



Programming manual
ecomatBasic
CR413S

GB

Contents

1	Preliminary note	8
1.1	Legal and copyright information	8
1.2	Purpose of the document	8
1.3	Symbols used	9
1.4	Warnings	9
1.5	Overview: ifm user documentation	9
1.6	Overview: CODESYS documentation	10
1.7	CODESYS	10
1.8	Change history	10
1.8.1	Changes 06 / 2024	10
2	Safety instructions	11
2.1	Required background knowledge	11
2.2	Note!	11
2.3	Start-up behaviour of the controller	12
2.4	Cyber security	12
2.4.1	Access protection	13
2.4.1.1	User administration in the project	13
2.4.1.2	Password of the controller	13
2.4.1.3	Protection against brute force attacks	13
2.4.1.4	Identification of the controller, preventing login on the wrong controller	13
3	Intended use	15
3.1	Predictive incorrect use	15
4	System description	16
4.1	System (in general)	17
4.1.1	System context of the controller	17
4.1.2	Integrated programmable logic controllers	17
4.1.3	Diagnostics	18
4.1.3.1	Cyclic diagnostics	18
4.1.3.2	Self-test at restart (start-up test)	18
4.2	Time response	19
4.2.1	Time-related behaviour IEC application	19
4.2.2	Configuration time	20
4.3	Hardware description	21
4.3.1	Hardware structure	21
4.3.1.1	Overview: Hardware	21
4.3.1.2	Note on wiring	21
4.3.1.3	Available memory	21
4.3.2	Device supply (technology)	23
4.3.2.1	Block diagram of the supply	23
4.3.2.2	Voltage ranges	23
4.3.2.3	Start conditions	24
4.3.2.4	Behaviour in case of voltage dip	24
4.3.2.5	Switching on and off	24
4.3.2.6	Sleep mode	24
4.3.3	Inputs (technology)	25
4.3.3.1	Cross references inputs	25
4.3.3.2	Types of inputs	25
4.3.4	Outputs (technology)	31
4.3.4.1	Cross references outputs	31
4.3.4.2	Output types	31
4.3.5	Output groups	36
4.3.5.1	Switching off output groups via IEC application	36
4.3.5.2	Output group type OutputGroup-A	36
4.3.5.3	Output group type OutputGroup-B	37
4.3.6	Feedback in case of externally supplied outputs	38
4.3.6.1	Outputs switched off due to error	39
4.3.7	Influence of actuators on the output diagnostics	39
4.3.8	Monitoring the processor temperature	40

4.4	Interfaces	40
4.4.1	CAN: Interfaces and protocols	40
4.4.2	Ethernet interface	41
4.4.3	Service communication	41
4.4.3.1	Service communication via CAN	42
4.4.3.2	Service communication via Ethernet	42
4.4.3.3	ifm ecologOne interface	42
4.4.3.4	IoT service interface	42
4.5	Software description	42
4.5.1	Overview: Software	42
4.5.1.1	Software on the PC	43
4.5.1.2	Software in the controller	43
4.5.2	Software modules for the device	44
4.5.2.1	Bootloader	44
4.5.2.2	Operating system	44
4.5.2.3	Application	44
4.5.2.4	Libraries	45
5	Installation	46
5.1	System requirements	47
5.1.1	Hardware	47
5.1.2	Software	47
5.2	Carry out installation	47
5.2.1	Installing the CODESYS Installer	47
5.2.2	Installing CODESYS Development System	48
5.2.3	ifm package	48
5.2.3.1	Components of the package	48
5.2.3.2	Install package (PC/laptop)	49
5.2.3.3	Update package (PC/laptop)	49
5.2.3.4	Uninstall package (PC/laptop)	50
5.2.3.5	Checking and updating installed add-ons	50
6	Getting started	52
6.1	Using the CODESYS operating instructions	53
6.2	Starting CODESYS	53
6.3	Creating a CODESYS project	53
6.3.1	New project with ecomatBasic	53
6.3.2	Overview: Project structure with ecomatBasic	54
6.4	Updating a CODESYS project from SP11 to SP18	54
6.5	Configuring the programming interface	56
6.5.1	Configure Ethernet interface	56
6.5.1.1	Set communication path of PLC	56
6.5.2	Configuring the CAN interface	57
6.5.2.1	Set communication path of PLC	57
6.5.2.2	Configuring the CAN adapter	58
6.5.3	Check the communication path (blinking test)	59
6.5.4	Display of the controller searched for	59
6.6	Activate the access protection for a project	60
7	System configuration	61
7.1	Preparing PLCs	61
7.1.1	Managing device users	61
7.1.1.1	Opening the user configuration	62
7.1.1.2	Assigning a password	62
7.1.2	Create a user in the CODESYS project	63
7.1.3	Sign the user in to the CODESYS project	63
7.1.4	Enter information about applications	63
7.1.5	Assigning the memory allocation - PLC_1 / PLC_2	64
7.1.5.1	Displaying and setting the memory allocation in CODESYS	64
7.1.6	Assigning inputs/outputs – PLC_1 / PLC_2	64
7.1.6.1	Mapping I/Os in CODESYS	65
7.1.7	Manage files	67
7.1.8	User-defined data	67

7.2	Configuring PLCs	68
7.2.1	Configure task processing	68
7.2.2	Add function libraries to the application	69
7.3	Use memory	69
7.3.1	Data exchange between PLC_1 and PLC_2;	69
7.3.2	Stack	70
7.3.2.1	Example: High stack consumption	70
7.3.2.2	Example: Low stack consumption	71
7.4	Configure IEC watchdog	71
7.5	Configure interfaces	71
7.5.1	Configuring interfaces in CODESYS	72
7.5.1.1	Downloading an interface configuration from the device	72
7.5.1.2	Opening an interface configuration	73
7.5.1.3	Loading an interface configuration to the device	73
7.5.1.4	Saving an interface configuration	73
7.5.2	Configuring the Ethernet address	73
7.5.2.1	Address assignment in Ethernet networks	74
7.5.2.2	Using Modbus	75
7.5.3	Configuring CAN interfaces in the device tree	76
7.5.3.1	via system configuration: CANopen Manager	77
7.5.3.2	via system configuration: CANopen device	79
7.5.3.3	via system configuration: J1939 manager	79
7.5.3.4	via function block: RAW-CAN	81
7.5.4	Interface configuration file comconf.cfg	81
7.5.4.1	Parameter comconf.cfg	81
7.5.4.2	Starting with factory settings	83
7.5.4.3	Starting with changed settings	83
7.5.5	Module slot detection	84
7.5.5.1	Example	84
7.5.5.2	Behaviour in case of an error	85
7.6	Configuring inputs and outputs	86
8	Programming of the standard application	87
8.1	Objects of a standard PLC application	87
8.2	Creating a standard PLC application	88
8.2.1	Supported programming languages	88
8.2.2	Data types	88
8.2.3	Supported variable types	88
8.2.3.1	Non-volatile data	89
8.2.3.2	Function description of non-volatile data	90
8.2.4	Configuring and using inputs and outputs	91
8.2.4.1	Accessing inputs and outputs	91
8.2.4.2	Setting the operating mode of inputs and outputs	91
8.2.4.3	Setting the filter of inputs and outputs	92
8.2.4.4	Default behaviour of the inputs and outputs in case of an error	92
8.2.4.5	Standard diagnostic limit values of inputs	92
8.2.4.6	Standard diagnostic limit values of outputs	93
8.2.4.7	Call sequence for diagnostic function blocks for inputs and outputs	94
8.2.5	Note the following when programming	94
8.3	Using ifm function libraries	94
8.3.1	Access to inputs	95
8.3.2	Access to outputs	95
8.3.3	Control device	95
8.3.4	Read device information	95
8.4	Using CAN	95
8.4.1	Use RawCAN (CAN Layer 2)	95
8.4.1.1	RawCAN: Control CAN network nodes	96
8.4.1.2	RawCAN: Send and receive CAN messages	96
8.4.1.3	RawCAN: Request and send remote CAN messages	96
8.4.2	Use CANopen	96
8.4.2.1	CANopen: Send and receive SDO	97
8.4.2.2	CANopen: Network Management (NMT)	97

8.4.3	Use SAE J1939	97
8.5	Using Ethernet	97
8.5.1	SysSocket interface	97
8.5.2	IoT service interface - sockets	98
8.5.3	Modbus	99
8.6	Error in the IEC application	99
9	Set-up and maintenance	100
9.1	Establishing a connection to the device	101
9.1.1	Connection via Ethernet	101
9.1.2	Connection via CAN	101
9.2	Check the operating system version of the device	101
9.2.1	Check the operating system version of the device	101
9.2.2	Check the hardware version of the device	102
9.3	Update the operating system of the device	102
9.3.1	Update the operating system of the device with ifm ecologOne	102
9.4	Transfer CODESYS project to device	102
9.4.1	Loading the PLC_1 application to the device	103
9.4.2	Loading the PLC_2 application to the device	103
9.4.3	Delete the application program on the device	104
9.5	Data transmission for series production	104
9.5.1	Data transmission with ifm ecologOne	105
9.5.2	Transmission of the files with CODESYS	106
9.5.3	Data transmission with TFTP	106
9.5.4	Files for series production	107
9.6	Documentation of the overall application	107
9.6.1	Documenting the serial number and operating system version	107
9.6.2	Read the device information	108
9.6.2.1	Application information (appinfo.txt)	108
9.6.2.2	Hardware information (devinfo.txt)	108
9.6.2.3	Software information (swinfo.txt)	109
9.7	CODESYS Debugging	110
10	Operation	112
10.1	Operating states	113
10.1.1	Overview of operating mode states	113
10.2	Status LEDs	114
10.2.1	Status LED: system ifm operating system (SYS0+SYS1)	115
10.2.2	Status LED: system PLC (SYS0, SYS1)	115
10.2.3	Status LED: System bootloader (SYS0)	116
10.2.4	Status LED: Sleep mode (SYS0)	116
10.2.5	Status LED Ethernet interface (ETH)	116
10.2.6	Controlling user LEDs in the applications	116
10.3	Reset	117
10.3.1	Reset behaviour of the system	117
10.3.2	CODESYS reset requirements (PLC_1)	118
10.3.3	Executing reset variants	118
10.3.4	Reset application (warm)	118
10.3.5	Reset application (cold)	118
10.3.6	Reset application (origin)	119
11	ifm function libraries	120
11.1	General information	120
11.2	Device library	121
11.2.1	Library ifmDevice.library	121
11.2.1.1	CAN_BAUDRATE (ENUM)	122
11.2.1.2	CAN_CHANNEL (ENUM)	122
11.2.1.3	CANconstants (GVL)	122
11.2.1.4	SysInfo (GVL)	122
11.2.1.5	SysInfoStruct (STRUCT)	122
11.2.1.6	LED_COLOUR (ENUM)	122
11.2.1.7	LED_FLASH_FREQ (ENUM)	123
11.2.1.8	Reset	124

11.2.1.9	RESET_TYPE (ENUM)	125
11.3	CAN libraries	126
11.3.1	Library ifmRawCAN.library	126
11.3.1.1	CAN_Enable	127
11.3.1.2	CAN_Recover	129
11.3.1.3	CAN_RemoteRequest	131
11.3.1.4	CAN_RemoteResponse	133
11.3.1.5	CAN_Rx	135
11.3.1.6	CAN_RxMask	137
11.3.1.7	CAN_RxRange	139
11.3.1.8	CAN_RxRangeExt	141
11.3.1.9	CAN_Status	143
11.3.1.10	CAN_Tx	145
11.3.1.11	CAN_Info (GVL)	147
11.3.1.12	CAN_BUS_STATE (STRUCT)	147
11.3.1.13	BUS_STATE (ENUM)	147
11.3.2	Library ifmCANopenManager.library	148
11.3.2.1	COP_GetNodeState	149
11.3.2.2	COP_SDORead	151
11.3.2.3	COP_SDOwrite	153
11.3.2.4	COP_SendNMT	155
11.3.2.5	NMT_SERVICE (ENUM)	157
11.3.2.6	NMT_STATES (ENUM)	157
11.4	Input and output libraries	158
11.4.1	Library ifmConfigSwThreshold.library	158
11.4.1.1	ConfigSwThreshold	159
11.4.2	Library ifmFastInput.library	162
11.4.2.1	FastCount	163
11.4.2.2	IncEncoder	165
11.4.2.3	Period	167
11.4.2.4	COUNT_DIRECTION (ENUM)	170
11.4.2.5	ENCODER_RESOLUTION (ENUM)	170
11.4.2.6	FREQ_SENSE_PERIODS (ENUM)	170
11.4.2.7	MODE_FAST_COUNT (ENUM)	170
11.4.2.8	MODE_INC_ENCODER (ENUM)	170
11.4.2.9	MODE_PERIOD (ENUM)	171
11.4.3	Library ifmIOcommon.library	172
11.4.3.1	Input	173
11.4.3.2	Output	175
11.4.3.3	SetLED	178
11.4.3.4	SupplySwitch	180
11.4.3.5	SystemSupply	181
11.4.3.6	Temperature	183
11.4.3.7	GetComResetState	185
11.4.3.8	FILTER_INPUT (ENUM)	186
11.4.3.9	FILTER_OUTPUT (ENUM)	186
11.4.3.10	MODE_INPUT (ENUM)	187
11.4.3.11	MODE_OUTPUT (ENUM)	187
11.4.3.12	MODE_SYSTEM_SUPPLY (ENUM)	188
11.4.3.13	MODE_TEMPERATURE (ENUM)	188
11.4.3.14	SYS_VOLTAGE_CHANNEL (ENUM)	188
11.4.4	Library ifmIOconfigDiagProt.library	189
11.4.4.1	ConfigDiagLevel	190
11.4.4.2	ConfigDiagProt	193
11.4.4.3	DEST (ENUM)	195
11.4.4.4	eDIAG_PROT_MODE (ENUM)	195
11.4.5	Library ifmOutGroup	196
11.4.5.1	OutputGroup	197
11.4.5.2	FILTER_OUTPUT_GROUP (ENUM)	200
11.4.5.3	MODE_OUTPUT_GROUP (ENUM)	200
11.4.6	Library ifmOutPWM	201
11.4.6.1	CurrentControl	202

11.4.6.2	PWM1000	205
11.4.6.3	MODE_CURRENT_CONTROL (ENUM)	207
11.4.6.4	MODE_PWM (ENUM)	207
11.5	Help function libraries	208
11.5.1	Library ifmSysInfo.library	208
11.5.1.1	GetInfo	209
11.5.1.2	aSysInfoList (GVL)	211
11.5.1.3	SYS_INFO (STRUCT)	212
12	Troubleshooting	213
12.1	Error classes	213
12.2	Error messages	213
12.3	Messages / diagnostic codes of the function blocks	214
12.4	Removing a faulty IEC application	214
12.5	Finding errors in the IEC application	214
12.6	Exporting the log file of the IEC application	215
13	Appendix	217
13.1	Wiring	218
13.2	Filter	221
13.2.1	Filter times of the inputs	221
13.2.2	Filter times of the outputs	222
13.3	Using the operators	223
13.4	Behaviour in case of floating point operations	226
13.4.1	Behaviour in case of specific arguments	226
13.4.2	Behaviour in case of conversions	226
13.5	Behaviour in case of integer operations	227
13.6	Supported SysSocket functions	228
13.7	Directory structure and file overview	230
13.8	Overview of user rights	231
13.8.1	ifm rights	232
13.8.2	CODESYS rights	232
13.9	Task configuration example	234
13.9.1	Initial situation	234
13.9.2	Setting the task configuration	235
13.9.3	Assigning the CAN manager to the bus task	236
13.9.4	Mapping of the CAN variables	237
13.9.5	Calling CAN POUs	239
13.9.6	Task monitoring	239
13.10	ifm behaviour models for function blocks	240
13.10.1	General	240
13.10.2	Behaviour model ENABLE	240
13.10.2.1	Response to errors	240
13.10.3	Behaviour model EXECUTE	240
13.10.3.1	Response to errors	241
	Glossary	243
	Index	244

1 Preliminary note

1.1	Legal and copyright information	8
1.2	Purpose of the document	8
1.3	Symbols used.....	9
1.4	Warnings	9
1.5	Overview: ifm user documentation	9
1.6	Overview: CODESYS documentation	10
1.7	CODESYS.....	10
1.8	Change history	10
1.8.1	Changes 06 / 2024	10

1.1 Legal and copyright information

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

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

- AS-i is the property of AS-International Association, (→ www.as-interface.net)
- CAN is the property of Robert Bosch GmbH, Germany (→ www.bosch.de)
- CAN is the property of CiA (CAN in Automation e.V.), Germany (→ www.can-cia.org)
- CODESYS™ is the property of CODESYS GmbH, Germany (→ www.codesys.com)
- DeviceNet™ is the property of ODVA™ (Open DeviceNet Vendor Association), USA (→ www.odva.org)
- EtherNet/IP® is the property of → ODVA™
- EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
- IO-Link® is the property of PROFIBUS Nutzerorganisation e.V., Germany (→ www.io-link.com)
- ISOBUS is the property of AEF - Agricultural Industry Electronics Foundation e.V., Germany (→ www.aef-online.org)
- Microsoft® is the property of Microsoft Corporation, USA (→ www.microsoft.com)
- Modbus® is the property of Schneider Electric SE, France (→ www.schneider-electric.com)
- PROFIBUS® is the property of PROFIBUS Nutzerorganisation e.V., Germany (→ www.profibus.com)
- PROFINET® is the property of → PROFIBUS Nutzerorganisation e.V., Deutschland
- Windows® is the property of → Microsoft Corporation, USA

1.2 Purpose of the document

These instructions apply to:

- ecomatBasic CR413S from firmware version V4.1.x.x

These instructions describe the following topics:

- Configuration of the device using CODESYS
- Programming of the device's internal programmable logic controllers using the CODESYS programming system

- Description of the device-specific CODESYS function libraries
- Updating the firmware of the device

1.3 Symbols used

- ✓ Requirement
- Instructions
- ▷ Reaction, result
- [...] Designation of keys, buttons or indications
- Cross-reference
-  Important note
Non-compliance may result in malfunction or interference.
-  Information
Supplementary note

1.4 Warnings

Warnings indicate the possibility of personal injury and damage to property. This enables safe product handling. Warnings are graded as follows:

	WARNING Warning of serious personal injury ▷ If the warning is not observed, fatal and serious injuries are possible.
	CAUTION Warning of minor to moderate personal injury ▷ If the warning is not observed, minor to moderate injuries are possible.
	ATTENTION Warning of damage to property ▷ If the warning is not observed, damage to property is possible.

1.5 Overview: ifm user documentation

The ifm user documentation for the device consists of the following modules:

Document	Content / Description
Data sheet	Technical data
Operating instructions	• Instructions for installation, electrical installation and commissioning • Technical data
Programming manual (this manual)	• Creation of a CODESYS project with this device • Target system configuration with CODESYS • Programming the device-internal PLC using CODESYS • Description of the device-specific CODESYS function libraries
Release notes	Additional information on the device, the software version and known issues

If any documents are not available, they can be requested from ifm or downloaded from the ifm website:

documentation.ifm.com

1.6 Overview: CODESYS documentation

CODESYS GmbH provides the following user documentation for programming the PLC of the device:

Document	Content / description
Online help	• Context-sensitive help • Description of the CODESYS programming system • Description of the components and libraries
CODESYS installation and first steps	• Remarks about the installing of the programming system CODESYS • First steps for handling the CODESYS programming system

After the installation of the programming system CODESYS 3.5 all documents are stored on the hard disk of the PC/laptop and can be accessed:

- Online help:
`...\Program Files\CODESYS [Version]\CODESYS\Online Help`
- CODESYS installation and first steps:
`...\Program Files\CODESYS [Version]\CODESYS\Documentation`

1.7 CODESYS

- Use the CODESYS programming system version V3.5 SP18 patch 0 (64 bits) to program the device's internal programmable logic controllers.

1.8 Change history

1.8.1 Changes 06 / 2024

- New document created

2 Safety instructions

- The unit described is a subcomponent for integration into a system.
 - The system architect is responsible for the safety of the system.
 - The system architect undertakes to perform a risk assessment and to create documentation in accordance with legal and normative requirements to be provided to the operator and user of the system. This documentation must contain all necessary information and safety instructions for the operator, the user and, if applicable, for any service personnel authorised by the architect of the system.
- Read this document before setting up the product and keep it during the entire service life.
- The product must be suitable for the corresponding applications and environmental conditions without any restrictions.
- Only use the product for its intended purpose (→ Installation instructions / Operating instructions, Intended use).
- If the operating instructions or the technical data are not adhered to, personal injury and/or damage to property may occur.
- The manufacturer assumes no liability or warranty for any consequences caused by tampering with the product or incorrect use by the operator.
- Improper or non-intended use may lead to malfunctions of the device, to unwanted effects in the application or to a loss of the warranty claims.
- Installation, electrical connection, set-up, programming, configuration, operation and maintenance of the product must be carried out by personnel qualified and authorised for the respective activity.
- Observe applicable documents.

2.1 Required background knowledge

This document is for specialists. Specialists are people who are qualified by their appropriate training and their experience to see risks and to avoid possible hazards that may be caused during operation or maintenance of the device.

For programming these people should also have knowledge of control technology experience in PLC programming to IEC 61131-3.

The document gives information concerning the correct handling of the product.

2.2 Note!

There is no warranty for any properties that is based on information, notes and examples provided in this manual. The drawings, representations and examples imply no responsibility for the system and no application-specific particularities.

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

**WARNING**

Failure to observe the instructions in this manual may result in property damage or physical injury!

ifm electronic gmbh accepts no liability for this.

- ▶ 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 qualification and training required to perform this work.
- ▶ Please note the technical data of the affected devices!
The current data sheet can be found on the ifm website.
- ▶ Observe the installation and connection conditions as well as the intended use of the devices concerned!
→ Operating instructions supplied or on the ifm website
- ▶ Please note the corrections and notes in the release notes for the hardware, software and documentation available on the ifm website: www.ifm.com

ATTENTION

The applications do not function properly or the configuration is incorrect.

- ▷ Risk of material damage.
- ▷ Faulty or undesired control function.
- ▶ Test and ensure the function of the application and the correctness of the configuration.
- ▶ The system manufacturer is responsible for the functioning of the application and the configuration.

2.3 Start-up behaviour of the controller

ATTENTION

Unintentional and dangerous start of machine or plant sections.

- ▷ Risk of material damage.
- ▶ The programmer must implement a restart disable against unintentional start.

A restart can, for example, be caused by:

- Voltage restoration after power failure
- Voltage pulses on the supply
- Start from sleep mode

This is how you implement restart protection:

- ▶ Actuator start-up only after additional approval, e.g. via key-operated switch.

2.4 Cyber security



- ▶ Please observe the following document: "Operating instructions (supplement) - Cyber Security Whitepaper"

ATTENTION

If the device is operated in an unprotected network environment.

- ▷ Unauthorised data access (read or write) is possible.
- ▷ Unauthorised manipulation of the device function is possible.
- ▶ Check and restrict access options to the device.
- ▶ Restrict access to authorised persons.
- ▶ Do not connect the device to open networks or the internet.
- ▶ If access from the internet should be necessary, it is mandatory to choose a secure method to connect the device (e.g. VPN).
- ▶ Use encrypted data transmission (e. g. https / TLS).

2.4.1 Access protection

The access protection of the controller is implemented by means of the following mechanisms:

2.4.1.1 User administration in the project

Each part of a CODESYS project can be protected against non authorised access by using corresponding settings in the CODESYS user administration: Create a user in the CODESYS project



To ensure the access protection, the user must create a corresponding CODESYS user administration for each project.

2.4.1.2 Password of the controller

[Managing device users \(→ 61\)](#)

2.4.1.3 Protection against brute force attacks

To protect against brute force attacks, the device has a function that fends off potential attackers:

Any login attempts with incorrect login data on the device (CODESYS, ifm ecologOne, TFTP, IoT-Core, etc.) will cause a login timeout.

If several incorrect logins occur in succession, the login timeout will be extended to up to one minute.

The device ignores any login attempts during the active login timeout. It is not possible to log in during this time, not even with correct login data.

You can only log on to the device once the login timeout has expired.

Full control functionality is still guaranteed during the active login timeout.

If the login timeout is active, error messages, e.g. if you try to log on again, will appear depending on the tool used: Busy, Server-Busy, Too Many Requests, etc.

2.4.1.4 Identification of the controller, preventing login on the wrong controller

- ▶ Ensure that the correct controller is used for communication within the network, especially if several controllers are operated in one machine/system.

To identify a controller, prevent login on the wrong controller and successfully log on to the correct controller:

- ▶ Manually check the serial number of the controller: [Documenting the serial number and operating system version \(→ 107\)](#).

- ▶ Perform a blinking test (winking function): [Check the communication path \(blinking test\)](#) (→ [59](#)).
- ▶ Use different passwords for different controllers.
- ▶ Operate the controllers in different networks or sub-networks.

3 Intended use

This device is used to control processes in applications. In addition, with CODESYS, device 2 receives programmable logic controllers (PLCs) that can be programmed independently:

3.1 Predictive incorrect use

**WARNING**

Use of the device in applications with functional safety requirements.

- ▷ Risk of personal injuries and/or damage to property.
- ▶ Only use the device in applications with functional safety requirements in combination with the CP100S software package.
- ▶ Adhere to the the relevant user documentation.

4 System description

4.1	System (in general)	17
4.1.1	System context of the controller	17
4.1.2	Integrated programmable logic controllers	17
4.1.3	Diagnostics	18
4.2	Time response	19
4.2.1	Time-related behaviour IEC application	19
4.2.2	Configuration time	20
4.3	Hardware description	21
4.3.1	Hardware structure	21
4.3.2	Device supply (technology)	23
4.3.3	Inputs (technology)	25
4.3.4	Outputs (technology)	31
4.3.5	Output groups	36
4.3.6	Feedback in case of externally supplied outputs	38
4.3.7	Influence of actuators on the output diagnostics	39
4.3.8	Monitoring the processor temperature	40
4.4	Interfaces	40
4.4.1	CAN: Interfaces and protocols	40
4.4.2	Ethernet interface	41
4.4.3	Service communication	41
4.5	Software description	42
4.5.1	Overview: Software	42
4.5.2	Software modules for the device	44

4.1 System (in general)

4.1.1 System context of the controller

The device is a subcomponent for integration into a system.

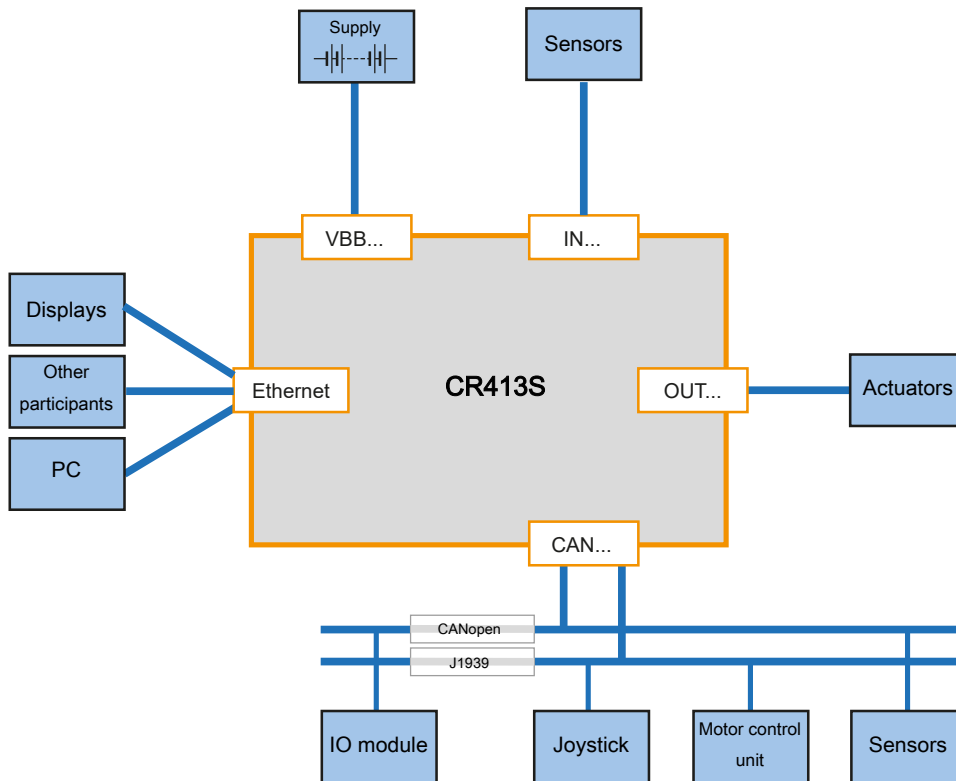


Fig. 1: System context

The elements with an orange frame in the graphics are a part of the controller.

All other elements are external and can be connected to the controller via interfaces and connections.

4.1.2 Integrated programmable logic controllers

The device features 2 separate controllers `PLC_1` and `PLC_2`.

The integrated controllers are free from mutual interference ("freedom of interference": memory, time-dependent behaviour and inputs and outputs).

Data can be exchanged between the controllers (→ [Data exchange between PLC_1 and PLC_2](#); § 69).

The controllers are each to be configured and programmed separately in CODESYS V3.5.



Before the programming of the application may even begin:

- Distribute resources to the two PLCs (memory, inputs, outputs, user LEDs) (→ [Preparing PLCs](#) § 61).

4.1.3 Diagnostics

The controller has the following internal diagnostics:

- [Cyclic diagnostics](#) (→ [18](#))
- [Self-test at restart \(start-up test\)](#) (→ [18](#))

The controller provides function libraries to create application-specific diagnostics for the IEC application.

4.1.3.1 Cyclic diagnostics

The internal cyclic diagnostics performed by the controller at start-up and during operation include:

- diagnosis of the supply for valid voltage values
- diagnosis of the processing including monitoring of the memory
- diagnosis of the inputs
- diagnosis of the outputs and output groups

The diagnostics are designed such that errors are detected and result in a shutdown depending on the error classes (→ [Error classes](#) [213](#)).

ATTENTION

Danger due to unintentional deactivation of all outputs if monitoring routines detect a system error:

- ▷ Risk of material damage.
- ▷ The device switches off the energy for all outputs of the affected output groups.
- ▶ Please note: [Output groups](#) (→ [36](#))
- ▶ Please note: [Wiring](#) (→ [218](#))

Watchdog

The watchdog is a cyclic diagnosis executed in the system in several stages:

- IEC task-related watchdog
This watchdog works in the ifm operating system and is executed separately for each PLC. Each IEC task is monitored individually.
If an error occurs, the system only deactivates the affected PLC and the corresponding outputs.
Error class = B
- External watchdog
If an error occurs, this watchdog puts the entire system into the "safe state" (emergency stop). The output groups and all outputs of the group go to logical "0".
Error class = B

→ chapter [Error classes](#) (→ [213](#))

To eliminate the fault:

- ▶ Rebooting the PLC is necessary via voltage on/off.

4.1.3.2 Self-test at restart (start-up test)

After the restart, the controller performs the following internal self-tests (start-up test) once:

- entire diagnostic path of the supply

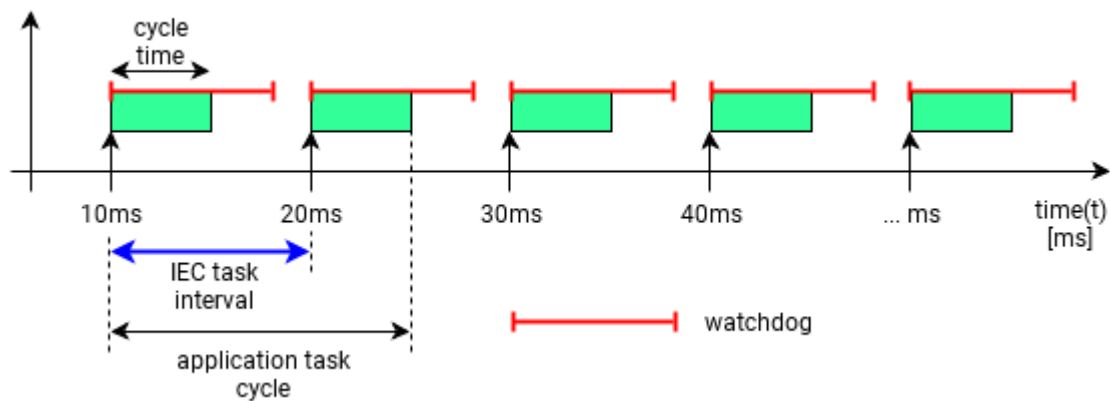
- entire diagnostic path of the watchdogs in the companion chip
- Function of the outputs and output groups and their diagnostic path
- function of processing / controller incl. memory and peripherals and their diagnostic path

Inputs can only be partially tested during the start-up test. The signal path cannot be tested, since the connected external signals (sensors, switches, signal cables, etc.) are required for this (§

4.2 Time response

4.2.1 Time-related behaviour IEC application

IEC applications have an IEC task interval in which the IEC application task is called and a cycle time (execution time of the task).



