ifm electronic

CE

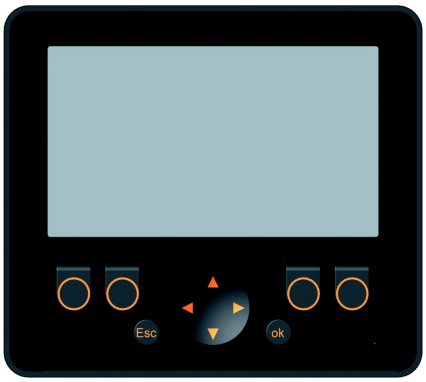
ecomat mobile

Original Programming Manual
BasicDisplay

ecomat100
CR0452

Runtime system v03.02
CODESYS® v2.3

English



7391002_03_UK 2015-04-07

Contents

1 About this manual

Contents

202

1.1 Copyright

6088

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

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

1.2 Overview: documentation modules for ecomatmobile devices

17405

The documentation for **ecomatmobile** 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] > CR0452 > [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] > CR0452 > [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] > CR0452 > [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] > CR0452 > [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 **ecomatmobile** at:
→ **ecomatmobile** 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
	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?

204
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 → Annex.

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 country > [Data sheet search] > (Article no.) > [Operating instructions]

1.6 History of the instructions (CR0452)

15324

What has been changed in this manual? An overview:

Date	Theme	Change
2014-03-24	Visualisation limits	Information concerning the permissible drawing area
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"
2015-01-13	Structure of documentation for error codes, system flags	• error flags: now only in the annex, chapter ***System flags*** • CAN / CANopen errors and error handling: now only in the system manual "Know-How" • error codes, EMCY codes: now in the annex, chapter ***Error tables***
2015-03-10	Available memory	Description improved

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 bodily injury!
ifm electronic gmbh does not assume any liability in this regard.

- ► The acting person must have read and understood the safety instructions and the corresponding chapters in this manual before working on and with this device.
- ► The acting person must be authorised to work on the machine/equipment.
- ► The acting person must have the qualifications and training required to perform this work.
- ► Adhere to the technical data of the devices!
  You can find the current data sheet on **ifm's** homepage 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

⚠ 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.
- ► Additionally 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.
- ► Additionally 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.

3 System description

Contents

975

3.1 Information about the device

15407

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

- BasicDisplayXL: CR0452

The display is part of the family of the BasicController: CR040n, CR041n, CR043n.

3.1.1 Accessories

15406

A wide range of accessories is available for the BasicDisplay. Examples:

EC0404	Frame for front panel mounting of CR0452
EC0406	RAMmount set for using CR0452 as a desktop unit
EC0452	Cable for power supply and CAN between the display and the BasicController when the cover EC0402 is used
EC0454	5 m cable for power supply and CAN between the display and the BasicController
---	"Maintenance Tool" software for updating firmware, runtime system and application program → ecomatmobile DVD "Software, tools and documentation"

You can find accessories for the article on ifm's website:
→ www.ifm.com > Select your country > [Data sheet search] > article no. > [Accessories]

3.2 Hardware description

Contents

14081

3.2.1 Hardware setup

Contents

15269

Protection IP 65
on the front panel when mounted: Protection IP 67

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 159) (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.

Interfaces

15383
15272

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	Interface do not exist	Interface do not exist	Interface do not exist
	CANopen			
	SAE J1939			

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!

Colour display of the CR0452

15258

Designation	Data
Technology	TFT
Screen diagonal	4.3" (10.9 cm)
Aspect ratio	16:9
Resolution	480 x 272 pixels
Colour depth	8 bits = 256 colours via defined colour palette ► Create image as a 256-colour file!
Background illumination	LED dimmable in 1 % steps Setting a) can be changed temporarily and b) can be preset (stored)

Operating elements of CR0452

15260

The display is fitted with the following operating elements:

- 4 function keys [F1]...[F4]
  backlit with LEDs
- 1 rocker switch
  as a combination of 4 independent keys
  backlit with LEDs
- 1 [OK] key
  backlit with LED
- 1 [ESC] key
  backlit with LED

All keys work independently of each other.
The device detects several simultaneously pressed keys and evaluates them.

Key LEDs dimmable

8369

► All operating elements are backlit with LEDs.

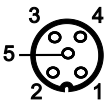
Night design of the operating elements:

- The LEDs for all operating elements can only be dimmed together:
  ▶ Flag KEY_BACKLIGHT_CTRL

Connection on the rear panel of the housing

8351

M12 connector, A-coded, for supply and CAN:

Illustration	Pin	Designation	Note
	1	n.c.	----
	2	VBB	8...32 V DC
	3	GND	terminal 31
	4	CAN_H	
	5	CAN_L	

3.2.2 Status LED

7998

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

LED colour	Flashing frequency	Description
Off	permanently off	no operating voltage
Red / green	briefly on	INIT state, reset checks
Green	5 Hz	no runtime system loaded
Green	2 Hz	RUN state: application is running
Green	permanently on	STOP state: application is stopped
Red	5 Hz	STOP state with error: application is stopped reason: undervoltage
Red	10 Hz	STOP state with error: application is stopped Cause: exceeded timeout of the application or visualisation: ▶ Delete the application! ▶ PowerOn reset ▶ Reload the application into the device
Red	permanently on	FATAL ERROR: application is stopped Cause: software watchdog has failed ▶ PowerOn reset If without success: ▶ Goto Bootloader ▶ PowerOn reset ▶ Reload the BasicSystem into the device ▶ Reload the application into the device If without success: ▶ Hardware error: send device to ifm!

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

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

15270
15271

The device is equipped with only one CAN interface.
The interface can be used with the following functions:
- RAW-CAN (Layer 2): CAN on level 2 (→ chapter ***Function elements: RAW-CAN (Layer 2)*** (→ page 54))
- CANopen master / CANopen slave (→ chapter ***Function elements: CANopen*** (→ page 80))
- CANopen network variables (via CODESYS) (→ chapter ***Network variables*** (→ page 44))
- SAE J1939 (for drive management, → chapter ***Function elements: SAE J1939*** (→ page 125))
- Bus load detection
- Error frame counter
- Download interface
- 100 % bus load without package loss

15272

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

CAN interface	CAN 1	CAN 2	CAN 3	CAN 4
Default download ID	ID 127	ID 126	ID 125	ID 124
CAN protocols	CAN Layer 2	Interface do not exist	Interface do not exist	Interface do not exist
	CANopen			
	SAE J1939			

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 no	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 38)

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

15274
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.

The visualisation pages and embedded graphics are part of the CODESYS application program.

Libraries

15409

ifm electronic offers a series of libraries (*.LIB) suitable for each device, containing the program modules for the application program. Examples:

Library	Use
ifm_CR0452_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

Detail information → *ifm libraries for the device CR0452* (→ page 45)

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] > CR0452 > [Operating instructions]
→ ecomatmobile 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!

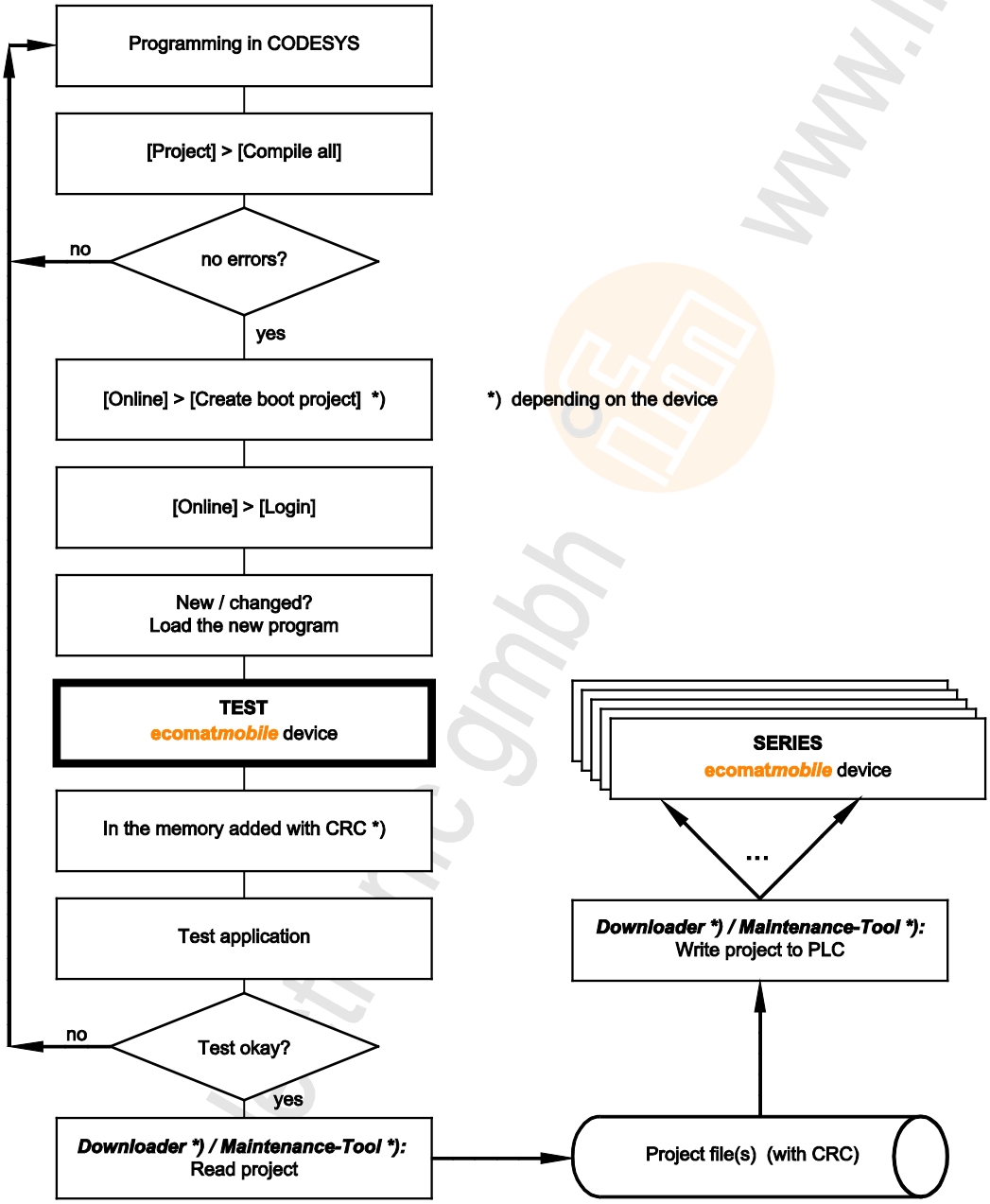
- ► When designing the application program the above-mentioned recommendations must be complied with and tested.
- ► If necessary, the cycle time must be optimised by restructuring the software and the system set-up.

