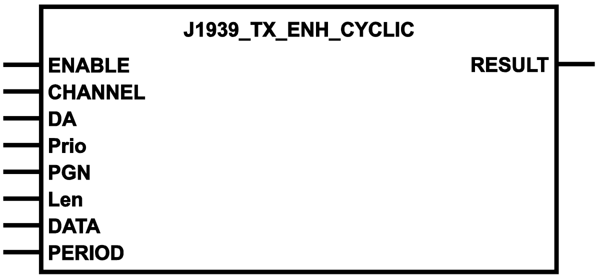
Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
242	F2	Error: setting is not possible
250	FA	Error: FiFo is full – data was lost

J1939_TX_ENH_CYCLIC

7716

= J1939 TX enhanced cyclic
Unit type = function block (FB)
Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7718

J1939_TX_ENH_CYCLIC serves for cyclic transmitting of CAN messages.
Otherwise, the FB corresponds to **J1939_TX_ENH** (→ page 157).

► Set the period duration via the parameter PERIOD.

> ⓘ If a period is too short, this could lead to a high bus load!
> The bus load can affect the performance of the complete system.

Parameters of the inputs

7719

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
DA	BYTE := 249	DA = **D**estination **A**ddress of the ECU PGN > 61139: parameter DA is ignored
Prio (optional use of the parameter)	BYTE := 3	message priority permissible values = 0…7
PGN	DWORD	PGN = **P**arameter **G**roup **N**umber Permissible = 0...262 143 = 0x00000000...0x0003FFFF
Len (optional use of the parameter)	BYTE := 8	number of the bytes to be transmitted permissible values = 0...8
DATA	ARRAY [0..7] OF BYTE	data to be sent (1...8 bytes)
PERIOD	TIME	period duration

Parameters of the outputs

7720

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
8	08	function block is active
242	F2	Error: setting is not possible

J1939_TX_ENH_MULTI

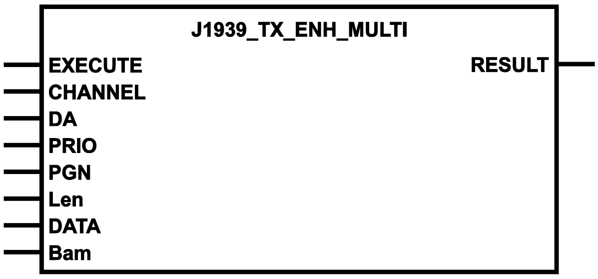
7699

= J1939 TX enhanced multiframe message

Unit type = function block (FB)

Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7705

The transmission of multi-frame messages is carried out via J1939_TX_ENH_MULTI.

The FB corresponds to **J1939_TX_ENH** (→ page 157). In addition, it can be determined whether the transmission shall be executed as BAM (**B**roadcast **A**nnounce **M**essage).

Parameters of the inputs

7712

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
DA	BYTE := 249	DA = **D**estination **A**ddress of the ECU PGN > 61139: parameter DA is ignored
Prio (optional use of the parameter)	BYTE := 3	message priority permissible values = 0…7
PGN	DWORD	PGN = **P**arameter **G**roup **N**umber Permissible = 0...262 143 = 0x00000000...0x0003FFFF
Len (optional use of the parameter)	BYTE := 8	number of the bytes to be transmitted permissible values = 0...8
DATA	ARRAY [0..1784] OF BYTE	data to be sent (1...1785 bytes)
Bam (optional use of the parameter)	BOOL := FALSE	BAM = **B**roadcast **A**nnounce **M**essage = message to all participants TRUE: multi-frame transmission as BAM message to all participants FALSE: automatic; message only to target address

Parameters of the outputs

7714

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
8	08	function block is active
65	41	Error: transmission is not possible
242	F2	Error: setting is not possible

Function elements: SAE J1939 diagnosis

Contents

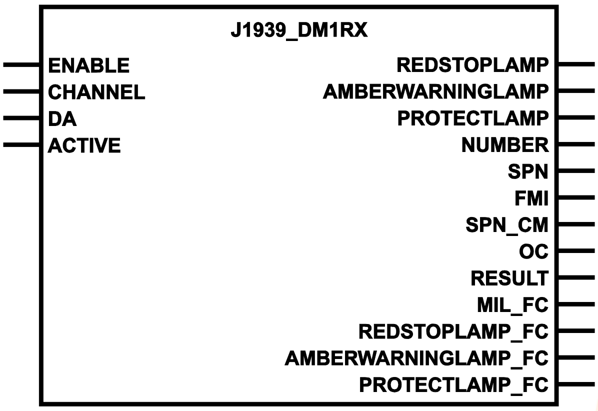
15085

J1939_DM1RX

14977

= J1939 Diagnostic Message 1 RX
Unit type = function block (FB)
Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7761

J1939_RX_DM1 receives diagnostic messages DM1 or DM2 from other ECUs.

Parameters of the inputs

14979

Parameter	Data type	Description
ENABLE	BOOL := FALSE	TRUE: execute this function element FALSE: unit is not executed > Function block inputs are not active > Function block outputs are not specified
CHANNEL	BYTE	CAN interface (1...n) depending on the device
DA	BYTE	DA = **D**estination **A**ddress of the ECU from where the DTCs are to be retrieved. DA = 254: read DTCs from the device itself
ACTIVE	BOOL	TRUE: Read active DTCs (DM1) FALSE: Read previously active DTCs (DM2)

Parameters of the outputs

14980

Parameter	Data type	Description
REDSTOPLAMP	BOOL	red stop lamp (for older projects only) TRUE: ON FALSE: OFF
AMBERWARNINGLAMP	BOOL	Amber warning lamp (for older projects only) TRUE: ON FALSE: OFF
PROTECTLAMP	BOOL	protect lamp (for older projects only) TRUE: ON FALSE: OFF
NUMBER	BYTE	number of the DTCs received (0...8)
SPN	WORD	**S**uspect **P**arameter **N**umber (→ J1939 specification)
FMI	BYTE	**F**ailure **M**ode **I**ndicator (→ J1939 specification) permissible values = 0...31 = 0x00...0x1F
SPN_CM	BOOL	conversion method (→ J1939 specification)
OC	BYTE	occurrence count
RESULT	BYTE	feedback of the function block (possible messages → following table)
MIL_FC	BYTE	Status of the electronic component: Malfunction indicatation light status and flash code: 0 = off 1 = on 2 = flash slowly 3 = flash quickly
REDSTOPLAMP_FC	BYTE	Status of the electronic component: red stop light status and flash code: 0 = off 1 = on 2 = flash slowly 3 = flash quickly
AMBERWARNINGLAMP_FC	BYTE	Status of the electronic component: Yellow warning light status and flash code: 0 = off 1 = on 2 = flash slowly 3 = flash quickly
PROTECTLAMP_FC	BYTE	Status of the electronic component: protection light status and flash mode: 0 = off 1 = on 2 = flash slowly 3 = flash quickly

Possible results for RESULT:

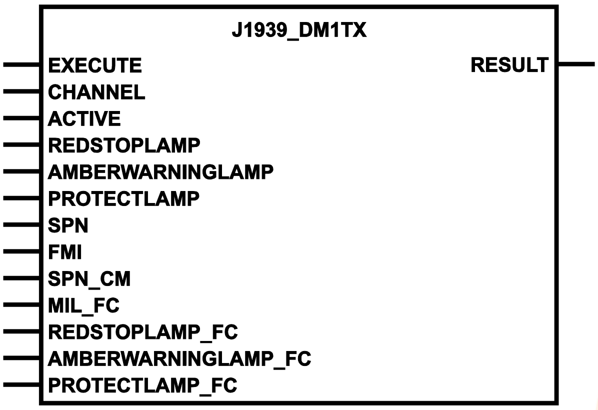
Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
8	08	FB is active – no data was received
242	F2	Error: setting is not possible

J1939_DM1TX

14993

= J1939 Diagnostic Message 1 TX
Unit type = function block (FB)
Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

7747

With J1939_TX_DM1 (DM = **D**iagnostic **M**essage) the controller can only transmit an active error message to the CAN stack.

> This message is stored in the hardware configuration.

> The message is marked "active" and transmitted once per second as DM1.

> If the error has already occurred, the event counter is incremented.
> The event counter is managed by the CAN stack.

> A disjunction of all bits of the trouble codes is executed. As soon as a bit is set in one of the trouble codes, it is equally set in the lamp state.

Upon arrival of a request at DM2, the CAN stack can read the according information from the hardware configuration and transmit it.

> When a DM3 message arrives, all inactive errors are deleted in the error memory in the hardware configuration.