The cycle time may fluctuate during operation. The determination of the cycle time can be done by measuring the processing time, e.g. in CODESYS Task Monitoring.

The Application Task Cycle is defined as follows:

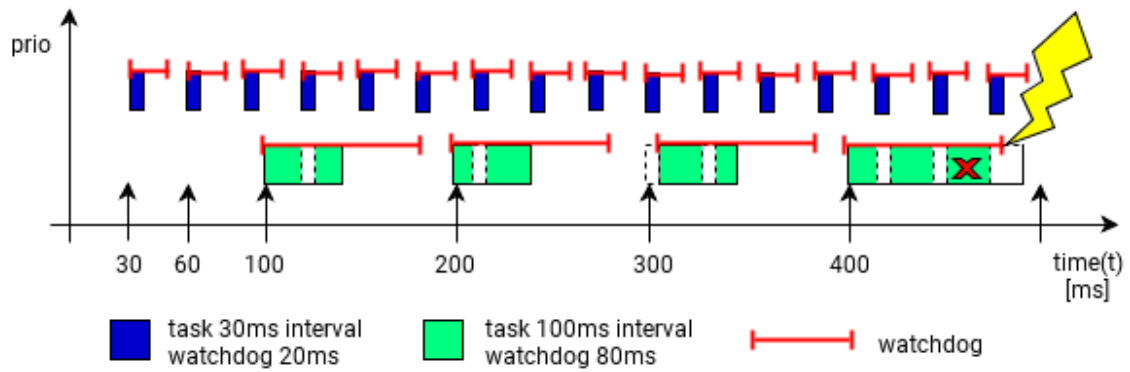
$$\text{APPTCY} = \text{IEC Task Interval} + \text{Cycle TimeMax}$$

In the supplied standard package, the CODESYS package features a template containing one task with standard settings per PLC and one program without function.

During the projecting time, the project engineer can adapt the template to the conditions of the overall application. ([→ Configure task processing](#) □ 68)

Within the scope of the adaptation to the overall application, the project engineer can create more than one IEC application.

The system is pre-emptive, so that the task with higher priority will interrupt the task with lower priority.



Processing time of the IEC application

$$T_{\text{Max}}(\text{IEC}) = 2 * \text{IEC Task Interval}_{\text{Max}}$$

$$T_{\text{Nom}}(\text{IEC}) = \text{IEC Task Interval}_{\text{Max}} + \text{Cycle Time}_{\text{Max}}$$

$$T_{\text{Min}}(\text{IEC}) = 0\text{ms}$$

4.2.2 Configuration time

In the application, the configuration of system components can be done, e.g. mode settings. After a change in the configuration, it can take up to 20 ms until the new values are valid.

4.3 Hardware description

4.3.1 Hardware structure

4.3.1.1 Overview: Hardware

System overview

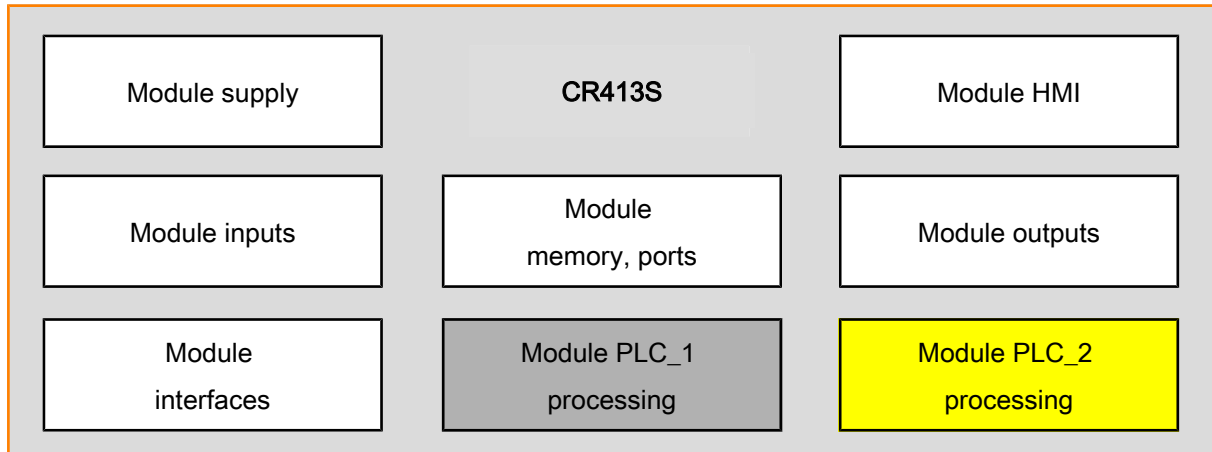


Fig. 2: System overview

Supply	Device supply (technology) (→ □ 23)
Inputs	Inputs (technology) (→ □ 25)
Interfaces	Interfaces (→ □ 40)
Memory, ports	Available memory (→ □ 21)
Processing	Integrated programmable logic controllers (→ □ 17)
HMI	Status LEDs (→ □ 114)
Outputs	Outputs (technology) (→ □ 31)

4.3.1.2 Note on wiring

For further information:

- → Operating instructions
- → Technical data

Designations of inputs and outputs

Inputs and outputs are assigned in groups.

The identifier of an input or output results from the following principle:

1. Type (IN, OUT)
2. Group number (00...xx)
3. Channel number (00...xx)

Examples:

- IN0002 = input | group 00 | channel 02 in this group
- OUT0507 = output | group 05 | channel 07 in this group

4.3.1.3 Available memory

Memory allocation

The controller subdivides the memory according to IEC61131-1 to store the user data into:

- memory of the applications (parts are configurable):
 - IEC-Code PLC_1
 - IEC-Code PLC_2
- Application data memory (IEC data):
 - for volatile data (IEC RAM)
 - for non-volatile data (memory-remanent in case of voltage failure)

The device has the following additions:

- USER files (storage of application-specific data as files in the folder `data` , [User-defined data](#) (→ [67](#)))
- IEC memory bytes (permanent storage of application-specific data at application-specific addresses)

Program memory	RAM	non-volatile memory
physical: 9.0 MB available: 5.5 MB	physical: 2.7 MB available: 1.5 MB	physical: 10 kbytes available: 10 kB
IEC code PLC_1 (configurable)	IEC RAM PLC_1 (configurable)	IEC retain PLC_1 (2.5 kB)
IEC code PLC_2 (configurable)	IEC RAM PLC_2 (configurable)	IEC memory bytes PLC_1 (2.5 kB)
User files (physical: 768 kB)		IEC retain PLC_2 (2.5 kB)
		IEC memory bytes PLC_2 (2.5 kB)

Fig. 3: Memory areas of the controller

Memory allocation variants

The user can select from the pre-defined configurations of the memory layout. The configuration enables optimum division between the device-internal PLCs.

[Device Variant] (memory configuration)	IEC-Code PLC_1	IEC-RAM PLC_1	IEC-Code PLC_2	IEC-RAM PLC_2	Memconfig-file
3583 kB PLC_1 flash 1023 kb PLC_2 flash	3.5 Mbytes	1024 kBytes	1.0 Mbytes	512 kBytes	CRxxxS-mem- conf-3583S-1023.ifm
2303 kB PLC_1 flash 2303 kb PLC_2 flash (Default)	2.25 Mbytes	768 kBytes	2.25 Mbytes	768 kBytes	CRxxxS-mem- conf-2303S-2303.ifm
1023 kB PLC_1 flash 3583 kb PLC_2 flash	1.0 Mbytes	512 kBytes	3.5 Mbytes	1024 kBytes	CRxxxS-mem- conf-1023S-3584.ifm



CODESYS projects reserve fixed memory areas in the RAM memory of PLC_1 and PLC_2 for project information and IO mapping.

The size of the reserved memory areas depends on the project.



Description of the allocation of the memory layout in CODESYS: [Assigning the memory allocation - PLC_1 / PLC_2](#) (→ [64](#))

4.3.2 Device supply (technology)

4.3.2.1 Block diagram of the supply

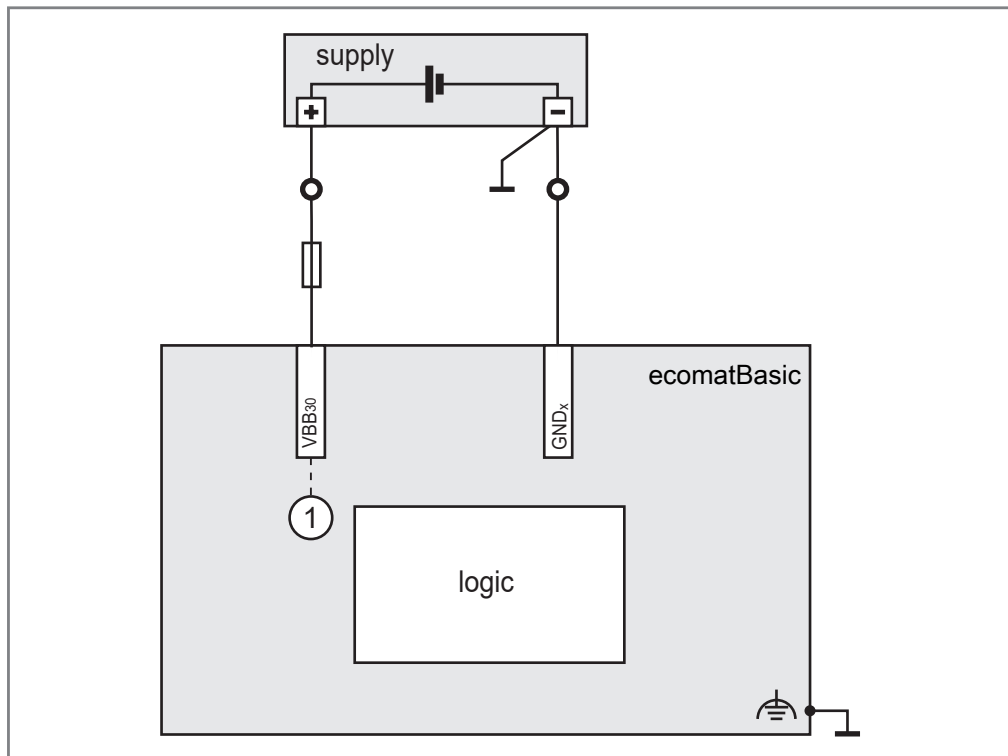


Fig. 4: Block diagram of the supply

1: VBB30: Voltage value of the device supply

4.3.2.2 Voltage ranges

The system monitors the voltage ranges.

The voltages mentioned here apply to the specified range of $\pm 1\%$ (at 36 V).

Voltage [V]		Description
from	to	
0	< +5.5	Undervoltage VBB30: If the controller was in the OPERATING mode, the controller will shut down.
+5.5	< +8.0	Limited operating range If the controller was in the OPERATING mode, it will continue to run without restrictions in the event of voltage dips in this range.
+8.0	+32.0	Regular operating voltage Nominal operating range All functions available VBB30 ≥ 8 V: Controller starts
> +32.0	+36.0	Overvoltage (protected) The device is not damaged by the voltage deviation. If this condition on VBB30 in the OPERATING mode lasts longer than 10 s, the controller will change to the FATAL ERROR state. If this condition on VBB0...n in the OPERATING mode lasts longer than 10 s, the corresponding output group will change to the COMPONENT ERROR state.

Voltage [V]		Description
from	to	
> +36.0		Overvoltage (unprotected) In this area, the device is no longer protected and the behaviour is not predictable. The device can be destroyed by this voltage. ► If such voltages are likely to occur in an application, provide for external protection.

ATTENTION

Overvoltage > 36 V (unprotected)

- ▷ Risk of material damage.
- ▷ Uncontrollable and unpredictable dangerous behaviour of the controller, e.g. all outputs activated.
- Provide external protection against overvoltage.

4.3.2.3 Start conditions

The device only starts when a voltage ≥ 8 V is applied to the VBB30 supply connection.

4.3.2.4 Behaviour in case of voltage dip

In the event of a voltage dip (e.g. due to motor start) $V_{BB30} < 5.5$ V, the controller will shut down. If the VBB30 does not drop to 0 V, the controller remains switched off after a recurring voltage $V_{BB30} \geq 8$ V.

The behaviour in case of voltage dip can be changed by the application:

- Call up the Reset function with `RESET_TYPE = POWER_RECOVER` once after starting the application. [Reset](#) (→ [124](#))
- ▷ The controller automatically performs a power-on reset if $V_{BB30} \geq 8$ V..

4.3.2.5 Switching on and off**Sequence when switching on:**

- The system recognises the applied voltage $V_{BB30} \geq 8$ V, boots and starts.

Sequence when switching off:

- The system recognises a drop in the applied voltage $V_{BB30} < 5.5$ V.
- Running tasks are aborted.
- The system switches off completely.



Data loss is possible in case of non-volatile data (retain data).

All data modified since the last successful saving of the non-volatile data will be lost: [Function description of non-volatile data](#) (→ [90](#))

4.3.2.6 Sleep mode

When the VBB30 power supply is switched on, the controller can be set to a power-saving mode by the IEC application:

- The sleep mode is activated with the FUN reset (→ [Reset](#) [124](#)).
- The hardware is almost switched off.
- Sleep mode is indicated by the blue SYS0 LED.
- After switching VBB30 off and on again, the controller will reboot (PowerOn Reset).

4.3.3 Inputs (technology)

4.3.3.1 Cross references inputs

Further information on the use of inputs:

- [Configuring and using inputs and outputs \(→ 91\)](#)

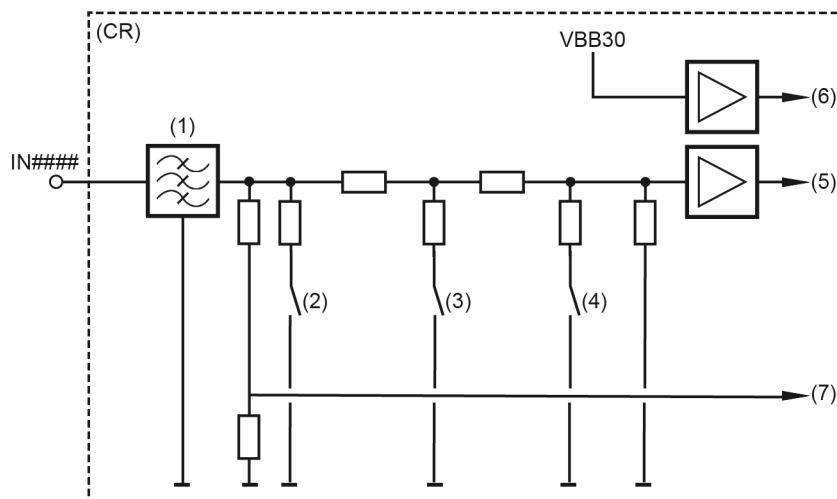
4.3.3.2 Types of inputs

We differentiate between the following input types:

Input type IN MULTIFUNCTION-A

Digital and analogue inputs

Principle circuit IN MULTIFUNCTION-A



IN####: Connection of multi-function input no. ####

CR: Unit

1: Input filter

2: analogue current measurement (integrated overload protection against currents > 23 mA)

3: digital input, plus-switching

4: analogue voltage measurement

5: measured value (mission signal)

6: reference voltage for ratio

7: internal monitoring signal (monitor signal)

Input configuration: [Configuring and using inputs and outputs \(→ 91\)](#)

Possible operating modes		Technical data
IN_DIGITAL_CSI	Digital input	plus-switching (B_L)
IN_DIGITAL_CSI_NA-MUR	Digital input	plus-switching (B_L) for Namur sensors
IN_VOLTAGE_10	Analogue voltage measurement	0...10 000 mV
IN_VOLTAGE_32	Analogue voltage measurement	0...32 000 mV
IN_VOLTAGE_RATIO	Analogue voltage measurement, ratiometric to the reference voltage	0...1000 ‰
IN_CURRENT_CSI	Analogue current measurement	0...20 000 μ A



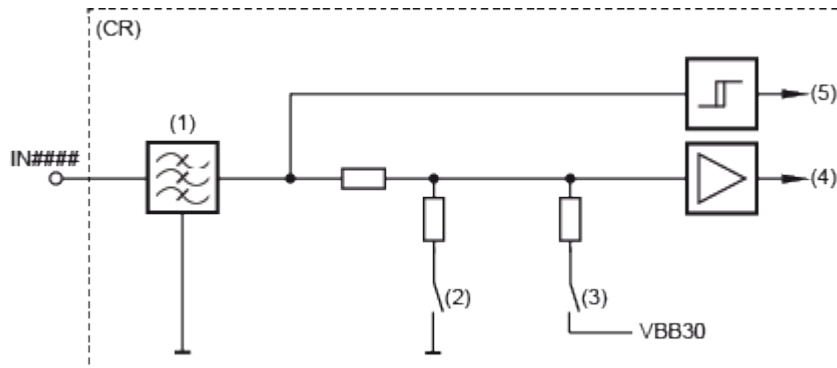
The signal of the measured value of an IN MULTIFUNCTION-A input (mission signal) is compared to the internal monitoring signal (monitor signal) and checked for plausibility.

The plausibility check is not performed for IN MULTIFUNCTION-A inputs with operating mode IN_CURRENT_CSI.

Input type IN FREQUENCY-B

Digital and fast inputs

Principle circuit IN_FREQUENCY-B



IN####: Connection of frequency / counting input no. ####
 CR: Unit
 1: Input filter
 2: CSI digital input; plus-switching
 3: CSO digital input; minus-switching
 4: measured value (analogue)
 5: measured value (digital)

Input configuration: [Configuring and using inputs and outputs \(→ 91\)](#)

Possible operating modes		Technical data
IN_DIGITAL_CSI	Digital input	plus-switching (B_L)
IN_DIGITAL_CSO	Digital input	minus-switching (B_H)
IN_DIGITAL_CSI_NAMUR	Digital input	plus-switching (B_L) for Namur sensors
IN_DIGITAL_CSI_BLANKING	Digital input	plus-switching (B_L) Blanking signal for fail-safe inputs
IN_VOLTAGE_10	Analogue voltage measurement	0...10 000 mV
IN_COUNT_CSI	Pulse counter	plus-switching (B_L)
IN_COUNT_CSO	Pulse counter	minus-switching (B_H)
IN_FREQUENCY_CSI	Frequency measurement	plus-switching (B_L)
IN_FREQUENCY_CSO	Frequency measurement	minus-switching (B_H)
IN_INC_ENCODER_CSI	Evaluate the input pair for position encoder	plus-switching (B_L)
IN_INC_ENCODER_CSO	Evaluate the input pair for position encoder	minus-switching (B_H)
IN_PERIOD_RATIO_CSI	Measuring of frequency, cycle time and pulse/pause ratio	plus-switching (B_L)
IN_PERIOD_RATIO_CSO	Measuring of frequency, cycle time and pulse/pause ratio	minus-switching (B_H)
IN_PERIOD_RATIO_US_CSI	Measuring of frequency, cycle time and pulse/pause ratio	plus-switching (B_L)
IN_PERIOD_RATIO_US_CSO	Measuring of frequency, cycle time and pulse/pause ratio	minus-switching (B_H)
IN_PHASE_CSI	Evaluate the input pair for the phase position	plus-switching (B_L)
IN_PHASE_CSO	Evaluate the input pair for the phase position	minus-switching (B_H)



- The internal resistance R_i of the signal source must be significantly lower than the input resistance R_{input} the input used (voltage matching principle).

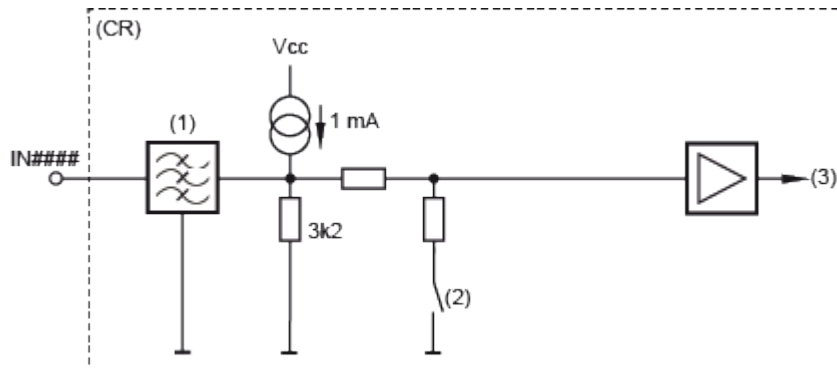
Otherwise the input signal of the fast input can be distorted (low-pass characteristic).

- If operating mode `IN_DIGITAL_CSO` and input signal voltage > `VBB30`: The voltage measurement at the input becomes inaccurate and the overvoltage detection triggers early.

Input type IN RESISTOR-A

Digital inputs and resistance measurement

Block diagram IN RESISTOR-A



IN####: Connection digital / resistance measurement input no. ####
 CR Unit
 1: Input filter
 2: CSI digital input; plus-switching
 3: measured value (analogue)

Input configuration: [Configuring and using inputs and outputs \(→ 91\)](#)

Possible operating modes		Technical data
IN_DIGITAL_CSI	Digital input	plus-switching (B_L)
IN_DIGITAL_CSI_NAMUR	Digital input	plus-switching (B_L) for Namur sensors
IN_RESISTOR	Resistance measurement (to GND)	minus-switching (B_H)

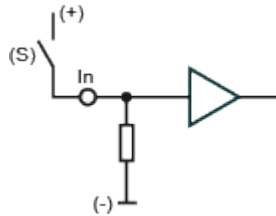
Digital input block diagram plus/minus-switching

Digital input plus-switching

CSI = Current Sinking

Principle circuit plus-switching digital input (BL) for positive sensor signal:

input = open => signal = low (GND);



In: connection of digital input n

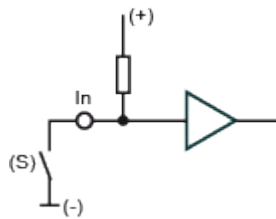
S: sensor

Digital input minus-switching

CSO = current sourcing

Block diagram, minus-switching digital input (BH) for negative sensor signal:

input = open => signal = high (supply);



In: connection of digital input n

S: sensor

4.3.4 Outputs (technology)

4.3.4.1 Cross references outputs

Further information on the use of outputs:

- [Configuring and using inputs and outputs \(→ 91\)](#)

4.3.4.2 Output types

We differentiate between the following output types:

Output type OUT PWM-n-A

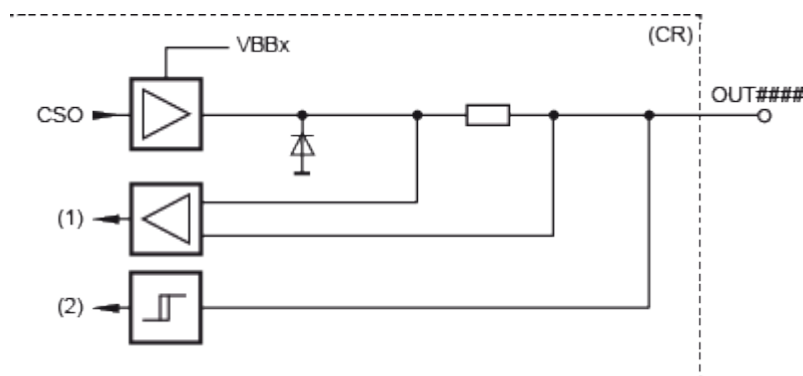
n = current rating

Example: n = 25 $\Rightarrow I_{\max} = 2.5 \text{ A}$

Digital output or

analogue output with pulse width modulation (PWM), optionally current-controlled (PWM_I)

Block diagram OUT PWM-n-A



OUT####: connection of PWM output no. ####

CR: Unit

1: measured value (analogue)

2: measured value (digital)

Output configuration: [Configuring and using inputs and outputs \(→ 91\)](#)

Possible operating modes		Technical data
OUT_DIGITAL_CSO	Digital output	Plus-switching (B _H)
OUT_PWM_CSO	Analogue output CSO with pulse width modulation, with current measurement	PWM _H Plus-switching (B _H)
OUT_CURRENT_CSO	Analogue output CSO with pulse width modulation, current-controlled	PWM _I Plus-switching (B _H)



The output diagnostics is internally limited to wire break, short circuit and overload.

The project engineer has to carry out external measures to avoid cross fault in cabling.

Reaction of the controller in case of an error: error class D, periphery error [Error classes \(→ 213\)](#)

Configuration and measurement via:

- FB Output ([→ Output ¶ 175](#)) for digital output
- FB PWM1000 ([→ PWM1000 ¶ 205](#)) for PWM
- FB CurrentControl ([→ CurrentControl ¶ 202](#)) for current control (PWM_I)

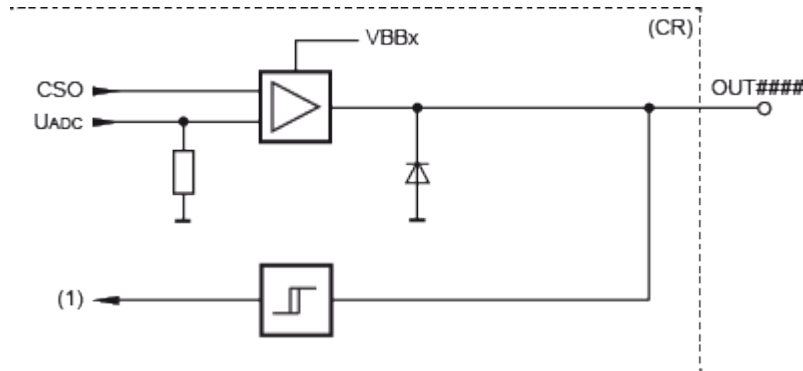
Output type OUT PWM-n-B

n = current rating

Example: n = 25 $\Rightarrow I_{\max} = 2.5 \text{ A}$

Digital output or
analogue output with pulse width modulation (PWM)

Block diagram OUT PWM-n-B



OUT####: connection of PWM output no. ####

CR: device

1: measured value (digital)

U_{ADC} : Current mirror

Output configuration: [Configuring and using inputs and outputs](#) ($\rightarrow$ [91](#))

Possible operating modes		Technical data
OUT_DIGITAL_CSO	Digital output	Plus-switching (B_H)
OUT_PWM_CSO	Analogue output CSO with pulse width modulation, with current measurement	PWM _H Plus-switching (B_H)



OUT PWM-n-B supports no dither functionality.

The current measurement does not detect any free-wheeling current.

Configuration and measurement via:

- FB Output ($\rightarrow$ [Output](#) [175](#)) for digital output
- FB PWM1000 ($\rightarrow$ [PWM1000](#) [205](#)) for PWM

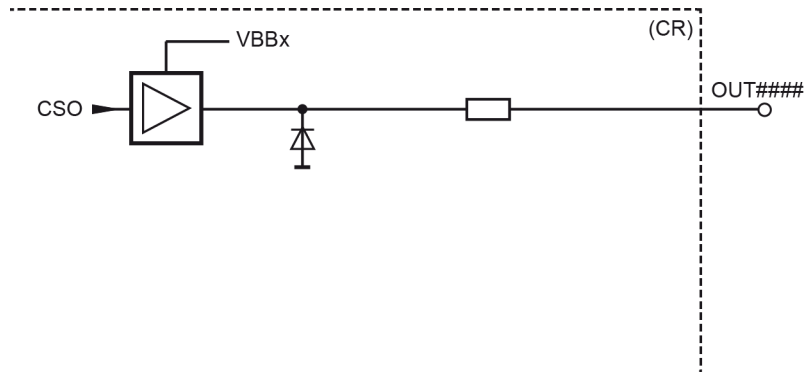
Output type OUT PWM-n-D

n = current rating

Example: n = 10 $\Rightarrow I_{\max} = 1.0 \text{ A}$

Digital output or analogue output with pulse width modulation (PWM)

Block diagram OUT PWM-n-D



OUT####: connection of PWM output no. ####

CR: device

Output configuration: [Configuring and using inputs and outputs \(→ 91\)](#)

Possible operating modes		Technical data
OUT_DIGITAL_CSO	Digital output	Plus-switching (B_H)
OUT_PWM_CSO	Analogue output CSO with pulse width modulation	PWM _H Plus-switching (B_H)

Configuration and measurement via:

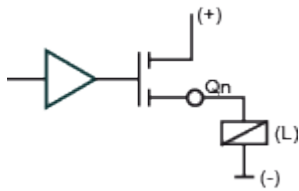
- FB Output ([→ Output 175](#)) for digital output
- FB PWM1000 ([→ PWM1000 205](#)) for PWM

Digital output block diagram plus/minus-switching

Digital output positive-switching

CSO = Current Sourcing

Block diagram plus-switching output (B_H) for positive output signal



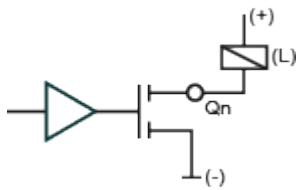
Qn: connection of output n

L: Load

Digital output minus-switching

CSI = Current Sinking

Block diagram minus-switching output (B_L) for negative output signal



Qn: connection of output n

L: Load

4.3.5 Output groups

The outputs of the device are assigned to output group.

The actual number of output groups and the type and number of associated outputs is specific to each device type. → Technical data

Each output group has its own VBBx supply voltage:

- VBB₀ supplies the outputs OUT₀₀₀₀...OUT_{000x}
- VBB₁ supplies the outputs OUT₀₁₀₀...OUT_{010x}
- etc.

The supply voltage of an output group can be switched off via the corresponding group switch. If the group switch is turned off, the corresponding outputs will be deactivated.

Preset: The group switch switches on as soon as the PLC switches to the RUN state.

4.3.5.1 Switching off output groups via IEC application

During the program sequence, the output switches are under the user's full software control.

During the program sequence, the output switches can be disconnected from power via FB OutputGroup (→ [OutputGroup](#)  197) and thus the associated outputs can be switched off.

ATTENTION

Unintentional and dangerous start of machine or plant sections.

▷ Risk of material damage.

▶ The programmer must implement a restart disable against unintentional start.

4.3.5.2 Output group type OutputGroup-A

The device has output groups of the **OutputGroup-A** type. → Technical data

The supply voltages of the output groups of type OutputGroup-A are switched off using integrated hardware group switches (semiconductor switches).

Operating mode for the FB [OutputGroup](#) (→  197): OUT_DIGITAL

Output deactivation OutputGroup-A

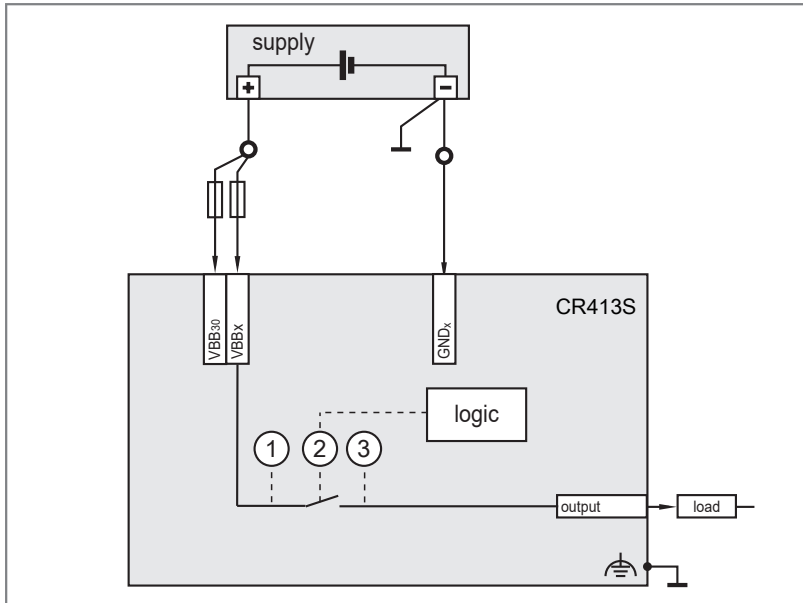


Fig. 5: Basic structure of the output cut-off for output groups type A

- 1: VbbxVoltage: Voltage value before the group switch x
- 2: VBBx_SW: Hardware group switch of output group x
- 3: GroupVoltage: Voltage value after the group switch x

4.3.5.3 Output group type OutputGroup-B

The device has output groups of the **OutputGroup-B** type. → Technical data

The supply voltages of the output groups of type OutputGroup-B are switched off using virtual software group switches.