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]
→ ecomatmobile 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)

1076

Premise: a valid runtime system is installed.

This state is passed through after every power on reset:

> The runtime system is initialised.

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

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

> The LED lights 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 171).

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

15277

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

If the maximum watchdog time (application program: 100 ms; visualisation: 1 200 ms) is exceeded:
> the device changes to the "Timeout Error" state
> all processes are stopped (reset)
> all outputs are switched off
> the screen goes black
> 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 unsuccessful:
- ▶ Goto Bootloader
- ▶ PowerOn Reset
- ▶ Reload the runtime system into the device
- ▶ Reload the application program into the device

If unsuccessful:
- ▶ Hardware error: return device to ifm!

Visualisation limits

8337

Embedded displays, used e.g. in ecomatmobile devices, cannot provide the full colour scope of bitmap graphics because the available power reserves are restricted. Nevertheless, the following preparations enable bitmap images in the device:

- Correct selection of the motifs,
- clever shifting of colours or
  clever compilation of a colour palette and
- the correct scaling of the bitmaps before using them in the device.

→ *Performance limits of the device* (→ page 27)

8465

Parameter	Limitation CR0451	Limitation CR0452
File type	Bitmap (*.bmp)	
File name	Only small letters, naming convention = 8.3	
Image size	320 x 240 pixels	480 x 272 pixels
Colours	8 bit = 28 colours = 256 colours can be represented	
Required memory space	≤ 76 Kbytes, depending on the image content for RLE compression	

Table: specifications for the start image

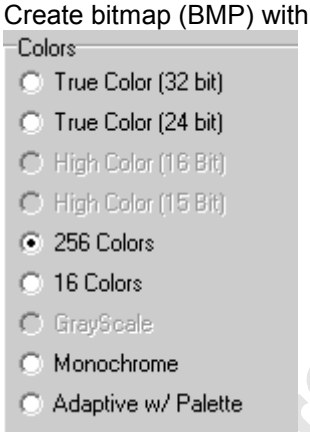
The graphics used in the project may be larger than the specified image size. In this case, however, only a (selectable) section of the image will be visible.

8464

Colours:

The device supports 2^8 (= 256) colour nuances.

► Create bitmap (BMP) with 256 colours:



Resample / scale image

3117

If an image is loaded in the device which does not meet the requirements for size or colour, it is resized before it is displayed and the colours used are "checked".

Each time the image is opened, it must be resampled. This often leads to much longer times to change from one image to the other. Corrective measures:

► First carry out all transformations of the bitmap or the image in an image processing program on your computer.
Only for BasicDisplay: The colour palette is adapted when the image is integrated into the project by CODESYS. On the device itself no adaptations will be made (size, scaling, colour).

► Only save the suitably transformed images in the visualisation of the device.

Limitations for visualisations

8319

Designation	Limitation
Length of character strings	≤ 80 characters
Length of path names	not relevant
Number of visualisation pages	≤ 15
Number of graphical objects per visualisation page	≤ 20
Number of bitmaps per project	≤ 256 depending on the size and the available flash memory for the application
Number of character sets per project	Character sets are permanently stored and cannot be changed.
Number of POUs [1]) per project	≤ 512

[1]) POU (**P**rogram **O**rganization **U**nit) = function, function block or program block

Because of the limited memory:
- avoid grouping of elements
- avoid visualisation as a master slide
- avoid visualisation of master background slides

CODESYS visualisation elements

453

Bitmap graphics (BMP) → chapter ***Visualisation limits*** (→ page 28)

Not all CODESYS functions can be executed successfully on this device:

Visualisation element	Functional safety for the PDM	
Polyline	o	A polyline is to consist of maximum 5 anchor points; not scalable A polyline is not to enclose any area.
Curve	--	Not supported
Rectangle	+	No problems known
Rounded rectangle	--	Not supported
Circle, ellipse	+	No problems known
Polygon	o	A polygon is to consist of maximum 10 anchor points; not scalable
Pie chart	--	Not supported
Visualisation	--	Not supported
Button	--	Not supported
Table	--	Not supported
Scroll bar	--	Not supported
Trend curve	--	Not supported
Alarm table	--	Not supported
Scales	+	Create scale as BMP file
Pointer instrument	+	Represent pointer instrument as BMP file with a superposed CoDeSys polygon
Bar graph	+	Create scale as BMP file Represent value as a superposed CoDeSys rectangle
Histogram	+	Create scale as BMP file Represent value as a superposed CoDeSys rectangle
Graphic file	+	Up to 256 per project possible • BMP [1]) • BMP RLE compressed • TIFF [1]) • JPEG [1])
Graphics scaling mode	o	Supported for circle, ellipse, rectangle, line, polyline, polygon
ActiveX element	--	Not supported
Pointer diagram	--	Not supported
Edit tools	--	Not supported

+ can be used without problem
o can be used with restrictions
-- cannot be used

[1]) During the integration into the project the file is converted into an RLE compressed bitmap. From the CODESYS version 2.3.9.24 an additional dialogue opens when an image is integrated. Using this dialogue the file is adapted to the Colour palette used for the device. During the colour conversion you can select between 'most similar colour' (deactivate [Dithering]) and 'Dithering'.

Drawing area:

- The left upper corner marks the home position (0,0) of the virtual and physical drawing area.
- Virtual drawing area = 2,560 x 1,536 pixels
  (enlarges the physical drawing area)
- Elements in the virtual drawing area are not calculated.

Drawing area

15987

- The left upper corner marks the home position (0,0) of the virtual and physical drawing area.
- Virtual drawing area (X/Y coordinates) = -32768...+32767
  (enlarges the physical drawing area)
- All objects including their outer dimensions must be within the borders of the virtual drawing area even after scaling or shifting!
  Otherwise the visualisation will not be correct any more.
- Elements in the virtual drawing area are not calculated.

Texts

8436

- The smallest font size which is clearly visible on the device is 11 point.
- Permissible fonts:
  - Arial (standard)
  - Lucida Console
- Permissible font size [Pixel] and font weight:
  - Arial: 11 (standard), 16, 24, 32 (all only normal)
  - Lucida Console: 16, 24, 48*) (all only normal)

  *) Lucida Console in the font size 48 only has the following characters:
  - numbers 0 to 9
  - special characters + - . : %
  - space.
- Permissible effects:
  - none (standard)
- The following text scripts are ignored:
  - Western
  - Hebrew
  - Arabic
  - Greek
  - Turkish
  - Baltic
  - Central European
  - Cyrillic
  - Vietnamese

Movement of elements

7392

Image and text elements can be moved on the display in a defined manner.

Element movement	Description
Rotate	Rotate the element around a defined pivot point Indicate the angle of rotation ▪ angle of rotation in [degree] ▪ positive value = rotation clockwise ▪ negative value = rotation anticlockwise
Shift	Shifting of the element: ▪ horizontal ▪ vertical ▪ only within the drawing area ▪ max. until leaving the drawing area
Relative shifting of • rectangle • ellipse / circle	Each edge of the element can be shifted by a specified number of pixels via an INT type variable: ▪ basic position of the 4 edges = zero ▪ new value shifts this edge by the specified value Shift direction for value > 0: ▪ horizontal edge down ▪ vertical edge to the right Shift direction for value < 0: ▪ horizontal edge up ▪ vertical edge to the left

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 64)
- ***CAN_RX_RANGE_FIFO*** (→ page 68)

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

1016

The device configurations described in the corresponding installation instructions or in the ***Annex*** (→ page 187) 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.

4.1 Set up the runtime system

Contents

14091

4.1.1 Reinstall the runtime system

14635
8486

On delivery of the **ecomatmobile** 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_CR0452_Vxxyyzz.RESX`),
- PLC configuration (`ifm_CR0452_Vxx.CFG`),
- device library (`ifm_CR0452_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. "`CR0452`") 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 **ecomatmobile** 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!

3084

When the operating system software or the CODESYS runtime system is considerably improved, ifm releases a new version. The versions are numbered consecutively (V01, V02, V03, ...).

Please see the respective documentation for the new functions of the new software version. Note whether special requirements for the hardware version are specified in the documentation.

If you have a device with an older version and if the conditions for the hardware and your project are OK, you can update your device to the new software version.

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 163)	Delivers information about the system software of the device: • software name, • software version, • build number, • build date

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_CR0452_Vxxyyzz.LIB` manually!