Parameters of the inputs

14995

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
ACTIVE	BOOL	TRUE: DTC is active Cyclically transmitted (1x per second) as DM1 FALSE: DTC is no longer active Saved in the hardware configuration Transmitted as DM2 when requested
REDSTOPLAMP	BOOL	red stop lamp (for older projects only) TRUE: ON FALSE: OFF
AMBERWARNINGLAMP	BOOL	Amber warning lamp (for older projects only) TRUE: ON FALSE: OFF
PROTECTLAMP	BOOL	protect lamp (for older projects only) TRUE: ON FALSE: OFF
SPN	WORD	**S**uspect **P**arameter **N**umber (→ J1939 specification)
FMI	BYTE	**F**ailure **M**ode **I**ndicator (→ J1939 specification) permissible values = 0...31 = 0x00...0x1F
SPN_CM	BOOL	conversion method (→ J1939 specification)
MIL_FC	BYTE	Status of the electronic component: Malfunction indicatation light status and flash code: 0 = off 1 = on 2 = flash slowly 3 = flash quickly
REDSTOPLAMP_FC	BYTE	Status of the electronic component: red stop light status and flash code: 0 = off 1 = on 2 = flash slowly 3 = flash quickly
AMBERWARNINGLAMP_FC	BYTE	Status of the electronic component: Yellow warning light status and flash code: 0 = off 1 = on 2 = flash slowly 3 = flash quickly
PROTECTLAMP_FC	BYTE	Status of the electronic component: protection light status and flash mode: 0 = off 1 = on 2 = flash slowly 3 = flash quickly

Parameters of the outputs

7750

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	data was marked "active" in the error memory
242	F2	Error: setting is not possible

J1939_DM1TX_CFG

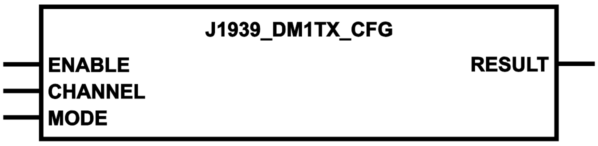
15424

= J1939 Diagnostic Message 1 TX configurable

Unit type = function block (FB)

Unit is contained in the library `ifm_J1939_NT_V02.00.02.LIB` or higher

Symbol in CODESYS:



Description

15426

As from runtime system V03.00.03 the CAN stack automatically sends a DM1 message every second as soon as the FB **J1939_ENABLE** (→ page 140) is called for the corresponding CAN interface.

► Use the FB J1939_DM1TX_CFG if you do not want the CAN stack to automatically and cyclically transmit DM1 messages.

The FB offers the following modes for cyclic transmission of DM1 messages:

MODE = 0 (preset)	The CAN stack sends DM1 "zero active faults" messages in compliance with standards every second. A manual transmission of DM1 messages via the FB **J1939_DM1TX** (→ page 166)is possible.
MODE = 1	The CAN stack does not send DM1 "zero active faults" messages. DM2 requests are answered automatically. A manual transmission of DM1 messages via the FB**J1939_DM1TX** (→ page 166) is possible.
MODE = 2	The CAN stack does not send cyclic DM1 "zero active faults" messages Nor does the CAN stack automatically reply to DM2 requests.

Parameters of the inputs

15427

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
CHANNEL	BYTE	CAN interface (1...n) depending on the device
MODE	BYTE := 0	Operating mode of the function block allowed = 0...2 (→ Description of the FB)

Parameters of the outputs

15429

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

Possible results for RESULT:

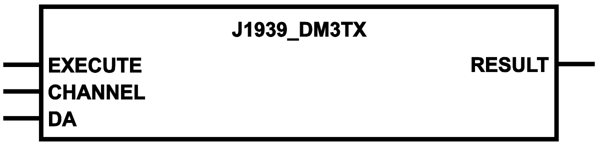
Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
242	F2	Error: setting is not possible

J1939_DM3TX

15002

= J1939 Diagnostic Message 3 TX
Unit type = function block (FB)
Unit is contained in the library `ifm_J1939_NT_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

15004

With J1939_DM3TX (DM = **D**iagnostic **M**essage) you can delete the inactive DTCs on another device.

> As soon as a DM3 message is received, all inactive errors in the error memory are deleted in the hardware configuration.

Parameters of the inputs

15006

Parameter	Data type	Description
EXECUTE	BOOL := FALSE	FALSE ⇨ TRUE (edge): execute function element once otherwise: function element is not active. A function element already started is processed.
CHANNEL	BYTE	CAN interface (1...n) depending on the device
DA	BYTE	DA = **D**estination **A**ddress of the ECU on which the DTCs are to be deteled. DA = 254: delete DTCs (DM2) in the device itself

Parameters of the outputs

15008

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
242	F2	Error: setting is not possible

5.2.5 Function elements: processing input values

Contents

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.

FASTCOUNT

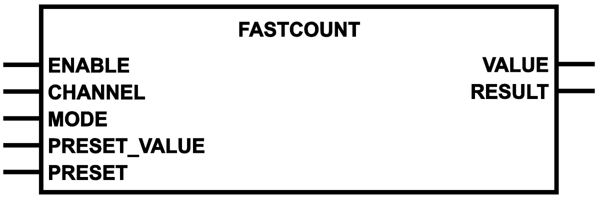
8112

= Fast Count

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

8114

FASTCOUNT operates as counter block for fast input pulses (up to 30 kHz).

This FB detects pulses at the fast input channels (→ data sheet).

> Overflow or underflow of the counter value is not detected.

Parameters of the inputs

16729

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > initiated processes continue in the background > FB outputs are not updated
CHANNEL	BYTE	Number of the fast input channel 12...15 for the inputs IN12...IN15
MODE	BYTE	Operating mode of the function block: 0 = 0x00 = stop counter 21 = 0x15 = upwards counter 22 = 0x16 = downwards counter
PRESET_VALUE	DWORD	counter start value
PRESET	BOOL	TRUE (for only 1 cycle): load the start value PRESET_VALUE FALSE: counter is active

Parameters of the outputs

8116

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

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	function block is active – valid values not yet available
130	82	channel setting is invalid
132	84	mode setting is invalid

INC_ENCODER

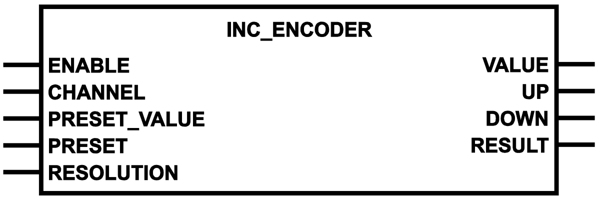
8134

= Incremental Encoder

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

8135

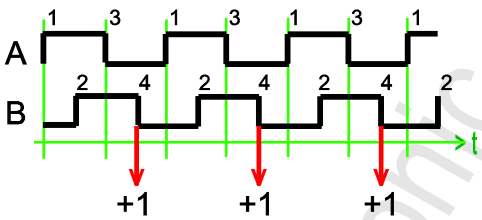
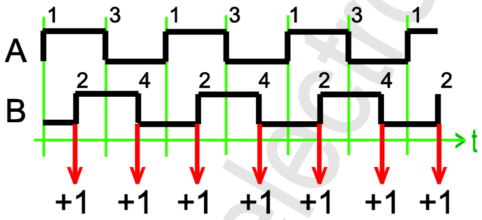
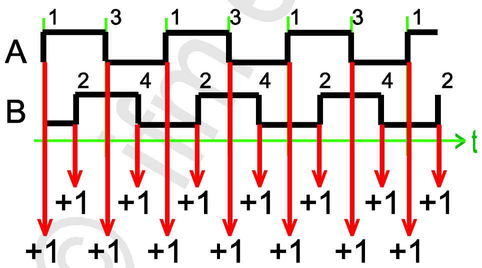
INC_ENCODER handles up/down counter functions for the evaluation of encoders.

Two frequency inputs form the input pair which is evaluated by means of the FB.
Permissible limit frequency = 0...1 000 Hz

Via PRESET_VALUE the counter can be set to a preset value. The value is adopted if PRESET is set to TRUE. Afterwards, PRESET must be set to FALSE again for the counter to become active again.

The current counter value is available at the output VALUE. The outputs UP and DOWN indicate the last counting direction of the counter. The outputs are TRUE if the counter has counted in the corresponding direction. If the counter was not changed since the last call of the FB, both the outputs are FLASE.

On input RESOLUTION the resolution of the encoder can be evaluated in multiples:
1 = normal resolution (-536 870 912...536 870 911, identical with the resolution of the encoder),
2 = double evaluation of the resolution (-1 073 741 824...1 073 741 823),
4 = 4-fold evaluation of the resolution (-2 147 483 648...2 147 483 647).
All other values on this input mean normal resolution.

	RESOLUTION = 1 counts for every fourth edge (= resolution of the encoder).
	RESOLUTION = 2 counts for every second edge
	RESOLUTION = 4 counts for every rising and falling edge

Parameters of the inputs

10259

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > initiated processes continue in the background > FB outputs are not updated
CHANNEL	BYTE	Number of the input channel pair 12 = channel pair 0 = inputs I12 + I13 14 = channel pair 1 = inputs I14 + I15
PRESET_VALUE	DINT	counter start value
PRESET	BOOL	TRUE (for only 1 cycle): load the start value PRESET_VALUE FALSE: counter is active
RESOLUTION	BYTE	evaluation of the encoder resolution: 01 = counts for every fourth edge (= resolution of the encoder) 02 = counts for every second edge 04 = counts for every rising and falling edge All other values count as "01".

Parameters of the outputs

8138

Parameter	Data type	Description

VALUE	DINT	if RESOLUTION = 1:
		VALUE = –536 870 912...536 870 911
		(= ¼ area of DINT)
		if RESOLUTION = 2:
		VALUE = –1 073 741 824...1 073 741 823
		(= ½ area of DINT)
		if RESOLUTION = 4:
		VALUE = –2 147 483 648…2 147 483 647
		(= area of DINT)

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
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	function block is active – valid values not yet available
130	82	channel setting is invalid
138	8A	resolution setting is invalid

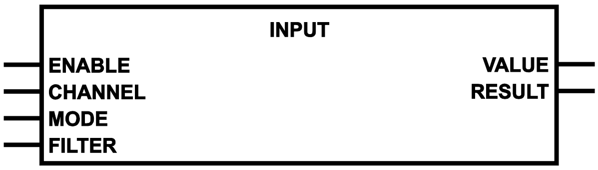
INPUT

8103

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

8104

INPUT enables determining the state at the input channels (→ data sheet). The FB provides the current state at the selected channel.