Output groups of type OutputGroup-B do not have a hardware group switch.

Operating mode for the FB **OutputGroup** (→ [197](#)): **OUT_DIGITAL_VIRT**

Output deactivation OutputGroup-B

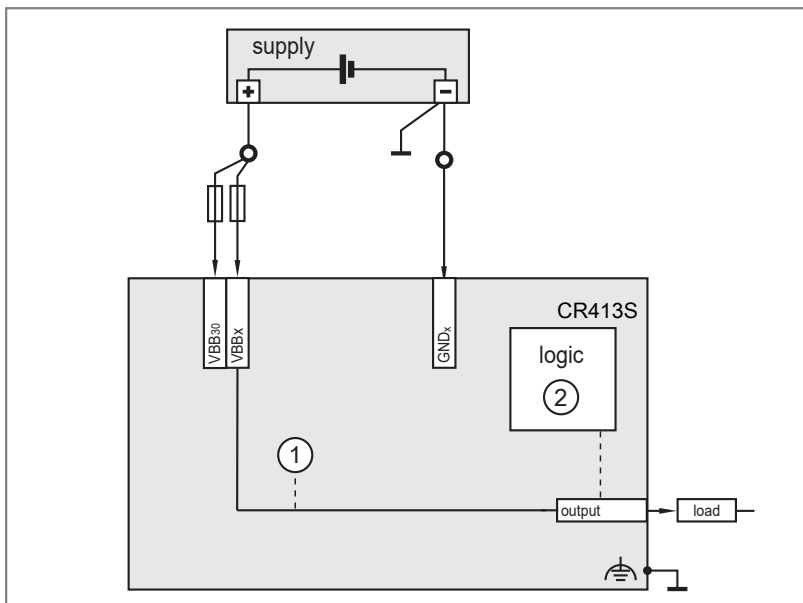


Fig. 6: Basic structure of the output deactivation for output groups type B

- 1: VbbxVoltage = GroupVoltage: Voltage value of the output group x
- 2: VBBx_SW: Virtual software group switch of the output group x

4.3.6 Feedback in case of externally supplied outputs

In some applications, actuators are not only controlled by outputs of the device, but also by external switches.

The external switches can lead to the outputs being fed back.

As soon as an incorrect voltage is measured at the controller, e.g. due to power feedback, the controller will report an error class depending on the selected configuration of the `ConfigDiagLevel` of the output and the output group and will switch off the output and the output group.

The relevant diagnosis for the output group is the analogue diagnosis `GroupVoltage` (measured voltage value after the group switch). For the outputs, the digital diagnosis `Status Feedback` is relevant.



In case of externally supplied outputs, perform the following measures to prevent feedback:

- ▶ Do not apply any external voltage to the outputs.
- ▶ Decouple the externally supplied outputs via blocking diodes (→ Figure below: V1 and V2) so that no external voltage can be applied to the output terminal of the controller.
- ▷ Any power feedback to the outputs is prevented.
- ▷ The controller does not go to the configured error class when switch S1 is closed.
- ▷ The output group does not switch off when switch S1 is closed.

OutputGroup-A

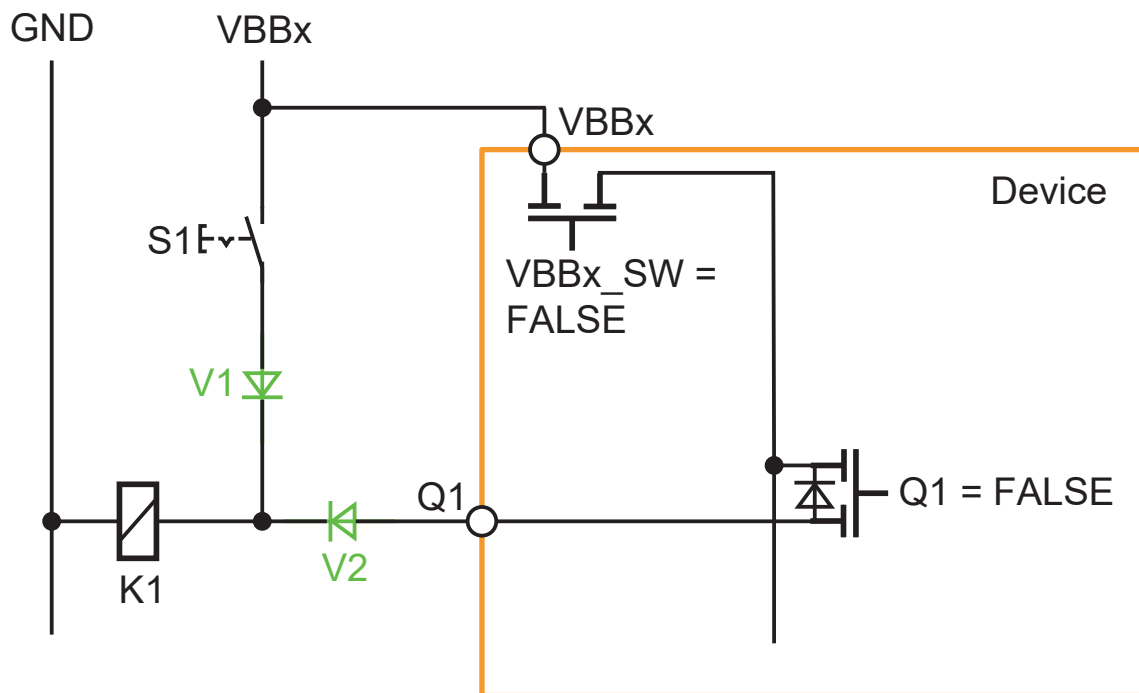


Fig. 7: Example: Wiring with blocking diodes to prevent power feedback with output group type OutputGroup-A

- S1: Possible additional external switch
V1, V2: Blocking diodes to be used to prevent power feedback
K1: Load at the device output, e.g. relay

OutputGroup-B

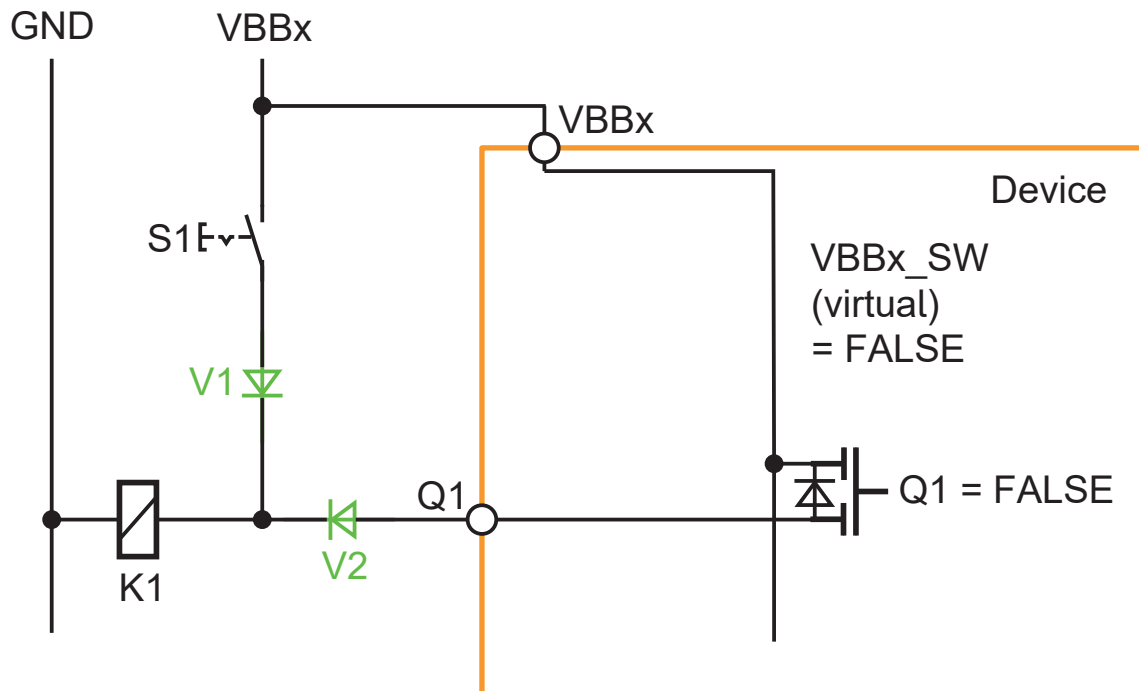


Fig. 8: Example: Wiring with blocking diodes to prevent feedback with output group type OutputGroup-B

S1: Possible additional external switch
 V1, V2: Blocking diodes to be used to prevent power feedback
 K1: Load at the device output, e.g. relay

4.3.6.1 Outputs switched off due to error

All outputs are switched off and the controller reports the error to the IEC application.

To reboot the device:

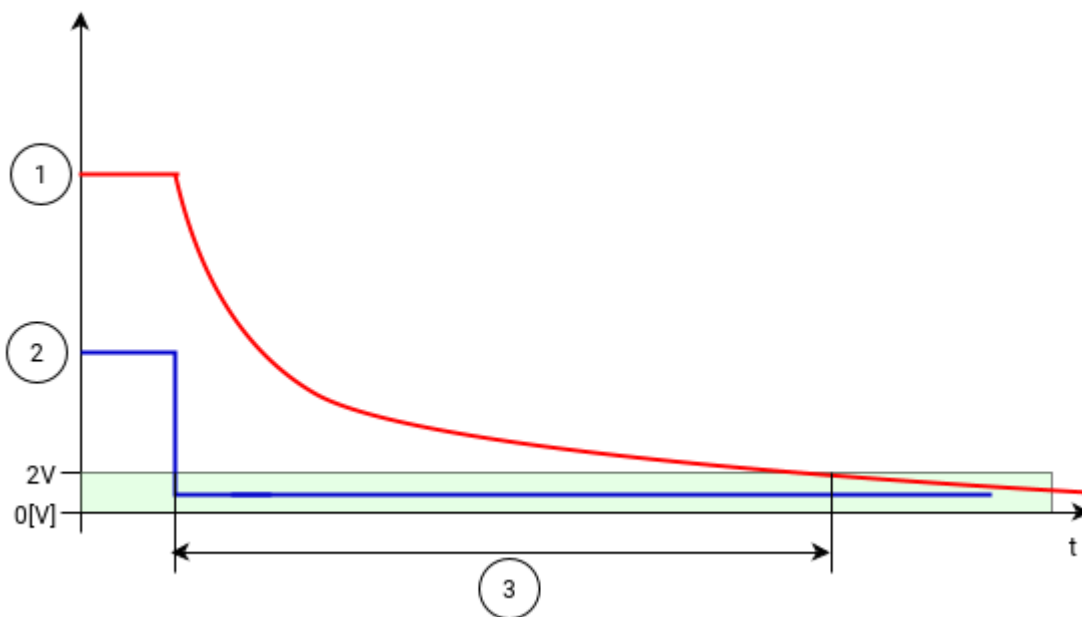
- Eliminate the cause of the error
- Carry out a Power-On-Reset

OR error handling in the IEC application:

- Eliminate the cause of the error
- Correct the error status of the output group via `xResetError`

4.3.7 Influence of actuators on the output diagnostics

The voltage at the output must have fallen below the threshold value of the diagnostic status feedback 10 ms (standard value of the DetectionTime) after switch-off of the output (→ Data sheet). Otherwise a STUCK_AT_HIGH error is triggered.



Capacitive or inductive loads can trigger a STUCK_AT_HIGH error at switch-off of an output (2) if the discharge (1) lasts longer than the set DetectionTime (3). As a result, the entire output group switches off.

- Use the function block ConfigDiagLevel (→ [ConfigDiagLevel](#) □ 190) to set the DetectionTime (3) of the output to a value suitable for the load.

4.3.8 Monitoring the processor temperature

The following limit values apply for monitoring the processor temperature:

Limit value type	Diagnostic code on FB Temperature	Input limit value	Output limit value
Lower limit value Temperature in the lower critical range	DIAG_LOW_TEMPERATURE	$\leq -45\text{ °C}$	$\geq -40\text{ °C}$
Upper limit value Temperature in the upper critical range	DIAG_HIGH_TEMPERATURE	$\geq +115\text{ °C}$	$\leq +110\text{ °C}$

The diagnostic message at the diagnostic output of the FB [Temperature](#) (→ □ 183) is triggered when the input limit value is violated and disappears when the output limit value is exceeded/fallen below (hysteresis).

4.4 Interfaces

The device includes the interfaces described in the following.



Position of the connections on the device and technical data: → Operating instructions, technical data

4.4.1 CAN: Interfaces and protocols



- Familiarise yourself with the following CODESYS functions:
CAN-based field buses:
→ Online help > Fieldbus support > CAN-based fieldbuses

The device has 2 independent CAN interfaces.

Each CAN interface supports the following protocols:

- RawCAN (CAN Layer 2)
- CANopen Manager
- CANopen device
- J1939 Manager

4.4.2 Ethernet interface

ATTENTION

If the device is operated in an unprotected network environment.

- ▷ Unauthorised data access (read or write) is possible.
- ▷ Unauthorised manipulation of the device function is possible.
- ▶ Check and restrict access options to the device.
- ▶ Restrict access to authorised persons.
- ▶ Do not connect the device to open networks or the internet.
- ▶ If access from the internet should be necessary, it is mandatory to choose a secure method to connect the device (e.g. VPN).
- ▶ Use encrypted data transmission (e. g. https / TLS).

The Ethernet interface supports the following standards:

- Transmission rate 10/100 Mbits/s

The Ethernet interface supports the following protocols:

- TCP/IP
- UDP/IP
- Modbus/TCP Slave
- Modbus/TCP Master
- Network variables UDP

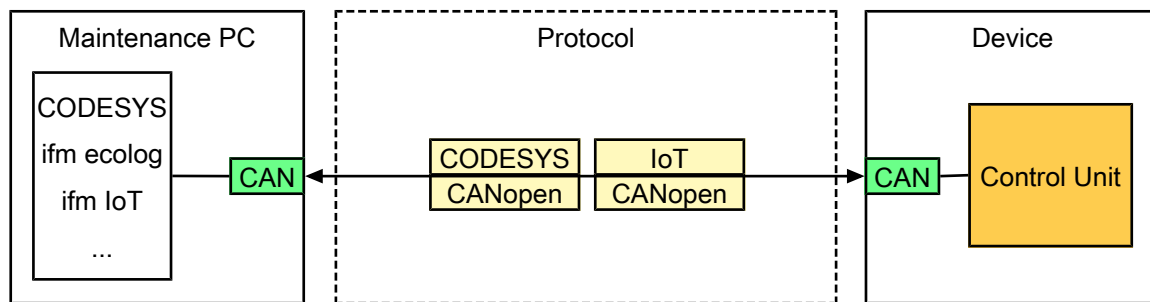
Connections and data → Technical data

4.4.3 Service communication

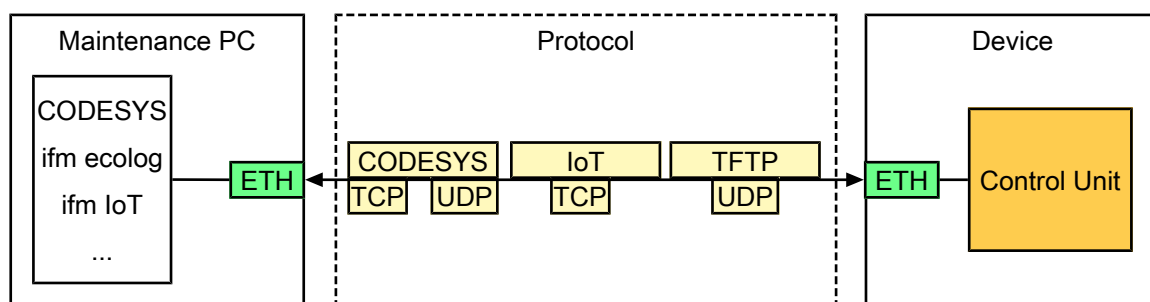
Service communication is used for bidirectional data transmission between a PC (programming device) and the device for service purposes. Project planning data, operating system updates, configuration files, retaining data, CODESYS applications, debug information, etc. are transferred.

The following hardware interfaces are available for service communication between a PC and the device.

4.4.3.1 Service communication via CAN



4.4.3.2 Service communication via Ethernet



4.4.3.3 ifm ecologOne interface

The device provides an interface for the maintenance tool ifm ecologOne . [ifm ecologOne \(→ 43\)](#)

4.4.3.4 IoT service interface

The devices have the technical requirements for using an IoT service interface, e.g. for software updates via a connected ecomatDisplay.

To use the IoT service interface with the ecomatDisplay, the separate ifm IoT Controller Library is required.



IoT service communication via CAN uses the CANopen block transfer with a high bus load.

The IoT service communication can block CAN messages with a lower priority on the CAN bus for transmission.

CAN messages with a higher priority can block the IoT service communication, which can lead to timeouts.

► Set the machine to a maintenance state before using IoT service communication.

4.5 Software description

4.5.1 Overview: Software

A distinction is made between the following software components:

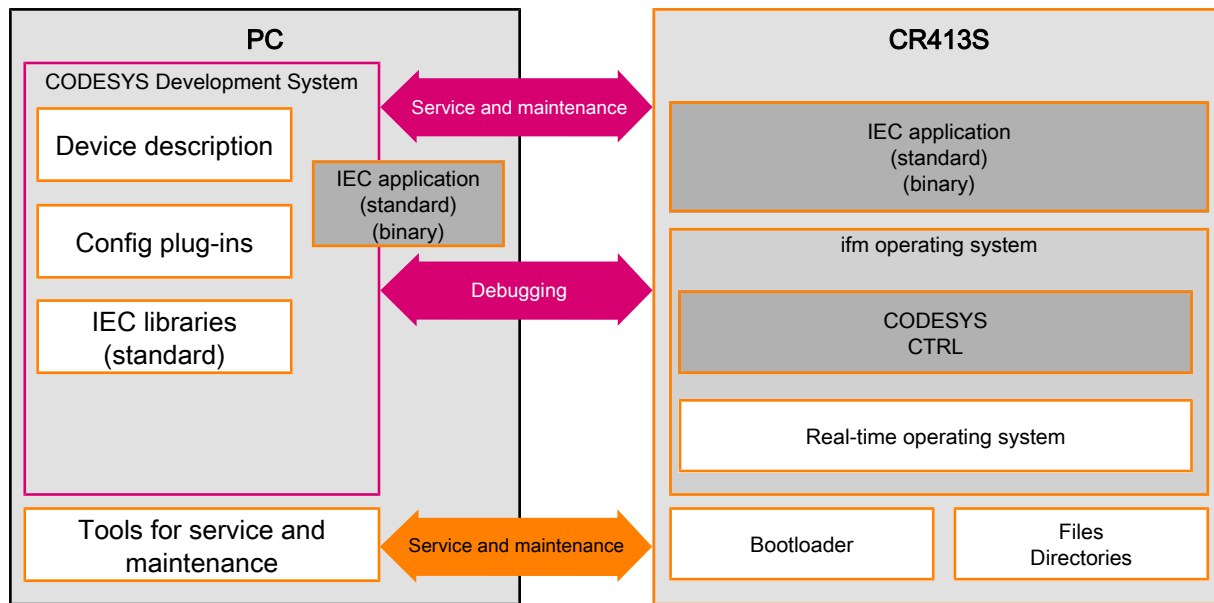


Fig. 9: Software overview

4.5.1.1 Software on the PC

CODESYS

The CODESYS Development System programming environment for creating and debugging the two applications is installed on the PC. The controller supports service and maintenance via CODESYS or via other tools.

The CODESYS functions are extended with plug-ins. This provides additional setting options for memory, communication interfaces and inputs/outputs.

For each derivative, ifm provides suitable device descriptions for the CODESYS Development System.

IEC libraries for the applications provide CODESYS and the programmer with access to the functions of the controller.

Licensing

With the purchase of a device, the buyer also acquires a valid licence for the use of the CODESYS 3.5 programming system.

ifm ecologOne

ifm ecologOne offers maintenance functions and functions for uploading and downloading projects and system files for the device.

ifm Maintenance Tool

ifm Maintenance Tool offers maintenance functions and functions for uploading and downloading projects and system files for devices with Ethernet interface.

4.5.1.2 Software in the controller

The controller processes the applications by means of several software components.

Among other things, the ifm operating system forms the runtime environment that executes the two applications of PLC_1 and PLC_2.

The programmer can store files and directories on the controller and use them in the application.

The bootloader is a fallback level for cases in which the ifm operating system is no (longer) available or it is corrupt.

4.5.2 Software modules for the device

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

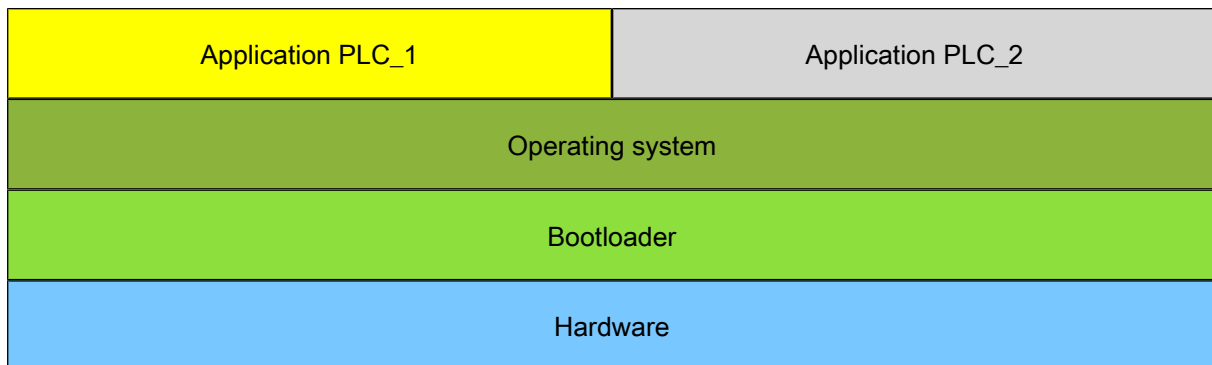


Fig. 10: Software modules

Software module	Can the user change the module?	By means of what tool?
Application with libraries: • for PLC_1 • for PLC_2	Yes	• CODESYS • Tools for service and maintenance
ifm operating system	• Upgrade yes • Downgrade no	• CODESYS • Tools for service and maintenance
Bootloader	No	---
(Hardware)	No	---

4.5.2.1 Bootloader

The bootloader is a start program with which the operating system can be reloaded on the device.



Only execute the bootloader update when explicitly requested by ifm!

4.5.2.2 Operating system

The operating system (also firmware or runtime system) is the basic program in the device and establishes the connection between the hardware of the device and the application.

The device is supplied with the installed operating system (→ [Check the operating system version of the device](#) □ 101).

The operating system can be downloaded at ifm.com.

4.5.2.3 Application

The application (also IEC application or application program) is software that is programmed into the machine by the manufacturer specifically for the application.

The application usually contains logic sequences, limit values and expressions to control the corresponding inputs and outputs, calculations and decisions.

The application can be loaded into a PLC without affecting the operating system.

ATTENTION

The applications do not function properly or the configuration is incorrect.

- ▷ Risk of material damage.
- ▷ Faulty or undesired control function.
- ▶ Test and ensure the function of the application and the correctness of the configuration.
- ▶ The system manufacturer is responsible for the functioning of the application and the configuration.

4.5.2.4 Libraries

ifm electronic provides the following function libraries for the programming of the device under CODESYS 3.5:

Name	Description
ifmCANOpenManager	Functions for use of the CAN interfaces as CANopen Manager
ifmDevice	Data structures, enumeration types and global variables
ifmFastInput	Functions to access the fast inputs of the device
ifmIOcommon	Functions for access to the inputs and outputs of the device
ifmIOconfigDiagProt	Functions to configure the I/O-related diagnostic and protective functions
ifmOutGroup	Functions to control output group switches
ifmOutHBridge	Functions to access H-bridge outputs
ifmOutPWM	Functions to access PWM outputs
ifmRawCAN	Functions for use of the CAN interfaces as CAN Layer 2
ifmSysInfo	Functions to set / read system information
ifmTypes	Global types and interfaces for other ifm libraries
ifmConfigSwThreshold	Functions for configuring the switching thresholds of inputs



▶ Information on use: [Using ifm function libraries \(→ 94\)](#)



▶ Detailed information about the ifm function libraries: [ifm function libraries \(→ 120\)](#)

5 Installation

5.1	System requirements	47
5.1.1	Hardware.....	47
5.1.2	Software	47
5.2	Carry out installation	47
5.2.1	Installing the CODESYS Installer.....	47
5.2.2	Installing CODESYS Development System	48
5.2.3	ifm package.....	48

This chapter describes the installation of the software components that are necessary to program the device.

5.1 System requirements

Under which conditions can and may this device be programmed and operated?

5.1.1 Hardware

- One device CR413S
- A PC for the CODESYS programming system An Internet connection is recommended during the software installation.
- An Ethernet connection between CODESYS-PC and the USB device (→ Operating instructions)
- or-
- A CAN connection between the CODESYS PC and the device (→ Operating instructions)
 - A CAN adapter for the USB interface of the PC

5.1.2 Software

The following software components are required to configure and program the device's internal PLCs:

Component	of	Description	Version	Download and installation information
CODESYS Development System	CODESYS	Programming software CODESYS for PLC programming according to standard IEC 61131-3	V3.5 SP18 patch 0 (64 bits)	Installing CODESYS Development System (→ 48)
CODESYS Installer	CODESYS	Manages the local installations of different CODESYS versions and supports their deployment and update. The ifm package for the device is also installed with the CODESYS Installer.	Current version, at least V2.2.2	Installing the CODESYS Installer (→ 47)
[CR413S]_V4.1.x.x.zip	ifm	ifm package: software package for the device CR413S, including: Components of the package (→ 48)	V4.1.x.x	Components of the package (→ 48)



The features and functions described in this manual can only be obtained by using the software components in the versions stated here.

5.2 Carry out installation

Install the software components described at [Software \(→ 47\)](#) on the PC:

5.2.1 Installing the CODESYS Installer



- Observe the CODESYS help at CODESYS Installer : [content.helpme-codesys.com/de/CODESYS Installer/](https://content.helpme-codesys.com/de/CODESYS%20Installer/)
- Download the CODESYS Installer at: store.codesys.com/de/codesys-installer.html
- Install the CODESYS Installer on the PC.
- Observe the manufacturer's installation instructions.

5.2.2 Installing CODESYS Development System

To install the "CODESYS Development System" software:



Administrator rights are required for installation on the PC.



Depending on the PC, CODESYS can take up to 1 hour or more to install without the installation progress being recognisable at all times.

► Wait and do not cancel the installation prematurely.

Recommended:

- Install the programming system CODESYS V 3.5 SP18 patch 0 (64 bits) using the tool CODESYS Installer.
- Observe the notes in the CODESYS Installer help.
- ▷ CODESYS is installed on the PC.

Alternatively, e.g. if installation via CODESYS Installer is not possible:

- Download CODESYS V3.5 SP18 patch 0 (64 bits) from www.ifm.com and install it manually.
- Observe the manufacturer's installation instructions: → CODESYS installation and first steps
- ▷ CODESYS is installed on the PC.

5.2.3 ifm package



- Familiarise yourself with the following CODESYS functions:
Package Manager: → Online help > CODESYS Development System > Manage packages and licenses

5.2.3.1 Components of the package

A software package is provided by ifm for programming the device's internal PLCs.

The software package can be downloaded after registration at www.ifm.com.

The package contains the following components:

Data name / path	Description
CRxxxx_[Version].zip	Software package (ZIP file). The version follows the version of the operating system.
CODESYS_Package	Folder for corresponding CODESYS device packages and plug-ins
ifm_Controller_[Version].package	CODESYS device package (device description, libraries, etc.)
V[CODESYS-Version]-SIL2.installation-config	Profile for the CODESYS installer
Plugin	Folder with the CODESYS plug-ins
CODESYS_Safety_SIL2_[Version].package	Package "CODESYS Safety SIL2 Plugin"
ifm_CODESYS_local_gateway_plugin_[Version].package	Plug-in for gateway configuration for service communication via CAN interface
ifm_comconfig_plugin_[Version].package	Plugin for interface configuration
ifm_iomapping_plugin_[Version].package	Plug-in for configuring the inputs and outputs
Device	Folder for corresponding files that can be loaded to the device.
boot	Folder for the bootloader
boot.ifm	Bootloader binary file
cfg	Folder with default configuration files

Data name / path	Description
comconf.cfg	Communication configuration (default)
CRxxxx-iomapping.cfg	Input / output resources (default)
CRxxxx-memconf-1023S-3584.ifm	Memory configuration 3
CRxxxx-memconf-2303S-2303.ifm	Memory configuration 2 (default)
CRxxxx-memconf-3583S-1023.ifm	Memory configuration 1
compat	Folder for the compatibility file
CRxxxx-compat.ifm	Compatibility file
os	Folder for the operating system
ifmOS.ifm	Operating system binary file
sis	Folder for hardware description
CRxxxx-[HW-VERSION]-sisys.ifm	One file per supported hardware version
Licences	Folder for licence information

5.2.3.2 Install package (PC/laptop)

To install a package:

Requirements

- CODESYS 3.5 SP18 patch 0 (64 bits) is installed on the PC/laptop.
- The software package for the device CR413S is saved and unpacked on the PC/laptop.
- CODESYS is closed.

1 Start the CODESYS Installer

- ▶ In Windows: Start the [CODESYS Installer] as administrator.
- ▷ The [CODESYS Installer] appears.
- ▷ The window displays the installed CODESYS components.

2 Install package

- ▶ Click on [Install File].
 - ▷ The file explorer is displayed.
- ▶ Select the requested *.package file.
- ▶ Click on [Open] and confirm with [OK].
- ▶ Click on [Continue].
 - ▷ The package is installed.
 - ▷ A success message appears.
- ▶ Click on [OK].
 - ▷ The window [CODESYS Installer] shows the installed packages.
- ▶ Optional: To install another package, proceed again as described.
- ▶ Close the CODESYS Installer.
- ▶ Start CODESYS.
 - ▷ The installed package is now available.

5.2.3.3 Update package (PC/laptop)

To update a package:

1 Uninstall the old version of the package (optional)

- ▶ Uninstalling a package ([→ Uninstall package \(PC/laptop\)](#) □ 50)

2 Install the new version of the package

- ▶ Installing a package ([→ Install package \(PC/laptop\)](#) □ 49)

3 Update the device

- ▶ In the device tree: Highlight the node [ecomatBasic (ecomatBasic)].
- ▶ Select [Project] > [Update device...].
 - ▷ A dialogue window appears.
- ▶ Click on [Update device] to start the update process.
 - ▷ CODESYS loads new device libraries.
 - ▷ Device tree view is updated.
- ▶ Click on [Close] to quit the Package Manager.
- ▶ Save the project.

5.2.3.4 Uninstall package (PC/laptop)

To uninstall a package:

Requirements:

- CODESYS is closed.

1 Start the CODESYS Installer

- ▶ In Windows: Start the [CODESYS Installer] as administrator.
 - ▷ The [CODESYS Installer] appears.
 - ▷ The window displays the installed CODESYS components.

2 Uninstall package

- ▶ Select the requested package in the [AddOns] area.
 - ▷ Information on the selected package will be displayed.
- ▶ Select the package version to be uninstalled.
- ▶ Click on [Uninstall] and confirm with [OK].
 - ▷ The selected package version will be removed.
- ▶ Close the [CODESYS Installer].

5.2.3.5 Checking and updating installed add-ons

The file `V[CODESYS-Version]-SIL2.installation-config` from the ifm package contains the CODESYS Installer profile for the device.

The CODESYS Installer profile lists all CODESYS add-ons and versions thereof that must be installed and used for the device. Additional add-ons are possible.

To verify whether the installed CODESYS add-ons correspond to the mandatory add-ons and versions:

- ▶ In CODESYS Installer, click on the [Export Configuration] button to export the current CODESYS Installer profile.
- ▶ Use a comparison tool for text files to compare the exported CODESYS Installer profile with the profile contained in the ifm package.

In the event of deviations, if add-ons are missing or if the wrong versions are installed:

- ▶ Install all the prescribed add-on versions (see CODESYS Installer profile).

If a prescribed version of a CODESYS add-on is lower than the currently installed version:

- ▶ Uninstall the currently installed add-on version, as CODESYS does not allow you to install lower versions of its add-ons at a later stage.
- ▶ Install the required add-on version.