NOTE

The software versions suitable for the selected target must always be used:
- runtime system (`ifm_CR0452_Vxxyyzz.RESX`),
- PLC configuration (`ifm_CR0452_Vxx.CFG`),
- device library (`ifm_CR0452_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. "`CR0452`") 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

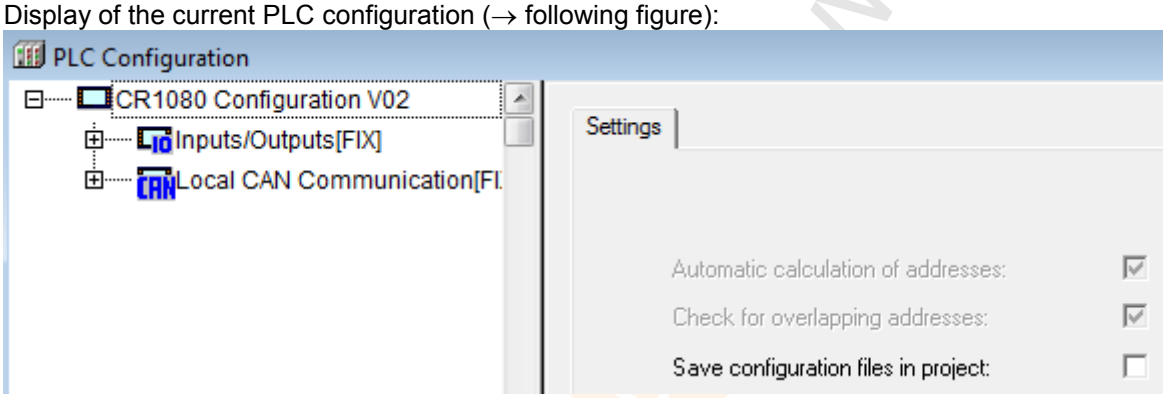
The PLC configuration is automatically loaded with the target system. The PLC configuration maps the contents of the file CR0452.cfg in CODESYS. Like this, the programmer has easy access to predefined system and error flags, inputs and outputs as well as to the CAN interfaces of the device.

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

► Click on the tab [Resources] in CoDeSys:

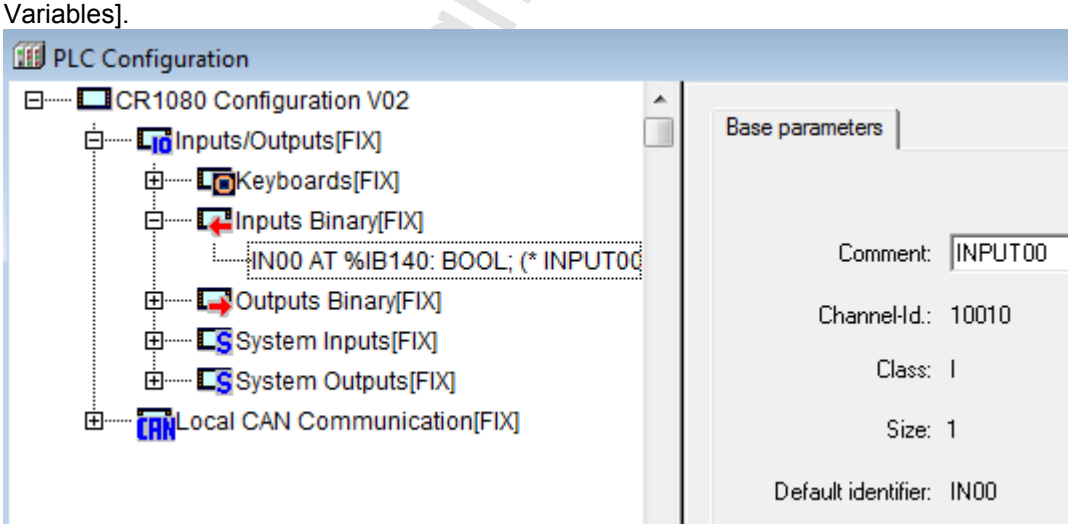
► Double-click on [PLC Configuration] in the left column.

> 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)

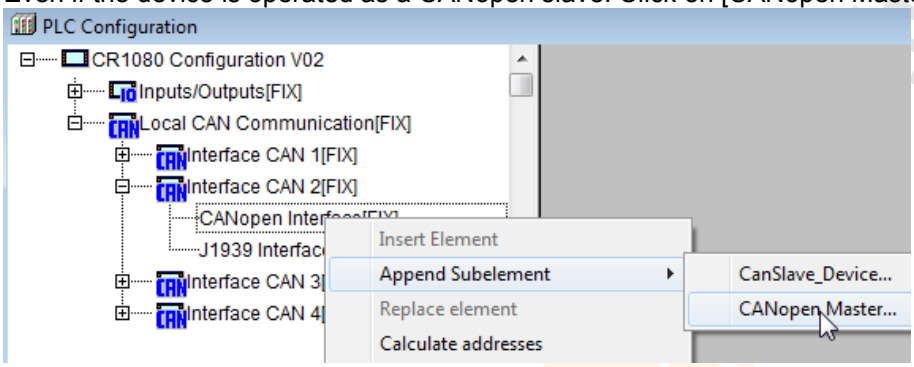
10080

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...]:

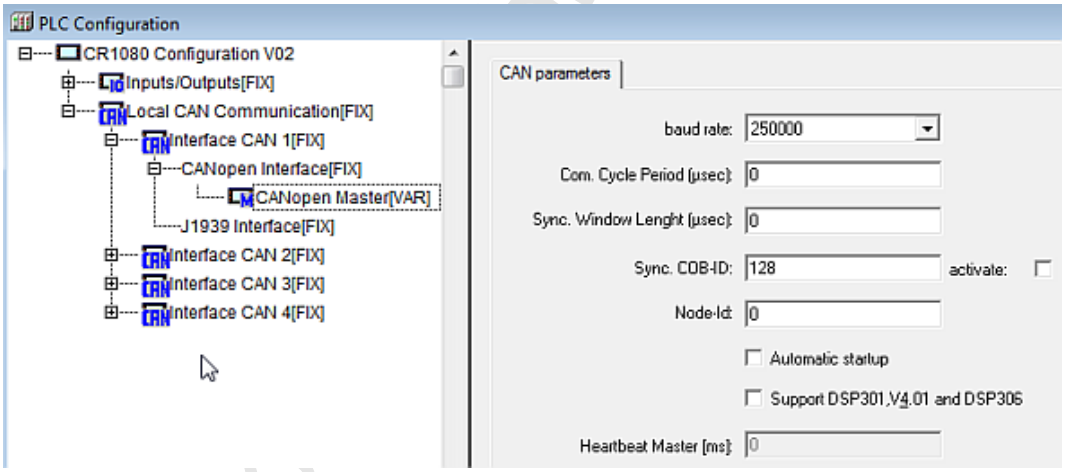


Info

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 81) 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 187)) have defined addresses which cannot be shifted.

4.4 Variables

Contents

3130

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

14486

The device supports the following types of variables:

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

→ CODESYS programming manual
→ ecomatmobile DVD "Software, tools and documentation"

4.4.1 Retain variables

8672

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

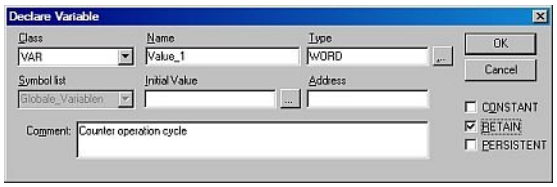
14166

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

4.4.2 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 CR0452

Contents

14235

Legend for ..._Vxxyyzz.LIB:

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

Here you will find a list of the ifm function elements matching this device, sorted according to the CODESYS libraries.

5.1.1 Required libraries

15300

If you do not want to base the initial programming of this device on an ifm template, you should be sure to integrate at least the following libraries into your project:

`ifm_CR0452_Vxxyyzz.LIB`	Device library
`ifm_CR0452_Init_Vxxyyzz.LIB`	Initialises the device screen

Libraries required for network variables

15304

If you want to work with network variables, you will need the following libraries in addition:

`ifm_NetVarLib_NT_Vxxyyzz.LIB`	Support of network variables (ifm library)
`3S_CANopenNetVar.LIB`	Support of network variables (3S library)

5.1.2 Library ifm_CR0452_Vxxyyzz.LIB

15284

This is the device library.

This ifm library contains the following function blocks:

Function element	Short description
FLASH_INFO (→ page 158)	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)
FLASH_READ (→ page 159)	Transfers different data types directly from the flash memory to the RAM
GET_APP_INFO (→ page 160)	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 161)	Delivers information about the device hardware: • ifm article number (e.g. CR0403), • article designation, • unambiguous serial number, • hardware revision, • production date
GET_IDENTITY (→ page 162)	Reads the identification of the application stored in the device (has previously been saved by means of ***SET_IDENTITY*** (→ page 170))
GET_SW_INFO (→ page 163)	Delivers information about the system software of the device: • software name, • software version, • build number, • build date
GET_SW_VERSION (→ page 164)	Delivers information about the software versions stored in the device: • BasicSystem version, • bootloader version, • SIS version, • application program version, • user flash version
MEM_ERROR (→ page 165)	Signals errors in some parameters or in the memory (Re-)initialisation of system resources
MEMCPY (→ page 166)	Writes and reads different data types directly in the memory
OHC (→ page 168)	Adjustable operating hours counter (0...3)
SET_IDENTITY (→ page 170)	Sets an application-specific program identification
SET_LED (→ page 171)	Change the frequency and color of the status LED in the application program
SET_PASSWORD (→ page 173)	Sets a user password for access control to program and memory upload
TIMER_READ_US (→ page 174)	Reads out the current system time in [µs] Max. value = 1h 11min 34s 967ms 295µs

5.1.3 Library ifm_CR0452_Init_Vxxyyzz.LIB

15286

This ifm library contains the following function blocks:

Function element	Short description
BASICDISPLAY_INIT (→ page 182)	Initialises the screen of the BasicDisplay in the first PLC cycle

5.1.4 Library ifm_PDMsmart_util_Vxxyyzz.LIB

15289

This ifm library contains the following function blocks:

Function element	Short description
GET_TEXT_FROM_FLASH (→ page 176)	Reads texts of the type STRING from the flash memory via ***FLASH_READ*** (→ page 159)
NORM_DINT (→ page 178)	Normalises a value [DINT] within defined limits to a value with new limits
NORM_REAL (→ page 179)	Normalises a value [REAL] within defined limits to a value with new limits
PDM_PAGECONTROL (→ page 183)	Controls invoking of certain visualisation pages
TOGGLE (→ page 180)	Setting and resetting of a Boolean variable with only one input bit

5.1.5 Library ifm_RAWCan_NT_Vxxyyzz.LIB

14715

This ifm library contains the following function blocks:

Function element	Short description
CAN_ENABLE (→ page 55)	Initialises the indicated CAN interface Configures the CAN baud rate
CAN_RECOVER (→ page 56)	Activate / deactivate the automatic bus off handling Restart the CAN interface in case of bus off
CAN_REMOTE_REQUEST (→ page 77)	Send a corresponding request and return the response of the other device as a result
CAN_REMOTE_RESPONSE (→ page 78)	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 61)	Configures a data receive object and reads out the receive buffer of the data object
CAN_RX_ENH (→ page 62)	• 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 64)	• 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 66)	• 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 68)	• 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 57)	= Set CAN download ID Sets the download identifier for the CAN interface
CAN_STATUS (→ page 58)	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 71)	Transfers a CAN data object (message) to the configured CAN interface for transmission at each call
CAN_TX_ENH (→ page 72)	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 74)	Cyclically transfers a CAN data object (message) to the configured CAN interface for transmission CAN-specific characteristics can be set

5.1.6 Library ifm_CANopen_NT_Vxxyyzz.LIB

14914

This ifm library contains the following function blocks:

Function element	Short description
CANOPEN_ENABLE (→ page 81)	Initialises the indicated CANopen master interface Configures the CAN baud rate
CANOPEN_GETBUFFERFLAGS (→ page 83)	= CANopen get buffer flags Provides information on the buffer flags The flags can be reset via the optional inputs.
CANOPEN_GETEMCYMESSAGES (→ page 120)	= 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 122)	= 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 116)	= 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 117)	= 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 90)	= CANopen slave get network management state Signals the network operating status of the node
CANOPEN_GETODCHANGEDFLAG (→ page 94)	= Get object directory changed flag Reports any change of value for a particular object directory entry
CANOPEN_GETSTATE (→ page 85)	= CANopen set state Request the parameters of the master, a slave device or a specific node in the network
CANOPEN_GETSYNCSTATE (→ page 112)	= 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 91)	= 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 95)	= CANopen read object directory Reads configuration data from the object directory of the device
CANOPEN_SDOREAD (→ page 99)	= CANopen read SDO Reads an "Expedited SDO" = Expedited Service Data Object
CANOPEN_SDOREADBLOCK (→ page 101)	= 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 103)	= CANopen read SDO multi Reads the indicated entry in the object directory of a node in the network
CANOPEN_SDOWRITE (→ page 105)	= SDO write Writes an "Expedited SDO" = Expedited Service Data Object
CANOPEN_SDOWRITEBLOCK (→ page 107)	= 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 109)	= CANopen write SDO multi Writes in the indicated entry in the object directory of a node in the network
CANOPEN_SENDEMCYMESSAGE (→ page 123)	= 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 87)	= CANopen set state Set the parameters of the master, a slave device or a specific node in the network
CANOPEN_SETSYNCSTATE (→ page 114)	= CANopen set SYNC state Switch the SYNC functionality on and off
CANOPEN_WRITEOBJECTDICT (→ page 96)	= CANopen write object directory Writes configuration data into the object directory of the device

5.1.7 Library ifm_J1939_NT_Vxxyyzz.LIB

14912

This ifm library contains the following function blocks:

Function element	Short description
J1939_DM1RX (→ page 150)	J1939 Diagnostic Message 1 RX Receives diagnostic messages DM1 or DM2 from other ECUs
J1939_DM1TX (→ page 152)	J1939 Diagnostic Message 1 TX Transmit an active error message to the CAN stack
J1939_DM1TX_CFG (→ page 155)	J1939 Diagnostic Message 1 TX configurable CAN stack does not send cyclic DM1 "zero active faults" messages
J1939_DM3TX (→ page 156)	J1939 Diagnostic Message 3 TX Deletes inactive DTCs (DM2) on a device
J1939_ENABLE (→ page 126)	Initialises the J1939 stack
J1939_GETDABYNAME (→ page 128)	= Get destination arbitrary name Determine the target address of one or several participants by means of the name information
J1939_NAME (→ page 130)	Give the device a name for identification in the network
J1939_RX (→ page 137)	Receives a single frame message Shows the message last read on the CAN bus
J1939_RX_FIFO (→ page 138)	= J1939 RX with FIFO Receives all specific messages and successively reads them from a FiFo
J1939_RX_MULTI (→ page 140)	= J1939 RX multiframe message Receives multiframe messages
J1939_SPEC_REQ (→ page 134)	= J1939 specific request Requests and receives a specific message from another controller
J1939_SPEC_REQ_MULTI (→ page 135)	= J1939 specific request multiframe message Requests and receives a specific multiframe message from another controller
J1939_STATUS (→ page 132)	Shows relevant information on the J1939 stack
J1939_TX (→ page 142)	Sends individual single frame messages
J1939_TX_ENH (→ page 143)	= J1939 TX enhanced Sends individual single frame messages Can also be set: transmission priority, data length
J1939_TX_ENH_CYCLIC (→ page 145)	= J1939 TX enhanced cyclic Cyclically sends single frame messages Can also be set: transmission priority, data length, period
J1939_TX_ENH_MULTI (→ page 147)	= J1939 TX enhanced Multiframe Message Sends individual multiframe messages

5.2 ifm function elements for the device CR0452

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

8354
7556

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

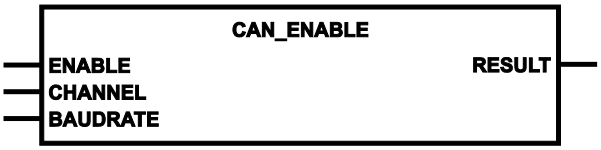
CAN_ENABLE

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:

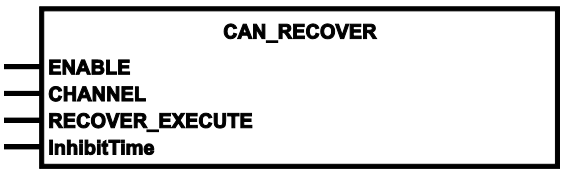
Value dec	Value hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
8	08	function block is active
9	09	CAN is not active
242	F2	Error: setting is not possible

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	Value 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

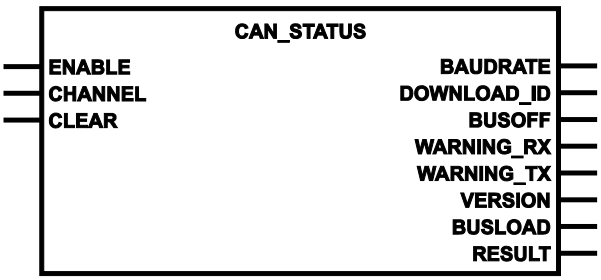
CAN_STATUS

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	Value 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^{11} IDs): 0...2 047 = 0x0000 0000...0x0000 07FF extended Frame (2^{29} 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	Value 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

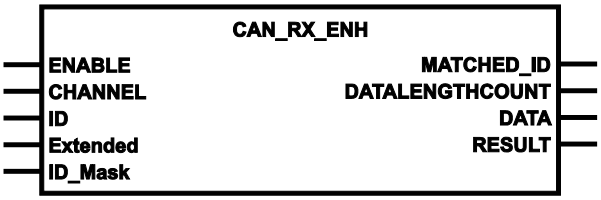
7606

= 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 61)):
- 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:

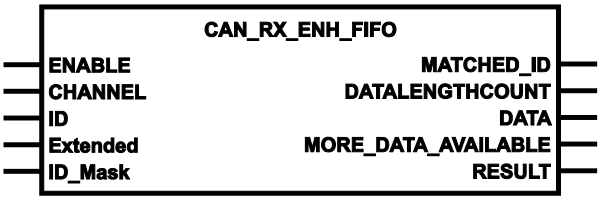
Value dec	Value hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
5	05	FB is being processed – still receiving
9	09	CAN is not active
242	F2	Error: setting is not possible

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:



Description

7616

In addition, CAN_RX_ENH_FIFO provides a FiFo for the received data (as opposed to ***CAN_RX_ENH*** (→ page 62)). 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.

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:

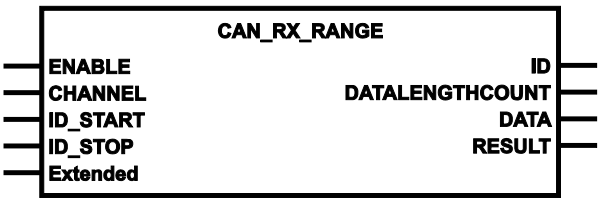
Value dec	Value hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
5	05	FB is being processed – still receiving
9	09	CAN is not active
242	F2	Error: setting is not possible
250	FA	Error: FiFo is full – data was lost

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^{11} IDs): 0...2 047 = 0x0000 0000...0x0000 07FF extended Frame (2^{29} 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:

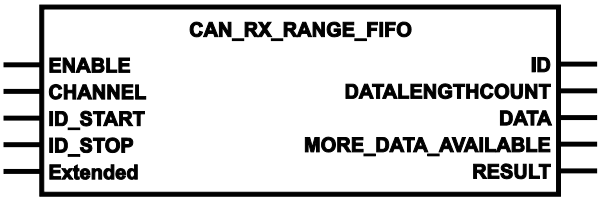
Value dec	Value hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
5	05	FB is being processed – still receiving
9	09	CAN is not active
242	F2	Error: setting is not possible

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 66).
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^{11} IDs): 0...2 047 = 0x0000 0000...0x0000 07FF extended Frame (2^{29} 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	Value 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

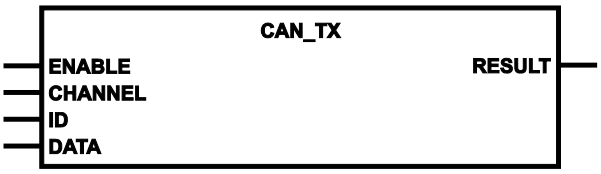
15055

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^{11} IDs): 0...2 047 = 0x0000 0000...0x0000 07FF extended Frame (2^{29} 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	Value 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

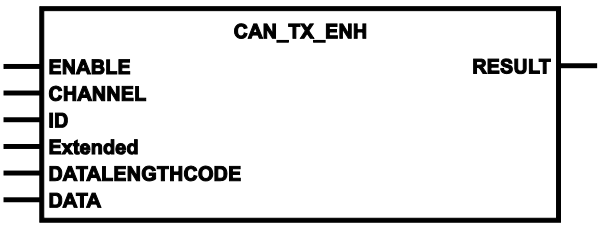
7558

= 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 71) 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:

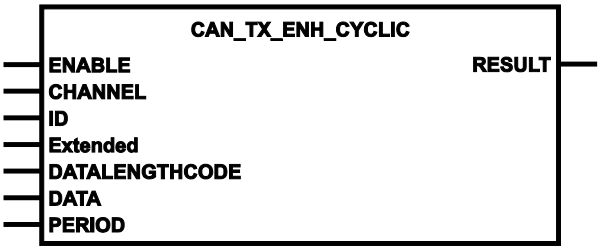
Value dec	Value hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
242	F2	Error: setting is not possible
250	FA	Error: FiFo is full – data was lost

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 72).