The measurement and the output value result from the operating mode indicated via MODE:
- binary input plus switching (BL) for positive sensor signal (with/without diagnosis)
- binary input minus switching (BH) for negative sensor signal
- analogue input 0...20 mA
- analogue input 0...10 V
- analogue input 0...32 V
- analogue input ratiometric 0...32 V
- analogue input resistance measurement 16...3 600 Ω (CR04nn from on HW release AD: 16...30 000 Ω)

! The operating mode should not be changed during operation.

The analogue values are provided as standardised values.

Parameters of the inputs

15879

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		
CHANNEL	BYTE	Number of input channel 0...15 for the inputs IN00...IN15		
MODE	BYTE	operating mode of the input channel:		
		0 = 0x00	off	
		1 = 0x01	(only for binary evaluated inputs) binary input, plus switching (BL)	
		3 = 0x03	voltage input	0…10 000 mV
		6 = 0x06	voltage input, ratiometric	0...1 000 ‰
		7 = 0x07	current input	0…20 000 µA
		9 = 0x09	voltage input	0…32 000 mV
		10 = 0x0A	(only for analogue evaluated inputs) binary input, plus switching (BL)	
		11 = 0x0B	(only for analogue evaluated inputs) binary input, plus switching (BL) with diagnosis (Namur)	
		12 = 0x0C	binary input, minus switching (BH)	
		18 = 0x12	resistance input	16…3 600 Ω from HW state AD: 16...30 000 Ω
FILTER	BYTE	filter for the measurement on the input: valid= 0...8 recommended = 4 → chapter **Configure the software filters of the inputs** (→ page 51)		

Parameters of the outputs

8106

Parameter	Data type	Description
VALUE	WORD	current value or status of the input channel (according to the selected operating mode MODE)
RESULT	BYTE	feedback of the function block (possible messages → following table)

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	function block is active – valid values not yet available
130	82	channel setting is invalid
132	84	mode setting is invalid
136	88	filter setting is invalid
141	8D	wire break occurred
142	8E	short to supply voltage occurred
144	90	Current at the input is too high

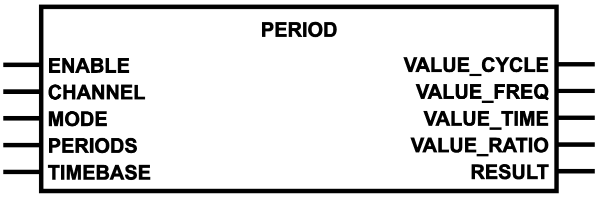
PERIOD

8122

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

15850

PERIOD measures the frequency in [Hz] or the period time (cycle time) in [µs] or the phase shift in [°] on the specified channel, depending on the set operating mode:

MODE dec	hex	Description
0	00	no measurement
14	0E	Frequency measurement Count the positive edges for a certain time.
19	13	Period duration measurement (better replace by MODE = 20!) Measure the time interval between two positive edges. Specify the average value over a certain number of periods.
20	14	Period duration and ratio measurement Measure the time interval between two positive edges. Specify the average value over a certain number of periods.
25	19	(LZS version 03.02.zz or higher) Phase shift (0...359°) between channel A and channel B of an input channel pair (message makes only sense if no great jumps > 179° can occur in the system)

> The operating mode should not be changed during operation.

> If MODE=19 or MODE=20 or MODE=25:
> Permissible input frequency = 0.1...3 000 Hz
> If the load is too high the cycle time can get unacceptably long.
> → chapter **Performance limits of the device** (→ page 39)

Parameters of the inputs

17816

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > initiated processes continue in the background > FB outputs are not updated
CHANNEL	BYTE	(MODE =14 / 20) Number of the fast input channel 12...15 for the inputs IN12...IN15 (MODE = 25) Number of the fast input A channel 12 / 14 for the inputs IN12 / IN14 B channel = A channel + 1
MODE	BYTE	Operating mode of the function block: 0 = 0x00 = no measurement 14 = 0x0E = frequency measurement 19 = 0x13 = interval measurement 20 = 0x14 = invertal and ratio measurement 25 = 0x19 = phase shift of two input signals
PERIODS	BYTE	Number of periods to be averaged (1...4) • if MODE = 14 / 19 / 20 ⇨ average arithmetically • if MODE = 25 ⇨ average geometrically • if PERIODS = 1 ⇨ no averaging
TIMEBASE	TIME	(only relevant if MODE = 14) time for counting the edges in [ms] permissible values = 1...2 000

Parameters of the outputs

8125

Parameter	Data type	Description
VALUE_CYCLE	DWORD	period duration of the input signal in [µs]
VALUE_FREQ	REAL	frequency of the input signal in [Hz]
VALUE_TIME	TIME	time elapsed since the last positive edge
VALUE_RATIO	WORD	mark-to-space ratio of the input signal in [‰]
RESULT	BYTE	feedback of the function block (possible messages → following table)

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	function block is active – valid values not yet available
130	82	channel setting is invalid
132	84	mode setting is invalid
137	89	value for PERIODS or TIMEBASE is invalid
146	92	Period duration is too long

5.2.6 Function elements: output functions

Contents

15075
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.

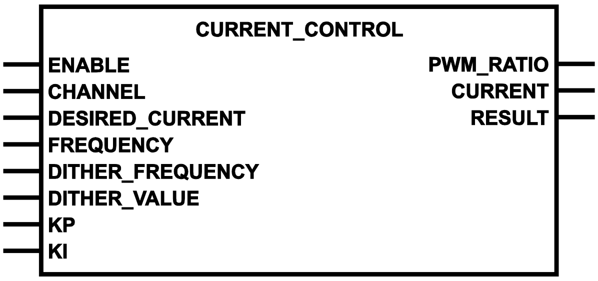
CURRENT_CONTROL

8082

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

8086

CURRENT_CONTROL operates as a current controller for the PWMI outputs.

The controller operates in dependence of the period duration 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. Depending on the requested controller behaviour the values can gradually be incremented (controller is stronger / faster) or decremented (controller is weaker / slower).

> At the desired value 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.

Depending on the controller hardware used, a different teach performance has to be noted.

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 ‰.

> With PWM frequencies below 100 Hz plus additional dither, the current control can no longer reach the indicated accuracy (→ data sheet).

Parameters of the inputs

17890

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > initiated processes continue in the background > FB outputs are not updated
CHANNEL	BYTE	Number of the current-controlled output channel 0...1 for the outputs OUT00...OUT01
DESIRED_CURRENT	WORD	desired current value of the output in [mA]
FREQUENCY	WORD	Permissible PWM frequency at the output in [Hz] allowed = 20...250 = 0x0014...0x00FA
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

Parameters of the outputs

8088

Parameter	Data type	Description
PWM_RATIO	WORD	for monitoring purposes: display PWM pulse ratio 0...1000 ‰
CURRENT	WORD	only available for current controlable outputs: current output current in [mA]
RESULT	BYTE	feedback of the function block (possible messages → following table)

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	function block is active – valid values not yet available
128	80	undervoltage on VBBx
129	81	overvoltage on VBBx
130	82	channel setting is invalid
131	83	value for DESIRED_CURRENT is invalid
134	86	dither setting is invalid

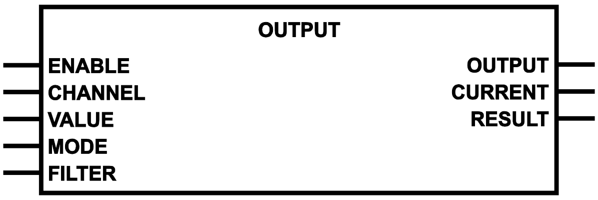
OUTPUT

8078

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

8079

OUTPUT assigns an operating mode to an output channel (→ data sheet). The FB enables the status detection on the selected output channel.

The measurement and the output value result from the operating mode indicated via MODE:
- binary output, plus switching (BH) with/without diagnostic function
- binary output, plus switching (BH) with diagnostic function and protection

> ! The operating mode should not be changed during operation.

Parameters of the inputs

17871

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
CHANNEL	BYTE	Number of the output channel 0...7 for the outputs OUT00...OUT07
VALUE	BOOL	TRUE: activate output FALSE: deactivate output
MODE	BYTE	Operating mode of the output: 0 = 0x00 = off 2 = 0x02 = binary output plus switching 15 = 0x0F = binary output plus switching with diagnosis 16 = 0x10 = binary output plus switching with diagnosis and protection
FILTER	BYTE	! only for outputs with current feedback: filter for the measurement on the output: valid = 0...8 recommended = 4 → chapter **Configure the software filters of the outputs** (→ page 55) ! For outputs without current feedback: FILTER = 0 or: do not set the parameter FILTER!

> ! The operating mode should not be changed during operation.

Parameters of the outputs

8081

Parameter	Data type	Description
OUTPUT	BOOL	TRUE: output is activated FALSE: output is deactivated
CURRENT	WORD	only available for current controlable outputs: current output current in [mA]
RESULT	BYTE	feedback of the function block (possible messages → following table)

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	function block is active – valid values not yet available
128	80	undervoltage on VBBx
129	81	overvoltage on VBBx
130	82	channel setting is invalid
132	84	mode setting is invalid
136	88	filter setting is invalid
141	8D	a wire break was detected (for binary output, plus switching (BH) with diagnosis)
142	8E	a short circuit was detected (for binary output plus switching (BH) with diagnosis)
145	91	current at the output is too high (for binary output plus switching (BH) with diagnosis and protection)

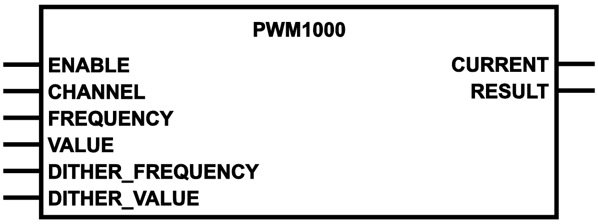
PWM1000

8060

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

8062

PWM1000 handles the initialisation and parameter setting of the PWM outputs.