6 Getting started

6.1	Using the CODESYS operating instructions	53
6.2	Starting CODESYS	53
6.3	Creating a CODESYS project	53
6.3.1	New project with ecomatBasic	53
6.3.2	Overview: Project structure with ecomatBasic	54
6.4	Updating a CODESYS project from SP11 to SP18.....	54
6.5	Configuring the programming interface	56
6.5.1	Configure Ethernet interface	56
6.5.2	Configuring the CAN interface	57
6.5.3	Check the communication path (blinking test)	59
6.5.4	Display of the controller searched for.....	59
6.6	Activate the access protection for a project	60

6.1 Using the CODESYS operating instructions

This manual exclusively describes the integration, configuration and programming of the device using the CODESYS programming system.

The CODESYS terminology is used to describe user actions and components of the user interface.

Standard CODESYS functions and mechanisms are not described. At the beginning of each section, there is a reference to the corresponding chapters of the CODESYS online help.

To access the CODESYS online help:

- ▶ Start CODESYS.
 - ▷ The CODESYS user interface appears.
- ▶ Press [F1].
 - ▷ The online help of the CODESYS programming system appears.



- ▶ Familiarise yourself with the CODESYS programming system! This applies in particular to the following topics:
 - Designations and functions of the elements of the user interface
 - Basic menu functions
 - Programming techniques and data management mechanisms
 - Fieldbus support

6.2 Starting CODESYS

- ✓ The software components have been installed correctly. → Installation chapter
- ▶ Start CODESYS. Double-click on the 3.5 SP18 patch 0 (64 bits) CODESYS icon.
 - ▷ CODESYS starts.
 - ▷ The CODESYS user interface appears.

6.3 Creating a CODESYS project



- ▶ Familiarise yourself with the following CODESYS functions:
 Creating a project → Online help > CODESYS Development System > Creating and Configuring a Project

Managing a project: → Online help > CODESYS Development System > Protecting and Saving Projects

6.3.1 New project with ecomatBasic

- ✓ All required software components are correctly installed. [Installation](#) (→ [46](#))
- ✓ CODESYS successfully started.
- ▶ Select [File] > [New Project...].
 - ▷ The window [New Project] appears.
- ▶ Set the following values:
 - [Category]: Select [ifm Controller].
 - [Templates]: Select the template matching the device: [ifm ecomatBasic]
 - Name: Enter the project name
 - [Location]: Select the storage location of the project file.
- ▶ Click on [OK] to confirm the entered values.

- ▷ CODESYS creates a new project with the selected template.
- ▷ The window [Geräte] shows the device tree of the project.
- ▶ Select [File] > [Save Project].
- ▷ CODESYS saves the project.

6.3.2 Overview: Project structure with ecomatBasic

A CODESYS project contains all components to configure, manage and program the ecomatBasic. All components of a project are shown in the [Device] window in a hierarchic tree view.

CODESYS projects with an ecomatBasic have the following structure:

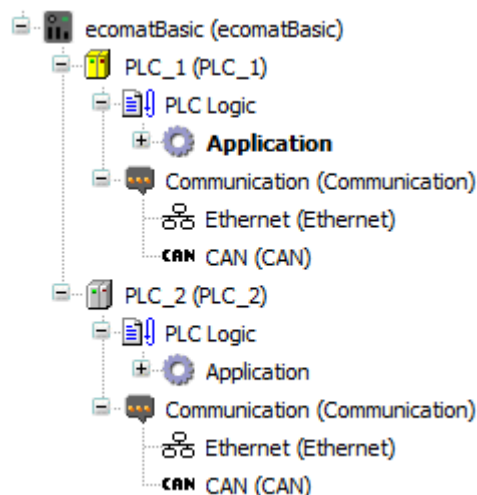


Fig. 11: CODESYS project structure ecomatBasic

[ecomatBasic (ecomatBasic)]	Logical father controller, offers access to the general settings of the ecomat-Basic.
[PLC_1 (PLC_1)]	Contents and settings of the PLC_1
[PLC_2 (PLC_2)]	Contents and settings of the PLC_2
[PLC Logic]	Contains the IEC applications
[Application]	Includes the objects that are required for a control program and the visualisation.
[Communication]	Provides access to the configuration options of the communication interfaces.
[Ethernet] [CAN]	Available communication interfaces.

6.4 Updating a CODESYS project from SP11 to SP18

To update a CODESYS project that was created with SP11 to SP18:

- ✓ All required software components for CODESYS 3.5 SP18 patch 0 (64 bits) are correctly installed. [Installation \(→ 46\)](#)
- ✓ The mandatory CODESYS add-on versions are installed: [Checking and updating installed add-ons \(→ 50\)](#)
- ✓ A backup of the CODESYS-SP11 project exists.
- ▶ Open the CODESYS SP11 project in CODESYS SP18.
- ▶ One or more error messages may appear: Close the error messages with [OK].
 - ▷ The [Project Inspection Overview] window appears.
- ▶ Select [Continue with this Version].

- ▶ Click on [Finish].
 - ▷ The [Project Environment] window appears.

Select versions

- ▶ From the [Compiler version] tab, in the [Action] list, select [Update to 3.5.18.0].
- ▶ From the [Visualization profile] tab, in the [Action] list, select the latest version (not relevant for controller projects without visualisation).
- ▶ From the [SoftMotion] tab, in the [Action] list, select the latest version.
- ▶ If the [Device versions] tab displays: Set the version of the associated add-on or the next lower version for the devices (CODESYS Installer profile, [Checking and updating installed add-ons](#) (→ [50](#))).



- ▶ If the [Library versions] tab displays: Do NOT update any libraries.

- ▶ Click on [OK].
 - ▷ The project is updated.
- ▶ If at any time the [Storage Format] window appears: Update the [Storage Format].
 - ▷ The project opens.
 - ▷ Devices that were not found and missing libraries are marked.

Update devices and set memory configuration

Update the 3 main devices to the version of the package `ifm_CODESYS_Vx.x.x.x.package`. The 3rd and 4th digits of the version may be lower or higher than the version of the package. Select devices with the desired memory configuration.

- ▶ Right-click on [CRxxxS/ecomatBasic/ecomatController] > [Update Device] to update the root device to [ecomatBasic/ecomatController].
- ▶ Double-click on the root device.
- ▶ Open the [Device Variant] tab.
- ▶ Select a suitable memory configuration for the project: [Displaying and setting the memory allocation in CODESYS](#) (→ [64](#))

Update libraries

- ▶ Compile the project with [F11].
- ▶ In the [Library Manager], check each PLC for underlined libraries, expand down to the underlined library at the lowest level and ensure that the right version is installed.
- ▶ Open the [Placeholders] window in the Library Manager.
- ▶ Change all libraries which can be allocated to a CODESYS add-on to the version of the same add-on in the file `V[CODESYS-Version]-SIL2.installation-config` or to the next lower version.

For CODESYS libraries that cannot be allocated to a CODESYS add-on:

- ▶ If a library version $\geq 4.x.x.x$ exists: Select the latest version.
- ▶ Otherwise: Select the version of the Service Pack (SP18) or lower.

For libraries still marked as invalid (e.g. from other manufacturers):

- ▶ Install if necessary and set the right version for the project.

Check added devices and install EDS files

- ▶ Install suitable device descriptions (e.g. EDS) for devices added to the device tree such as CANopen devices, J1939 devices or Modbus devices.

Correct the program code

- ▶ Compile the project with [F11].
- ▶ If necessary, make changes to the program code.

Compile and load the project

- ▶ Compile the project with [F11].
- ▶ If no errors are displayed: Save the project and load it to a device as required.

6.5 Configuring the programming interface

The internal PLCs are programmed via (position of the connections: → Operating instructions):

- Ethernet interface (→ [Configure Ethernet interface](#)  56)
- CAN interface (→ [Configuring the CAN interface](#)  57)

6.5.1 Configure Ethernet interface



- ▶ Recommendation: If possible, use service communication via Ethernet, as this offers fast data transmission.



Device and PC/laptop can be coupled directly or indirectly via an Ethernet network.

- ▶ Only use the recommended accessories for connection of the Ethernet interfaces! (→ Operating instructions)
- ▶ For network connection, an experienced user or system administrator should set up the network addresses and the configuration.

ATTENTION

If the device is operated in an unprotected network environment.

- ▷ Unauthorised data access (read or write) is possible.
- ▷ Unauthorised manipulation of the device function is possible.
- ▶ Check and restrict access options to the device.
- ▶ Restrict access to authorised persons.
- ▶ Do not connect the device to open networks or the internet.
- ▶ If access from the internet should be necessary, it is mandatory to choose a secure method to connect the device (e.g. VPN).
- ▶ Use encrypted data transmission (e. g. https / TLS).

6.5.1.1 Set communication path of PLC

To configure the communication path between the CODESYS programming system and the device's internal PLC via an Ethernet interface:

Preparations

- CODESYS PC/laptop and Ethernet interface of the device are connected.

- Optional: Adjust IP settings of the Ethernet interface.
- The Ethernet interface of the device is activated using the SERVICE parameter in the interface configuration for CODESYS. (standard setting for devices with Ethernet interface)

1 Select the communication settings

- ▶ In the device tree: Double-click on [ecomatBasic (ecomatBasic)]
- ▶ In the editor window: Select the [Communication Settings] tab.
 - ▷ The editor window shows the communication settings.

2 Select the gateway

- ▶ Select the requested gateway in the [Gateway] list.
 - ▷ The list shows the selected gateway.

3 Set the communication path

- ▶ Click on the [Scan Network] button.
 - ▷ The [Select Device] window appears.
 - ▷ CODESYS scans the network for devices.
 - ▷ The window shows the network path and the detected devices.
- ▶ Select the node of the requested device and click [OK].
 - ▷ The device login window appears.
- ▶ Log in using the `admin` user. (No password is set by default.)
 - ▷ The communication path to the device-internal PLC is set.
 - ▷ CODESYS can transfer data to the device-internal PLC.
 - ▷ CODESYS can transfer data from the PLC to the PC.

6.5.2 Configuring the CAN interface



- ▶ Recommendation: If possible, use service communication via Ethernet, as this offers fast data transmission.

6.5.2.1 Set communication path of PLC

To configure the communication path between the CODESYS programming system and the device's internal PLC via a CAN interface:

Requirements



- ▶ The service communication is processed with a lower priority than the process CAN communication.
 - ▶ Service communication cannot be carried out if there are too many non-received CAN messages on the CAN bus. Consequence: CODESYS cannot connect via the CAN interface.
 - ▶ Ensure that all sent CAN messages are received and processed in the IEC application using the Rx function blocks.
- A CAN adapter is connected to a USB interface of the CODESYS PC, e.g. ifm CANfox (article no. EC2112).
 - The CAN adapter is ready for operation and connected to the CAN connection of the device.
 - The CAN interface of the device is activated using the SERVICE parameter in the interface configuration for CODESYS. (Default setting for devices without Ethernet interface)

- The CAN adapter is activated and configured in CODESYS: [Configuring the CAN adapter](#) (→ [58](#))
- The project is open in CODESYS.

1 Select the communication settings

- ▶ In the device tree: Double-click on [ecomatBasic (ecomatBasic)]
- ▶ In the editor window: Select the [Communication Settings] tab.
 - ▷ The editor window shows the communication settings.

2 Select the gateway

- ▶ Select the requested gateway in the [Gateway] list.
 - ▷ The list shows the selected gateway.

3 Set the communication path

- ▶ Click on the [Scan Network] button.
 - ▷ The [Select Device] window appears.
 - ▷ CODESYS scans the network for devices.
 - ▷ The window shows the network path and the detected devices.
- ▶ Select the node of the requested device and click [OK].
 - ▷ The device login window appears.
- ▶ Log in using the `admin` user. (No password is set by default.)
 - ▷ The communication path to the device-internal PLC is set.
 - ▷ CODESYS can transfer data to the device-internal PLC.
 - ▷ CODESYS can transfer data from the PLC to the PC.

6.5.2.2 Configuring the CAN adapter

A USB-CAN adapter is used to access the device from the PC via the CAN interface for project planning, e.g. ifm CANfox (article no. EC2112).

To configure the CAN adapter for establishing a connection with the device:

- ▶ In the device tree of CODESYS: Double click on ecomatBasic (ecomatBasic).
 - ▷ The tab of the device appears in the editor window.
- ▶ In the editor window: Click on [CODESYS Local Gateway].
 - ▷ The [CODESYS Local Gateway] view appears.

If no device is connected yet or the default configuration is to be restored:

- ▶ Open the standard configuration by clicking on [Restore Default Configuration].
 - ▷ The default configuration is displayed.

Or, if a device is already connected:

- ▶ Download the configuration from the connected device by clicking on [Get Active Configuration].
 - ▷ The active configuration is displayed.

Configuring the gateway:

- ▶ Enter the required driver in the list [CAN Driver]. The CANfox driver can be used universally for many CAN adapters.
- ▶ Enter the baud rate in the input field [Baudrate].

- ▶ Enter a unique node ID for the adapter in the CAN network in the input field [Gateway Node ID].
- ▶ If required, activate the option [Blocktransfer] for faster data transfer.
- ▶ To save the configuration, click on the [Save Gateway] button.
 - ▷ The gateway configuration has been changed and saved.

Activating the gateway configuration:

- ▶ In the Windows notification area next to the clock (Systray), click on the  [CODESYS Gateway SysTray] icon.
- ▶ Click on [Stop Gateway] in the context menu.
 - ▷ The gateway is stopped.
- ▶ Click again on the [CODESYS Gateway SysTray] icon.
- ▶ Click on [Start Gateway] in the context menu.
 - ▷ The gateway is started with the saved configuration.

6.5.3 Check the communication path (blinking test)

To check the set communication path, execute a blinking test of the device:

- ▶ In the device tree: Double-click on the [Device] symbol (CR413S)
- ▶ In the editor window: Select the [Communication] tab.
- ▶ Activate [Scan Network...].
 - ▷ The [Scan Network] window appears.
- ▶ Select the gateway node and start the scan process with [Scan Network]
 - ▷ CODESYS scans the network for devices.
 - ▷ Window shows the network path and detected devices.
- ▶ Select the node of the device
- ▶ Click on the [Wink] button
- ▷ The SYS0-/SYS1-LEDs of the connected device flash in a colour sequence when the connection is established.
The colour sequence is: green-red-yellow, 2 seconds per colour and an overall duration of 20 seconds.

6.5.4 Display of the controller searched for

If a controller has been searched for and found using the CODESYS function [Scan Network], the search result is displayed as follows, depending on the type of controller and the service interface settings.



The search result displayed does not always correspond to the interface actually used for service communication:

Devices with an Ethernet connection always display the Ethernet interface, regardless of the settings in `comconf.cfg`.

Ethernet service interface:

PLC	Search result	Example
PLC_1	ARTIKELNUMMER SIL IP:IP-Adresse SN:Seriennummer [CODESYS Routing Info]	CR721S SIL IP:192.168.82.247 SN:000000861268 [078E.B000.1350]

PLC	Search result	Example
PLC_2	ARTIKELNUMMER STD IP:IP-Adresse SN:Seriennummer [CODESYS Routing Info]	CR721S STD IP:192.168.82.247 SN:000000861268 [078E.B000.1350]

6.6 Activate the access protection for a project



► Familiarise yourself with the following CODESYS functions!

- Protecting and saving a project:
→ Online help > CODESYS Development System > Projecting and Saving Projects

The user can use a password to protect the device from unauthorised access.

- Select [Project] > [Project Settings].
 - ▷ The [Project Properties] window appears.
- Select [Safety].
- Enable the [project file encryption] checkbox.
- Enter the requested password in the field [New Password].
- Enter the entered password again in the field [Confirm new password].
- Select [OK] to activate the access protection for the project.
- ▷ Access protection is activated. Project is encrypted.

7 System configuration

7.1	Preparing PLCs	61
7.1.1	Managing device users	61
7.1.2	Create a user in the CODESYS project	63
7.1.3	Sign the user in to the CODESYS project.....	63
7.1.4	Enter information about applications.....	63
7.1.5	Assigning the memory allocation - PLC_1 / PLC_2	64
7.1.6	Assigning inputs/outputs – PLC_1 / PLC_2	64
7.1.7	Manage files.....	67
7.1.8	User-defined data.....	67
7.2	Configuring PLCs	68
7.2.1	Configure task processing.....	68
7.2.2	Add function libraries to the application	69
7.3	Use memory	69
7.3.1	Data exchange between PLC_1 and PLC_2;	69
7.3.2	Stack	70
7.4	Configure IEC watchdog	71
7.5	Configure interfaces	71
7.5.1	Configuring interfaces in CODESYS.....	72
7.5.2	Configuring the Ethernet address	73
7.5.3	Configuring CAN interfaces in the device tree	76
7.5.4	Interface configuration file comconf.cfg.....	81
7.5.5	Module slot detection	84
7.6	Configuring inputs and outputs	86

7.1 Preparing PLCs

7.1.1 Managing device users

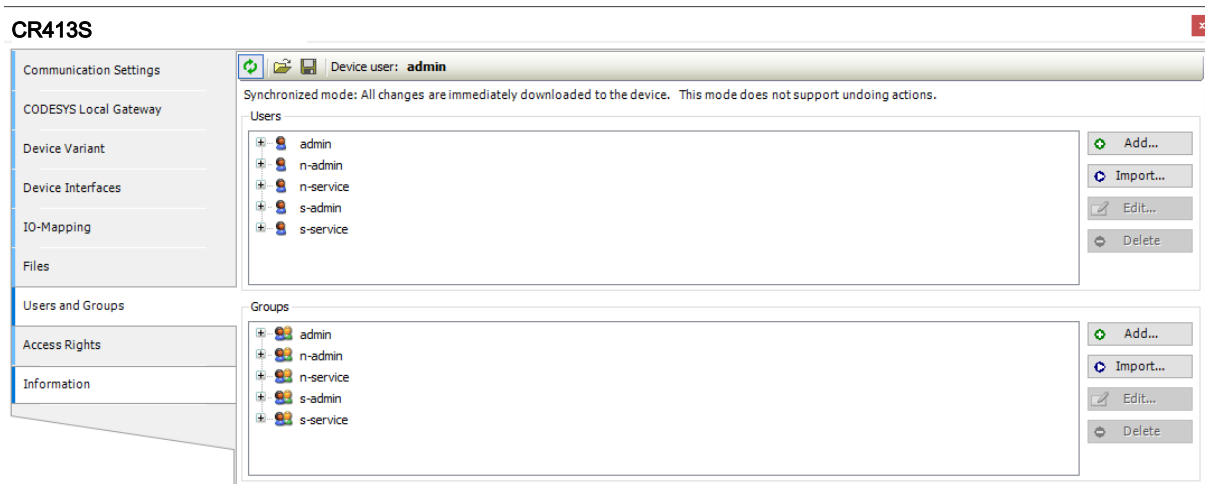


Fig. 12: Users and groups

7.1.1.1 Opening the user configuration

✓ The communication path to the device is set: [Configuring the programming interface \(→ 56\)](#)

► In the CODESYS device tree: Double click on the ecomatBasic icon (ecomatBasic)

► In the editor window: Select the [Users and Groups] tab.

► Click on the  button.

▷ The user configuration of the device is synchronised with the editor.

▷ The editor displays the user configuration.

▷ The following users are available:

- admin
- n-admin
- n-service
- s-admin
- s-service



Information about the user authorisations: [Overview of user rights \(→ 231\)](#)



By default, the users do not have a password.

The project engineer must assign a password to each user to prevent unauthorised access to the project and the controller.

7.1.1.2 Assigning a password



► Recommendation: Set the password of the `admin` user last.

✓ The user configuration is open and synchronised with the device: [Opening the user configuration \(→ 62\)](#)

► Double-click on one of the shown users.

▷ The [Edit user] window appears.

► Enter the password (max. 32 characters) and confirm.

► Click on the [OK] button.

▷ The password is synchronised with the device.

▷ The window closes.

► Click on [File] > [Save project] to save the changes in the project.



(User) groups are not supported.



For this action, you need to sign in as device user.

For more user convenience:

► Create a user in the project and sign the user in once:

[Create a user in the CODESYS project \(→ 63\)](#)

[Sign the user in to the CODESYS project \(→ 63\)](#)

7.1.2 Create a user in the CODESYS project

To ensure that the user data does not need to be entered again when working with the CODESYS project, store it in the project:



- ▶ Make sure to use the correct lower-case and upper-case letters.

- ▶ Menu [Project] > [Project Settings] > [Users and Groups] > [Add...]
- ▶ Enter:
 - [Login name:] s-admin
 - [Full name:] s-admin
 - [Description:] Administrator (e.g.)
 - [Password:] the desired password (must be identical with the password of the device user s-admin)
 - [Password:] repeat the desired password
 - [Activated:] activate click on
 - [OK]
- ▶ If you have not yet done so, sign in as Owner to be able to create the new user [Sign the user in to the CODESYS project \(→ 63\)](#)
- ▷ The user has been created in the CODESYS project.

7.1.3 Sign the user in to the CODESYS project

Sign in to the project after saving the user details in the CODESYS project:



- ▶ Make sure to use the correct lower-case and upper-case letters.

- ▶ Double-click on [Current User: ...] in status bar
 - ▷ The user change dialogue appears
- ▶ Click on [Logon as another user...]
 - ▷ The login dialogue appears
- ▶ Sign in as Owner:
 - [User name:] Owner
 - [Password:] No password
 - Click on[OK]
- ▶ Double-click again on [Current User: ...] in status bar
- ▶ Sign in as s-admin:
 - [User name:] s-admin
 - [Password:] the corresponding password
 - Click on[OK]
- ▷ Login as s-admin is taking place.
- ▷ It is necessary to sign in one more time after restarting the project.
- ▷ The login details will be stored.

7.1.4 Enter information about applications

To identify and better differentiate between the applications in PLC_1 and PLC_2, enter meaningful information for both applications as follows:

- ▶ Right mouse click on [Application] > [Properties] > [Information] tab

- ▶ Enter meaningful texts at:
[Author]
[Version]
[Description], e.g. PLC control cabinet front or PLC control cabinet rear
- ▶ Close the window by clicking [OK]
- ▶ Save the project with [File] > [Save Project]
- ▷ After translating and loading to the corresponding PLC: The file `swinfo.txt` contains the corresponding information stored on the controller. (Software information (swinfo.txt))

7.1.5 Assigning the memory allocation - PLC_1 / PLC_2

This chapter describes how to assign the memory allocation to the device's internal PLCs PLC_1 and PLC_2.



When creating a project using the project template, the following memory layout is set by default:
[2303 kB PLC_1 flash | 2303 kB PLC_2 flash] = 2.25 MB PLC_1 / 2.25 MB PLC_2

7.1.5.1 Displaying and setting the memory allocation in CODESYS

- ▶ In the device tree of CODESYS: Double click on ecomatBasic (ecomatBasic).
- ▶ In the editor window: Select the [Device Variant] tab.
 - ▷ The editor window shows the settings for memory allocation.
 - ▷ The currently set memory allocation is highlighted in bold.

[Device Variant] (memory configuration)	IEC-Code PLC_1	IEC-RAM PLC_1	IEC-Code PLC_2	IEC-RAM PLC_2	Memconfig-file
3583 kB PLC_1 flash 1023 kb PLC_2 flash	3.5 Mbytes	1024 kBytes	1.0 Mbytes	512 kBytes	CRxxxS-mem- conf-3583S-1023.ifm
2303 kB PLC_1 flash 2303 kb PLC_2 flash (Default)	2.25 Mbytes	768 kBytes	2.25 Mbytes	768 kBytes	CRxxxS-mem- conf-2303S-2303.ifm
1023 kB PLC_1 flash 3583 kb PLC_2 flash	1.0 Mbytes	512 kBytes	3.5 Mbytes	1024 kBytes	CRxxxS-mem- conf-1023S-3584.ifm

- ▶ Highlight the required memory partition.
- ▶ Click on the button [Apply to project configuration].
 - ▷ The changed memory allocation is transferred to the CODESYS project.
- ▶ Click on the button [Apply to target].
 - ▷ The memory allocation is downloaded to the device.



When loading the memory allocation to the device, both IEC applications (PLC_1 application and PLC_2 application) are deleted!

7.1.6 Assigning inputs/outputs – PLC_1 / PLC_2

This chapter describes the assignment of inputs and outputs to the device-internal PLCs using the CODESYS plug-in for I/O assignment.



When creating a project using the project template, all inputs and outputs are assigned to PLC_2 by default.



If an input/output is accessed that is not assigned to the respective PLC, the diagnostic message DIAG_ACCESS appears at the corresponding function block.

The message only appears if there is not already an error, since the error message has a higher priority than the diagnostic message.

Access to the input/output is blocked by the system in all cases.



More information: [Configuring and using inputs and outputs \(→ 91\)](#)

7.1.6.1 Mapping I/Os in CODESYS

Before programming the application, map the inputs and outputs to the device-internal PLCs as described in the following steps.

- ▶ In the device tree of CODESYS: Double click on ecomatBasic (ecomatBasic).
 - ▷ The tab of the device appears in the editor window.
- ▶ In the editor window: Click on [IO-Mapping].
 - ▷ The [IO-Mapping] view appears.
- ▶ Either download the I/O assignment from the connected device (→ [Downloading an I/O mapping from the device 66](#)) or open an I/O assignment saved on the computer (→ [Opening an I/O mapping 66](#)), e.g. from the ifm package.
 - ▷ The I/O assignment is displayed in the [Configurator] tab.

Section	Element	[Resource]	[PLC_1]	[PLC_2]	[Default Value]
[Local_IO]	[Inputs]	[INxxxx]	O	O	
		...	...	...	
	[Outputs]	[OUTxxxx]	O	O	
		...	O	O	
[System_IO]	[System Inputs]	[VBB30]	O	O	
		[TemperatureChip0]	O	O	
	[System Outputs]	[VBBx_SW]	O	O	[On]
		...	O	O	[On]
[HMI]	[User_LEDs]	[UserLEDx]	O	O	
		...	O	O	

Tab. 1: Possible settings:

- ▶ In the columns [PLC_1] and [PLC_2]: Select the required inputs and outputs to assign them to the corresponding internal PLC.
- ▶ Optional: Set values in the [Default Value] column.
- ▶ Optional: In the [File-Editor] tab, you can view and edit the same I/O mapping as an I/O mapping configuration file.
- ▶ Optional: Check or note down the checksum values [Header CRC] and [File CRC].

ATTENTION

Faulty configuration

- ▷ Malfunction is possible. The plugin does not validate the configuration.
- ▶ Before loading the configuration to the device: Make sure that the entered configuration is correct.

- ▶ Save the I/O mapping (→ [Saving an I/O mapping 66](#)) or load it to the device (→ [Loading an I/O mapping to the device 66](#)).

Downloading an I/O mapping from the device

- ▶ In the device tree of CODESYS: Double click on ecomatBasic (ecomatBasic).
 - ▷ The tab of the device appears in the editor window.
- ▶ In the editor window: Click on [IO-Mapping].
 - ▷ The [IO-Mapping] view appears.
- ▶ Click on the [Load from device] button.
- ▷ The I/O mapping saved on the device is downloaded from the device and displayed in the [Configurator] area.

Opening an I/O mapping

- ▶ In the device tree of CODESYS: Double click on ecomatBasic (ecomatBasic).
 - ▷ The tab of the device appears in the editor window.
- ▶ In the editor window: Click on [IO-Mapping].
 - ▷ The [IO-Mapping] view appears.
- ▶ Click on the [Open] button.
- ▶ Select a saved I/O mapping.
- ▷ The saved I/O mapping is loaded and displayed in the [Configurator] area.

Loading an I/O mapping to the device

- ▶ In the device tree of CODESYS: Double click on ecomatBasic (ecomatBasic).
 - ▷ The tab of the device appears in the editor window.
- ▶ In the editor window: Click on [IO-Mapping].
 - ▷ The [IO-Mapping] view appears.
- ▶ Load or open an I/O mapping from the connected device and edit it if necessary.

ATTENTION

Faulty configuration

- ▷ Malfunction is possible. The plugin does not validate the configuration.
- ▶ Before loading the configuration to the device: Make sure that the entered configuration is correct.

- ▶ Click on the [Transfer to device] button.
- ▷ The I/O mapping is loaded to the connected device.
- ▷ After a power-on reset of the device, the I/O mapping becomes active.



More information: [Configuring and using inputs and outputs](#) (→ [91](#))

Saving an I/O mapping

- ▶ In the device tree of CODESYS: Double click on ecomatBasic (ecomatBasic).
 - ▷ The tab of the device appears in the editor window.
- ▶ In the editor window: Click on [IO-Mapping].
 - ▷ The [IO-Mapping] view appears.
- ▶ Load or open an I/O mapping from the connected device and edit it if necessary.
- ▶ Click on the [Save as] button.
- ▶ Select a memory location and file name for the I/O mapping.

▷ The I/O mapping will be saved on the computer.



More information: [Configuring and using inputs and outputs](#) (→ [91](#))

7.1.7 Manage files



When the transmission starts, the entire controller goes into the UPDATE mode.

In this mode, all outputs of the device will be switched off and both applications (PLC_1 und PLC_2) will be stopped. (→ [Operating states](#) [113](#))



For this action, you need to sign in as device user.

For more user convenience:

- ▶ Create user in project and perform user login only once
Create a user in the CODESYS project
[Sign the user in to the CODESYS project](#) (→ [63](#))

To transfer files between a PC and the device:

Requirements:

- The device is connected to the PC via an internal service interface.

1 Selecting file view

- ▶ In the device tree: Double click on [ecomatBasic (ecomatBasic)].
- ▶ In the editor window: Select the [Files] tab.
- ▶ Click on the [Refresh] icon.
 - ▷ The editor window shows the folder structure on the PC on the left and the folder structure on the device on the right

2 Transferring a file from PC to the device

- ▶ Highlight the file on the left.
- ▶ Select the device's target directory on the right
- ▶ Start transfer using the [>>] button
 - ▷ The file is transferred to the device.

3 Transferring a file from the device to the PC

- ▶ Highlight the file on the right
- ▶ Select the PC's target directory on the left.
- ▶ Start the transfer by clicking on the [<<] button.
 - ▷ The file is transmitted to the sensor.

7.1.8 User-defined data



- ▶ Familiarise yourself with the following CODESYS functions:
CODESYS Symbol Configuration: → Online help > CODESYS Communication > Symbol Configuration



- ▶ Use the device user `admin` to transfer user-defined files to the device.

A maximum of 2 symbol files of the CODESYS symbol configuration can be saved as user-defined files (user files) in the directory `data` on the device.

7.2 Configuring PLCs



- Familiarise yourself with the following CODESYS functions:
Generic device editor:
→ Online Help > CODESYS Development System > Reference user interface > Objects > Object 'Device' and Generic device editor

The device-internal PLCs are configured using the "Generic device editor" of the CODESYS programming system. The programmer can access the device editor via the nodes `PLC_1` and `PLC_2` in the device tree.

To configure the device-internal PLCs:

- In the device tree: Double-click on `[PLC_1]` or `[PLC_2]`.
 - ▷ The editor window shows the device editor of the selected device-internal PLC.
- Configure the selected PLC.
- Save the project to adopt the changes.



- Add meaningful information to the applications: Enter information about applications

7.2.1 Configure task processing



- Familiarise yourself with the following CODESYS functions:
Task configuration:
→ Online help > CODESYS Development System > Programming of Applications > Task Configuration

Parameters control the processing of the tasks. The user can set the parameters for each task:

- Create new task. (maximum 4 per PLC)
- Configure task properties:
 1. [Priority]: admissible = 0 (high) ... 3 (low)
(select a priority for each task)
 - 2nd [Typ]: Cyclical
 3. [Interval]: Interval of the task call-ups in [ms]
The interval time must be longer than the processing time of the task.
- Recommended: Activate watchdog: Configure IEC watchdog
The watchdog time must be shorter than the interval time.
The watchdog time must be longer than the processing time of the task.
- Assign the sub-program with POU's to the newly created task.

Requirements for task interval time and watchdog time:

	minimum admissible	maximum admissible
Task interval time	4 ms	2000 ms
Watchdog time	2 ms	1998 ms

When the project is created, CODESYS automatically creates the following task using the project template:

Name	Description	Default settings
Task	Task for the processing of the main program [PLC_PRG (PRG)]	Priority: 1 Type: Cyclical Interval: t#10ms Watchdog activated Time: t#8ms Sensitivity: 1



By default, CODESYS assigns the CAN communication to the task with the shortest interval time. If this is not wanted or if the CAN buses are operating at very high capacity:

- ▶ Configuring task properties:
 1. [Priority]: high
 2. [Type]: Cyclical
 3. [Interval]: requested cycle time (=transmission interval)
- ▶ Assign sub-programs with the POU's for CAN communication to the CAN tasks.