► 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	Value 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

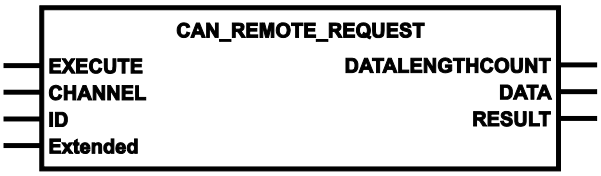
CAN_REMOTE_REQUEST

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	Value 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

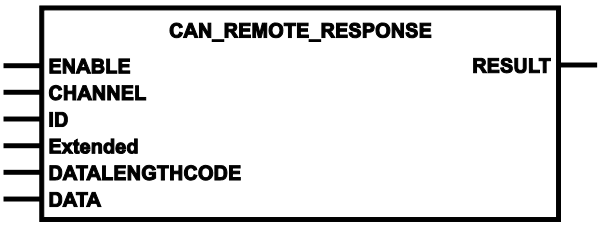
CAN_REMOTE_RESPONSE

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	Value 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 81) 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 55) 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	Value 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

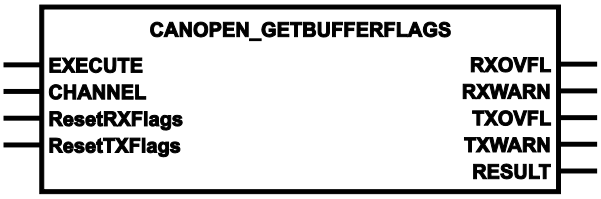
7890

= 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	Value 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

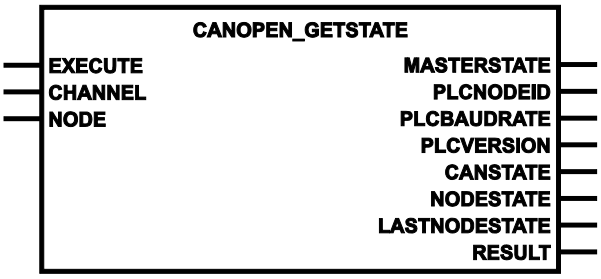
7865

= 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

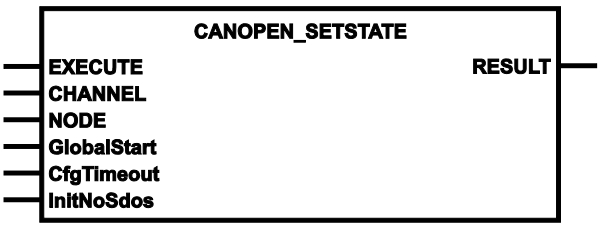
7858

= 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 91). 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	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

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	Value 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 = 1...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
043	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	Value 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

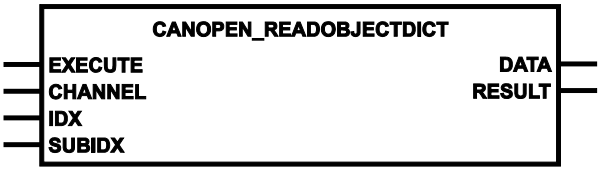
7933

= 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	Value 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

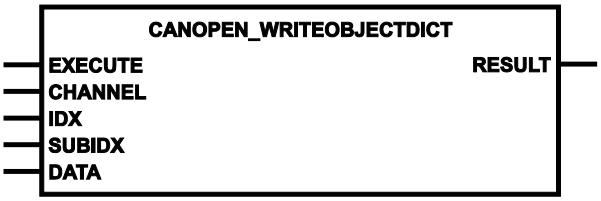
7940

= 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	Value 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

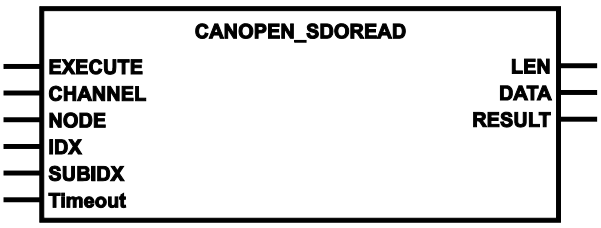
7791

= 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
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	buffer overflow – too many data bytes were received

CANOPEN_SDOREADBLOCK

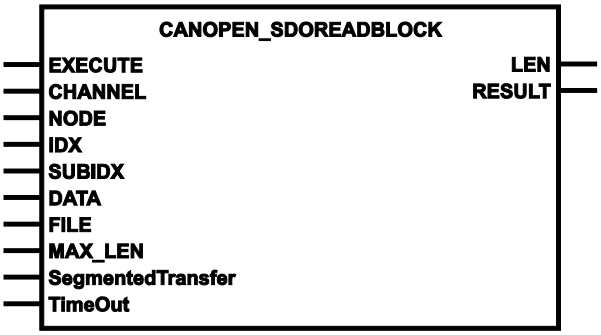
14942

= 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 ℹ 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 ℹ 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 ℹ 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	Value 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

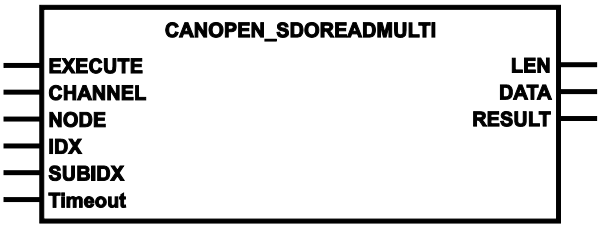
7806

= 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

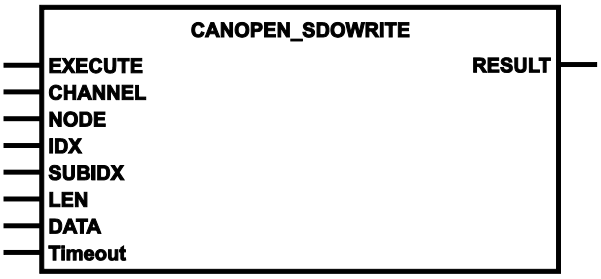
7825

= 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	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

CANOPEN_SDOWRITEBLOCK

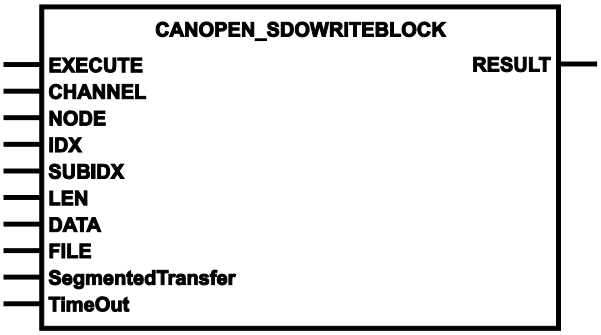
14961

= 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	Value 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

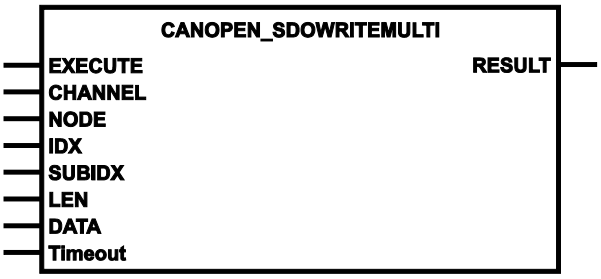
7832

= 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	Value 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	Value 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 33)
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	FB is active – not yet processed
242	F2	Error: setting is not possible

CANOPEN_GETGUARDHBSTATSLV

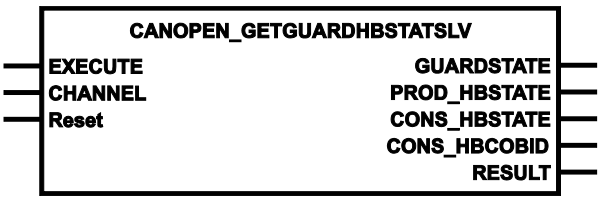
7902

= 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	Value 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	Value 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	Value 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

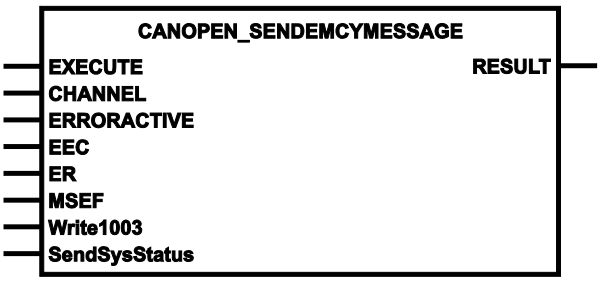
7908

= 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	Value 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