The FB enables a simple use of the PWM FB in the device. For each channel an own PWM frequency, the mark-to-space ratio and the dither can be set.

The PWM frequency FREQUENCY can be directly indicated in [Hz] and the mark-to-space ratio VALUE in steps of 1 ‰.

Parameters of the inputs

17875

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > initiated processes continue in the background > FB outputs are not updated
CHANNEL	BYTE	Number of the PWM output channel 0...11 for the outputs OUT00...OUT11
FREQUENCY	WORD	PWM frequency in [Hz] allowed = 20...250 = 0x0014...0x00FA
VALUE	WORD	PWM value (mark-to-space ratio) in [‰] allowed = 0...1 000 = 0x0000...0x03E8 Values > 1 000 are regarded as = 1 000
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

Parameters of the outputs

8523

Parameter	Data type	Description
CURRENT	WORD	only available for current controlable outputs: current output current in [mA]
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	function block is active – valid values not yet available
128	80	undervoltage on VBBx
130	82	channel setting is invalid
131	83	value for VALUE is invalid
133	85	value for FREQUENCY is invalid
134	86	dither setting is invalid

5.2.7 Function elements: system

Contents

15067

Here we show you **ifm** functions that enable you to
- manage memory contents
- read information from software and hardware
- set or read various data and parameters

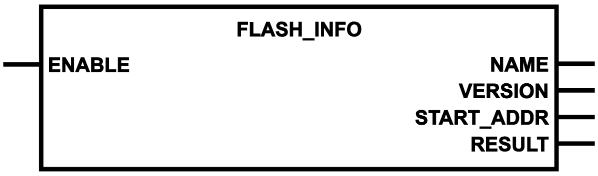
FLASH_INFO

11580

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

11588

FLASH_INFO reads the information from the user flash memory:
- name of the memory area (user defined),
- software version,
- start address (for simple reading with IEC structure).

Parameters of the inputs

11589

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

11590

Parameter	Data type	Description
NAME	STRING(24)	Name of the memory area (user defined)
VERSION	STRING(24)	Software version
START_ADDR	DWORD	Start address of the data
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
157	9D	Software header invalid (CRC error)

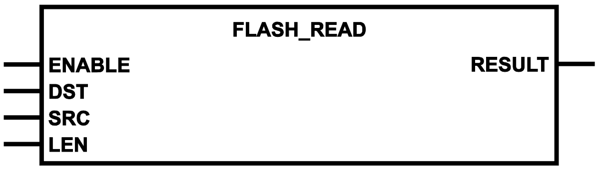
FLASH_READ

8147

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

11579

FLASH_READ 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 address resulting from SRC + LEN must be ≤ 65 408.

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

Parameters of the inputs

8148

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	destination address ⓘ Determine the address by means of the operator ADR and assigne it to the FB!
SRC	DWORD	relative start address in the memory valid = 0...65 407 = 0x0000 0000...0x0000 FF7F
LEN	WORD	number (≥ 1) of the data bytes to be transmitted

Parameters of the outputs

8152

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
152	98	inadmissible memory area: • invalid source address • invalid destination address • invalid number of bytes

GET_APP_INFO

11581

= get application information
Unit type = function block (FB)
Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

11593

GET_APP_INFO provides information about the application software stored in the device:
- name (= file name of the CODESYS project),
- version (= from CODESYS menu [Project] > [Project Info] > [Version]),
- unambiguous CoDeSys build number,
- CoDeSys build date.

Parameters of the inputs

11594

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

11595

Parameter	Data type	Description
NAME	STRING(24)	Name of the application
VERSION	STRING(24)	Version of the application program
BUILD_NUM	STRING(24)	Unique CODESYS build number (e.g.: "45")
BUILD_DATE	STRING(24)	CODESYS build date (e.g.: "20111006123800")
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

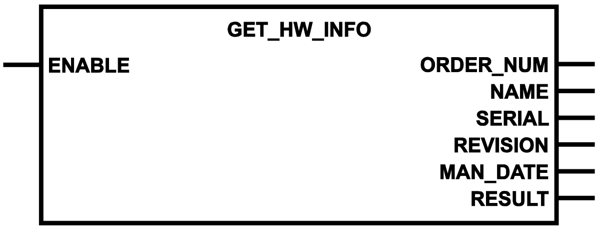
Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid

GET_HW_INFO

11582

= get hardware information
Unit type = function block (FB)
Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

1599

GET_HW_INFO provides information about the hardware of the device:
- **ifm** article number (e.g. CR0403),
- article designation,
- unambiguous serial number,
- hardware revision,
- production date.

Parameters of the inputs

11600

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

11601

Parameter	Data type	Description
ORDER_NUM	STRING(24)	ifm article no. (e.g.: CR0403)
NAME	STRING(24)	Article designation (e.g.: "BasicController 12/12")
SERIAL	STRING(24)	Serial number of the device (e.g.: "000045784")
REVISION	STRING(24)	Hardware revision level of the device (e.g.: "V01.00.01")
MAN_DATE	STRING(24)	Date of manufacture of the device (e.g.: "20111007123800")
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

GET_IDENTITY

8166

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

15411

GET_IDENTITY reads the identification stored in the device (has previously been saved by means of **SET_IDENTITY** (→ page 201)).

Parameters of the inputs

8167

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

8168

Parameter	Data type	Description
APP_IDENT	STRING(80)	identifier of the application as a string of max. 80 characters, e.g.: "Crane1704"
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

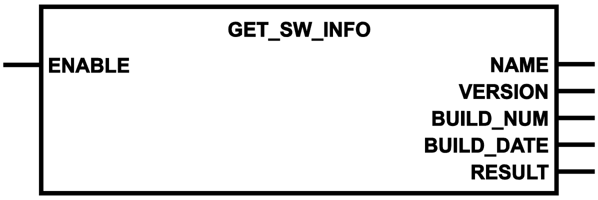
Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
155	9B	value could not be read

GET_SW_INFO

11583

= get software information
Unit type = function block (FB)
Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

11596

GET_SW_INFO provides information about the system software of the device:
- software name,
- software version,
- build number,
- build date.

Parameters of the inputs

11597

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

11598

Parameter	Data type	Description
NAME	STRING(24)	Name of the system software (e.g.: "BasicSystem")
VERSION	STRING(24)	Version of the system software (e.g.: "V02.00.03")
BUILD_NUM	STRING(24)	Build number of the system software (e.g.: "45")
BUILD_DATE	STRING(24)	Build date of the system software (e.g.: "20111006123800")
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

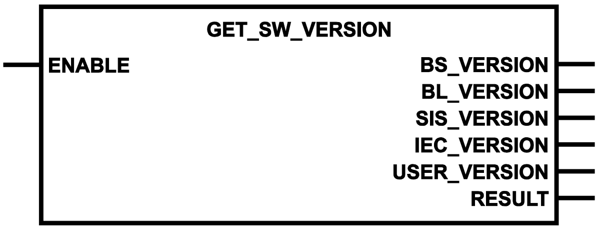
Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid

GET_SW_VERSION

14763

= get software version
Unit type = function block (FB)
Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

14765

GET_SW_VERSION provides information on the software in the device:
- BasicSystem version
- bootloader version
- SIS version
- IEC application program version
- IEC user flash version

Parameters of the inputs

14766

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

14767

Parameter	Data type	Description
BS_VERSION	STRING(24)	Basic system version
BL_VERSION	STRING(24)	Bootloader version
SIS_VERSION	STRING(24)	SIS version (SIS = **S**ystem **I**nformation **S**ervice)
IEC_VERSION	STRING(24)	IEC application program version
USER_VERSION	STRING(24)	IEC user flash version
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

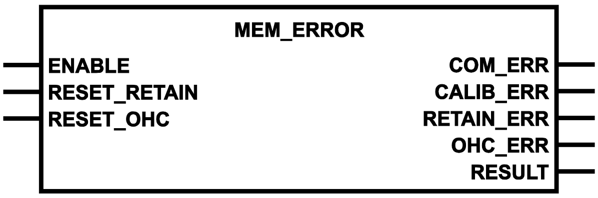
Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid

MEM_ERROR

14770

= Memory Error
Unit type = function block (FB)
Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

14772

MEM_ERROR signals errors in some parameters or in the memory.
The memory areas can be deleted via the corresponding FB inputs.

Parameters of the inputs

14773

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
RESET_RETAIN	BOOL	TRUE: Delete non-volatile retain memory FALSE: No changes to memory contents
RESET_OHC	BOOL	TRUE: Delete non-volatile OHC memory FALSE: No changes to memory contents

Parameters of the outputs

14774

Parameter	Data type	Description
COM_ERR	BOOL	Download ID and baud rate are set to default values (download parameters got lost)
CALIB_ERR	BOOL	Calibration values are invalid (analogue inputs, PWM outputs, system voltages)
RETAIN_ERR	BOOL	Retain memory is invalid (e.g. partially deleted due to strong magnetic field)
OHC_ERR	BOOL	OHC values are invalid (e.g. partially deleted due to strong magnetic field)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex		Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid

MEMCPY

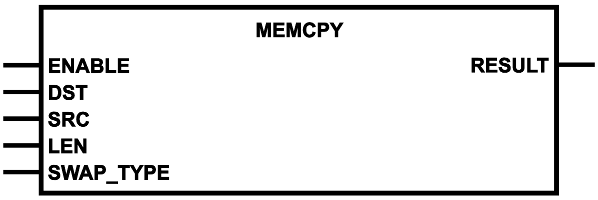
8160

= memory copy

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

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 FB!