▶ Please note regarding the applications of the PLCs:

- ▶ Multitasking for accessing inputs and outputs with function blocks is only supported to a limited extent.
- ▶ Only create full and parameter setting access to an input from one task!!
For further accesses from other tasks, only MODE_INPUT = MONITOR (→ [MONITOR operating mode for inputs and outputs](#) 91) may be used. (MODE_INPUT (ENUM) (→ 187))
- ▶ Only create full and parameter setting access to an input from one task!
For access from other tasks, only the MODE_OUTPUT = MONITOR (→ [MONITOR operating mode for inputs and outputs](#) 91) may be used. (MODE_OUTPUT (ENUM) (→ 187))



More information on task configuration:

Task configuration example

7.2.2 Add function libraries to the application

The ifm package includes ifm function libraries for the programming of the device under CODESYS. The libraries are installed in CODESYS together with the ifm package.

Users can add individual libraries to an application they need for the programming.



- ▶ Familiarise yourself with the following CODESYS functions:
Library manager:
→ Online help > CODESYS Development System > Using Libraries > Adding a Library to the Application

7.3 Use memory

7.3.1 Data exchange between PLC_1 and PLC_2;



- ▶ Familiarise yourself with the following CODESYS functions: EVC, Exchange Variable Connection and EVL, Exchange Variable List.

Data exchange between the PLCs is only permitted via the data of PLC_1.

- Recommendation: Perform data exchange between PLC_1 and PLC_2 with EVC (Exchange Variable Connection) and EVL (Exchange Variable List):

PLC	Mechanism	Description
PLC_2	EVC	The variables that are to be used from the PLC_1 application in the PLC_2 application are selected in an editor.
PLC_1	EVL	Special type of GVL (Global Variable List) for declaring variables that are used to exchange non safe data between the PLC_1 application and the PLC_2 application. All variables of the EVL are stored in the non safe memory area. This enables write access from the PLC_2 application.

7.3.2 Stack

For each IEC task, 2.5 KB of stack are available to the IEC application.

Variables of IEC functions (inputs, outputs, local variables) are stored on the stack.

Especially nested calls of functions as well as functions with strings, arrays and structures occupy a lot of stack. (→ [Example: High stack consumption](#) □ 70)



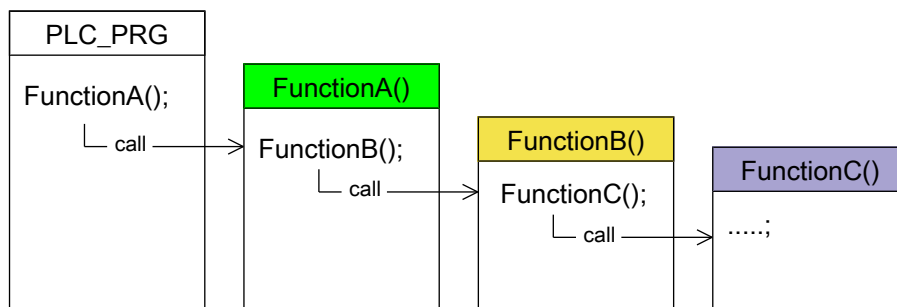
The stack consumption is monitored. If the maximum available stack is exceeded, the machine goes into FATAL_ERROR.

Programming measures for stack optimisation:

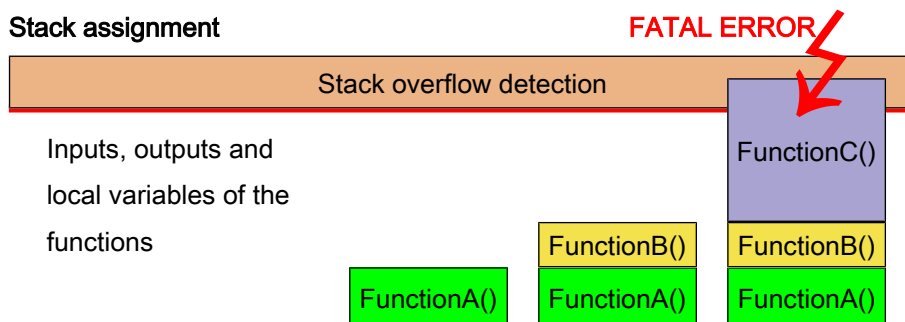
- Divide nested function calls into several individual calls. (→ [Example: Low stack consumption](#) □ 71)
- Save local variables as global variables.
- Convert functions into function blocks.

7.3.2.1 Example: High stack consumption

Program structure: Nested function calls



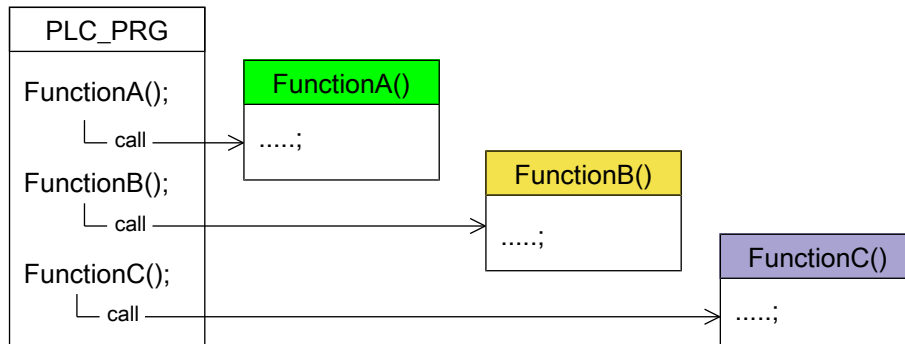
Stack assignment



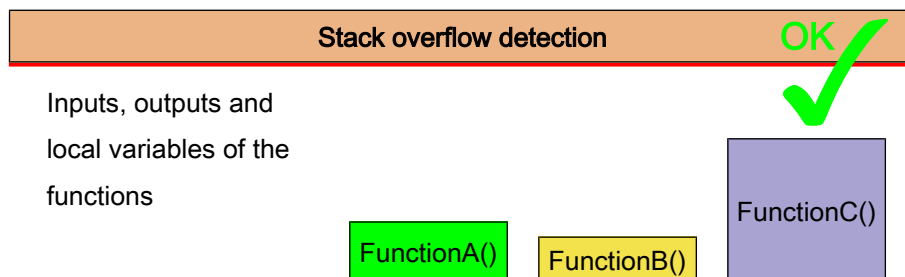
Nested function calls result in high stack consumption and can lead to a FATAL ERROR due to a stack overflow.

7.3.2.2 Example: Low stack consumption

Program structure: Sequential function call (single calls)



Stack assignment



Successively executed single calls of functions relieve the stack.

7.4 Configure IEC watchdog



- Familiarise yourself with the following CODESYS functions:

Watchdog:

→ Online help > CODESYS Development System > Programming of Applications > Task Configuration > Creating a Task Configuration > 'Configuration' tab

Task Configuration:

→ Online help > CODESYS Development System > Programming of Applications > Task Configuration

To configure the IEC watchdog of a task:

- Open the task configuration. Configure task processing
- Enable the watchdog with the option field [Enable]
- Enter watchdog [Time]
- Set the [Sensitivity]
- ▷ Watchdog is configured



- The watchdog time must be shorter than the interval time.
- The watchdog time must be longer than the processing time of the task.

7.5 Configure interfaces

7.5.1 Configuring interfaces in CODESYS

- ▶ In the device tree of CODESYS: Double click on ecomatBasic (ecomatBasic).
 - ▷ The tab of the device appears in the editor window.
- ▶ In the editor window: Click on [Device Interfaces].
 - ▷ The [Device Interfaces] view appears.
- ▶ Load the interface configuration from the connected device (→ [Downloading an interface configuration from the device](#) □ 72) or load an interface configuration saved on the computer (→ [Opening an interface configuration](#) □ 73), e.g. from the ifm package.
- ▶ Configure the interfaces in the [Configurator] area.

ATTENTION

Faulty configuration

- ▷ Malfunction is possible. The plugin does not validate the configuration.
- ▶ Before loading the configuration to the device: Make sure that the entered configuration is correct.

- ▶ Optional: In the [File-Editor] tab, you can view and edit the same interface configuration as a configuration file.
- ▶ Optional: Check or note down the checksum values [Header CRC] and [File CRC].
- ▶ Load the interface configuration to the connected device (→ [Loading an interface configuration to the device](#) □ 73) or save it on the computer (→ [Saving an interface configuration](#) □ 73).

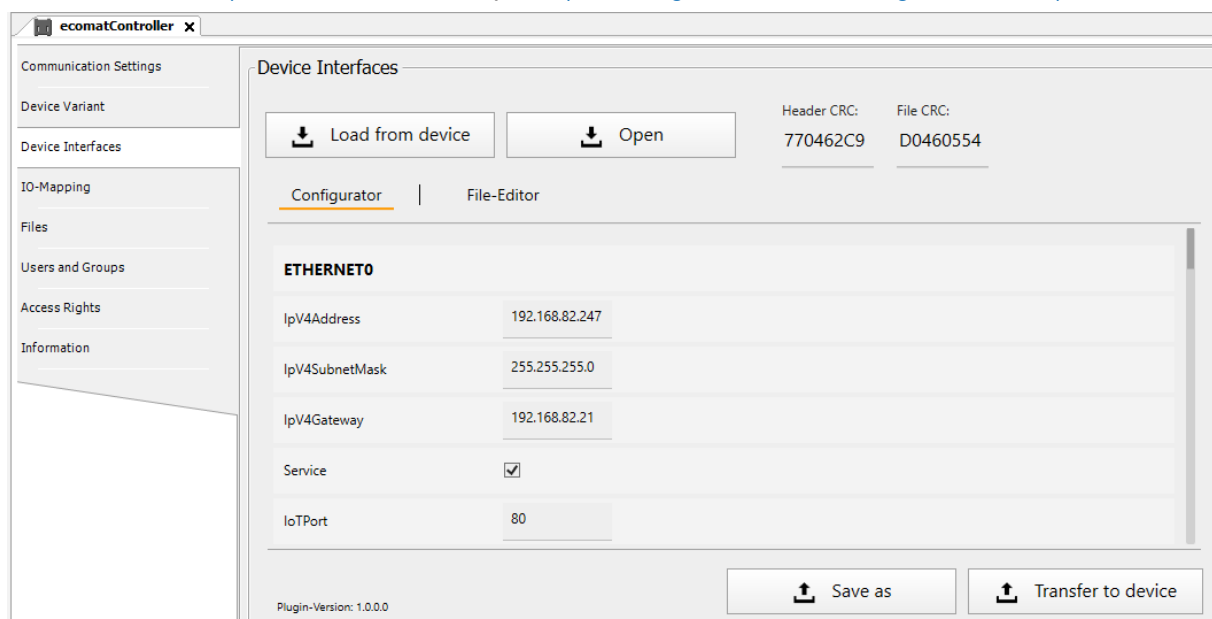


Fig. 13: Interface plugin

7.5.1.1 Downloading an interface configuration from the device

- ▶ In the device tree of CODESYS: Double click on ecomatBasic (ecomatBasic).
 - ▷ The tab of the device appears in the editor window.
- ▶ In the editor window: Click on [Device Interfaces].
 - ▷ The [Device Interfaces] view appears.
- ▶ Click the [Load from device] button.

- ▷ The interface configuration saved on the device is downloaded from the device and displayed in the [Configurator] area.

7.5.1.2 Opening an interface configuration

- ▶ In the device tree of CODESYS: Double click on ecomatBasic (ecomatBasic).
 - ▷ The tab of the device appears in the editor window.
- ▶ In the editor window: Click on [Device Interfaces].
 - ▷ The [Device Interfaces] view appears.
- ▶ Click the [Open] button.
- ▶ Select a saved interface configuration.
- ▷ The saved interface configuration is loaded and displayed in the [configurator] area.

7.5.1.3 Loading an interface configuration to the device

- ▶ In the device tree of CODESYS: Double click on ecomatBasic (ecomatBasic).
 - ▷ The tab of the device appears in the editor window.
- ▶ In the editor window: Click on [Device Interfaces].
 - ▷ The [Device Interfaces] view appears.
- ▶ Load or open an interface configuration from the connected device and edit it if necessary.

ATTENTION

Faulty configuration

- ▷ Malfunction is possible. The plugin does not validate the configuration.
- ▶ Before loading the configuration to the device: Make sure that the entered configuration is correct.

- ▶ Click the [Transfer to device] button.
- ▷ The interface configuration is loaded to the connected device.
- ▷ After a power-on reset, the interface configuration becomes active unless the device is started with factory settings.

7.5.1.4 Saving an interface configuration

- ▶ In the device tree of CODESYS: Double click on ecomatBasic (ecomatBasic).
 - ▷ The tab of the device appears in the editor window.
- ▶ In the editor window: Click on [Device Interfaces].
 - ▷ The [Device Interfaces] view appears.
- ▶ Load or open an interface configuration from the connected device and edit it if necessary.
- ▶ Click the [Save as] button.
- ▶ Select a memory location and file name for the interface configuration.
- ▷ The interface configuration will be saved on the computer.

7.5.2 Configuring the Ethernet address

- ▶ Use the ifm plug-in [Device Interfaces] in CODESYS to configure the device interfaces: [Configuring interfaces in CODESYS \(→ 72\)](#)

Factory setting of the Ethernet interface:

Setting	Value
IP address	192.168.82.247
Subnet mask	255.255.255.0
Gateway address	192.168.82.21

ATTENTION

If the device is operated in an unprotected network environment.

- ▷ Unauthorised read or write access to data is possible.
- ▷ Unauthorised manipulation of the device function is possible.
- ▶ Check and restrict access options to the device.
- ▶ Restrict access to authorised users.
- ▶ Choose a secure method to connect with the device (e.g. VPN).
- ▶ Use encrypted data transmission (e.g. https / TLS).



- ▶ Receiving too many data packages may lead to data loss!

7.5.2.1 Address assignment in Ethernet networks

In the Ethernet network every IP address **MUST** be unique.

The following IP addresses are reserved for network-internal purposes and are therefore not allowed as an address for participants: nnn.nnn.nnn.0 | nnn.nnn.nnn.255.

Only network participants whose subnet mask is identical and whose IP addresses are identical with respect to the subnet mask can communicate with each other.

Rule:

If part of the subnet mask = 255, the corresponding IP address parts must be identical.

If part of the subnet mask = 0, the corresponding IP address parts must be different.

If the subnet mask = 255.255.255.0, 254 participants communicating with each other are possible in the network.

If the subnet mask = 255.255.0.0, then $256 \times 256 = 65\,024$ participants communicating with each other are possible in the network.

In the same physical network different subnet masks of the participants are allowed. They form different groups of participants which cannot communicate with groups of participants having other subnet masks.



- ▶ In case of doubt or problems please contact your system administrator.

Examples:

Participant A IP address	Participant A Subnet mask	Participant B IP address	Participant B Subnet mask	Communication of participants possible?
192.168.82.247	255.255.255.0	192.168.82.10	255.255.255.0	Yes, 254 participants possible
192.168.82. 247	255.255.255.0	192.168.82. 247	255.255.255.0	No (same IP address)

Participant A IP address	Participant A Subnet mask	Participant B IP address	Participant B Subnet mask	Communication of participants possible?
192.168.82.247	255.255. 255 .0	192.168.82.10	255.255. 0 .0	No (different subnet mask)
192.168. 82 .247	255.255.255.0	192.168. 116 .10	255.255.255.0	No (different IP address range: 82 vs. 116)
192.168.222.213	255.255.0.0	192.168.222.123	255.255.0.0	Yes, 65 024 participants are possible
192.168.111.213	255.255.0.0	192.168.222.123	255.255.0.0	Yes, 65 024 participants are possible
192.168.82.247	255.255.255.0	192.168.82. 0	255.255.255.0	no; the whole network is disturbed be- cause the IP address xxx.xxx.xxx.0 is not allowed

7.5.2.2 Using Modbus



- Familiarise yourself with the following CODESYS functions:
Modbus master: → Online Help > Fieldbus Support > Modbus Configurator > Modbus Master

Modbus slave device: → Online Help > Fieldbus Support > Modbus Configurator > Modbus Slave Device

To use the Modbus network protocol, set the configuration via the device tree as follows:

Adding an Ethernet adapter to an Ethernet

- In the CODESYS device tree: Right-click on [Communication] > [Ethernet].
- Select [Add Device...].
 - ▷ The window [Add Device] appears.
- Select <All vendors> in the [Vendor] list.
- In the list below, select [Fieldbuses] > [Ethernet Adapter] > Ethernet [Adapter] > [Ethernet].
- Conform the selection with [Add Device]
- Close the window [Add Device] with the [Close] button.
- ▷ The Ethernet adapter is attached.

Attaching a Modbus TCP master

- In the CODESYS device tree: Right-click on [Communication] > [Ethernet] > [Ethernet_1].
- Select [Add Device...].
 - ▷ The window [Add Device] appears.
- Select <All vendors> in the [Vendor] list.
- In the list below, select [Fieldbuses] > [Modbus] > Modbus [TCP Master] > Modbus [TCP Master].
- Conform the selection with [Add Device]
- Close the window [Add Device] with the [Close] button.
- ▷ The TCP master mode is attached.

Attaching a Modbus TCP slave device

- In the CODESYS device tree: Right-click on [Communication] > [Ethernet] > [Ethernet_1] > [Modbus_TCP_Master].
- Select [Add Device...].
 - ▷ The window [Add Device] appears.
- Select <All vendors> in the [Vendor] list.

- ▶ In the list below, select [Fieldbuses] > [Modbus] > [Modbus TCP Slave Device] > Modbus [TCP Slave Device.]
- ▶ Conform the selection with [Add Device]
- ▶ Close the window [Add Device] with the [Close] button.
- ▷ The TCP Slave Device mode is added to the Modbus TCP Master.

Configuring a Modbus TCP slave device

- ▶ In the CODESYS device tree: Double-click on [Communication] > [Ethernet] > [Ethernet_1] > [Modbus_TCP_Master] > [Modbus_TCP_Slave].
 - ▷ The configuration window for the Modbus_TCP_Slave appears.
- ▶ In the following tabs, set the parameters as required:
 - [General]: Enter the IP address of the Modbus_TCP_Slave. Optional: Enter [Response Timeout] and [Port].
 - [Modbus Slave Channel]: Add the required Modbus channels (Read Holding Registers).
 - [Modbus TCPSlave I/O Mapping]: Define variables and assign the values of the Modbus registers to them.



- ▶ For more information about the Modbus settings, the Modbus communication and the register configuration of the respective Modbus slave: → Operating instructions of the Modbus slave.
- ▶ After completing the Modbus configuration, evaluate the variables in the application program.

7.5.3 Configuring CAN interfaces in the device tree

The CAN interfaces are configurable as follows:

- via system configuration:
 - CANopen
 - SAE J1939
- via function block:
 - RAW-CAN

The following entries are available at **Vendor = 3S**:

- CIA CANopen
 - CIA CANopenManager
 - +– CANopen_Manager
 - CIA Local Device
 - +– CANopen Device
- SAE J1939
 - SAE J1939 Manager
 - +– J1939_Manager



When configuring CANopen and SAE J1939 via system configuration, please note the following:

- ▶ Use at least version 4.1 of the CODESYS add-ins for configuring CANopen and SAE J1939. If necessary, update the version using the CODESYS Installer.
- ▶ Select the task suitable for CAN communication globally at [PLC settings] under [Bus cycle options] or locally in the Manager component at [IO-Mapping] under [Bus cycle options]. Otherwise, CODESYS will attach the communication stack to the task with the shortest task interval (setting: [<unspecified>]). This might lead to an unexpected time response of the CAN bus.
- ▶ Protocol messages (e.g. SYNC, PDO, PGN/SPN, etc.) can only be processed at the interval of the set task.
- ▶ The task interval should not exceed half of the bus interval (SYNC, repetition time, etc.).
- ▶ In the settings of the CANopen manager, the validity of the time settings can be verified via [Check and Fix Configuration].



As of CODESYS version 3.5 SP17, the J1939 configurator is no longer supplied with a parameter group / signal database. The old database is no longer supported.

CODESYS is able to install a DBC database in the J1939 Manager.

Some J1939 clients also have a DBC file that can be imported into CODESYS.

A database can also be purchased. (Recommended)

If no database is installed, parameter groups and signals can also be configured manually on the [User-defined] tab. (Not recommended, because it is error-prone)

7.5.3.1 via system configuration: CANopen Manager

In the CODESYS device tree, you will find the following entry under each PLC:
[Communication] > [CAN]



- ▶ Configure each interface only at ONE position!

• Add CAN bus

- ▶ In the CODESYS device tree: Right-click on [Communication] > [CAN].
- ▶ Select [Add Device...].
 - ▷ The [Add Device] window appears.
- ▶ Under [Device]: Vendor: Select [ifm electronic].
- ▶ In the list below: Select [ifmCANBus].
- ▶ Confirm the selection with [Add Device].
- ▶ Close the [Add Device] window with the [Close] button.

• Assign CAN interface

- ▶ In the CODESYS device tree: Double-click [Communication] > [CAN] > [ifmCANBus].
- ▶ Tab [General] > [General] > [Network]:
Assign this setting to a CAN interface with ▲/▼.
- ▶ Select the required [Baudrate (bit/s)] value from the list.

• Add CANopen manager

- ▶ In the CODESYS device tree: Right-click on [Communication] > [CAN] > [ifmCANBus].
- ▶ Select [Add Device...].

- ▷ The [Add Device] window appears.
- ▶ Under [Device]: [Vendor]: Select [<All vendors>].
- ▶ In the list below: Select [Fieldbuses] > [CiA CANopen] > [CiA CANopenManager] > [CANopenManager].
- ▶ Confirm the selection with [Add Device].
- ▶ Close the [Add Device] window with the [Close] button.

- **Configure CANopen manager**

- ▶ In the CODESYS device tree: Double-click on [Communication] > [CAN] > [ifmCANBus] > [CANopen_Manager].
- ▶ Tab [General] > [General] > [Node ID]:
Assign a node ID to this interface with ▲/▼.
- ▶ Select the other parameters as required, e.g.:
Configure the heartbeat protocol under [Guarding]:
- ▶ Tick the box to enable [Heartbeat Producing].
- ▶ Set the [Node ID] and [Producer Time (ms)] parameters.
Configure the sync protocol in the [Sync] section:
- ▶ Tick the box to enable [Sync Producing] if required.
- ▶ Set the parameters [COB-ID (Hex)], [Cycle Period (µs)] and [Window Length (µs)].
In the [Time] section: The time protocol is not supported.
- ▷ By selecting [File] > [Save Project] from the menu, the values become valid.



The sync protocol triggers the receiving/sending of data of the CANopen devices (input: SDO 16#1800/ output: SDO 16#1400).

- **Attach CANopen remote device**

- ▶ In the CODESYS device tree: Right-click on [Communication] > [CAN] > [ifmCANBus] > [CANopen_Manager].
- ▶ Select [Add Device...].
 - ▷ The [Add Device] window appears.
- ▶ Under [Device]: [Vendor]: Select [<All vendors>].
- ▶ In the list below: Select a remote device under [Fieldbuses] > [CiA CANopen] > [CiA Remote Device].
- ▶ Confirm the selection with [Add Device].
- ▶ Close the [Add Device] window with the [Close] button.

- **Configure CANopen remote device**

- ▶ In the CODESYS device tree: Double-click remote device under [Communication] > [CAN] > [ifmCANBus] > [CANopen_Manager].
- ▶ Select/set the parameters as required



When activating the option field [Factory Settings]: Each time the controller is switched on or the program is downloaded, the settings are reset to factory settings. Thereby, user settings can be overwritten. The type and the scope of reset settings depend on the device.

- ▷ By selecting [File] > [Save Project] from the menu, the values become valid.

7.5.3.2 via system configuration: CANopen device

In the CODESYS device tree, you will find the following entry under each PLC:
[Communication] > [CAN]

These entries are equivalent.

- **Add CAN bus**

- ▶ In the CODESYS device tree: Right-click on [Communication] > [CAN].
- ▶ Select [Add Device...].
 - ▷ The [Add Device] window appears.
- ▶ Under [Device]: Vendor: Select [ifm electronic].
- ▶ In the list below: Select [ifmCANBus].
- ▶ Confirm the selection with [Add Device].
- ▶ Close the [Add Device] window with the [Close] button.

- **Assign CAN interface**

- ▶ In the CODESYS device tree: Double-click [Communication] > [CAN] > [ifmCANBus].
- ▶ Tab [General] > [General] > [Network]:
Assign this setting to a CAN interface with ▲/▼.
- ▶ Select the required [Baudrate (bit/s)] value from the list.

- **Add CANopen device**

- ▶ In the CODESYS device tree: Right-click on [Communication] > [CAN] > [ifmCANBus].
- ▶ Select [Add Device...].
 - ▷ The [Add Device] window appears.
- ▶ Under [Device]: Vendor: select <Alle Hersteller>.
- ▶ In the list below: Select [Fieldbuses] > [CiA CANopen] > [Local Device] > [CANopenDevice].
- ▶ Confirm the selection with [Add Device].
- ▶ Close the [Add Device] window with the [Close] button.

- **Set CANopen device parameters**

- ▶ In the CODESYS device tree: Double-click on [Communication] > [CAN] > [Local Device] > [CANopenDevice].
- ▶ Tab [General] > [General] > [Node ID]:
Assign a node ID to this interface with ▲/▼.
- ▶ Select the further parameters according to the requirements, e.g.: [Heartbeat Producing]
 - ▷ By selecting [File] > [Save Project] from the menu, the values become valid.



The correct baud rate and node ID must be set in the CANopen device so that the CANopen master will recognise the device.

7.5.3.3 via system configuration: J1939 manager

In the CODESYS device tree, you will find the following entry under each PLC:
[Communication] > [CAN]

These entries are equivalent.

- **Add CAN bus**

- ▶ In the CODESYS device tree: Right-click on [Communication] > [CAN].
- ▶ Select [Add Device...].
 - ▷ The [Add Device] window appears.
- ▶ Under [Device]: Vendor: Select [ifm electronic].
- ▶ In the list below: Select [ifmCANBus].
- ▶ Confirm the selection with [Add Device].
- ▶ Close the [Add Device] window with the [Close] button.

- **Assign CAN interface**

- ▶ In the CODESYS device tree: Double-click [Communication] > [CAN] > [ifmCANBus].
- ▶ Tab [General] > [General] > [Network]:
Assign this setting to a CAN interface with ▲/▼.
- ▶ Select the required [Baudrate (bit/s)] value from the list.

- **Add J1939 Manager**

- ▶ In the CODESYS device tree: Right-click on [Communication] > [CAN] > [ifmCANBus].
- ▶ Select [Add Device...].
 - ▷ The [Add Device] window appears.
- ▶ Under [Device]: Vendor: select <Alle Hersteller>.
- ▶ In the list below: Select [Fieldbuses] > [SAE J1939] > [J1939 Manager] > [J1939_Manager] .
- ▶ Confirm the selection with [Add Device].
- ▶ Close the [Add Device] window with the [Close] button.

- **Configure J1939 Manager**

- ▶ In the CODESYS device tree: Double-click on [Communication] > [CAN] > [J1939_Manager].
- ▶ Tab [General] > [Database] > [Database]:
Select the required database from the list.
Default = J1939Default
 - ▷ By selecting [File] > [Save Project] from the menu, the values become valid.



Users can use their own databases.

These databases must be located at the following path: C:\ProgramData\CODESYS\J1939 Databases

The ProgramData directory is hidden by default.

- **Add a J1939-ECU**

- ▶ In the CODESYS device tree: Right-click on [Communication] > [CAN] > [ifmCANBus] > [J1939_Manager].
- ▶ Select [Add Device...].
 - ▷ The [Add Device] window appears.
- ▶ Under [Device]: Vendor: select <Alle Hersteller>.
- ▶ In the list below: Select [Fieldbuses] > [J1939] > [J1939_ECU].
- ▶ Confirm the selection with [Add Device].

► Close the [Add Device] window with the [Close] button.

• Set J1939-ECU parameters

- In the CODESYS device tree: Double-click on [Communication] > [CAN] > [J1939_Manager] > [J1939_ECU].
- In the [General] tab, under [General], make the following settings based on the application:

Application	[Local device]		Significance [Preferred Address]
• Receiving broadcast data of the ECU • No transmission	☐	Deactivated	Address of the ECU from which the data is to be received
• Sending data (broadcast and P2P) • Receiving P2P data	☒	Activated	Address of the ifm controller

- Add parameter groups in the tab [TX signals] by clicking on [Add PG].
 - ▷ By selecting [File] > [Save Project] from the menu, the values become valid.

7.5.3.4 via function block: RAW-CAN

The library ifmRawCAN.library (→ [Library ifmRawCAN.library](#) ▢ 126) features several function blocks for this application.

7.5.4 Interface configuration file comconf.cfg

- Use the ifm plug-in [Device Interfaces] in CODESYS to configure the device interfaces: [Configuring interfaces in CODESYS](#) (→ ▢ 72)

The interface configuration of the device is saved in the file `comconf.cfg`.

The `comconf.cfg` file is located in the `/com` file directory of the device.

7.5.4.1 Parameter comconf.cfg

The file `comconf.cfg` contains the following sections and parameters:

Section [ETHERNETx]

This area contains the settings for an Ethernet interface of the device. If the device has several Ethernet interfaces, the area is available once for each interface.

Key	Value range	Description
IpV4Address	Valid IPv4 address	Acceptance only if IP address and subnet mask are correct.
IpV4SubnetMask	Valid IPv4 subnet mask	
IpV4Gateway	Valid IPv4 address	Optional
Service	TRUE, FALSE	Optional; TRUE: CODESYS service communication is activated on this interface. FALSE: CODESYS service communication is deactivated on this interface.
SlotConfig	TRUE, FALSE	Optional; TRUE: IP address setting with module slot detection: The value from the SLOT-CONFIG section is added to the last octet of the IP address. FALSE: IP address setting WITHOUT module slot detection. (Module slot detection)

Key	Value range	Description
IoTService	TRUE, FALSE	Optional; TRUE: The ifm IoT service interface is activated. FALSE: The ifm IoT service interface is deactivated.
IoTPort	Valid port	Optional; Ethernet port of the ifm IoT service interface

Section [CANx]

This area contains the settings for a CAN interface of the device. If the device has several CAN interfaces, the area is available once for each interface.

Key	Value range	Description
Baud rate	10000* 20000* 33300 50000* 83300 100000* 125000* 250000* 500000* 666600 800000* 1000000*	CAN baud rate Only baud rates marked with * can be used for service communication.
NodeId	1...127	Node ID of the CODESYS service communication
Service	TRUE, FALSE	Optional; TRUE: CODESYS service communication on the interface is activated. With CAN, only the interface that is read out first is activated as the service interface (if several are set to TRUE). FALSE: CODESYS service communication on the interface is activated.
SlotConfig	TRUE, FALSE	Optional; TRUE: Address setting with module slot detection: Value from SLOT-CONFIG is added to the specified NodeID. FALSE: Address setting without slot position detection. (Module slot detection)
IoTService	TRUE, FALSE	Optional; TRUE: The ifm IoT service interface is activated. FALSE: The ifm IoT service interface is deactivated.
IoTNodeId	1...127	Optional; CANopen NodeID of the ifm IoT service interface 123 – x = channel x = 0...maximum number of CAN channels - 1

Section [SLOT-CONFIG]

This area contains the settings for module slot detection (Slot-Config).

Key	Value range	Description
BIT0	IN0000...IN9999	LSB0 of the Slot-Config (Module slot detection) The input type must be IN-FREQUENCY-x.
BIT1	IN0000...IN9999	LSB1 of the Slot-Config (Module slot detection) The input type must be IN-FREQUENCY-x.

Key	Value range	Description
BIT2	IN0000...IN9999	LSB2 of the Slot-Config (Module slot detection) The input type must be IN-FREQUENCY-x.
BIT3	IN0000...IN9999	LSB3 of the Slot-Config (Module slot detection) The input type must be IN-FREQUENCY-x.

7.5.4.2 Starting with factory settings

To start the switched-off device with the factory settings (the user-defined settings in the file `comconf.cfg` are not taken into account):

- Apply the `TRUE` signal to the `RESET-COM` connection and switch on the device at the same time.
- After the start-up: Apply the `FALSE` signal to `RESET-COM`.

Content of `comconf.cfg` with factory settings

The `comconf.cfg` file has the following factory settings:

```
[ETHERNET0]
IPv4Address=192.168.82.247
IPv4SubnetMask=255.255.255.0
IPv4Gateway=192.168.82.21
Service=true
IoTPort=80
IoTService=true
SlotConfig=false
```

```
[CAN0]
Baudrate=250000
NodeId=127
Service=false
IoTNodeId=123
IoTService=false
SlotConfig=false
```

```
[CAN1]
Baudrate=250000
NodeId=126
Service=false
IoTNodeId=122
IoTService=false
SlotConfig=false
```

```
[SLOT-CONFIG]
BIT0=IN0400
BIT1=IN0401
BIT2=IN0500
BIT3=IN0501
```

7.5.4.3 Starting with changed settings

To start the switched-off device with the contents of the modified file `comconf.cfg`:

- Apply the `FALSE` signal to the `RESET-COM` connection and switch on the device at the same time.