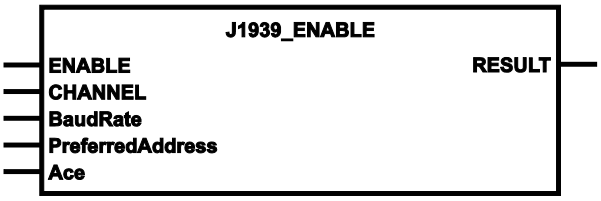
15077

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	Value 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

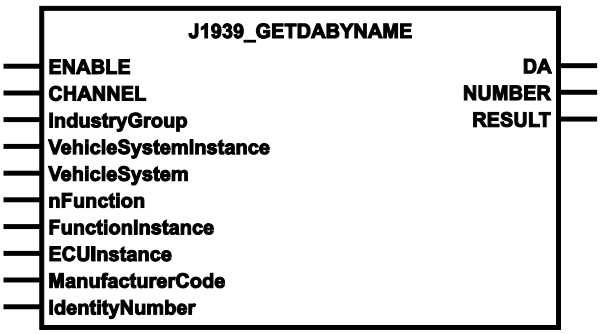
7664

= 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)) 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:

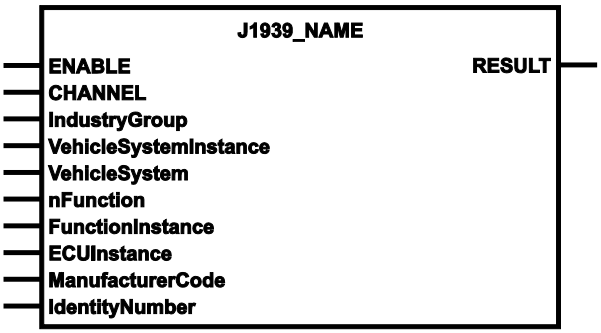
Value dec	Value hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
8	08	function block is active
242	F2	Error: setting is not possible

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)) 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	Value 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

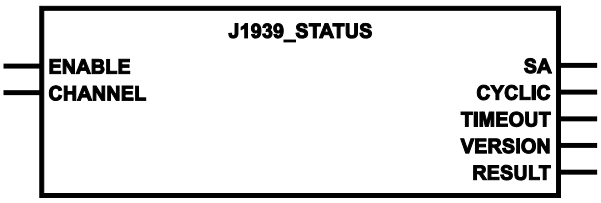
J1939_STATUS

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	Value 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

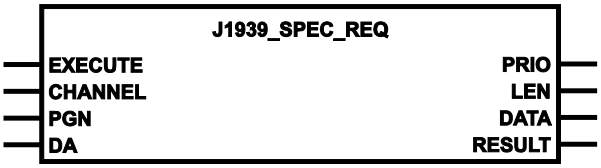
15023

= 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 allowed = 1...??? = 0x00000001...0x???
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

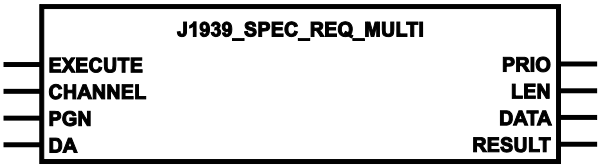
15033

= 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 allowed = 1...??? = 0x00000001...0x???
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

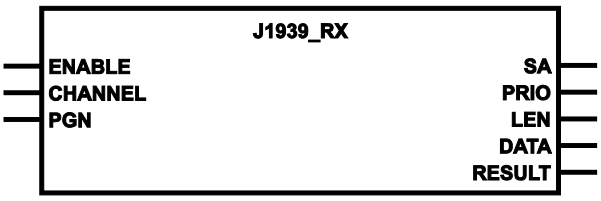
15081

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

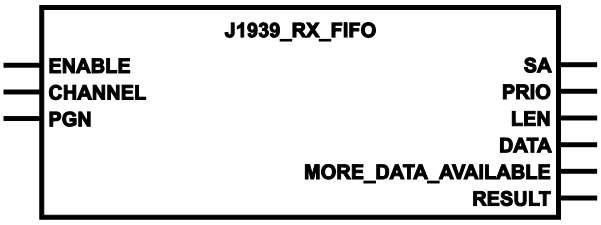
7732

= 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:

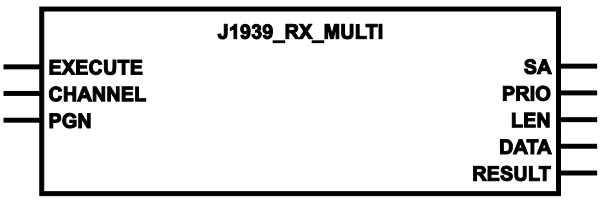
Value dec	Value hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
5	05	FB is active – no data received yet
242	F2	Error: setting is not possible
250	FA	Error: FiFo is full – data was lost

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	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
242	F2	Error: setting is not possible

Function elements: transmit SAE J1939

Contents

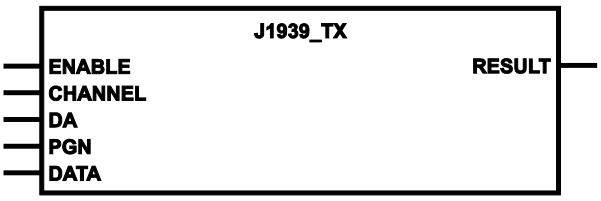
15083

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:

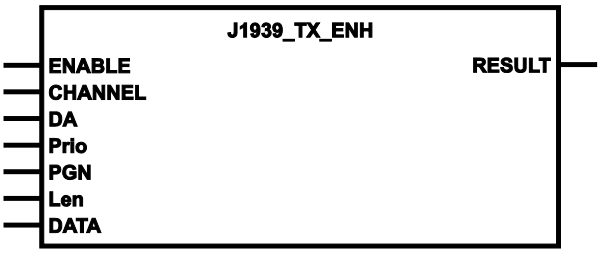
Value dec	hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
242	F2	Error: setting is not possible
250	FA	Error: FiFo is full – data was lost

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 147).

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:

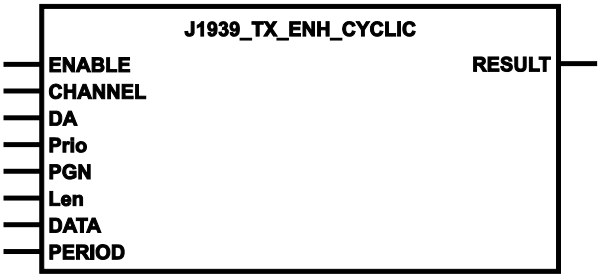
Value dec	Value hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
242	F2	Error: setting is not possible
250	FA	Error: FiFo is full – data was lost

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 143).

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

J1939_TX_ENH_MULTI

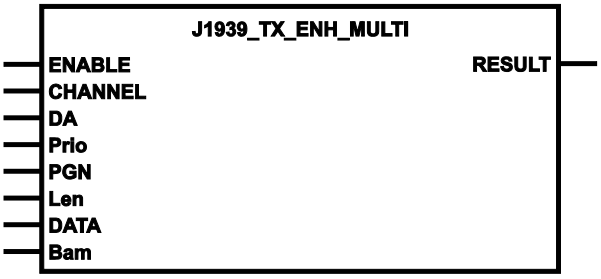
7699

= 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 143). 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	Value 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

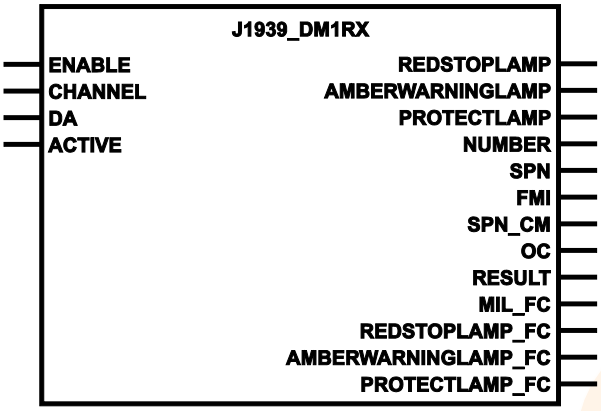
15085

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
FMI	BYTE	**F**ailure **M**ode **I**ndicator permissible values = 0...31 = 0x00...0x1F
SPN_CM	BOOL	conversion method
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	Value 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

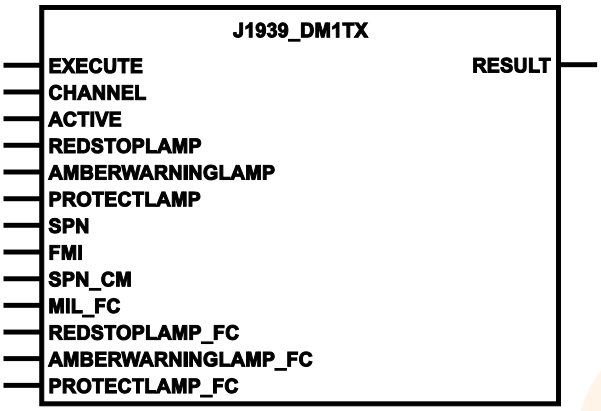
14993

= 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
FMI	BYTE	**F**ailure **M**ode **I**ndicator permissible values = 0...31 = 0x00...0x1F
SPN_CM	BOOL	conversion method
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	Value 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

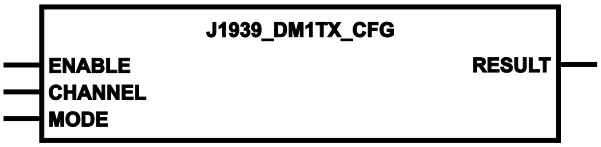
15424

= 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 126) 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 152)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 152) 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	Value hex	Description
0	00	FB is inactive
1	01	function block execution completed without error
242	F2	Error: setting is not possible

J1939_DM3TX

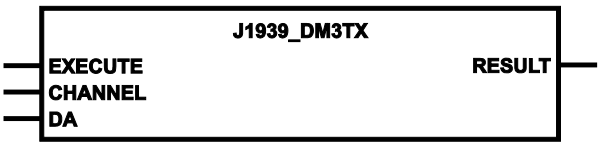
15002

= 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	Value 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: 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