> 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.

Parameters of the inputs

8162

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	destination address ! Determine the address by means of the operator ADR and assigne it to the FB!
SRC	DWORD	source address
LEN	WORD	number (≥ 1) of the data bytes to be transmitted
SWAP_TYPE	BYTE	Swap the byte sequence: 0 = no swapping e.g.: 1A 2B 3C 4D ⇨ 1A 2B 3C 4D 1 = swap 2 bytes (WORD, INT, ...) e.g.: 1A 2B 3C 4D ⇨ 2B 1A 4D 3C ! LEN must be a multiple of 2! 2 = swap 4 bytes (DWORD, DINT, REAL, TIME, ...) e.g.: 1A 2B 3C 4D ⇨ 4D 3C 2B 1A ! LEN must be a multiple of 4!

Parameters of the outputs

8163

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

Possible results for RESULT:

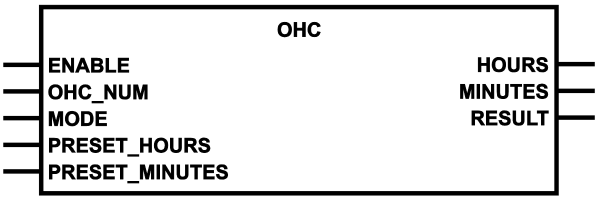
Value dec	Value hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
152	98	inadmissible memory area: • invalid source address • invalid destination address • invalid number of bytes
156	9C	inadmissible values: • invalid value for SWAP_TYPE • LEN does not match SWAP_TYPE

OHC

14777

= Operating Hours Counter
Unit type = function block (FB)
Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

14778

OHC provides 4 operating hours counters for universal use.
However, for hardware version < AD: only 2 operating hours counters possible.
Valid counting range: 0:00...4 294 967 295:59 hours (= 490 293 years, 25 days, 15 hours)

> If hardware version of device < AD:
> reset the memory area for OHC once:
> ▶ In the FB **MEM_ERROR** (→ page 196), set input RESET_OHC = TRUE!
> \> Only now can the operating hours counters be used.

Parameters of the inputs

14779

Parameter	Data type	Description
ENABLE	BOOL	TRUE: execute this function element FALSE: unit is not executed > initiated processes continue in the background > FB outputs are not updated
OHC_NUM	BYTE	Operating Hours Counter Number of the counter (0...3)
MODE	BYTE	Operating mode of the counter Permissible values = 0 = stop counter 1 = continue counting at the last stored value 2 = reset counter 3 = preset counter with the following values
PRESET_HOURS	DWORD	Preset hours (0...4 294 967 295 = 0x0000 0000...0xFFFF FFFF)
PRESET_MINUTES	BYTE	Preset minutes (0...59 = 0x00...0x3B)

Parameters of the outputs

14780

Parameter	Data type	Description
HOURS	DWORD	Counter value hours (0...4 294 967 295 = 0x0000 0000...0xFFFF FFFF)
MINUTES	BYTE	Counter value minutes (0...59 = 0x00...0x3B)
RESULT	BYTE	feedback of the function block (possible messages → following table)

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
130	82	Counter number in OHC_NUM is invalid
131	83	Preset value is invalid
132	84	mode setting is invalid
158	9E	Remanent memory is invalid (CRC error)

SET_IDENTITY

8174

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

8535

SET_IDENTITY sets an application-specific program identification.

Using this FB, a program identification can be created by the application program.

► This identification can be read in order to identify the loaded program:
  • via the software "Maintenance Tool"
  • in the application program via the FB **GET_IDENTITY** (→ page 193)

Parameters of the inputs

8175

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
APP_IDENT	STRING(80)	identifier of the application as a string of max. 80 characters, e.g.: "Crane1704" Reset with APP_IDENT = ""

Parameters of the outputs

8176

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

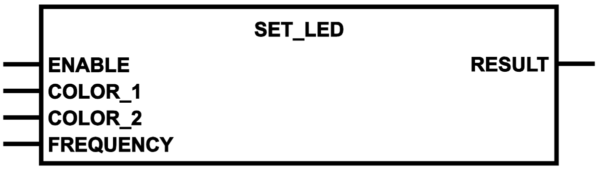
SET_LED

8052

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

8054

Via SET_LED frequency and color of the status LED can be changed in the application program.

> ! If the flashing mode is changed in the application program, the default setting table is no longer valid (→ chapter **Status LED** (→ page 28)).

Parameters of the inputs

8223

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
COLOR_1	BYTE	LED color for "switched on" color constant from the data structure "System LED Color"; allowed: 00 = LED_BLACK (= LED out) 01 = LED_RED 02 = LED_GREEN 03 = LED_YELLOW
COLOR_2	BYTE	LED color for "switched off" color constant from the data structure "System LED Color"; allowed: 00 = LED_BLACK (= LED out) 01 = LED_RED 02 = LED_GREEN 03 = LED_YELLOW
FREQUENCY	BYTE	LED flashing frequency Frequency constant from the data structure "System LED Frequency"; allowed: 00 = LED_0HZ = permanently ON 01 = LED_05HZ = flashes at 0.5 Hz 02 = LED_1HZ = flashes at 1 Hz 04 = LED_2HZ = flashes at 2 Hz 10 = LED_5HZ = flashes at 5 Hz

Parameters of the outputs

8227

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
2	02	function block is active (action not yet completed)
133	85	value for FREQUENCY is invalid
151	97	value for color is invalid

SET_PASSWORD

8178

Unit type = function block (FB)

Unit is contained in the library `ifm_CR2530_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

8179

SET_PASSWORD sets a user password for program and memory upload via the maintenance tool.

If the user password is active, reading of the application program or the data memory via the maintenance tool is only possible if the correct password has been entered.

If an empty string (default condition) is assigned to the PASSWORD input, the password is reset. Than an upload of the application software or of the data memory is possible at any time.

> ! The password is reset when loading a new application program.

Parameters of the inputs

8180

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
PASSWORD	STRING(16)	password If PASSWORD = "", than access is possible without enter of a password

Parameters of the outputs

8181

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

Possible results for RESULT:

Value dec	hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid

TIMER_READ_US

8219

Unit type = function block (FB)
Unit is contained in the library `ifm_CR2530_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

8220

Parameter	Data type	Description
TIME_US	DWORD	current system time [µs]
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

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*** (→ page 37)

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".

6.3 Response to system errors

8504

In principle, the programmer is responsible to react to the error messages in the application program. An error description is provided via the error message.

> The system resets the error message as soon as the error causing state is not present anymore.

6.3.1 Example process for response to an error message

8505

The runtime system cyclically writes the system flag TEMPERATURE.

The application program detects the device temperature by retrieving the INT variable.
If permissible values for the application are exceeded or not reached:

> The application program deactivates the outputs.

► Rectify the cause of the error.

> The application program detects the temperature value which has returned to normal:
The machine / system can be restarted or operation can be continued.

6.4 CAN / CANopen: errors and error handling

19604

→ System manual "Know-How ecomat*mobile*"
→ chapter **CAN / CANopen: errors and error handling**

7 Appendix

Contents

1664

Additionally to the indications in the data sheets you find summary tables in the appendix.

7.1 System flags

8374

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 flags (symbol name)	Type	Description
SUPPLY_VOLTAGE_VBBx	WORD	supply voltage on VBBx in [mV] CR040n: x = 1, 2 CR041n: x = 1, 2 CR253n: x = 1, 2
SUPPLY_VOLTAGE_VBBS	WORD	supply voltage on VBBs in [mV]
SUPPLY_VOLTAGE_VU	WORD	internal supply voltage in [mV]
TEMPERATURE	INT	temperature in the device in [°C]

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

Inputs: addresses and variables

21043

IEC address	I/O variable	Remark
%IB0	IN00	Binary input / analogue input channel 0
%IB1	IN01	Binary input / analogue input channel 1
%IB2	IN02	Binary input / analogue input channel 2
%IB3	IN03	Binary input / analogue input channel 3
%IB4	IN04	Binary input / resistance input channel 4
%IB5	IN05	Binary input / resistance input channel 5
%IB6	IN06	Binary input channel 6
%IB7	IN07	Binary input channel 7
%IB8	IN08	Binary input channel 8
%IB9	IN09	Binary input channel 9
%IB10	IN10	Binary input channel 10
%IB11	IN11	Binary input channel 11
%IB12	IN12	Binary input / fast input channel 12
%IB13	IN13	Binary input / fast input channel 13
%IB14	IN14	Binary input / fast input channel 14
%IB15	IN15	Binary input / fast input channel 15
%IW8	SUPPLY_VOLTAGE_VBBS	Power supply VBBS voltage in [mV]
%IW9	SUPPLY_VOLTAGE_VU	Power supply VU voltage in [mV]
%IW10	SUPPLY_VOLTAGE_VBB1	Power supply VBB1 voltage in [mV]
%IW11	SUPPLY_VOLTAGE_VBB2	Power supply VBB2 voltage in [mV]
%IW12	TEMPERATURE	Temperature in [°C]