The FB [GetComResetState](#) (→ [185](#)) is used to analyse in the application which interface configuration was used to start.

7.5.5 Module slot detection

- Use the ifm plug-in [Device Interfaces] in CODESYS to configure the device interfaces: [Configuring interfaces in CODESYS](#) (→ [72](#))

Module slot detection (Slot-Config) enables the automatic assignment of unique IP addresses when using multiple controllers in an application with a shared Ethernet network.

Module slot detection (Slot-Config) enables the automatic assignment of unique CAN bus node IDs when using multiple controllers in an application with a shared CAN bus system.

Operating principle:

- The module slot number is coded at the participating controllers with the help of up to 4 defined control inputs of the type IN_FREQUENCY.
- The last octet of the set IP base address of the Ethernet interface is automatically incremented by the module location number.
- The set node ID of the CAN interface is automatically increased by the module position number.
- The maximum 4 controller inputs for coding the module slot number are configured in the `comconf.cfg` file in the [SLOT-CONFIG] area using the entries BIT0 to BIT3. The inputs specified there must be permanently connected to VBB30 (= 1) or GND (= 0).
- Module slot detection is only applied to interfaces that have the entry `SlotConfig=true` in the `comconf.cfg` file.
- If a module slot number exceeds the value range, the offset 0 is used.
- If an unavailable interface is selected, it is ignored.

7.5.5.1 Example

Sample extract from the interface configuration file `comconf.cfg`:

```
[ETHERNET0]
IPv4Address=192.168.82.240
IPv4SubnetMask=255.255.255.0
IPv4Gateway=192.168.82.21
Service=true
SlotConfig=true
```

```
[CAN0]
Baudrate=250000
NodeId=127
Service=true
```

```
[CAN1]
Baudrate=250000
NodeId=116
Service=false
SlotConfig=true
```

```
[SLOT-CONFIG]
BIT0=IN0400
BIT1=IN0401
BIT2=IN0500
BIT3=IN0501
```

- Section [ETHERNET0] :
The entry `SlotConfig=true` indicates that module slot detection is activated for the Ethernet0 interface.
- Areas [CAN0] :
If the entry `SlotConfig` is missing or `SlotConfig=false`, module slot detection for the CAN0 interface will be deactivated.
- Section [CAN1] :
The entry `SlotConfig=true` indicates that module slot detection is activated for the CAN1 interface.
- Section [SLOT-CONFIG] :
The controller inputs IN0000...IN0003 are assigned to the entries `BIT0...BIT3`.

Calculation of the module slot number:

BIT0 = IN0400 (24 V) = 1, corresponds to $2^0 = 1$

BIT1 = IN0401 (24 V) = 1, corresponds to $2^1 = 2$

BIT2 = IN0500 (0 V) = 0, corresponds to 0

BIT3 = IN0501 (0 V) = 0, corresponds to 0

The result is: Module slot number = $1 + 2 = 3$

The result is: Automatically assigned IP address of Ethernet0 = $192.168.82.240 + 3 = 192.168.82.243$

The result is: Automatically assigned node ID of CAN1 = $116 + 3 = 119$

With the same `comconf.cfg` file and different input circuits, the result is, for example:

BIT0 = IN0400 (0 V) = 0, corresponds to 0

BIT1 = IN0401 (24 V) = 1, corresponds to $2^1 = 2$

BIT2 = IN0500 (0 V) = 0, corresponds to 0

BIT3 = IN0501 (24 V) = 1, corresponds to $2^3 = 8$

The result is: Module slot number = $2 + 8 = 10$

The result is: Automatically assigned IP address of Ethernet0 = $192.168.82.240 + 10 = 192.168.82.250$

The result is: Automatically assigned node ID of CAN1 = $116 + 10 = 126$

7.5.5.2 Behaviour in case of an error

If errors occur during the calculation for module slot detection, the device behaves as described in the table:

Condition	Description	Description	Reaction	Message DIAG	Error class
Wrong input type	If at least one input has no binary evaluation	Startup	From the first invalid input type, these and the following will be ignored for the off-set	----	---

Condition	Description	Description	Reaction	Message DIAG	Error class
BIT0 not available	If BIT0 is not available in SLOT-CONFIG	Startup	Offset 0	---	----
BIT not consecutive	If bits are not defined as consecutive	Startup	From the first non-consecutive BIT, these and the following will be ignored for the offset	----	----
Offset exceeds value range (IP address)	If the offset by BIT-CONFIG exceeds the value range of the last octet ($IpV4Address + Offset > 254$)	Startup	offset 0 is used for the IP address	---	---
Offset exceeds value range (NodeId)	If the offset by the BIT-CONFIG exceeds the valid value range of the NodeId ($NodeId + Offset > 127$)	Startup	offset 0 is used for the NodeId	---	---
Offset exceeds value range (IoTNodeId)	If the offset by the BIT-CONFIG exceeds the valid value range of the IoTNodeID ($IoTNodeId + Offset > 127$)	Startup	offset 0 is used for the IoTNodeId	---	---
Input name invalid	If the input name is Invalid (e.g. Invalid: IN0, INPUT0,0,0001, AN0000, ...)	Startup	From the first invalid input name, these and the following will be ignored for the offset	---	---

7.6 Configuring inputs and outputs

The inputs and outputs can be configured using function blocks.

More information: [Configuring and using inputs and outputs \(→ 91\)](#)

8 Programming of the standard application

8.1	Objects of a standard PLC application	87
8.2	Creating a standard PLC application	88
8.2.1	Supported programming languages	88
8.2.2	Data types	88
8.2.3	Supported variable types	88
8.2.4	Configuring and using inputs and outputs	91
8.2.5	Note the following when programming	94
8.3	Using ifm function libraries	94
8.3.1	Access to inputs	95
8.3.2	Access to outputs	95
8.3.3	Control device	95
8.3.4	Read device information	95
8.4	Using CAN	95
8.4.1	Use RawCAN (CAN Layer 2)	95
8.4.2	Use CANopen	96
8.4.3	Use SAE J1939	97
8.5	Using Ethernet	97
8.5.1	SysSocket interface	97
8.5.2	IoT service interface - sockets	98
8.5.3	Modbus	99
8.6	Error in the IEC application	99

8.1 Objects of a standard PLC application

All objects of a PLC application are listed as sub-elements of the node [PLC_2 (PLC_2)] > [PLC Logic] > [Application] in the device tree.

In the basic configuration, a standard PLC application includes the following objects:

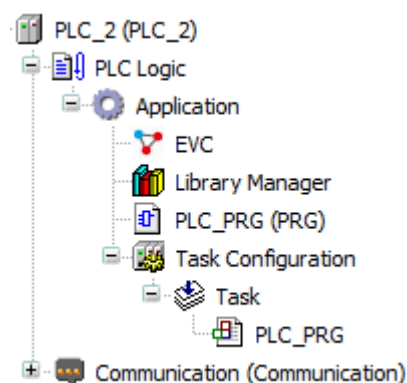


Fig. 14: Objects of a standard PLC application

[Application]	Container for objects of a PLC application
[EVC]	Exchange Variable Connection. Provides the PLC_2 application with variables from the PLC_1 application via a shared memory area. Data exchange between PLC_1 and PLC_2; (→ □ 69)
[Library Manager]	Provides access to standard and device-specific function libraries. Using ifm function libraries
[PLC_PRG (PRG)]	Provides access to the editor of the standard PLC application. Creating a standard PLC application
[Task Configuration]	Provides access to the settings of the task processing. Configure task processing

If necessary, the user can add further objects to the standard PLC application.

8.2 Creating a standard PLC application



- Familiarise yourself with the following CODESYS functions:
→ Online help > CODESYS Development System > Programming of Applications

CODESYS automatically creates the program block PLC_PRG (PRG) when the project is created [PLC_PRG (PRG)]. The function block is processed cyclically. Other programs are called in this function block.

To create a PLC application:

- In the device tree: Double click on [Application] > [PLC_PRG (PRG)]
 - ▷ The editor window shows the input mask of the selected programming language.
- Enter program code.

8.2.1 Supported programming languages

The following programming languages according to IEC 61131 may be used in standard program blocks:

- Function block diagram FUP/FBD
- Ladder diagram KOP/LD
- Structured text ST
- Sequential function chart AS/SFC
- Continuous function chart CFC

8.2.2 Data types



- Familiarise yourself with the following CODESYS functions:
Data types: → Online help > CODESYS Development System > Reference, Programming > Data Types

8.2.3 Supported variable types



- Familiarise yourself with the following CODESYS functions!
- Local variables
→ Online help > CODESYS Development System > Reference, Programming > Variable Types and special Variables > Local Variables– VAR
- Global variable list:
→ Online help > CODESYS Development System > Reference, Programming > Variable Types and special Variables > Global Variables - VAR_GLOBAL
- Network variables UDP
→ Online help > CODESYS Development System > Working with Control Networks > Network Variables
- Non-volatile variables - RETAIN PERSISTENT
→ Online help > Reference, Programming > Variable Types and special Variables > Remanent Variables - RETAIN, PERSISTENT

The device supports the following variable types:

Variable type	Declaration	Scope of validity	Memory behaviour
local	In the declaration part of the POU	Applies only to the POU in which it has been declared	volatile
locally PERSISTENT			non-volatile at: - Reset hot/cold - Download
global	In the global variable list (GVL)	Applies to all POUs of the project	volatile
global PERSISTENT			non-volatile at: - Reset hot/cold - Download
Network	In network variable lists	Values are available to all projects in the whole network if the variable is contained in their network variables lists. Only UDP network variables are supported.	volatile



For performance reasons, do not overuse 64 bit variables!

8.2.3.1 Non-volatile data

Different types of non-volatile data are supported:

- Retain variables / retain data (`VAR_PERSISTENT`): Variables declared as `VAR_PERSISTENT` generate non-volatile data.
- Memory bytes (`%MB` , `%MW` , etc.): Memory bytes are pre-defined non-volatile memory areas.



Please note: [Important information on memory bytes](#) (→ [89](#))

In the following, both types will be referred to as non-volatile data.

Non-volatile data includes the values that are stored in it during the following processes:

- Switching the device off and on
- Online rest (warm/cold) of the device
- Loading the PLC (download)

Typical uses for non-volatile variables are all variables whose values must not be lost when the device is switched off, e.g:

- Operating hours that are counted up and retained while the machine is in operation,
- position values of incremental encoders
- target values entered in the display unit
- machine parameters

Important information on memory bytes

ATTENTION

If a variable is declared at a memory address without a default initialisation value, the persistent data is overwritten with 0 each time Power-on reset of the device.

▷ Data loss may occur with memory bytes.

► In the declaration part of the POU, add the compiler attribute `{attribute 'no-init'}` above each variable that is not to be initialised implicitly. (→ CODESYS online help)

Programming example:

```

PROGRAM PLC_PRG
VAR
Var1 AT %MW0 : UINT; // This memory address will be set to 0 every power-on-
reset
{attribute 'no-init'}
Var2 AT %MW1 : UINT; // Will not be initialized
{attribute 'no-init'}
Var3 AT %MW2 : UINT; // Will not be initialized
END_VAR

```

Further possible solution:

- Use a pointer of the type or structure and assign it to the memory area.
- Declare the variable with `REFERENCE TO` to a memory bytes memory address.

Programming example:

```

Var3 : REFERENCE TO WORD := %MW2; // Diese Speicheradresse wird persistent
gespeichert

```



The reference only works for normal data types.

8.2.3.2 Function description of non-volatile data

After power-on, the non-volatile data is written from the non-volatile memory to the RAM buffer (0).

At the end of the cycle of the lowest priority task (1), non-volatile data is written into an RAM buffer (3, 4, 5).

The execution of the lowest priority task (1) is not interrupted by higher priority tasks (2) at the end of the cycle. This ensures the consistency of the non-volatile data.

The non-volatile data is written from the volatile RAM buffer to the non-volatile memory (7) at runtime in the OPERATING state if the data in the RAM buffer has changed.

Only the data of the last completely executed task will be stored in the RAM buffer (5). The data of an unfinished task (8) will not be stored.

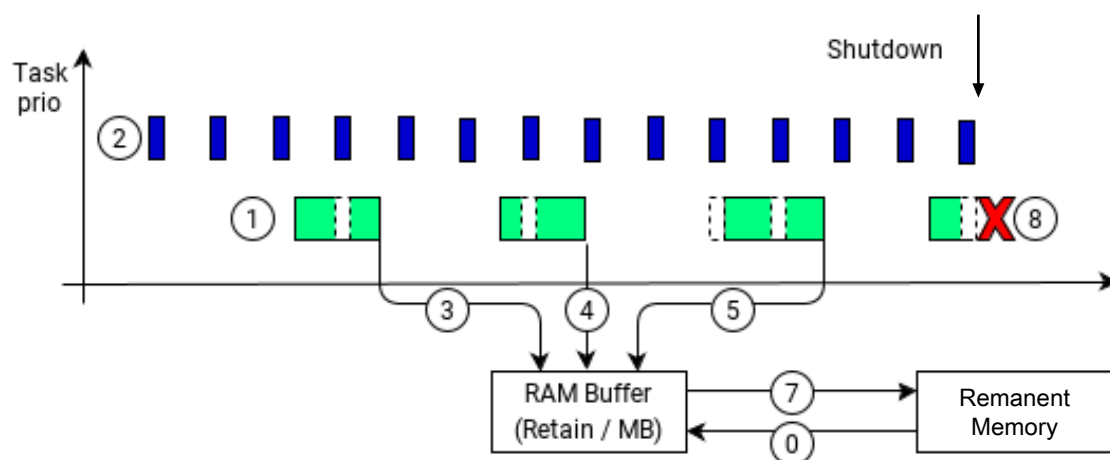


Fig. 15: Handling of non-volatile data



Data loss is possible with FATAL_ERROR.

- ▶ After a FATAL_ERROR and subsequent power-on reset of the controller, the last complete data written to the non-volatile memory is available.



Please note the following information:

- [Switching on and off](#) (→ [124](#))
- [Operating states](#) (→ [113](#))
- [Error classes](#) (→ [213](#))

8.2.4 Configuring and using inputs and outputs

8.2.4.1 Accessing inputs and outputs

The inputs and outputs of the device can be accessed using function blocks: [Using ifm function libraries](#) (→ [94](#))



For simple configuration and use of inputs and outputs of the PLC_2 (standard PLC):

- ▶ Use the IO-Wrapper.
- ▶ To do this, download the IO wrapper from ifm.com.

8.2.4.2 Setting the operating mode of inputs and outputs

The operating mode of an input, output or output group is set at the input `eMode` of the corresponding function block with ENUM values, e.g:

FB	FB input	ENUM
Input (→ Input 173)	<code>eMode</code>	MODE_INPUT (ENUM) (→ 187)
Output (→ Output 175)	<code>eMode</code>	MODE_OUTPUT (ENUM) (→ 187)
OutputGroup (→ OutputGroup 197)	<code>eMode</code>	MODE_OUTPUT_GROUP (ENUM)

If the `eMode` at the function block is changed:

- Pending diagnostic messages will be reset
- Measured values will be reset to 0 / FALSE
- Diagnostic settings will be set to standard values

MONITOR operating mode for inputs and outputs

Full access and parameter setting access via FB to a control input or a control output may only be established from a task and an application. ("main instance")

For reading and monitoring inputs and outputs from other tasks or applications, limited access can be made via FB using the MONITOR operating mode.

Function modules used with the MONITOR operating mode have the following properties and behaviour:

- Suitable for use in a PLC application or task to which the resource does not belong.
- No values, parameters, configurations or process data are written.
- Only the FB output data for reading and monitoring values is updated.
- Configuration error messages of the main instance FB are not displayed. All other (process) error messages of the main instance are displayed.
- The FB supplies default values until the main instance FB has reached the PREPARED state.

The operating mode MONITOR at an input FB or output FB is set using the ENUMS `MODE_x` (e.g. [MODE_INPUT \(ENUM\)](#) (→ [187](#))) at the FB input parameter `Mode`, e.g.:

► Set the `MODE_INPUT.MONITOR` at the instance of FB INPUT at the input parameter `Mode`.

8.2.4.3 Setting the filter of inputs and outputs

The filter of an input, output or output group is set at the input `eFilter` of the corresponding function block with ENUM values, e.g:

FB	FB input	ENUM
Input (→ Input 173)	<code>eFilter</code>	FILTER_INPUT (ENUM) (→ 186)
Output (→ Output 175)	<code>eFilter</code>	FILTER_OUTPUT (ENUM) (→ 186)
OutputGroup (→ OutputGroup 197)	<code>eFilter</code>	FILTER_OUTPUT_GROUP (ENUM) (→ 200)

Further information on the filters: [Filter](#) (→ [221](#))

8.2.4.4 Default behaviour of the inputs and outputs in case of an error



The following applies to inputs:

By default, automatic deactivation for exceeded measuring range is enabled → Technical data. The corresponding channel will be set to the safe substitute value (0 / FALSE) in case of an error.

- Automatic deactivation can be configured using the FB [ConfigDiagProt](#) (→ [193](#)).
- The threshold values for wire break and short-circuit detection can be configured using the FB [ConfigDiagLevel](#) (→ [190](#)).

For outputs, the following applies:

By default, automatic deactivation in the event of **stuck-at-high** and **excessive current** is enabled → Technical data. The corresponding channel switches off automatically in case of a fault.

- Automatic deactivation can be configured using the FB [ConfigDiagProt](#) (→ [193](#)).
- The threshold values for wire break and short-circuit detection can be configured using the FB [ConfigDiagLevel](#) (→ [190](#)).

8.2.4.5 Standard diagnostic limit values of inputs

In the following table, standard limit values of the diagnostic/error detection are indicated in the operating modes:

Operating mode	DetectionTime	Lower limit value	Upper limit value
IN_DIGITAL_CSI	10 ms	0% of VBB30	400% of VBB30
IN_DIGITAL_CSI_NAMUR	10 ms	1 V	95% of VBB30
IN_DIGITAL_CSO	10 ms	0% of VBB30	100% of VBB30
IN_VOLTAGE_10, IN_VOLTAGE_32, IN_CURRENT_CSI IN_RESISTOR	10 ms	0	maximum measuring range of the set operating mode (→ Technical data)
IN_VOLTAGE_RATIO	10 ms	0	1000 ‰

Changing the diagnostic limit values and the DetectionTime using the FB [ConfigDiagLevel](#) (→ [190](#)).

The following operating modes have no preset diagnostic functionality:

- IN_DIGITAL_CSI_BLANKING
- IN_COUNT_CSI
- IN_COUNT_CSO
- IN_FREQUENCY_CSI
- IN_FREQUENCY_CSO
- IN_INC_ENCODER_CSI
- IN_INC_ENCODER_CSO
- IN_PERIOD_RATIO_CSI
- IN_PERIOD_RATIO_CSO
- IN_PERIOD_RATIO_US_CSI
- IN_PERIOD_RATIO_US_CSO
- IN_PHASE_CSI
- IN_PHASE_CSO



The diagnostics for these operating modes can be done in the application, e.g. by evaluating the parameter `uiValueTime` (time since the last signal change at the input) on the corresponding function blocks.

8.2.4.6 Standard diagnostic limit values of outputs

In the following table, standard limit values of the diagnostic/error detection are indicated in the operating modes:

• OUT_DIGITAL_CSO • OUT_PWM_CSO • OUT_CURRENT_CSO • OUT_H_BRIDGE • OUT_SENSOR_05 • OUT_SENSOR_10	10 ms	0 mA	→ maximum switching current in the data sheet

Changing the diagnostic limit values and the DetectionTime using the FB [ConfigDiagLevel](#) (→ [190](#)).



In OUT_PWM_CSO, OUT_CURRENT_CSO and OUT_H_BRIDGE modes, the `uiDetectionTime` must be at least twice the PWM frequency. If dithering is used, 2 times the dither frequency must be used.

Example:

Frequency = 100 Hz

$uiDetectionTime(min) = 1 / 100 \text{ Hz} * 2 = 20 \text{ ms}$

The following operating modes have no preset diagnostic functionality:

- OUT_DIGITAL_CSI
- OUT_PWM_CSI
- OUT_ANALOG_10

8.2.4.7 Call sequence for diagnostic function blocks for inputs and outputs

The call sequence for diagnostic function blocks for inputs and outputs must be programmed as follows:

Inputs:

1. FB [Input](#) (→ [173](#))
2. FB [ConfigDiagProt](#) (→ [193](#)) (optional)
3. FB [ConfigSwThreshold](#) (→ [159](#)) (optional)
4. FB [ConfigDiagLevel](#) (→ [190](#)) (optional)

Outputs:

1. FB [Output](#) (→ [175](#)) / FB [PWM1000](#) (→ [205](#)) / FB [CurrentControl](#) (→ [202](#))
2. FB [ConfigDiagProt](#) (→ [193](#)) (optional)
3. FB [ConfigDiagLevel](#) (→ [190](#)) (optional)



Do not execute the next block until the previous block has been completely processed (`xPrepared` = TRUE or `xDone` = TRUE).

8.2.5 Note the following when programming

Observe the following notes when programming the IEC application in CODESYS:

- ▶ The attribute `pack_mode` is not supported and must not be used.
- ▶ The `SysFileGetTime` function is not supported and must not be used.
- ▶ After opening a file on the device for reading or writing (library: `SysFile`), close the file again directly using the IEC application to prevent data loss and file corruption. Switching off the device during a write process can damage a file.
- ▶ Please note: [Using the operators](#) (→ [223](#))
- ▶ Please note: [Behaviour in case of integer operations](#) (→ [227](#))
- ▶ Please note: [Behaviour in case of floating point operations](#) (→ [226](#))
- ▶ Please note: [Troubleshooting](#) (→ [213](#))



Restrictions on the use of Datasource connections

Only 1 communication channel is available on PLC_1. If the PLC_1 project uses a Datasource connection, simultaneous service communication with CODESYS is not possible. If the connection to the data source is disconnected (e.g. by stopping/disconnecting the PLC handler or HMI), service communication is possible again.

2 channels are available on PLC_2. This means that 1 data source connection and 1 service communication is possible at the same time.

8.3 Using ifm function libraries

ifm electronic provides the following function libraries for the programming of the device under CODESYS 3.5: [Libraries](#) (→ [45](#))



- ▶ Detailed information about the ifm function libraries: [ifm function libraries](#) (→ [120](#))

8.3.1 Access to inputs

To access the inputs of the device, the following functional elements are available:

Function block	Brief description
Input (→ 173)	assigns an operating mode to an input channel and provides the current state at the selected channel
FastCount (→ 163)	Counter block for fast input pulses
IncEncoder (→ 165)	up/down counter function for the evaluation of encoders
Period (→ 167)	- measures at the indicated channel: the frequency and the cycle period (cycle time) in [μs], - measures at the indicated channel pair: the phase shift in [°] between channel A and channel B

8.3.2 Access to outputs

To access the outputs of the device, the following functional elements are available:

Module	Brief description
Output (→ 175)	assigns an operating mode to an output channel provides the current state of the selected channel.
OutputGroup (→ 197)	controls the activation status of an output group and provides diagnostic information about the group and the connected outputs. Using the FB, an output group including the corresponding outputs can be switched on or off.
PWM1000 (→ 205)	initialises and configures a PWM-capable output channel the pulse-pause ratio can be specified in 1 % steps
CurrentControl (→ 202)	Current controller for a PWMi output channel

8.3.3 Control device

The following function elements are available to control the device:

Function block	Brief description
SetLED (→ 178)	Change the frequency and the colour of the status LED in the application program

8.3.4 Read device information

To read measured system values from the device, the following functional elements are available:

Function block	Brief description
SystemSupply (→ 181)	indicates the value of the system voltage
Temperature (→ 183)	indicates the value of the system temperature

8.4 Using CAN

8.4.1 Use RawCAN (CAN Layer 2)



► Observe the notes about task configuration!

In order to access one of the CAN interfaces configured for RawCAN operation, the following POU's are available.

Requirements:

- CAN interface is configured for operation as RawCAN (CAN Layer 2).

8.4.1.1 RawCAN: Control CAN network nodes

The following POU's are available to control a node in a CAN network:

Function block	Brief description
CAN_Enable (→ □ 127)	initialises the specified CAN interface configures the CAN baud rate
CAN_Recover (→ □ 129)	controls the processing of a failure of the specified CAN channel reset and restart the CAN interface in case of a failure of the CAN channel

8.4.1.2 RawCAN: Send and receive CAN messages

The following POU's are available to send or receive messages in a CAN network:



- ▷ The service communication is processed with a lower priority than the process CAN communication.
- ▷ Service communication cannot be carried out if there are too many non-received CAN messages on the CAN bus. Consequence: CODESYS cannot connect via the CAN interface.
- ▶ Ensure that all sent CAN messages are received and processed in the IEC application using the Rx function blocks.

Module	Brief description
CAN_Rx (→ □ 135)	configures a data receive object and reads the receive buffer of the data object
CAN_RxMask (→ □ 137)	receives CAN messages of a non-contiguous range the range is defined by a bit pattern and a bit mask
CAN_RxRange (→ □ 139)	receives CAN messages of a contiguous range the range is defined by an upper and a lower limit
CAN_Tx (→ □ 145)	asynchronous transmission of CAN messages

8.4.1.3 RawCAN: Request and send remote CAN messages

The following POU's are available to request remote messages in a CAN network or to send replies to a remote request:

Function block	Brief description
CAN_RemoteRequest (→ □ 131)	Send request for a remote message
CAN_RemoteResponse (→ □ 133)	Reply to the request of a remote message

8.4.2 Use CANopen



- ▶ Observe the notes about task configuration!

In order to access one of the CAN interfaces configured for CANopen operation, the following POU's are available in ifm libraries.

Further POU's are available in CODESYS libraries by CODESYS GmbH.

Requirements

- Device configured as CANopen Manager (master)
→ Chapter [via system configuration: CANopen Manager \(→ □ 77\)](#)

8.4.2.1 CANopen: Send and receive SDO

The following POU's are available to send or receive Service Data Objects (SDO):

Function block	Brief description
COP_SDOread (→ 151)	Read Service Data Object (SDO)
COP_SDOwrite (→ 153)	Write Service Data Object (SDO)

8.4.2.2 CANopen: Network Management (NMT)

The following POU's are available for the management of the CANopen network:

Function block	Brief description
COP_GetNodeState (→ 149)	Request state of one or several CANopen devices
COP_SendNMT (→ 155)	Send an NMT control command to a CANopen device

8.4.3 Use SAE J1939



► Observe the notes about task configuration!

To use the network protocol SAE J1939, the configuration is set via the device tree. (→ Chapter [via system configuration: J1939 manager](#) (→ [79](#)))

CODESYS GmbH provides the `IoDrvJ1939` library with additional functions.



As of CODESYS version 3.5 SP17, the J1939 configurator is no longer supplied with a parameter group / signal database. The old database is no longer supported.

CODESYS is able to install a DBC database in the J1939 Manager.

Some J1939 clients also have a DBC file that can be imported into CODESYS.

A database can also be purchased. (Recommended)

If no database is installed, parameter groups and signals can also be configured manually on the [User-defined] tab. (Not recommended, because it is error-prone)

8.5 Using Ethernet

8.5.1 SysSocket interface

The SysSocket interface provides the following sockets for implementation in the IEC application:

Sockets	Number	Ports
TCP	32	user-defined
UDP	8	user-defined

The following CODESYS functions reserve the following number of sockets:

CODESYS function	Number of sockets
Transmitter network variable list	1
Receiver network variable list	1
TCP connection with the CODESYS protocol (e.g. Data Source Manager)	2

CODESYS function	Number of sockets
UDP connection with the CODESYS protocol (e.g. Data Source Manager)	1
In Modbus TCP master mode, per Modbus slave	1
In Modbus TCP slave mode	1

The following sockets and ports are reserved for the system:

Sockets	Number	Ports
TFTP (within the system)	1	69
CODESYS communication UDP (within the system)	1	1740, 1741, 1742, 1743
CODESYS communication TCP (within the system)	2	11740
IoT Core:	1	80



Do not use the reserved ports for TFTP and CODESYS in the application!

The implemented Ethernet stack is an embedded stack with limited resources. There is only a defined number of available send/receive buffers. These buffers are shared by all sockets in the system. This means that only a part of the total available sockets can be used for the IEC application to ensure system function, e.g. update and CODESYS debugging.

Available buffers:

Buffers	Maximum number of
Send buffers	12
Receive buffers	32



► To ensure the system function, please observe the following:

- Recommendation: Only use a maximum of 16 receive sockets in the IEC application.
- When the send buffer is full, the SysSocket function will return ERR_SOCK_NOBUFFER. Respond to this event in the IEC application, e.g. by resending the message.
- The command FIONBIO of the function SysSocketIoctl only supports the call as non-blocking.
- When the RESET COM pin is enabled, the SysSocketSetIPAddress and SetSubnetMask functions report the error ERR_SOCK_NOTSOCKET.
- The UDP Segmented Telegram is not supported. If more than 1460 bytes of data are sent, the application must split them into several UDP packets.
- The CODESYS library SysSocket is released for use in the IEC application.
- The CODESYS library UDP is intended exclusively for CODESYS internal use.
- The CODESYS library TCP is intended exclusively for CODESYS internal use.



The following SysSocket functions have a timeout parameter and block the IEC task for the defined timeout period: SysSockGetRecvSizeUdp, SysSockPing, SysSockSelect. A timeout period that is set too long will trigger the watchdog and lead to FATAL_ERROR.

- Set the timeout period considering the task interval and the watchdog time: timeout period < watchdog time < task interval

8.5.2 IoT service interface - sockets

The IoT service interface does not have its own sockets, but uses the freely available sockets of the SysSocket interface.

If the IoT service interface is to be used:

- Ensure that sufficient free sockets are available. Each incoming connection requires a free socket.

8.5.3 Modbus



The following restrictions apply when using a Modbus master or slave device:

A CODESYS Modbus master or slave device can only be used exclusively on one PLC of the controller.

If a Modbus master or slave device is configured on a PLC, other SysSocket connections can only be used on the same PLC as the Modbus master/slave device.

Every active TFTP or Ethernet IoT connection temporarily interrupts the Modbus connection until the other connections are closed again.

- ▶ If a Modbus master or slave device is configured on a PLC: Do not use an Ethernet connection on the other PLC.
- ▶ Only use TFTP and Ethernet IoT connections temporarily for maintenance work and not during regular operation.
- ▶ Debugging in CODESYS is always possible on both controllers.

To use the Modbus, the configuration is set via the device tree: [Using Modbus \(→ 75\)](#)

CODESYS GmbH provides Modbus libraries.

8.6 Error in the IEC application

The operating system (ifmOS) treats the following errors from the IEC application and leads the causing PLC into Error class B ([→ Error classes 213](#)).



To improve the availability of the application, the project engineer should avoid the above-mentioned errors and avoid them e.g. by means of coding guidelines, review, test, etc.

Error in the application	Error class
Division by zero for Integer data type	B
Division by zero for Float data type	B
Memory access violation (writing)	B
Zero pointer exception	B

The operating system (ifmOS) has no error handling for the following arithmetic errors.



The project engineer must exclude the following errors by avoiding systematic errors and e.g. coding guidelines, review, test, etc. or allow the error calculations explicitly (e.g. subtraction of 2 values to get one absolute value).

Error in the application	Error class
Overflow (data types Integer and Float)	No error handling!
Underflow (data types Integer and Float)	No error handling!
Float Inexact Value	No error handling!