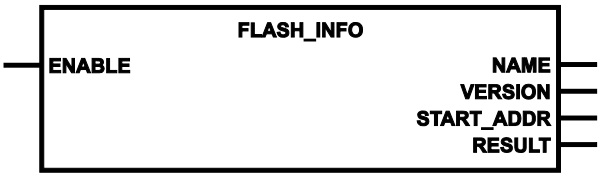
FLASH_INFO

11580

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0452_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	Value hex	Description
0	00	FB is inactive
1	01	FB execution completed without error – data is valid
157	9D	Software header invalid (CRC error)

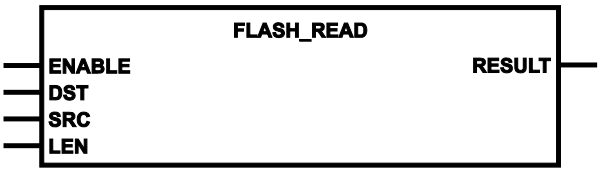
FLASH_READ

8147

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0452_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	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

GET_APP_INFO

11581

= get application information

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0452_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

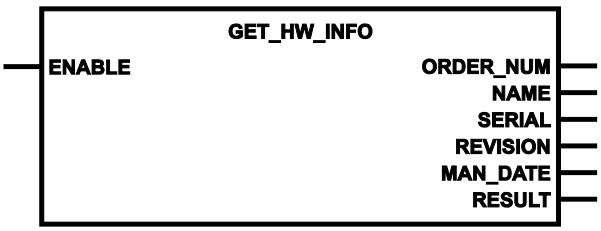
11582

= get hardware information

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0452_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	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_CR0452_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 170)).

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	Value 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

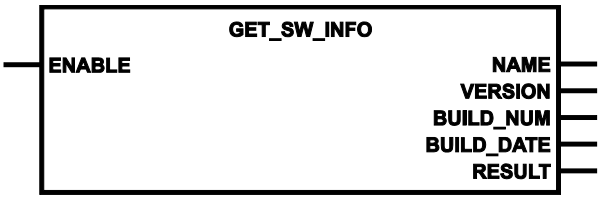
11583

= get software information

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0452_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

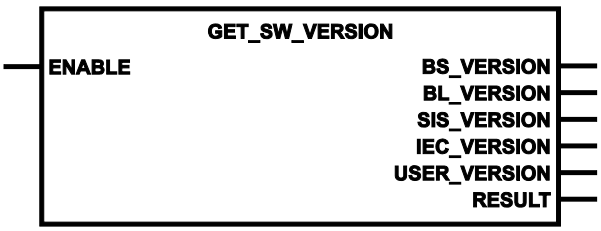
14763

= get software version

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0452_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

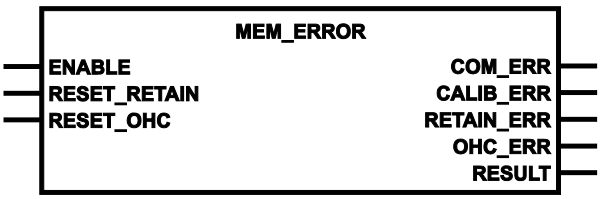
14770

= Memory Error

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0452_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

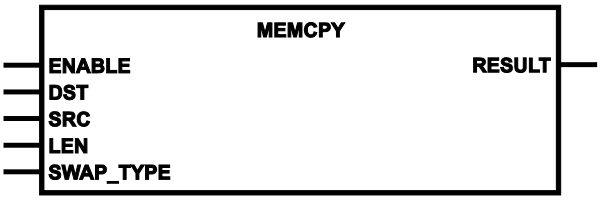
8160

= memory copy

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0452_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

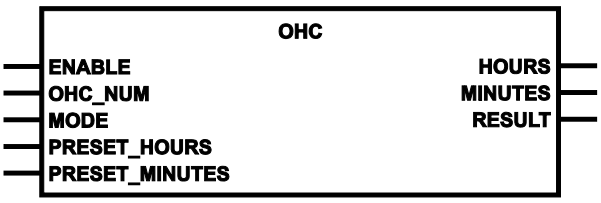
14777

= Operating Hours Counter

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0452_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 165), 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	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	Value 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_CR0452_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 162)

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

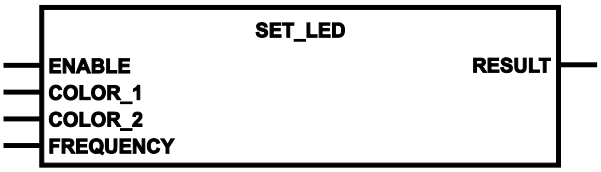
SET_LED

8052

Unit type = function block (FB)

Unit is contained in the library `ifm_CR0452_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 15)).

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	Value 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_CR0452_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	Value 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_CR0452_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

5.2.6 Function elements: graphics

Contents

15294

Function elements: graphics help

Contents

15296

Here you will find further ifm function elements that will support you while programming the graphical interface.

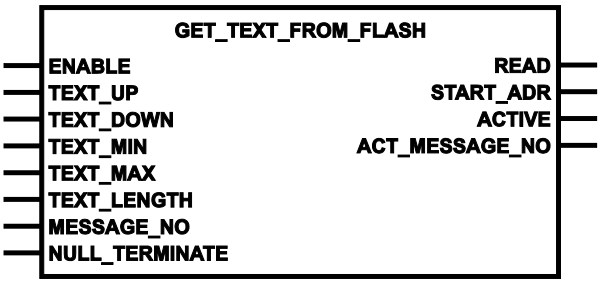
GET_TEXT_FROM_FLASH

3196

Unit type = function block (FB)

Unit is contained in the library `ifm_PDMsmart_UTIL_Vxxyyzz.Lib`

Symbol in CODESYS:



Description

11651

GET_TEXT_FROM_FLASH controls ***FLASH_READ*** (→ page 159) to directly read text of type STRING.

As opposed to PDM360 and PDM360compact, PDM360smart has no file system. Therefore flash memories or FLASH memories are recommended to store text messages. To read these memory areas FLASH_READ is needed.

To ensure reading of one or several texts, the start address of the text in the memory must be calculated. This calculation and setting/resetting of the ENABLE input are made in GET_TEXT_FROM_FLASH.

The texts in the memory must be organised according to the rules below:

Text length
The text length should be the same for all texts and is limited to max. 30 characters because of the display size of the device.

Text creation
The texts should be created using a spreadsheet program (e.g. Excel) and then saved in CSV format. This CSV file can be directly loaded to the requested memory area using the **ifm** maintenance tool. → **ecomat*mobile*** DVD "Software, tools and documentation".

A STRING is automatically terminated with a NULL byte by the programming system. Therefore a text of 30 characters uses 31 bytes in the memory. The FB takes this into account for the calculation. From the indicated flash memory of the device (→ chapter ***Available memory*** (→ page 12)) subtract 128 bytes for the header. Thus with text length of 30 characters 65 408 / 31 = 2 109 texts can be saved in the flash memory.

Parameters of the inputs

3302

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
TEXT_UP	BOOL	edge FALSE → TRUE: read next text
TEXT_DOWN	BOOL	edge FALSE → TRUE: read previous text
TEXT_MIN	WORD	lower limit for MESSAGE_NO
TEXT_MAX	WORD	upper limit for MESSAGE_NO
TEXT_LENGTH	BYTE	text length
MESSAGE_NO	WORD	text number
NULL_TERMINATE	BOOL	TRUE: string has null termination FALSE: string has no null termination

Parameters of the outputs

15596

Parameter	Data type	Description
READ	BOOL	Read command ► Set this signal to the input ENABLE of the FB FLASH_READ!
START_ADR	WORD	Calculated start address ► Set this signal to the input SCR of the FB FLASH_READ!
ACTIV	BOOL	TRUE: FB is active (if input ENABLE = 1)
ACT_MESSAGE_NO	WORD	Current text number

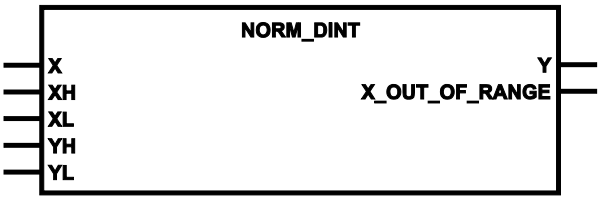
NORM_DINT

13240

Unit type = function block (FB)

Unit is contained in the library `ifm_PDMsmart_UTIL_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

3307

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

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

NOTE

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

Parameters of the inputs

3308

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

Parameters of the outputs

3309

Parameter	Data type	Description
Y	DINT	normalised value
X_OUT_OF_RANGE	BOOL	input value X is outside the defined value range XL/XH

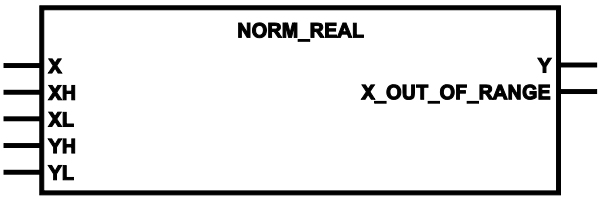
NORM_REAL

13244

Unit type = function block (FB)

Unit is contained in the library `ifm_PDMsmart_UTIL_Vxxyyzz.LIB`

Symbol in CODESYS:



Description

3310

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

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

NOTE

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

Parameters of the inputs

3311

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

Parameters of the outputs

3312

Parameter	Data type	Description
Y	REAL	normalised value
X_OUT_OF_RANGE	BOOL	input value X is outside the defined value range XL/XH

TOGGLE

13248

Unit type = function block (FB)

Unit is contained in the library `ifm_PDMsmart_UTIL_Vxxyyzz.LIB`

Symbol in CODESYS:

Description

3304

TOGGLE enables the setting and resetting of a Boolean variable via only one input bit.