Outputs: addresses and variables

21044

IEC address	I/O variable	Remark
%QB0	OUT00	Binary output / PWM output channel 0
%QB1	OUT01	Binary output / PWM output channel 1
%QB2	OUT02	Binary output / PWM output channel 2
%QB3	OUT03	Binary output / PWM output channel 3
%QB4	OUT04	Binary output / PWM output channel 4
%QB5	OUT05	Binary output / PWM output channel 5
%QB6	OUT06	Binary output / PWM output channel 6
%QB7	OUT07	Binary output / PWM output channel 7
%QB8	OUT08	Binary output / PWM output channel 8
%QB9	OUT09	Binary output / PWM output channel 9
%QB10	OUT10	Binary output / PWM output channel 10
%QB11	OUT11	Binary output / PWM output channel 11
%QB12	OUT12	Binary output channel 12
%QB13	OUT13	Binary output channel 13
%QB14	OUT14	Binary output channel 14
%QB15	OUT15	Binary output channel 15

7.2.2 Possible operating modes inputs/outputs

Contents

2386

Inputs: operating modes

15746

= this configuration value is default

Inputs	Possible operating mode		Set with function block	FB input	Value	
					dec	hex
IN00…IN03	off		INPUT	MODE	0	00
	voltage input	0…10 000 mV	INPUT	MODE	3	03
	voltage input ratiometric	0…1 000 ‰	INPUT	MODE	6	06
	current input	0…20 000 µA	INPUT	MODE	7	07
	voltage input	0…32 000 mV	INPUT	MODE	9	09
	binary input	plus switching	INPUT	MODE	10	0A
	binary input with diagnosis (Namur)	plus switching	INPUT	MODE	11	0B
	binary input	minus switching	INPUT	MODE	12	0C
IN04…IN05	off		INPUT	MODE	0	00
	binary input	plus switching	INPUT	MODE	10	0A
	binary input with diagnosis (Namur)	plus switching	INPUT	MODE	11	0B
	resistance input	16…30 000 ohms	INPUT	MODE	18	12
IN06…IN11	off		INPUT	MODE	0	00
	binary input	plus switching	INPUT	MODE	10	0A
	binary input with diagnosis (Namur)	plus switching	INPUT	MODE	11	0B
IN12…IN15	off		INPUT	MODE	0	00
	binary input, digital evaluation	plus switching	INPUT	MODE	1	01
	frequency measurement	0…30 000 Hz	PERIOD	MODE	14	0E
	period and ratio measurement	0.1...3 000 Hz	PERIOD	MODE	20	14
	Phase difference	0...359°	PERIOD	MODE	25	19
	upwards counter downwards counter	0…30 000 Hz	FASTCOUNT	MODE	21 22	15 16
	detect encoder	0…1 0000 Hz	INC_ENCODER			

Set operating modes with the following function block:

INPUT (→ page 176)	Assigns an operating mode to an input channel Provides the current state of the selected channel
FASTCOUNT (→ page 172)	Counter block for fast input pulses
INC_ENCODER (→ page 174)	Up/down counter function for the evaluation of encoders
PERIOD (→ page 179)	• measures at the indicated channel: the frequency and the period length (cycle time) in [µs], • measures at the indicated channel pair: the phase shift in [°] between channel A and channel B

Outputs: operating modes

15747

= this configuration value is default

Outputs	Possible operating mode		Set with function block	FB input	Value	
					dec	hex
OUT00 ...OUT01	off		OUTPUT	MODE	0	00
	binary output	plus switching	OUTPUT	MODE	2	02
	binary output with diagnosis	plus switching	OUTPUT	MODE	15	0F
	binary output with diagnosis and protection	plus switching	OUTPUT	MODE	16	10
	analogue output with pulse-width modulation		PWM1000			
	analogue current-controlled output		CURRENT_CONTROL			
OUT02 ...OUT07	off		OUTPUT	MODE	0	00
	binary output	plus switching	OUTPUT	MODE	2	02
	binary output with diagnosis	plus switching	OUTPUT	MODE	15	0F
	binary output with diagnosis and protection	plus switching	OUTPUT	MODE	16	10
	analogue output with pulse-width modulation		PWM1000			
OUT08 ...OUT09	off		OUTPUT	MODE	0	00
	binary output	plus switching	OUTPUT	MODE	2	02
	analogue output with pulse-width modulation		PWM1000			
	analogue output with pulse-width modulation, voltage-controlled	(at pins 25 + 43)	PWM1000			
OUT10 ...OUT11	off		OUTPUT	MODE	0	00
	binary output	plus switching	OUTPUT	MODE	2	02
	analogue output with pulse-width modulation		PWM1000			
OUT12 ...OUT15	binary output	plus switching	---	---	---	---

Set operating modes with the following function block:

OUTPUT (→ page 184)	Assigns an operating mode to an output channel Provides the current state of the selected channel
PWM1000 (→ page 186)	Initialises and configures a PWM-capable output channel the mark-to-space ratio can be indicated in steps of 1 ‰
CURRENT_CONTROL (→ page 182)	Current controller for a PWMi output channel

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

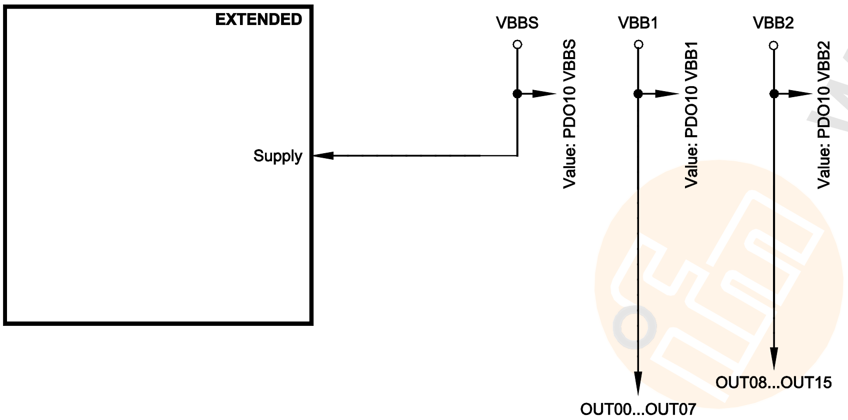
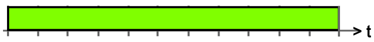


Figure: Block diagram of the supply

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)

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:

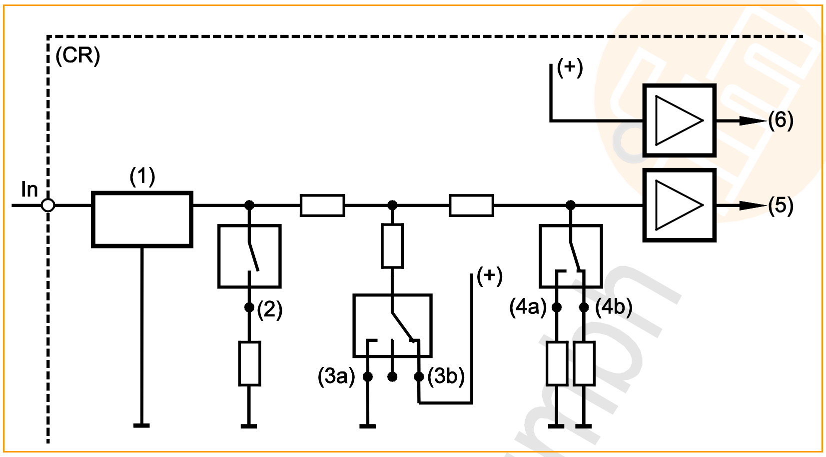
- current input 0...20 mA
- voltage input 0...10 V
- voltage input 0...32 V
- resistance measurement 16...30 000 Ω (measurement to GND)

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

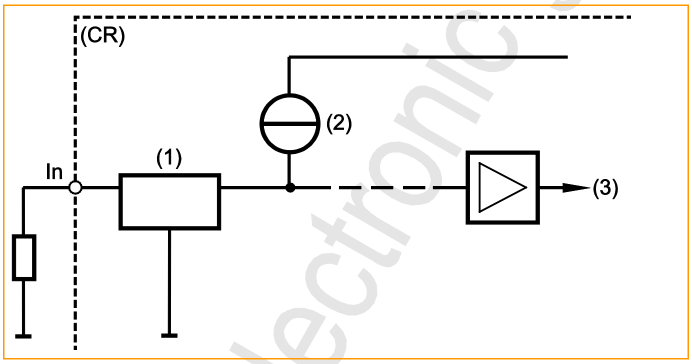


Figure: block diagram of the resistor survey input

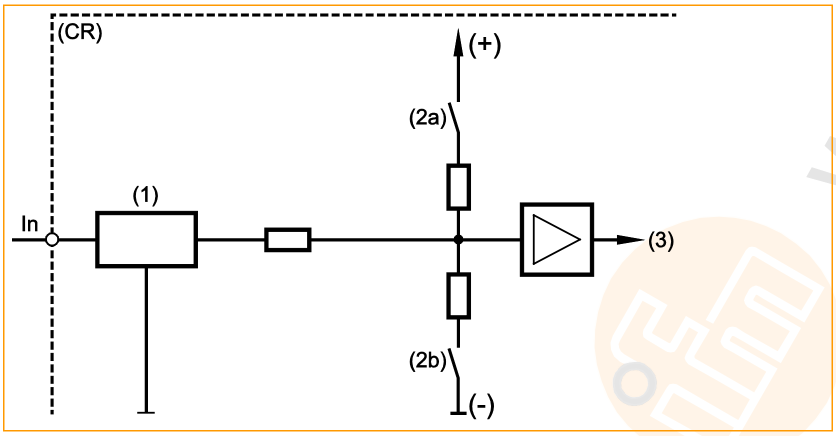
In = pin resistor survey input n
(CR) = device
(1) = input filter
(2) = constant-current source
(3) = voltage

Binary inputs

1015
7345

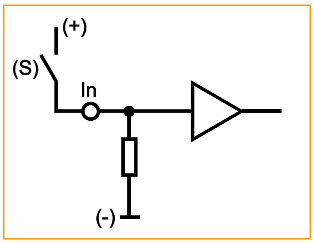
The binary input can be operated in following modes:
• binary input plus switching (BL) for positive sensor signal
• binary input minus switching (BH) for negative sensor signal

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



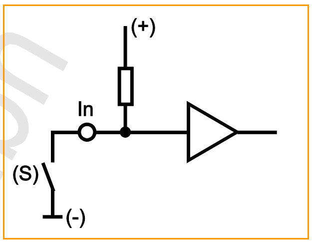
In = pin binary-input n
(CR) = device
(1) = input filter
(2a) = input minus switching
(2b) = input plus switching
(3) = voltage

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.