9 Set-up and maintenance

9.1	Establishing a connection to the device	101
9.1.1	Connection via Ethernet.....	101
9.1.2	Connection via CAN.....	101
9.2	Check the operating system version of the device.....	101
9.2.1	Check the operating system version of the device.....	101
9.2.2	Check the hardware version of the device	102
9.3	Update the operating system of the device.....	102
9.3.1	Update the operating system of the device with ifm ecologOne	102
9.4	Transfer CODESYS project to device	102
9.4.1	Loading the PLC_1 application to the device.....	103
9.4.2	Loading the PLC_2 application to the device.....	103
9.4.3	Delete the application program on the device.....	104
9.5	Data transmission for series production.....	104
9.5.1	Data transmission with ifm ecologOne.....	105
9.5.2	Transmission of the files with CODESYS	106
9.5.3	Data transmission with TFTP	106
9.5.4	Files for series production	107
9.6	Documentation of the overall application	107
9.6.1	Documenting the serial number and operating system version	107
9.6.2	Read the device information	108
9.7	CODESYS Debugging	110

9.1 Establishing a connection to the device

9.1.1 Connection via Ethernet

► Depending on the required topology, integrate the device into the Ethernet network via an Ethernet interface and observe the following:

- [Ethernet interface \(→ 41\)](#)
- [Configure Ethernet interface \(→ 56\)](#)
- [Configure interfaces \(→ 71\)](#)
 - [Configuring interfaces in CODESYS \(→ 72\)](#)
 - [Configuring the Ethernet address \(→ 73\)](#)
 - [Interface configuration file comconf.cfg \(→ 81\)](#)
 - [Module slot detection \(→ 84\)](#)
- [Using Ethernet \(→ 97\)](#)



► Before downloading a software component, read out the serial number and consult the network diagram to make sure that you are accessing the right controller.

9.1.2 Connection via CAN

► Integrate the device into a CAN network via a CAN interface and observe the following:

- [CAN: Interfaces and protocols \(→ 40\)](#)
- [Configuring the CAN interface \(→ 57\)](#)
- [Configure interfaces \(→ 71\)](#)
 - [Configuring interfaces in CODESYS \(→ 72\)](#)
 - [Configuring CAN interfaces in the device tree \(→ 76\)](#)
 - [Interface configuration file comconf.cfg \(→ 81\)](#)
 - [Module slot detection \(→ 84\)](#)
- [Using CAN \(→ 95\)](#)
- [Task configuration example \(→ 234\)](#)



► Before downloading a software component, read out the serial number and consult the network diagram to make sure that you are accessing the right controller.

9.2 Check the operating system version of the device

9.2.1 Check the operating system version of the device

To check the operating system version of the device:

- Connect to PLC: Set communication path of PLC.
- Copy the file `\info\swinfo.txt` to a local PC drive by clicking [`<<`]: Manage files.
- Open `swinfo.txt` in an editor, e.g. Notepad.

▷ Content of the opened file (example):

```
[ifmOS]
Version=V1.4.0.3
BuildDate=21.10.2016 15:11:14
```

```
[Bootloader]
Version=1.2.3.4
BuildDate=01.01.2017 23:10:15
```

- ▶ Check the information in the [ifmOS] area behind [Version=].
- ▷ If the version deviates from the required version, the operating system must be updated.

9.2.2 Check the hardware version of the device

To check the hardware version of the device:

- ▶ Connect to PLC: Set communication path of PLC.
- ▶ Copy the file `\info\devinfo.txt` to a local PC drive by clicking [<<]: Manage files.
- ▶ Open and check `devinfo.txt` in an editor, e.g. notepad
- ▷ `devinfo.txt` includes the following information: Hardware information (`devinfo.txt`)

9.3 Update the operating system of the device

ATTENTION

Interruption of the voltage supply to the VBB30 during the file transfer/update process

- ▷ An incomplete operating system update may destroy the device.
- ▶ During the update process, ensure the electrical power supply to the device at VBB30.
- ▶ Do not disconnect the device before the data transfer/update process is finished.



- ▶ Use the device user `admin` to update the operating system of the device.

- ▶ Before the update process, please ensure:
 - The device is connected to the power supply and switched on.
 - The device is connected to the PC via an Ethernet interface (if available) or via a CAN interface.

When loading the operating system to the controller, the controller verifies whether the software is compatible with the hardware. If they are not compatible, the transmission will stop.

9.3.1 Update the operating system of the device with ifm ecologOne

To update the operating system of the device with ifm ecologOne:

→ “ifm ecologOne” software manual

9.4 Transfer CODESYS project to device



- ▶ Familiarise yourself with the following CODESYS functions:
Translate project/application and transfer to device: → Online help > CODESYS Development System > Transferring applications to the PLC



- ▶ Before downloading a software component, read out the serial number and consult the network diagram to make sure that you are accessing the right controller.

When the application is loaded to the controller, the controller verifies whether the translated application is compatible with the installed firmware version and the configured memory layout. If they are not compatible, the transmission will stop.



- Observe the notes on the operating modes of the device's standard PLC. ([→ Operating states](#) [□ 113](#))

9.4.1 Loading the PLC_1 application to the device

ATTENTION

When CODESYS connects to a controller, diagnostic functions such as watchdog or memory protection will be disabled.

- ▷ Risk of material damage.
- Before connecting to the controller, the designer must ensure
 - that the machine/system does not pose a hazard under any circumstances.
 - that the machine/system is located in a protected and enclosed environment.
 - that there are no persons or objects in the operating range of the machine/system.



When the transmission starts, the entire controller goes into the UPDATE mode.

In this mode, all outputs of the device will be switched off and both applications (PLC_1 und PLC_2) will be stopped. ([→ Operating states](#) [□ 113](#))

To transfer the created application from `PLC_1` to the device as a boot project:

Requirements:

- The communication path has been set: Set communication path of PLC.
- The project has been tested.

1 Translate the application

- In the device tree: Highlight the desired application as active application.
- Translate the active application with [Build] > [Generate Code].
- ▷ CODESYS generates program code.

2 Load the application to the device

- Change to debug mode with [SIL2] > [Enter debug mode...].
- Connect with the device via [Online] > [Login].
- ▷ Active application is loaded to the device (download).
- ▷ Application on the device is in the STOP state.

3 Start the application

- Start the application with [Debug] > [Start].
- ▷ Application goes to the RUN state.

9.4.2 Loading the PLC_2 application to the device



When the transmission starts, the entire controller goes into the UPDATE mode.

In this mode, all outputs of the device will be switched off and both applications (PLC_1 und PLC_2) will be stopped. ([→ Operating states](#) [□ 113](#))

To transfer the created application from `PLC_2` to the device as a boot project:

Requirements:

- The communication path has been set: [Configuring the programming interface](#) ([→ □ 56](#))
- The project has been tested.

1 Translate the application

- ▶ In the device tree: Highlight the desired application as active application.
- ▶ Translate the active application with [Build] > [Generate Code].
- ▷ CODESYS generates program code.

2 Load the application to the device

- ▶ Connect with the device via [Online] > [Login].
- ▷ Active application is loaded to the device (download).
- ▷ Application on the device is in the STOP state.

3 Start the application

- ▶ Start the application with [Debug] > [Start].
- ▷ Application goes to the RUN state.

9.4.3 Delete the application program on the device

To delete an application stored on the device:

1 Connect with the device

1. In the device tree: highlight application as active application.
2. Connect with the device with [Online] > [Login].

CODESYS is in the online mode.

2 Delete application

1. In the editor window: Select tab [Device] > [Applications].
2. Refresh view with [Refresh List].

List shows the applications that are stored on the device.

1. Delete all applications in the device with [Remove All].

OR:

Highlight requested application and delete with [Remove] from the device.

The selected application will be deleted.

9.5 Data transmission for series production

For the series production, application data and stored data can be transferred to the PC and then transferred from the PC to further devices.

The data transmission takes place in two steps:

1. Data backup from the device to the PC
2. Distribution of the backed up data to the target devices

When loading the operating system to the controller, the controller verifies whether the software is compatible with the hardware. If they are not compatible, the transmission will stop.

When loading the application to the controller, the compatibility of the application will be checked when it has been loaded. Starting an incompatible application is not possible.

If other files are being loaded, the compatibility will not be checked.



Procedure for copying retain data from the device to the PC:

- ▶ Switch off the device.
 - ▷ The current retain data is saved in the non-volatile memory of the device.
- ▶ Switch the device on again.
- ▶ Copy the retain data as required.



Only a single file can be accessed on the device at a time.

- ▷ If a file is opened multiple times on the device at the same time (e.g. with tftp, ecologOne, CODESYS, IoT), the process is cancelled and an error message appears:
 - tftp: SERVER_BUSY
 - CODESYS: ERR_PENDING
 - IoT: Errorcode 423 Filesystem busy
- ▶ Access a file on the device only one at a time and one way.
- ▶ After the error message appears, repeat the file access.

ATTENTION

If the controller is restarted and especially in case of a power-on reset from the debug mode:

- ▷ Risk of material damage.
- ▷ The controller passes into the operating mode.
- ▷ The controller application starts.
- ▶ Before restarting the controller, the designer must ensure
 - that the machine/system does not pose a hazard under any circumstances
 - that the machine/system is located in a protected and enclosed environment
 - that there are no persons or objects in the operating range of the machine/system
- ▶ Implement restart protection, start acknowledgement, etc. in the application.

ATTENTION

Incompatibility between files and hardware / software.

- ▷ Risk of material damage.
- ▶ The system manufacturer must ensure the compatibility of the following data:
 - iomapping.cfg to the device and the IEC application.
 - memconf.ifm to the device and the IEC application.
 - comconf.cfg to the device and the IEC application.
 - retain data and memory data to the IEC application. Recommendation: Use the checksum CRC32 in the file swinfo.txt for this purpose.
 - User files (which are read and written from the IEC application) to the IEC application.

9.5.1 Data transmission with ifm ecologOne

Procedure for data transmission with ifm ecologOne: → Software manual “ifm ecologOne”



When the transmission starts, the entire controller goes into the UPDATE mode.

In this mode, all outputs of the device will be switched off and both applications (PLC_1 und PLC_2) will be stopped. (→ [Operating states](#) □ 113)

9.5.2 Transmission of the files with CODESYS



When the transmission starts, the entire controller goes into the UPDATE mode.

In this mode, all outputs of the device will be switched off and both applications (PLC_1 und PLC_2) will be stopped. ([→ Operating states](#)  113)



Logging in as a device user is required for this action.
For greater ease of use:

- ▶ Create user in project and perform user login only once
Create a user in the CODESYS project
[Sign the user in to the CODESYS project](#) ([→](#)  63)

To transfer files between the PC and the device:

1 Selecting file view

- ▶ In the CODESYS device tree: Double click on the ecomatBasic icon (ecomatBasic)
- ▶ In the editor window: Select the [Files] tab.
- ▶ Click on the  [Refresh] icon
 - ▷ The editor window shows the folder structure on the PC on the left and on the device on the right

2 Transferring a file from PC to the device

- ▶ Highlight the file on the left
- ▶ Select device target directory on the right
- ▶ Start the transfer by clicking on the [>>] button
 - ▷ The file is transferred to the device

3 Transferring a file from the device to the PC

- ▶ Highlight the file on the right
- ▶ Select PC target directory on the left
- ▶ Start the transfer by clicking on the [<<] button
 - ▷ The file is transferred to the PC

9.5.3 Data transmission with TFTP



When the transmission starts, the entire controller goes into the UPDATE mode.

In this mode, all outputs of the device will be switched off and both applications (PLC_1 und PLC_2) will be stopped. ([→ Operating states](#)  113)

With the aid of the program `TFTP`, files can be transferred.

Transfer file from device to PC:

```
tftp -i IP address GET source target1
```

IP address = address of the source device, e.g. 192.168.82.247

Source = source file on the device

Target = target file on the PC

Transfer file from PC to device:

```
tftp -i IP address PUT source target
```

IP address = address of the source device, e.g. 192.168.82.247

Source = source file on the PC

Target = target file on the device

Example:

```
tftp -i 192.168.82.247 PUT [Windows-Pfad]\ifmOS.ifm /os/ifmOS.ifm
```

9.5.4 Files for series production

The following files must be transferred:

Data name / path	Description
apps - standard.app - safe.app	Folder PLC_2 application PLC_1 application
os ifmOS.ifm	Folder ifmOS
cfg - comconf.cfg - memconf.ifm - iomapping.cfg	Folder Communication configuration Memory configuration Input / output resources

The following file must be transferred according to the kind of application (retain data and free user data):

Data name / path	Description
retain - standard.ret - standard.mb - safe.ret - safe.mb	Folder Application Retain PLC_2 Application Memory Bytes PLC_2 Application Retain PLC_1 Application Memory Bytes PLC_1
data - symsapp.xml - symnapp.xml	Folder - Symbol-File PLC_1 - Symbol-File PLC_2

9.6 Documentation of the overall application**9.6.1 Documenting the serial number and operating system version**

All devices are labelled with a unique serial number.

- Before installing the devices, document and archive the serial number together with the operating system version, the machine ID and the location of the machine.

This is the only way to trace at a later date which device is installed in which machine or system and with which operating system version the device is operated.



- After updating the operating system, update the information on the operating system version in the documentation.

9.6.2 Read the device information

Information of the device is provided in different files on the controller in the directory `\info`. This information can be used to identify the device and to determine its version and software content.

The following files with information are included:

File name	Description
appinfo.txt	Application information (appinfo.txt)
devinfo.txt	Hardware information (devinfo.txt)
swinfo.txt	Software information (swinfo.txt)

- To check/document the information in the device:
- Connect to PLC: Set communication path of PLC.
- Copy the files that are listed in the table in the directory `\info` by clicking on [`<<`] to a local PC drive. Manage files.
- Open the files in a text editor and check their content



Further information about files and directory structure on the controller: Directory structure and file overview

9.6.2.1 Application information (appinfo.txt)

The file `appinfo.txt` contains the following information:

Section	Key	Description
Standard task X	Task type	PLC_2 application: Task type
Standard task X	Task priority	PLC_2 application: Task priority
Standard task X	Task interval	PLC_2 application: Task interval in ms (only if type = cyclic)
Standard task X	WatchdogEnable	PLC_2 application: Watchdog enabled (TRUE, FALSE)
Standard task X	WatchdogTime	PLC_2 application: Watchdog time in ms
Standard task X	WatchdogSensitivity	PLC_2 application: Watchdog sensitivity
Safe-Task-X	Task type	PLC_1 application: Task type
Safe-Task-X	Task priority	PLC_1 application: Task priority
Safe-Task-X	Task interval	PLC_1 application: Task interval in ms (only if type = cyclic)
Safe-Task-X	WatchdogEnable	PLC_1 application: Watchdog enabled (TRUE, FALSE)
Safe-Task-X	WatchdogTime	PLC_1 application: Watchdog time in ms
Safe-Task-X	WatchdogSensitivity	PLC_1 application: Watchdog sensitivity
Security	PasswordsActive	Passwords are set for all application users (TRUE, FALSE)

with X = 0...3, depending on the number of the CODESYS task



Read the files from the controller: [Read the device information \(→ 108\)](#)

9.6.2.2 Hardware information (devinfo.txt)

The file `devinfo.txt` contains the following information:

Section	Key	Description
IDENTIFICATION	Serial	Serial number of the device
DEVICE	Order no.	ifm article number of the device
DEVICE	Name	ifm article description of the device

Section	Key	Description
DEVICE	Revision	Production status of the device
MANUFACTURE	Date	Production data and time
HARDWARE	Version	Hardware version of the device
NETWORK	MAC	MAC address of the device
NETWORK	DNS	DNS name
CERTIFICATE	Certificate [0..9]	Certificates of the controller
CRC	CRC32	CRC32 over the entire previous ASCII range. IEEE 802.3 Standard Ethernet Polynomial. This section must be at the end of the file.



Read the files from the controller: [Read the device information \(→ 108\)](#)

9.6.2.3 Software information (swinfo.txt)

The file `swinfo.txt` contains an own section with current values for each of the following software components. If the corresponding software component is not included in the controller, there is no corresponding section in the file.

Software components:

- ifmOS
- Bootloader
- SIS-SYS
- comconf
- memconf
- iomapping
- StandardPLC
- SafePLC
- StandardRetain
- StandardMB
- SafeRetain
- SafeMB

The file `swinfo.txt` contains the following information for the above-mentioned software components:

Key	Description
Name	Software name (e.g. ifmOS)
Name	only for PLC_1 and PLC_2: Information from CODESYS project title
Description	only for PLC_1 and PLC_2: Information from Application Information Description (for both applications, separate information is possible)
Version	Software version number (e.g. V1.5.3.29)
Build date	Software build date (format DD.MM.YYYY, hh:mm:ss)
CodeCRC32	CRC32 via the software. Standard Ethernet Polynomial (04C11DB71H).
HeadCRC32	CRC32 via the header of the software of the Standard Ethernet Polynomial software (04C11DB71H).
Active	only for boot loader and ifmOS: indicates which software is executed (TRUE/FALSE).
CRC32	CRC32 over the entire previous ASCII range up to the [CRC] section (last character is]). IEEE 802.3 Standard Ethernet Polynomial (04C11DB71H). This section must be at the end of the file.



► The file shows the current values (versions, names, CRCs etc.) of the files in the controller.



Read the files from the controller: [Read the device information](#) (→ [108](#))

9.7 CODESYS Debugging



► Familiarise yourself with the following CODESYS functions:

- CODESYS debugging:
→ Online help > CODESYS Development System > Test and remove errors

ATTENTION

If CODESYS connects to a controller, the controller goes into **non-safe operation**.

▷ Risk of material damage.

- Before connecting to the controller, the designer must ensure
 - that the machine/system does not pose a hazard under any circumstances.
 - that the machine/system is located in a protected and enclosed environment.
 - that there are no objects in the operating range of the machine/system.

ATTENTION

If the controller is restarted and especially in case of a power-on reset from the debug mode:

▷ Risk of material damage.

▷ The controller passes into the operating mode.

▷ The controller application starts.

- Before restarting the controller, the designer must ensure
 - that the machine/system does not pose a hazard under any circumstances
 - that the machine/system is located in a protected and enclosed environment
 - that there are no objects in the operating range of the machine/system

► Implement restart protection, start acknowledgement, etc. in the application.

When debugging is enabled:

- CODESYS resets possible (→ [Reset](#) [117](#)):
 - reset (warm)
 - reset (cold)
 - reset (default)
- When executing a reset:
 - all resources opened by the IEC application in question (files, sockets, etc.) will be closed.
 - all inputs and outputs assigned to the IEC application in question will be switched off and put into default mode.
- It is possible to force values in the IO mapping.
- It is possible to start and stop the IEC application.
- Execution of the following debugging functions:
- use of up to 8 breakpoints per PLC

- step-by-step program processing (menu [Debug] > Step [Over], [Step Into, Step] [Out], [Run to Cursor])
- For IEC application exception: jump to the position in the editor



CODESYS debugging restrictions

- ▶ Use a maximum of 6 breakpoints for step-by-step program execution (2 breakpoints are used internally by CODESYS).
- ▶ When the error message [Breakpoint could not be set on target.] appears in CODESYS: Disable breakpoints or log out and log in again to release breakpoints.
- ▶ In step-by-step program execution, CODESYS uses 1 breakpoint internally for each CASE branch in CASE instructions:
- ▶ Debug CASE instructions with more than 8 branches only as follows:
 - Set breakpoint at the CASE instruction.
 - Determine the target branch using the variable value.
 - Set breakpoint in target branch.
 - Continue debugging with [Debug] > [Start].

10 Operation

10.1	Operating states.....	113
10.1.1	Overview of operating mode states.....	113
10.2	Status LEDs	114
10.2.1	Status LED: system ifm operating system (SYS0+SYS1).....	115
10.2.2	Status LED: system PLC (SYS0, SYS1).....	115
10.2.3	Status LED: System bootloader (SYS0)	116
10.2.4	Status LED: Sleep mode (SYS0)	116
10.2.5	Status LED Ethernet interface (ETH)	116
10.2.6	Controlling user LEDs in the applications	116
10.3	Reset.....	117
10.3.1	Reset behaviour of the system.....	117
10.3.2	CODESYS reset requirements (PLC_1)	118
10.3.3	Executing reset variants	118
10.3.4	Reset application (warm)	118
10.3.5	Reset application (cold).....	118
10.3.6	Reset application (origin)	119

10.1 Operating states

The following table describes the possible operating modes of the device:

Operating status	Description	Has an impact on...	condition of the inputs	condition of the outputs
POWER_OFF	The device is switched off. Initial condition.	Whole device	OFF	OFF
INIT	Starting the device: Initialising the software, checking the hardware.	Whole device	OFF	OFF
RUNTIME_OPERATING	Normal operating status of the controller. Executing the application.	Corresponding device-internal PLC	ACTIVE	ACTIVE
RUNTIME_STOP	Application is stopped after an error class B event (SERIOUS ERROR).	Corresponding device-internal PLC	INACTIVE	OFF
RUNTIME_DEBUG_RUN	Executing the application in the debug mode.	Corresponding device-internal PLC	ACTIVE	ACTIVE
RUNTIME_DEBUG_BP_HALT	The application is in the debug mode, stop through breakpoint or exception.	Corresponding device-internal PLC	INACTIVE	OFF
RUNTIME_DEBUG_STOP	The application is stopped in the debug mode.	Corresponding device-internal PLC	INACTIVE	OFF
SHUTDOWN	Shutting down the device: Closing the applications, storing non-volatile data and checking the hardware.	Whole device	OFF	OFF
UPDATE	Updating the device: loading data to the device.	Whole device	OFF	OFF
SYSTEM_STOP	The device is stopped after an error class A event A = (FATAL ERROR).	Whole device	OFF	OFF
SLEEP	Sleep mode The sleep mode is supported by devices with hardware version >= V1.0.5.2.	Whole device	OFF	OFF

Legend:

OFF: Inputs are in the default mode and outputs are switched off.

ACTIVE: The assigned inputs and outputs are operated with the configuration set by the IEC application (mode, filter, diagnostics, etc.).

INACTIVE: The assigned inputs include their preset configuration and will continue to be read. The diagnostic in the IEC application is suppressed. The hardware is still protected.

10.1.1 Overview of operating mode states

The following figure shows the possible operating modes of the device.

It contains:

- status of the application
- status of the process communication (inputs/outputs, CAN bus)
- status of the service communication (connection with the programming device)
- display of the LEDs SYS0 / SYS1

All states within **PLC_1 OPERATING** and **PLC_2 OPERATING** are independent and non-reactive in the states in the in the other PLC.

Any states outside of this, concern both PLCs at once.

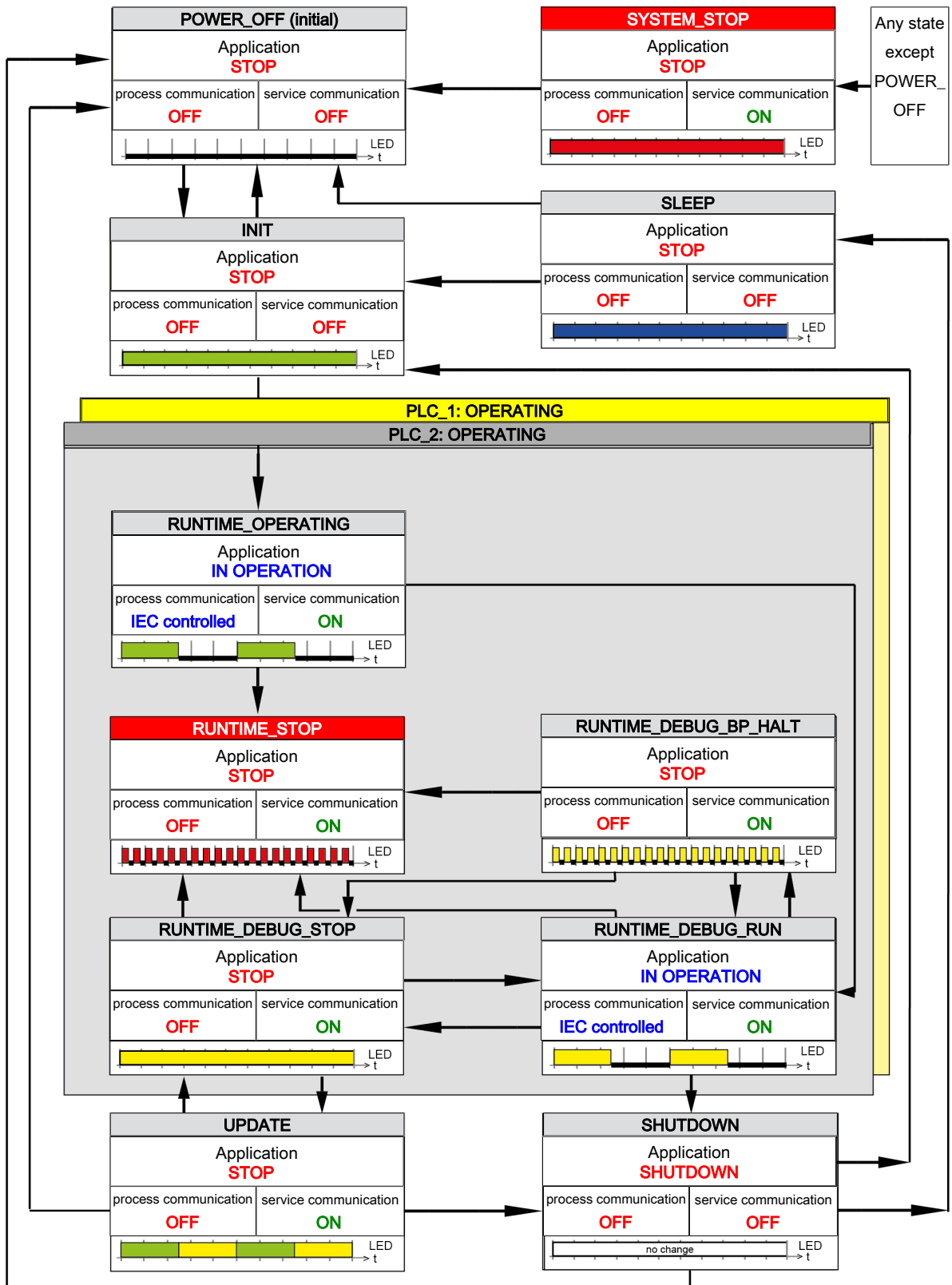


Fig. 16: Operating states

10.2 Status LEDs

The device has the following LEDs:

LED	Meaning
SYS0	- Status PLC_2 - Status of the ifm operating system - Status of the bootloader
SYS1	- Status PLC_1 - Status of the ifm operating system
ETH	Status of the Ethernet interface
APPx	User LED, freely usable LED in the application

10.2.1 Status LED: system ifm operating system (SYS0+SYS1)

For the status of the ifm operating system, both LEDs SYS0 and SYS1 are lit simultaneously:

LED colour	Indication	Description
Off	Permanently off	ifm operating system on the device: POWER_OFF
		
Green	Permanently on	ifm operating system on the device: INIT
		
Red	Permanently on	ifm operating system on the device: SYSTEM_STOP Error class = A
		
No change	No change	ifm operating system on the device: SHUTDOWN
	No change	
Green/yellow	Flashing with 2 Hz	ifm operating system on the device: UPDATE
		

10.2.2 Status LED: system PLC (SYS0, SYS1)

The SYS0 LED is for PLC_2.

The SYS1 LED is for PLC_1.

The status of one of the PLCs has no influence on the display of the other PLC.

LED colour	Indication	Description
green	Permanently on	RUNTIME_OPERATING no application loaded
		
green	flashing with 2 Hz	RUNTIME_OPERATING Application = RUN
		
Yellow	flashing with 2 Hz	RUNTIME_DEBUG_RUN Application = RUN
		
Yellow	Flashing with 10 Hz	RUNTIME_DEBUG_BP_HALT Application = STOP
		
Yellow	Permanently on	RUNTIME_DEBUG_STOP Application = STOP

LED colour	Indication	Description
		
red	Flashing with 10 Hz	RUNTIME_STOP Error class = B
		
(time frame = 200 ms)		

10.2.3 Status LED: System bootloader (SYS0)

Only LED SYS0 is used for the bootloader status.
LED SYS1 is switched off in these cases.



Only execute the bootloader update on the device when explicitly requested by ifm!

LED colour	Indication	Description
Green	Flashing with 5 Hz	No runtime system loaded
	 (time frame = 200 ms)	
Green/yellow	Flashing with 5 Hz	Bootloader update process active
	 (time frame = 200 ms)	
Red	Flashing with 5 Hz	► Contact ifm support.
	 (time frame = 200 ms)	

10.2.4 Status LED: Sleep mode (SYS0)

The SYS0 LED is for the sleep mode status only.

The SYS1 LED is switched off in these cases.

LED colour	Indication	Description
blue	Permanently on	Sleep mode is active
		

10.2.5 Status LED Ethernet interface (ETH)

The Ethernet interface displays its status as follows:

LED colour	Indication	Description
green	Permanently on	Ethernet connection is established No data traffic
		
green	flashes	Ethernet connection is established with data traffic
		

10.2.6 Controlling user LEDs in the applications

The LEDs APPx are for free use in the applications.

This is the function of the FB [SetLED](#) (→ [178](#)).

Possible colours: [LED_COLOUR \(ENUM\)](#) (→ [122](#))

Possible frequencies: [LED_FLASH_FREQ \(ENUM\)](#) (→ [123](#))

10.3 Reset

10.3.1 Reset behaviour of the system

ATTENTION

At the end of a factory reset, a PowerOn Reset (restart of the controller) is automatically carried out. The process takes some seconds.

- ▷ The device can be destroyed if the factory reset is interrupted or the device is manually restarted during the factory reset.
- ▶ Ensure power supply to the VBB30 during the entire factory reset and the automatically following power-on reset.
- ▶ Do not interrupt the factory reset and the automatically following PowerOn Reset.
- ▶ Wait for the factory reset and the automatically following PowerOn Reset.

The system shows the following behaviour for the different reset variants and system resources:

Resource State	CODESYS reset (warm) (only the PLC concerned)	CODESYS reset (cold) (only the PLC concerned)	CODESYS reset (origin) (only the PLC concerned)	PowerOn Reset (PLC_1 and PLC_2 application)	Factory reset (PLC_1 and PLC_2 application)
IEC application	is kept	is kept	is deleted	is kept	is deleted
IEC variables	are re-initialised	are re-initialised	are deleted	are re-initialised	are deleted
IEC-persistent (non-volatile variables)	are kept	are kept	are deleted	are kept	are deleted
IEC memory bytes	are kept	are kept	are kept	are kept	are deleted
user files	are kept	are kept	are kept	are kept	are deleted
iomapping.cfg	is kept	is kept	is kept	is kept	is deleted
comconf.cfg	is kept	is kept	is kept	is kept	is deleted (The comconf.cfg is restored with default values after deletion)
memconf.ifm	is kept	is kept	is kept	is kept	is deleted
passwd.ifm	is kept	is kept	is kept	is kept	is deleted
State after reset	RUNTIME_DEBUG_STOP	RUNTIME_DEBUG_STOP	RUNTIME_DEBUG_STOP	RUNTIME_OPERATING	RUNTIME_OPERATING No application (after automatic PowerOn Reset)



Definition "are re-initialised": The variable will be set to its initialisation value.

If the programmer does not assign an initialisation value to one of the variables, CODESYS will initialise this variable with the standard value (usually with "0").

10.3.2 CODESYS reset requirements (PLC_1)



The following applies to all CODESYS resets that are executed for PLC_1:

- ▶ Set PLC_1 to debug mode before the reset.
- ✓ The communication path to the device is set: [Configuring the programming interface](#) (→ [56](#))
- ▶ In the device tree: Right-click on the [Application] element below [PLC_1].
- ▶ Click on the [Set Active Application] menu item in the context menu.
 - ▷ The IEC application of [PLC_1] is set as the active application.
- ▶ Change to debug mode via [SIL2] > [Enter debug mode...].
 - ▷ PLC_1 is in debug mode.
- ▶ Trigger the desired reset.

10.3.3 Executing reset variants

The following reset variants can be called up as follows:

Calling component	CODESYS reset (warm)	CODESYS reset (cold)	CODESYS reset (origin) (Only the PLC concerned goes into the Update state)	PowerOn Reset (PLC_1 and PLC_2 application)	Factory reset (PLC_1 and PLC_2 application)
CODESYS in debug mode	Yes	Yes	Yes	No	No
IEC application with FUN reset	No	No	No	Yes	No
ifm Maintenance Tool	Yes	Yes	Yes	Yes	Yes
Yes = reset call possible No = reset call not possible					

10.3.4 Reset application (warm)

To reset the application:

- ▶ In the device tree: Select [Application] and set as active application, if not already done.
- ▶ Select [Online] > [Login].
 - ▷ CODESYS changes to the online mode.
- ▶ Select [Online] > [Reset warm] to reset the application.
 - ▷ Application changes to the STOP state.
 - ▷ Standard variables are newly initialised.
 - ▷ Retain variables keep their values.

10.3.5 Reset application (cold)

To reset the application:

- ▶ In the device tree: Select [Application].
 - ▶ Select [Online] > [Login].
- CODESYS changes to the online mode.
- ▶ Select [Online] > [Reset cold] to reset the application.

Application changes to the STOP state.

All variables are newly initialised

10.3.6 Reset application (origin)

To reset the application:

► In the device tree: Select [Application].

► Select [Online] > [Login].

CODESYS changes to the online mode.