The first rising edge on the input IN sets the output OUT to 'TRUE'.
The next rising edge resets the output back to 'FALSE'.
etc.

Parameters of the inputs

3305

Parameter	Data type	Description
IN	BOOL	edge FALSE → TRUE: setting / resetting of the output

Parameters of the outputs

3306

Parameter	Data type	Description
OUT	BOOL	1st edge on IN ⇒ TRUE 2nd edge on IN ⇒ FALSE 3rd edge on IN ⇒ TRUE ...

Function elements: graphical visualisation

Contents

15298

Here you will find ifm function elements for the following purposes:
- initialise device screen
- invoke visualisation pages

BASICDISPLAY_INIT

9310

Unit type = function (FUN) of type BOOL
Unit is contained in the library `ifm_CRRnnnn_Init_Vxxyyzz.LIB`

Symbol in CODESYS:

BASICDISPLAY_INIT

Description

9312

The function BASICDISPLAY_INIT initialises the screen of the BasicDisplay in the first PLC cycle. Without this initialisation the screen remains dark.

The function requires no parameter setting.

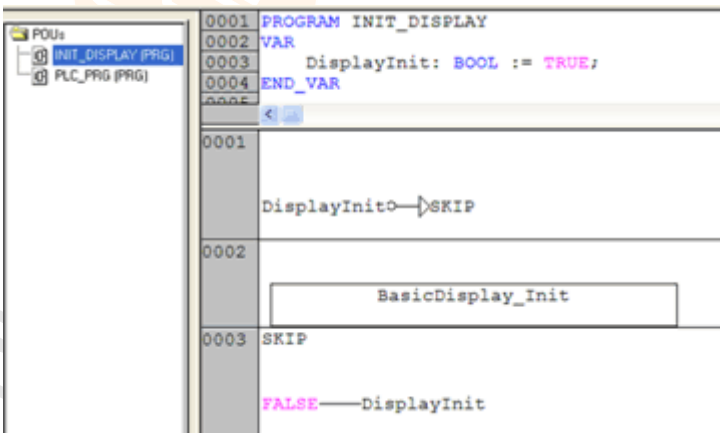
► Call the function only in the first PLC cycle!
Then skip the call.
→ the following example

Example: BasicDisplay_Init

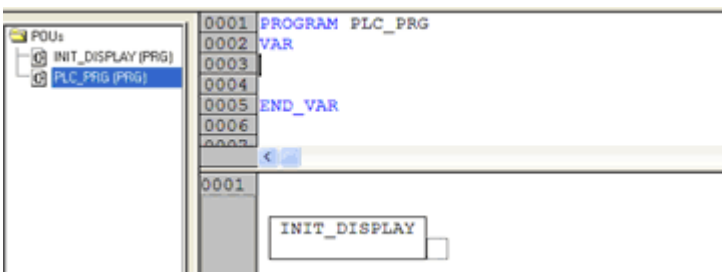
9314

► Generate the program (PRG) INIT_DISPLAY.

► Call the function (FUN) BASICDISPLAY_INIT so that it is only executed in the first PLC cycle.



► Call the program INIT_DISPLAY in the program PLC_PRG (together with the other initialisations, if any).



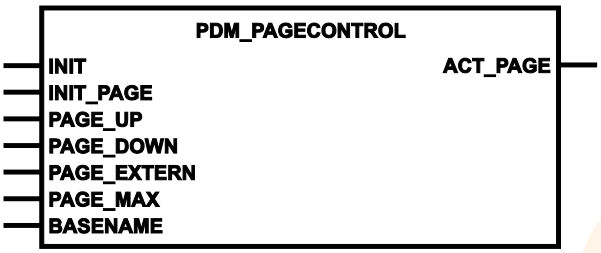
PDM_PAGECONTROL

3186

Unit type = program (PRG)

Unit is contained in the library	available for the following devices:
`ifm_PDM_UTIL_Vxxyyzz.LIB`	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
`ifm_PDMng_UTIL_Vxxyyzz.LIB`	• PDM360 NG: CR108n, CR120n
`ifm_PDMsmart_UTIL_Vxxyyzz.LIB`	• BasicDisplay: CR045n • PDM360smart: CR1070, CR1071

Symbol in CODESYS:



Description

3294

PDM_PAGECONTROL controls the opening of certain visualisation pages. In CoDeSys the visualisation pages are opened and feedback is given via the system variable CurrentVisu (type STRING[40]).

With this program it is possible to open a selected visualisati→ page or to scroll through the visualisations step by step.

Optimum use of the program is ensured when all visualisation names correspond to the same pattern, i.e. a combination of a basename followed by a 5-digit number (library version V04.00.07 or higher; before: 3-digit *)).

Example BASENAME = `PAGE`:
Visualisation name = `PAGE00001`, `PAGE00002`, `PAGE00003`, etc.

For the basename 1...35 capital letters (no special characters) are allowed. The visualisations should be numbered consecutively. The program creates the final visualisation name from the parameter BASENAME and the number or reads the number from the current visualisation name and provides it in the output parameter ACT_PAGE.

Instead of naming the visualisations with basename and consecutive number every visualisation can also be named individually, e.g. `SERVICE1`, `MOTORDATA2`, `CONFIGURATION3`. In this case, however, programming is more complex because basename and visualisation number must be assigned individually. Scrolling step by step is then very restricted.

ⓘ Use the letter `P` as BASENAME, your program is then compatible with the ifm templates.

*) ⓘ **Also note the new 5-digit numbering when naming your existing visualisation pages!**

Parameters of the inputs

3293

Parameter	Data type	Description
INIT	BOOL	TRUE (only for 1 cycle): Display is initialised with the initisalisation indicated in INIT_PAGE. FALSE: during further processing of the program
INIT_PAGE	WORD	visualisation number which is to be called with INIT
PAGE_UP	BOOL	edge FALSE → TRUE: increments the visualisation number
PAGE_DOWN	BOOL	edge FALSE → TRUE: decrements the visualisation number
PAGE_EXTERN	WORD	The indicated visualisati→ page is directly opened (independent of PAGE_UP / PAGE_DOWN). if PAGE_EXTERN = ACT_PAGE, then PAGE_EXTERN is reset "0"!
PAGE_MAX	WORD	maximum number of selectable visualisation pages
BASENAME	STRING [35]	Common part of the name of the visualisati→ page Visualisation pages are numbered by their names: eg. P00001. The following applies: • "P" = BASENAME (only capital letters!) • "00001" = visualisation number (5 digits!)

Parameters of the outputs

3295

Parameter	Data type	Description
ACT_PAGE	WORD	current visualisation number

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 24)

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 Annex

Contents

1664

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

7.1 System flags

15309
8440

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

► Für die Programmierung nur die Symbolnamen der Systemmerker nutzen!

System flags (symbol name)	Type	Description
KEY_F1	BOOL	function key 1
KEY_F2	BOOL	function key 2
KEY_F3	BOOL	function key 3
KEY_F4	BOOL	function key 4
KEY_UP	BOOL	navigation key [▲]
KEY_DOWN	BOOL	navigation key [▼]
KEY_LEFT	BOOL	navigation key [◀]
KEY_RIGHT	BOOL	navigation key [▶]
KEY_OK	BOOL	navigation key [OK]
KEY_ESC	BOOL	Navigation key [ESC]
KEY_CHANGED	BOOL	Pulse:Key status changed (= a button was pressed or released)
KEY_PRESSED	BOOL	A button was pressed
KEY_BACKLIGHT	BYTE	key background illumination (0...100 %) preset = 100 %
RT_F1	BOOL	pulse: function key 1 actuated
RT_F2	BOOL	pulse: function key 2 actuated
RT_F3	BOOL	pulse: function key 3 actuated
RT_F4	BOOL	pulse: function key 4 actuated
RT_UP	BOOL	pulse: navigation key [▲] actuated
RT_DOWN	BOOL	pulse: navigation key [▼] actuated
RT_LEFT	BOOL	pulse: navigation key [◀] actuacted
RT_RIGHT	BOOL	pulse: navigation key [▶] actuated
RT_OK	BOOL	pulse: navigation key [OK] actuated
FT_F1	BOOL	pulse: function key 1 released
FT_F2	BOOL	pulse: function key 2 released
FT_F3	BOOL	pulse: function key 3 released
FT_F4	BOOL	pulse: function key 4 released
FT_UP	BOOL	pulse: navigation key [▲] released
FT_DOWN	BOOL	pulse: navigation key [▼] released

System flags (symbol name)	Type	Description
FT_LEFT	BOOL	pulse: navigation key [◀] released
FT_RIGHT	BOOL	pulse: navigation key [▶] released
FT_OK	BOOL	pulse: navigation key [OK] released
SCREEN_BACKLIGHT	BYTE	LCD background illumination (0...100 %) preset = 100 %
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 [°C]

7.2 Error tables

Contents

19606

7.2.1 Error flags

19608

→ chapter ***System flags*** (→ page 187)

7.2.2 Errors: CAN / CANopen

19610
19604

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

EMCY codes: CANx

13094

ℹ The indications for CANx also apply to each of the CAN interfaces.

EMCY code object 0x1003		Object 0x1001	Manufactor specific information					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Description
00	80	11	--	--	--	--	--	CANx monitoring SYNC error (only slave)
00	81	11	--	--	--	--	--	CANx warning threshold (> 96)
10	81	11	--	--	--	--	--	CANx receive buffer overrun
11	81	11	--	--	--	--	--	CANx transmit buffer overrun
30	81	11	--	--	--	--	--	CANx guard/heartbeat error (only slave)

EMCY codes: system

8413

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	31	05						Supply voltage
00	42	09						Excess temperature

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

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

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 195)

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

E

F

G

H

I

J

K

L

M

10 Notizen • Notes • Notes

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

Stand: 2015-03-06 8310

www.ifm.com • E-Mail: info@ifm.com

Service-Hotline: 0800 16 16 16 4 (Germany only, Mo...Fr, 07.00...18.00 o'clock)

ifm Niederlassungen • Sales offices • Agences

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

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