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** (→ page 238)

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** (→ page 233)

> 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** (→ page 238)

► Configuration of each input is made via the PLC configuration:
→ chapter **Configure inputs of the integrated I/O module** (→ page 233)

Resistance measurement

9773

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

8972

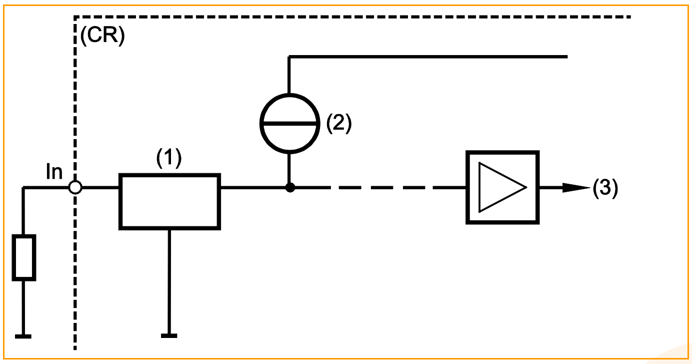


Figure: block diagram of the resistor survey input

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

8970

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

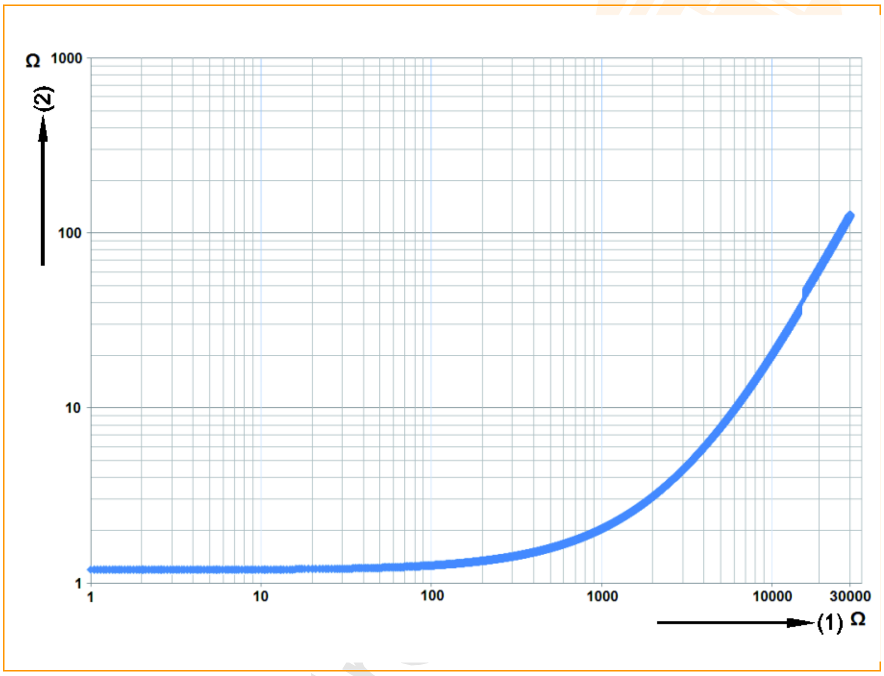


Figure: resolution dependent on the resistance value

(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.

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** (→ page 238)

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** (→ page 233)

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** (→ page 238)

► Configuration of each input is made via the PLC configuration:
→ chapter **Configure inputs of the integrated I/O module** (→ page 233)

Outputs of the integrated I/O module ExB01

Contents

16234

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** (→ page 238)

► Configuration of each output is made via the PLC configuration:
→ chapter **Configure outputs of the integrated I/O module** (→ page 235)

► ! 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:

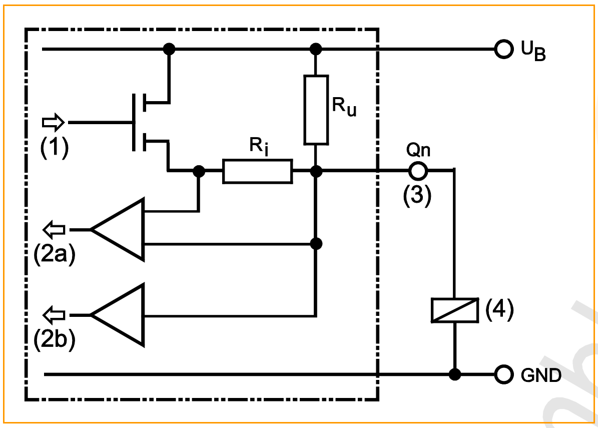


Figure: principle block diagram
(1) Output channel
(2a) Read back channel for diagnosis via current measuring
(2b) Read back channel for diagnosis via voltage measuring
(3) Pin output n
(4) Load

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).

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)

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** (→ page 238)

► Configuration of each output is made via the PLC configuration:
→ chapter **Configure outputs of the integrated I/O module** (→ page 235)

► 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:

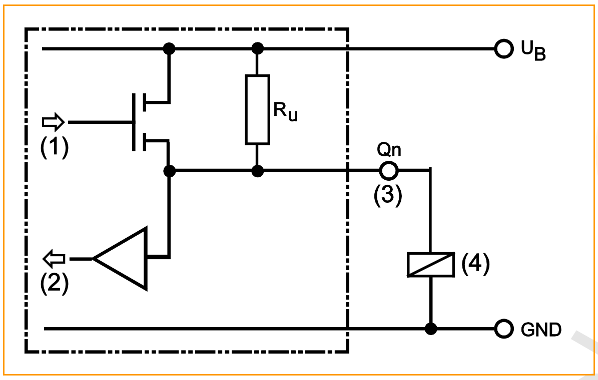


Figure: principle block diagram
(1) Output channel
(2) Read back channel for diagnosis
(3) Pin output n
(4) Load

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)

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** (→ page 238)

► Configuration of each output is made via the PLC configuration:
→ chapter **Configure outputs of the integrated I/O module** (→ page 235)

► 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** (→ page 238)

► Configuration of each output is made via the PLC configuration:
→ chapter **Configure outputs of the integrated I/O module** (→ page 235)

► 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** (→ page 238)

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

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	---	---

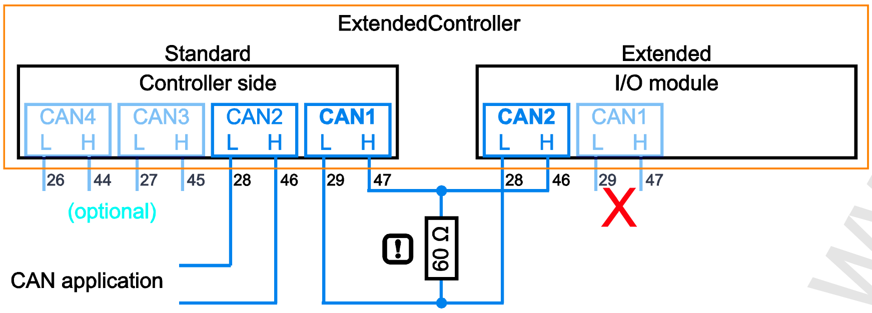
Standard baud rate = 250 Kbits/s

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!

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

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 (`CR2530 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:

```
CANopen Master[VAR]
   ExB01 (EDS)  [VAR]
      %QB256 Can-Output
      %IB256 Can-Input
```

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** (→ page 242)

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 **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.

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	

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)** (→ page 240)

> 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)** (→ page 240)

NAMUR diagnosis for binary signals of non-electronic switches:

► Equip the switch with an additional resistor connection!

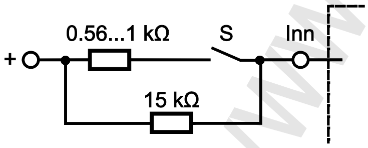
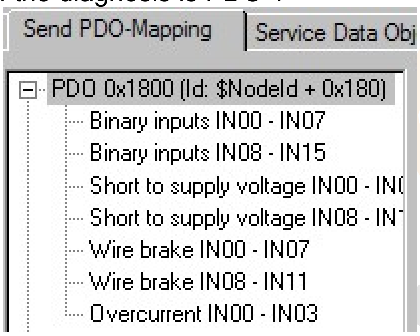


Figure: non-electronic switch S at input Inn

> The result of the diagnosis is PDO 1



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).

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)** (→ page 240)

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!

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

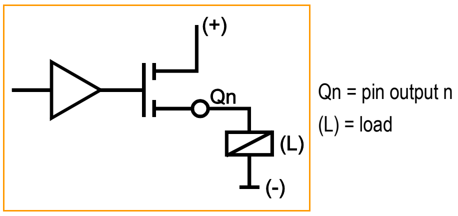
FILTER	Filter frequency [Hz]	Step response [ms] for ...			Remarks
		0…90 %	0…95 %	0…99 %	
fixed	52	7.2	9.4	14.4	

Binary outputs: configuration and diagnosis (I/O module ExB01)

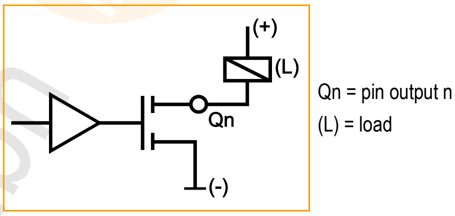
15882

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

15450



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



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 (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)** (→ page 241)

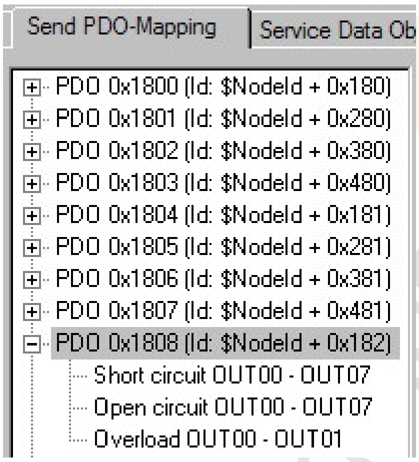
Binary outputs: Diagnosis (I/O module ExB01)

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)** (→ page 241)

> The result shows PDO 9:

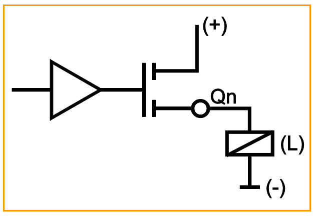


PWM outputs: I/O module ExB01

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.
  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)** (→ page 241)

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

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

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

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

7.3.3 Object directory of the integrated I/O module

Contents

15837

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

Object directory obligatory objects (index 0x1000…0x1FFF), overview

15979

Object directory Index	Sub-Idx	Parameter description	Parameter for	Preset parameter value	Change saved automatically?	When does the change become effective?
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

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

Object directory Index	Sub-Idx	Parameter description	Parameter for	Preset parameter value	Change saved automatically?	When does the change become effective?
	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	--	--

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

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

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

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

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).

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!

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

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

Index	S-Idx	Parameter name	Data type		Default	Details
	0x1	Consumer heartbeat time	rw	UDINT	0	Heartbeat monitoring time for the node Format: 0x0nntttt 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

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

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

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).

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

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

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

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

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

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

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

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

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

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

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

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

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	

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	

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

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

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

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

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

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)** (→ page 276)

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.

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	

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

7.4 Error tables

Contents

19606

7.4.1 Error flags

19608

→ chapter **System flags** (→ page 208)

7.4.2 Errors: CAN / CANopen

19610
19604

→ System manual "Know-How ecomat*mobile*"
→ chapter **CAN / CANopen: errors and error handling**

EMCY codes: CANx

13094

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

8412

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	I0 LSB	I0 MSB				Inputs interruption
08	21	03	I0 LSB	I0 MSB				Inputs short circuit
10	21	03	I0 LSB	I0 MSB				Excess current 4…20 mA
00	23	03	Q0 LSB	Q0 MSB				Outputs interruption
08	23	03	Q0 LSB	Q0 MSB				Outputs short circuit
00	31	05						Power supply VBBs
00	33	05						Terminal voltage VBBo
08	33	05						Output voltage VBBr
00	42	09						Excess temperature

In the CANopen stack none of these EMCY codes are fix implemented. Advice:

► Make these EMCY codes with CANOPEN_SENDEMCYMESSAGE.

8 Glossary of Terms

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.

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

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.

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.

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.

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

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** (→ page 289)

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.
The PDU is an item of the →CAN protocol →SAE J1939. PDU indicates a part of the destination or source address.

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 = PDU format (PF) + PDU source (PS)
The parameter group number is an item of the →CAN protocol →SAE J1939. PGN collects the address parts PF and PS.

Pictogram

Pictograms are figurative symbols which convey information by a simplified graphic representation.
(→ chapter **What do the symbols and formats mean?** (→ page 6))

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.

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.

rw

RW = read/ write
Bidirectional data transmission: Data can be read and also changed.

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?** (→ page 6))

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.

wo

WO = write only
Unidirectional data transmission: Data can only be changed and not read.

9 Index

A

B

C

D

J

L

M

N

O

W

10 Notizen • Notes • Notes

11 ifm weltweit • ifm worldwide • ifm à l’échelle internationale

Version: 2015-03-06 8310

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	ifm electronic gmbh Vertrieb Deutschland Niederlassung Nord • 31135 Hildesheim • Tel. 0 51 21 / 76 67-0 Niederlassung West • 45128 Essen • Tel. 02 01 / 3 64 75 -0 Niederlassung Mitte-West • 58511 Lüdenscheid • Tel. 0 23 51 / 43 01-0 Niederlassung Süd-West • 64646 Heppenheim • Tel. 0 62 52 / 79 05-0 Niederlassung Baden-Württemberg • 73230 Kirchheim • Tel. 0 70 21 / 80 86-0 Niederlassung Bayern • 82178 Puchheim • Tel. 0 89 / 8 00 91-0 Niederlassung Ost • 07639 Tautenhain • Tel. 0 36 601 / 771-0 ifm electronic gmbh • Friedrichstraße 1 • 45128 Essen
A	ifm electronic gmbh • 1120 Wien • Tel. +43 16 17 45 00
OFF	ifm efector pty ltd. • Mulgrave Vic 3170 • Tel. +61 3 00 365 088
B, L	ifm electronic N.V. • 1731 Zellik • Tel. +32 2 / 4 81 02 20
BR	ifm electronic Ltda. • 03337-000, Sao Paulo SP • Tel. +55 11 / 2672-1730
CH	ifm electronic ag • 4 624 Härkingen • Tel. +41 62 / 388 80 30
CN	ifm electronic (Shanghai) Co. Ltd. • 201203 Shanghai • Tel. +86 21 / 3813 4800
CND	ifm efector Canada inc. • Oakville, Ontario L6K 3V3 • Tel. +1 800-441-8246
CZ	ifm electronic spol. s.r.o. • 25243 Průhonice • Tel. +420 267 990 211
DK	ifm electronic a/s • 2605 BROENDBY • Tel. +45 70 20 11 08
E	ifm electronic s.a. • 08820 El Prat de Llobregat • Tel. +34 93 479 30 80
F	ifm electronic s.a. • 93192 Noisy-le-Grand Cedex • Tél. +33 0820 22 30 01
FIN	ifm electronic oy • 00440 Helsinki • Tel . +358 75 329 5000
GB, IRL	ifm electronic Ltd. • Hampton, Middlesex TW12 2HD • Tel. +44 208 / 213-0000
GR	ifm electronic Monoprosopi E.P.E. • 15125 Amaroussio • Tel. +30 210 / 6180090
H	ifm electronic kft. • 9028 Györ • Tel. +36 96 / 518-397
I	ifm electronic s.a. • 20041 Agrate-Brianza (MI) • Tel. +39 039 / 68.99.982
IL	Astragal Ltd. • Azur 58001 • Tel. +972 3 -559 1660
IND	ifm electronic India Branch Office • Kolhapur, 416234 • Tel. +91 231-267 27 70
J	efector co., ltd. • Chiba-shi, Chiba 261-7118 • Tel. +81 043-299-2070
MAL	ifm electronic Pte. Ltd • 47100 Puchong Selangor • Tel. +603 8063 9522
MEX	ifm efector S. de R. L. de C. V. • Monterrey, N. L. 64630 • Tel. +52 81 8040-3535
N	Sivilingeniør J. F. Knudtzen A/S • 1396 Billingstad • Tel. +47 66 / 98 33 50
NL	ifm electronic b.v. • 3843 GA Harderwijk • Tel. +31 341 / 438 438
P	ifm electronic s.a. • 4410-136 São Félix da Marinha • Tel. +351 223 / 71 71 08
PL	ifm electronic Sp. z o.o. • 40-106 Katowice • Tel. +48 32-608 74 54
RA, ROU	ifm electronic s.r.l. • 1107 Buenos Aires • Tel. +54 11 / 5353 3436
ROK	ifm electronic Ltd. • 140-884 Seoul • Tel. +82 2 / 790 5610
RP	Gram Industrial, Inc. • 1770 Mantilupa City • Tel. +63 2 / 850 22 18
RUS	ifm electronic • 105318 Moscow • Tel. +7 495 921-44-14
S	ifm electronic a b • 41250 Göteborg • Tel. +46 31 / 750 23 00
SGP	ifm electronic Pte. Ltd. • Singapore 609 916 • Tel. +65 6562 8661/2/3
SK	ifm electronic s.r.o. • 835 54 Bratislava • Tel. +421 2 / 44 87 23 29
THA	SCM Allianze Co., Ltd. • Bangkok 10 400 • Tel. +66 02 615 4888
TR	ifm electronic Ltd. Sti. • 34381 Sisli/Istanbul • Tel. +90 212 / 210 50 80
UA	TOV ifm electronic • 02660 Kiev • Tel. +380 44 501 8543
USA	ifm efector inc. • Exton, PA 19341 • Tel. +1 610 / 5 24-2000
ZA	ifm electronic (Pty) Ltd. • 0157 Pretoria • Tel. +27 12 345 44 49

Technische Änderungen behalten wir uns ohne vorherige Ankündigung vor.
We reserve the right to make technical alterations without prior notice.
Nous nous réservons le droit de modifier les données techniques sans préavis.