► Select [Online] > [Reset origin] to reset the application.

Application changes to the STOP state and is deleted.

All variables are newly initialised

Non-volatile variables will be set to 0.

PLC is reset to the original state.

11 ifm function libraries

11.1	General information.....	120
11.2	Device library	121
11.2.1	Library ifmDevice.library.....	121
11.3	CAN libraries	126
11.3.1	Library ifmRawCAN.library.....	126
11.3.2	Library ifmCANopenManager.library.....	148
11.4	Input and output libraries.....	158
11.4.1	Library ifmConfigSwThreshold.library	158
11.4.2	Library ifmFastInput.library.....	162
11.4.3	Library ifmIOcommon.library	172
11.4.4	Library ifmIOconfigDiagProt.library	189
11.4.5	Library ifmOutGroup.....	196
11.4.6	Library ifmOutPWM.....	201
11.5	Help function libraries.....	208
11.5.1	Library ifmSysInfo.library.....	208

This chapter contains the detailed description of the function libraries provided by ifm electronic for programming the device under CODESYS 3.5.

11.1 General information

General information about:

- [Messages / diagnostic codes of the function blocks \(→ 214\)](#)
- [ifm behaviour models for function blocks \(→ 240\)](#)

11.2 Device library

11.2.1 Library ifmDevice.library

The library contains the following:

- device-specific data structures
- device-specific enumeration types
- device-specific global variables and constants
- device-specific functions

11.2.1.1 CAN_BAUDRATE (ENUM)

Name	Description	Possible values		Data type	Value
CAN_BAUDRATE	Data transmission rate of the CAN interface	KBAUD_10	10 kilobaud	INT	10
		KBAUD_20	20 kilobaud	INT	20
		KBAUD_33	33.3 kilobaud	INT	33
		KBAUD_50	50 kilobaud	INT	50
		KBAUD_83	83.3 kilobaud	INT	83
		KBAUD_100	100 kilobaud	INT	100
		KBAUD_125	125 kilobaud	INT	125
		KBAUD_250	250 kilobaud	INT	250
		KBAUD_500	500 kilobaud	INT	500
		KBAUD_666	666.6 kilobaud	INT	666
		KBAUD_800	800 kilobaud	INT	800
		KBAUD_1000	1000 kilobaud	INT	1000

11.2.1.2 CAN_CHANNEL (ENUM)

Name	Description	Possible values		Data type	Value
CAN_CHANNEL	Identifier of the CAN interface	CHAN_0	CAN interface 0	INT	0
		...	...	INT	...
		CHAN_x	CAN interface x	INT	x

11.2.1.3 CANconstants (GVL)

Name	Description	Data type	Value
usiNumberCANItf	Number of the CAN interfaces of the devices	UINT	4

11.2.1.4 SysInfo (GVL)

Name	Description	Data type	Value
usiNumberOfSysInfo	Number of system components of the device	USINT	8
aSysInfoList	Variable with list of the system components aSysInfoList (GVL) (→ 211)	ARRAY[0..8] OF SysInfoStruct (STRUCT) (→ 122)	

11.2.1.5 SysInfoStruct (STRUCT)

Designation	Data type	Meaning	Possible values
eInfoType	INFO_TYPE	System component	E.g. FIRMWARE_DEVICE
sValue	STRING(255)	Value of the system component	e.g. 3.1
sName	STRING(32)	Name of the system component	E.g. FW Device

11.2.1.6 LED_COLOUR (ENUM)

Name	Description	Possible values		Data type	Value
LED_COLOUR	Colour of the LED (RGB code)	BLACK (OFF)	Off	UINT	0x00 0000
		WHITE	White	UINT	0xFF FFFF

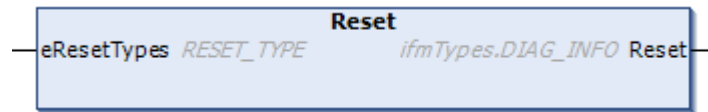
Name	Description	Possible values		Data type	Value
LED_COLOUR	Colour of the LED (RGB code)	RED	Red	UINT	0xFF 0000
		GREEN	Green	UINT	0x00 FF00
		BLUE	Blue	UINT	0x00 00FF
		YELLOW	Yellow	UINT	0xFF FF00
		MAGENTA	Magenta	UINT	0xFF 00FF
		CYAN	Cyan	UINT	0x00 FFFF

11.2.1.7 LED_FLASH_FREQ (ENUM)

Name	Description	Possible values		Data type	Value
LED_FLASH_FREQ	Flashing frequency of the status LED	FRQ_0Hz	Off	INT	0
		FRQ_05Hz	0.5 Hz	INT	1
		FRQ_1Hz	1 Hz	INT	2
		FRQ_2Hz	2 Hz	INT	4
		FRQ_5Hz	5 Hz	INT	7
		FRQ_10Hz	10 Hz	INT	8

11.2.1.8 Reset

Function block type: Function (FUN)
Library: ifmDeviceCR07nn.library
Symbol in CODESYS:



Description

The FUN reset is used to set or execute different types of PLC and system resets.

Input parameters

Parameter	Data type	Meaning	Possible values
eResetTypes	RESET_TYPE	Setting the reset type	RESET_TYPE (ENUM)

Output parameters

Parameter	Data type	Meaning	Possible values
Reset	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_NOT_SUPPORTED	Error: Invalid function calls; Function is not supported.
• ERR_ACCESS	Error: FB/function cannot access the required resource; Resource is blocked by another task.
• DIAG_ACCESS	FB/function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.

11.2.1.9 RESET_TYPE (ENUM)

Name	Description	Possible values		Value
RESET_TYPE	Reset mode	NONE	No action. Default value	0
		POWER	Power-on reset will be immediately executed.	1
		POWER_RECOVER	Restarting the device after it has been switched off (VBB < 5,5V), if VBB reaches more than 8V again.	2
		WARM *	Warm start (PLC reset) will be executed immediately.	3
		COLD *	Cold start (PLC reset) will be executed immediately.	4
		ORIGIN *	Reset to original state (PLC reset) will be executed immediately.	5
		FACTORY *	Factory reset is executed immediately for PLC_1 and PLC_2.	6
		SLEEP	The unit switches to sleep mode when the function is executed. After switching VBB30 off and on again, the controller will reboot (PowerOn Reset).	7
The reset modes marked with a star * are currently not supported.				

11.3 CAN libraries

11.3.1 Library ifmRawCAN.library

The library contains POUs and data structures for the programming of the CAN Layer 2 level of the CAN interfaces of the device under CODESYS.

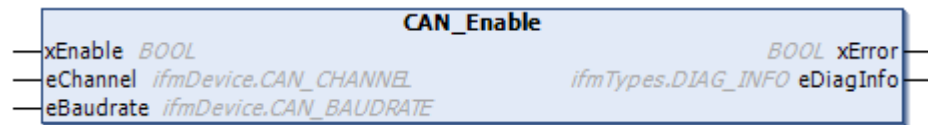
11.3.1.1 CAN_Enable

Function block type: Function block (FB)

Behaviour model: ENABLE

Library: ifmRawCAN.library

Symbol in CODESYS:



If the CAN interface is set as interface for service communication (in the file `comconf.cfg` entry `Service=TRUE`), the baud rate at the FB input `eBaudrate` must be identical! Otherwise: No communication is possible and error message `ERR_BAUDRATE_INVALID_OR_ALREADY_SET`

Description

The FB activates the CAN Layer 2 functions of a CAN interface with a certain transmission rate. Simultaneously the FB writes information about the current state of the CAN interface into the global variable CAN State.

Changes of the transmission rate or of the CAN interface are applied at once. All existing reception and send buffer storages are deleted.



The FB does not have any influence on a CANopen Manager / CANopen Device at the selected CAN interface. In this case the FB cannot change the transmission rate of the CAN interface.

Input parameters

Parameter	Data type	Description	Possible values	
xEnable	BOOL	Control activity of the FB	FALSE	Deactivate FB
			TRUE	Enable FB
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ □ 122)	
eBaudrate	CAN_BAUDRATE	Baud rate of the CAN channel	CAN_BAUDRATE (ENUM) (→ □ 122)	

Output parameters

Parameter	Data type	Description	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_BUS_OFF	Error: CAN interface is in the "BUS OFF" state

• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_INVALID_VALUE	Error: at least 1 invalid input parameter or invalid combination of input parameters; Function call has been stopped.
• ERR_BAUDRATE_INVALID_OR_ALREADY_SET	Error: The required baud rate cannot be set because it is invalid or a different baud rate has already been selected.
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!

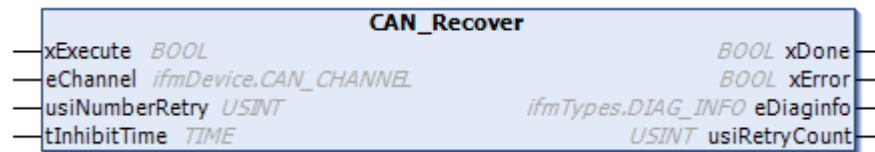
11.3.1.2 CAN_Recover

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifmRawCAN.library

Symbol in CODESYS:



Description

The FB controls the processing of a failure of the CAN channel.

The call of the FB triggers the following actions:

- If the CAN channel fails the CAN interface is reset and rebooted.
- All buffer storages are emptied.



If the CAN channel keeps failing after the maximum number of recovery attempts has been exceeded, the CAN bus remains in the error state.

- Call FB again to repeat the execution of the recovery function.

Input parameters

Parameter	Data type	Description	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE => TRUE	FB is executed once
			Other	No impact on FB processing
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ □ 122)	
usiNumberRetry	USINT	Max. number of retries	e.g. 4	
tInhibitTime	TIME	Time until the CAN interface is started again after the detection of a CAN bus failure	e.g. #2ms	

Output parameters

Parameter	Data type	Description	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	• FB successfully executed • FB can be called again
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	• An error has occurred • Action could not be executed • Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
usiRetryCount	USINT	Counter for retries carried out since the last activation of the FB		

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_INACTIVE_INTERFACE	Error: Selected CAN channel is deactivated.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_INVALID_VALUE	Error: at least 1 invalid input parameter or invalid combination of input parameters; Function call has been stopped.
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!

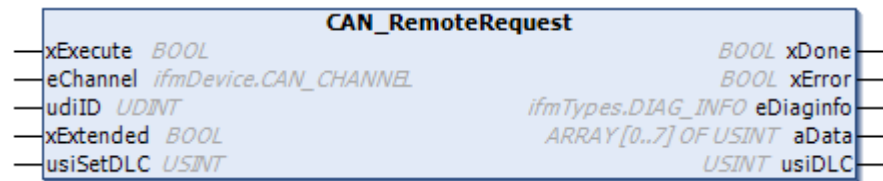
11.3.1.3 CAN_RemoteRequest

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifmRawCAN.library

Symbol in CODESYS:



Description

The FB sends the request for a CAN Remote message into a CAN network. The FB provides the data of the response message in an array. The FB supports standard and extended frames.

Input parameters

Parameter	Data type	Description	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE => TRUE	FB is executed once
			Other	No impact on FB processing
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ 122)	
udiID	UDINT	Identifier of the CAN message	- for Standard Frame (11 bits identifier): 0 ... 2047 - for Extended-Frame (29 bits identifier): 0 ... 536.870.911	
xExtended	BOOL	Requested frame type: - Standard frame (11 bits identifier) - Extended frame (29 bits identifier)	FALSE	Standard Frame*
			TRUE	Extended Frame
usiSetDLC	UINT	Number of the data bytes in the CAN message (DLC = Data Length Count)	0 ... 8	0 bytes* ... 8 bytes

* ... default value

Output parameters

Parameter	Data type	Description	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	- FB successfully executed - FB can be called again
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information

Parameter	Data type	Description	Possible values	
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
aData	ARRAY [0...7] OF USINT	Array for storage of the data received		
usiDLC	USINT	Number of the data bytes in the CAN message (DLC = Data Length Count)	0 ... 8	0 bytes* ... 8 bytes

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_BUSY	Status: FB/function is currently executed.
• ERR_BUFFER_OVERFLOW	Error: Transmission buffer full; CAN message cannot write to buffer storage and is not transmitted
• ERR_INVALID_VALUE	Error: at least 1 invalid input parameter or invalid combination of input parameters; Function call has been stopped.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INACTIVE_INTERFACE	Error: Selected CAN channel is deactivated.

11.3.1.4 CAN_RemoteResponse

Function block type: Function block (FB)

Behaviour model: ENABLE

Library: ifmRawCAN.library

Symbol in CODESYS:



Description

The FB replies as reaction to the request of a CAN Remote message and sends the data required into a CAN network.

As long as the FB is activated it responds to each remote request message (automatic reply).

Several FB calls are possible during one PLC cycle.

Input parameters

Parameter	Data type	Description	Possible values	
xEnable	BOOL	Control activity of the FB	FALSE	Deactivate FB
			TRUE	Enable FB
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ □ 122)	
udiID	UDINT	Identifier of the CAN message	- for Standard Frame (11 bits identifier): 0 ... 2047 - for Extended-Frame (29 bits identifier): 0 ... 536.870.911	
xExtended	BOOL	Requested frame type: - Standard frame (11 bits identifier) - Extended frame (29 bits identifier)	FALSE	Standard Frame*
			TRUE	Extended Frame
usiDLC	USINT	Number of the data bytes in the CAN message (DLC = Data Length Count)	0 ... 8	0 bytes* ... 8 bytes

* ... default value

Output parameters

Parameter	Data type	Description	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
uiRTR_Cnt	UINT	Number of received remote requests after the last FB call		

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_INACTIVE_INTERFACE	Error: Selected CAN channel is deactivated.
• ERR_BUFFER_OVERFLOW	Error: Transmission buffer full; CAN message cannot write to buffer storage and is not transmitted
• ERR_INVALID_VALUE	Error: at least 1 invalid input parameter or invalid combination of input parameters; Function call has been stopped.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!

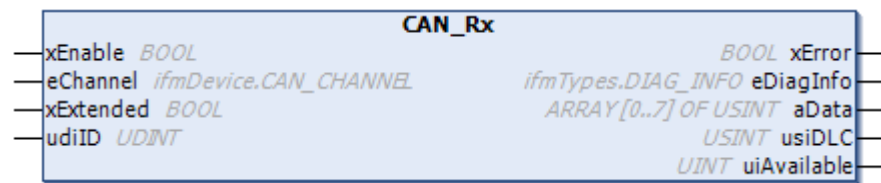
11.3.1.5 CAN_Rx

Function block type: Function block (FB)

Behaviour model: ENABLE

Library: ifmRawCAN.library

Symbol in CODESYS:



Description

The FB receives CAN messages with a defined identifier.

The FB receives all CAN messages with the indicated identifier between 2 FB calls and stores them in a FIFO buffer storage. The number of the received CAN messages is displayed. The CAN message received first is always provided on the output.

If there are several CAN messages in the FIFO buffer storage, the function block can be called until the output is `uiAvailable = 0` and all CAN messages have been read from the FIFO buffer storage.



The function block has the following behaviour after changing the receive ID during the runtime of the application

- ▶ The FB memory be not be reset completely. The outputs `aData` and `usiDLC` keep the last values. The `uiAvailable` counter is set to 0.
- ▶ Only use the FB with static (unchanged in operation) ID configurations at the inputs.
- ▶ Check the value of `uiAvailable` $\neq 0$ before using the data.

Input parameters

Parameter	Data type	Description	Possible values	
xEnable	BOOL	Control activity of the FB	FALSE	Deactivate FB
			TRUE	Enable FB
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ 122)	
xExtended	BOOL	Requested frame type: - Standard frame (11 bits identifier) - Extended frame (29 bits identifier)	FALSE	Standard Frame*
			TRUE	Extended Frame
udiID	UDINT	Identifier of the CAN message	- for Standard Frame (11 bits identifier): 0 ... 2047 - for Extended-Frame (29 bits identifier): 0 ... 536.870.911	

Output parameters

Parameter	Data type	Description	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed

Parameter	Data type	Description	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
aData	ARRAY [0...7] OF USINT	Array for storage of the data received		
usiDLC	USINT	Number of the data bytes in the CAN message (DLC = Data Length Count)	0 ... 8	0 bytes* ... 8 bytes
uiAvailable	UINT	- Number of received CAN messages since the last FB call - Current CAN message is taken into account	0	No CAN messages received between 2 FB calls
			n	n CAN messages received

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_INACTIVE_INTERFACE	Error: Selected CAN channel is deactivated.
• ERR_BUFFER_OVERFLOW	Error: Transmission buffer full; CAN message cannot write to buffer storage and is not transmitted
• ERR_INVALID_VALUE	Error: at least 1 invalid input parameter or invalid combination of input parameters; Function call has been stopped.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!

11.3.1.6 CAN_RxMask

Function block type: Function block (FB)

Behaviour model: ENABLE

Library: ifmRawCAN.library

Symbol in CODESYS:



Description

The FB receives CAN messages of a non-coherent area. The area is defined by a bit pattern and a bit mask.

The following rules apply to the bit mask:

0: The equivalent bit of the CAN identifier can be 0 or 1

1: The equivalent bit of the CAN identifier must have the same value as the bit in the bit pattern

Example:

Samples: 000 0010 0000

Form: 000 1111 1111

Result: xxx 0010 0000

All CAN messages with an identifier whose 8 least significant bits have the value "0010 0000" are received.

e.g. 110 0010 0000 000 0010 0000, 001 0010 0000



General behaviour of the FB: [CAN_Rx](#) (→ [135](#))



The function block has the following behaviour after changing the receive ID during the runtime of the application

- ▷ The FB memory be not be reset completely. The outputs `aData` and `usiDLC` keep the last values. The `uiAvailable` counter is set to 0.
- ▶ Only use the FB with static (unchanged in operation) ID configurations at the inputs.
- ▶ Check the value of `uiAvailable` $\neq 0$ before using the data.

Input parameters

Parameter	Data type	Description	Possible values	
xEnable	BOOL	Control activity of the FB	FALSE	Deactivate FB
			TRUE	Enable FB
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ 122)	
xExtended	BOOL	Requested frame type: - Standard frame (11 bits identifier) - Extended frame (29 bits identifier)	FALSE	Standard Frame*
			TRUE	Extended Frame
udiIDSet	UDINT	Preset bit pattern for the masking of the identifier of the CAN message	E.g. 000 0010 0000	

Parameter	Data type	Description	Possible values
udiIDMask	UDINT	Bit pattern of the required area 1 ... Bit is relevant for the selection 0 ... Bit is not relevant for the selection	E.g. 000 1111 1111

* ... default value

Output parameters

Parameter	Data type	Description	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
aData	ARRAY [0...7] OF USINT	Array for storage of the data received		
usiDLC	USINT	Number of the data bytes in the CAN message (DLC = Data Length Count)	0 ... 8	0 bytes* ... 8 bytes
uiAvailable	UINT	- Number of received CAN messages since the last FB call - Current CAN message is taken into account	0	No CAN messages received between 2 FB calls
			n	n CAN messages received
udiID	UDINT	Identifier of the CAN message	- for Standard Frame (11 bits identifier): 0 ... 2047 - for Extended-Frame (29 bits identifier): 0 ... 536.870.911	

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_INACTIVE_INTERFACE	Error: Selected CAN channel is deactivated.
• ERR_BUFFER_OVERFLOW	Error: Transmission buffer full; CAN message cannot write to buffer storage and is not transmitted
• ERR_INVALID_VALUE	Error: at least 1 invalid input parameter or invalid combination of input parameters; Function call has been stopped.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!

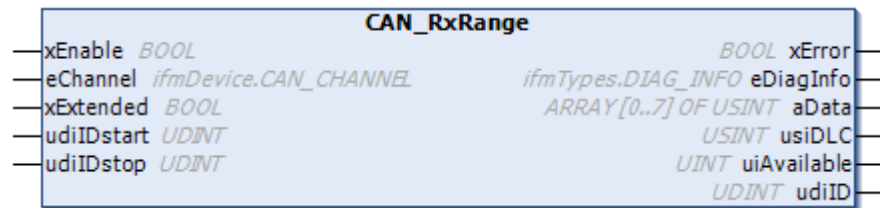
11.3.1.7 CAN_RxRange

Function block type: Function block (FB)

Behaviour model: ENABLE

Library: ifmRawCAN.library

Symbol in CODESYS:



Description

The FB receives CAN messages of a contiguous area with Standard Identifier (11 bit). The area is defined by an upper and lower limit.

The following rules apply to the definition of this area:

Lower and upper limit: 0 ... 2047 (11 bit identifier)

The value for the lower limit must be $\leq$ the value of the upper limit.

Example:

Lower limit: 000 0000 0010

Upper limit: 000 0000 1000

Result: All CAN messages with an identifier whose 4 least significant bits have a value between "0010" and "1000" are received.



Use the FB CAN_RxRange for CAN messages with Extended Identifier (29 bits).

▷ Data loss is possible.

▶ Use the following FB CAN_RxRange for CAN messages with Standard Identifier (11 bits):

▶ Set the building block input `xExtended` = FALSE (default setting).

▶ Use the following FB for CAN messages with Extended Identifier (29 bits): [CAN_RxRange](#) (→ [139](#))



General behaviour of the FB: [CAN_Rx](#) (→ [135](#))



The function block has the following behaviour after changing the receive ID during the runtime of the application

▷ The FB memory be not be reset completely. The outputs `aData` and `usiDLC` keep the last values. The `uiAvailable` counter is set to 0.

▶ Only use the FB with static (unchanged in operation) ID configurations at the inputs.

▶ Check the value of `uiAvailable` $\neq$ 0 before using the data.

Input parameters

Parameter	Data type	Description	Possible values	
xEnable	BOOL	Control activity of the FB	FALSE	Deactivate FB
			TRUE	Enable FB
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ 122)	

Parameter	Data type	Description	Possible values	
xExtended	BOOL	Requested frame type: - Standard frame (11 bits identifier) - Extended frame (29 bits identifier)	FALSE	Standard Frame* Required setting. Do not change!
			TRUE	Extended Frame Do not use! May lead to data loss!
udiIDStart	UDINT	Start of the required area	E.g. 000 0000 0010	
udiIDStop	UDINT	End of the required area	E.g. 000 0000 1000	

* ... default value

Output parameters

Parameter	Data type	Description	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
aData	ARRAY [0...7] OF USINT	Array for storage of the data received		
usiDLC	USINT	Number of the data bytes in the CAN message (DLC = Data Length Count)	0 ... 8	0 bytes* ... 8 bytes
uiAvailable	UINT	- Number of received CAN messages since the last FB call - Current CAN message is taken into account	0	No CAN messages received between 2 FB calls
			n	n CAN messages received
udiID	UDINT	Identifier of the CAN message	Standard Frame (11 bits identifier): 0 ... 2047	

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_INACTIVE_INTERFACE	Error: Selected CAN channel is deactivated.
• ERR_BUFFER_OVERFLOW	Error: Transmission buffer full; CAN message cannot write to buffer storage and is not transmitted
• ERR_INVALID_VALUE	Error: at least 1 invalid input parameter or invalid combination of input parameters; Function call has been stopped.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!

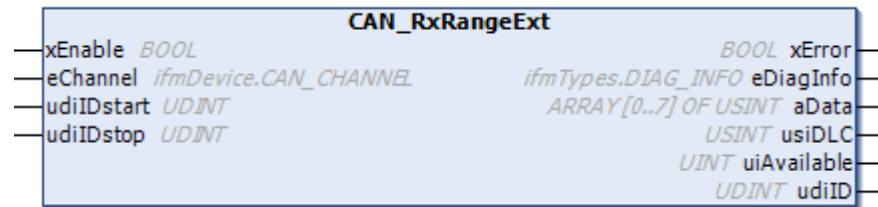
11.3.1.8 CAN_RxRangeExt

Function block type: Function block (FB)

Behaviour model: ENABLE

Library: ifmRawCAN.library

Symbol in CODESYS:



Description

The FB receives CAN messages of a contiguous area with Extended Identifier (29 bits). The area is defined by an upper and lower limit.

The following rules apply to the definition of this area:

- Lower and upper limit: 0 ... 536 870 911
- The value for the lower limit must be ≤ the value of the upper limit.

Example:

Lower limit: 0 0000 0000 0000 0000 0000 0000 0010

Upper limit: 0 0000 0000 0000 0000 0000 0000 1000

Result: All CAN messages with an identifier whose 4 least significant bits have a value between "0010" and "1000" are received.



General behaviour of the FB: [CAN_Rx \(→ 135\)](#)

FB for Standard Identifier (11 bits) [CAN_RxRange \(→ 139\)](#)



The function block has the following behaviour after changing the receive ID during the runtime of the application

- ▷ The FB memory be not be reset completely. The outputs `aData` and `uiDLC` keep the last values. The `uiAvailable` counter is set to 0.
- ▶ Only use the FB with static (unchanged in operation) ID configurations at the inputs.
- ▶ Check the value of `uiAvailable` ≠ 0 before using the data.

Input parameters

Parameter	Data type	Description	Possible values	
xEnable	BOOL	Control activity of the FB	FALSE	Deactivate FB
			TRUE	Enable FB
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ 122)	
udiIDStart	UDINT	Start of the required area	e.g. 0 0000 0000 0000 0000 0000 0000 0010	
udiIDStop	UDINT	End of the required area	e.g. 0 0000 0000 0000 0000 0000 0000 1000	

Output parameters

Parameter	Data type	Description	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
aData	ARRAY [0...7] OF USINT	Array for storage of the data received		
usiDLC	USINT	Number of the data bytes in the CAN message (DLC = Data Length Count)	0 ... 8	0 bytes* ... 8 bytes
uiAvailable	UINT	- Number of received CAN messages since the last FB call - Current CAN message is taken into account	0	No CAN messages received between 2 FB calls
			n	n CAN messages received
udiID	UDINT	Identifier of the CAN message	Extended-Frame (29 bits identifier) 0 ... 536.870.911	

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_INACTIVE_INTERFACE	Error: Selected CAN channel is deactivated.
• ERR_BUFFER_OVERFLOW	Error: Transmission buffer full; CAN message cannot write to buffer storage and is not transmitted
• ERR_INVALID_VALUE	Error: at least 1 invalid input parameter or invalid combination of input parameters; Function call has been stopped.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!

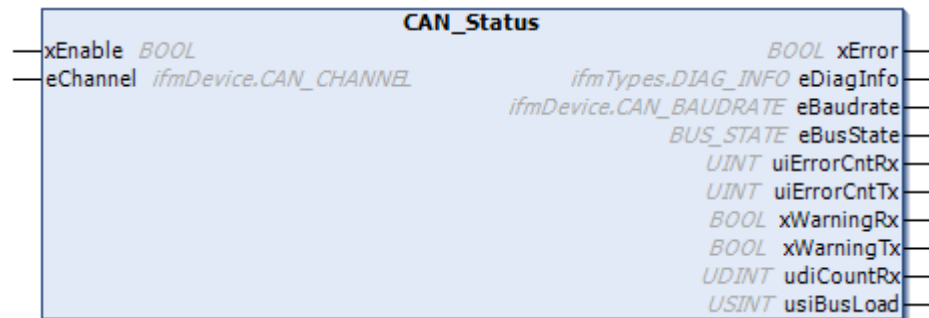
11.3.1.9 CAN_Status

Function block type: Function block (FB)

Behaviour model: ENABLE

Library: ifmRawCAN.library

Symbol in CODESYS:



Description

The FB reads the current status of the CAN network and provides the following status and diagnostic information:

- Baud rate
- Status of the CAN bus (status diagram)
- Counter Rx error
- Counter Tx error
- Warning Rx error
- Warning Tx error
- Counter of received CAN messages
- Bus load

If the value range of `udiCountRx` is exceeded, the counter starts at 0.

The bus load is calculated for standard frames. When using Extended Frames, the actual bus load may be higher than indicated on `usiBusLoad`.

Input parameters

Parameter	Data type	Description	Possible values	
xEnable	BOOL	Control activity of the FB	FALSE	Deactivate FB
			TRUE	Enable FB
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ 122)	

Output parameters

Parameter	Data type	Description	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	• An error has occurred • Action could not be executed • Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	

Parameter	Data type	Description	Possible values
eBaudrate	CAN_BAUDRATE	Baud rate of the CAN channel	CAN_BAUDRATE (ENUM) (→ 122)
eBusState	BUS_STATE	current state of the CAN interface	BUS_STATE (ENUM) (→ 147)
uiErrorCntRx	UINT	Error counter Rx messages	0...65535
uiErrorCntTx	UINT	Error counter Tx messages	0...65535
xWarningRx	BOOL	Rx error: Threshold for warning message exceeded (uiErrorCntRx > 96)	FALSE no warning
			TRUE warning
xWarningTx	BOOL	Tx error: Threshold for warning message exceeded (uiErrorCntTx > 96)	FALSE no warning
			TRUE warning
udiCountRx	UDINT	Number of detected CAN messages (independent of configured Rx messages)	0...4294967295
usiBusLoad	USINT	Bus load (in percent)	0...100

Diagnostic codes

• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!

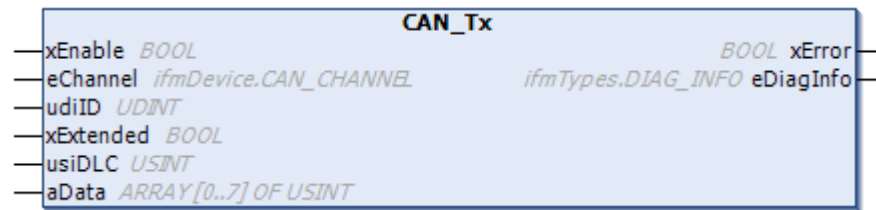
11.3.1.10 CAN_Tx

Function block type: Function block (FB)

Behaviour model: ENABLE

Library: ifmRawCAN.library

Symbol in CODESYS:



Description

By means of this FB CAN messages can be sent asynchronously. The FB writes the configured CAN message into the buffer storage of the selected CAN channel. When the CAN message is transmitted depends on the state of the CAN channel and the buffer storage. The FB and the PLC cycle do not have any influence on this.



The FB can be called several times during a PLC cycle.

The repeated call of the FB during a PLC cycle triggers a repeated transmission of the CAN message within the PLC cycle.

Input parameters

Parameter	Data type	Description	Possible values	
xEnable	BOOL	Control activity of the FB	FALSE	Deactivate FB
			TRUE	Enable FB
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ □ 122)	
udiID	UDINT	Identifier of the CAN message	- for Standard Frame (11 bits identifier): 0 ... 2047 - for Extended-Frame (29 bits identifier): 0 ... 536.870.911	
xExtended	BOOL	Requested frame type: - Standard frame (11 bits identifier) - Extended frame (29 bits identifier)	FALSE	Standard Frame*
			TRUE	Extended Frame
usiDLC	USINT	Number of the data bytes in the CAN message (DLC = Data Length Count)	0 ... 8	0 bytes* ... 8 bytes
aData	ARRAY [0...7] OF USINT	Array with the data to be sent		

* ... default value

Output parameters

Parameter	Data type	Description	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information

Parameter	Data type	Description	Possible values
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_INACTIVE_INTERFACE	Error: Selected CAN channel is deactivated.
• ERR_BUFFER_OVERFLOW	Error: Transmission buffer full; CAN message cannot write to buffer storage and is not transmitted
• ERR_INVALID_VALUE	Error: at least 1 invalid input parameter or invalid combination of input parameters; Function call has been stopped.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!

11.3.1.11 CAN_Info (GVL)

Name	Description	Data type	Possible values
CAN_State	State of the CAN channels	ARRAY[0...3] OF CAN_BUS_STATE	per array field: CAN_BUS_STATE (STRUCT) (→ 147)
Logger_Enabled	shows whether Log of the CAN channel is activated	ARRAY[0...3] OF BOOL	per array field
			FALSE deactivated
			TRUE activated

11.3.1.12 CAN_BUS_STATE (STRUCT)

Name	Data type	Description	Possible values
uiBaudrate	UINT	Baud rate of the CAN interface	CAN_BAUDRATE (ENUM) (→ 122)
eBusState	BUS_STATE	current state of the CAN interface	BUS_STATE (ENUM) (→ 147)
uiErrorCntRx	UINT	Error counter Rx messages	0...65535
uiErrorCntTx	UINT	Error counter Tx messages	0...65535
xWarningRx	BOOL	Rx error: Threshold for warning message exceeded (uiErrorCntRx > 96)	FALSE no warning
			TRUE warning
xWarningTx	BOOL	Tx error: Threshold for warning message exceeded (uiErrorCntTx > 96)	FALSE no warning
			TRUE warning
udErrorCntTx	UDINT	Number of detected CAN messages (independent of configured Rx messages)	0...4294967295

11.3.1.13 BUS_STATE (ENUM)

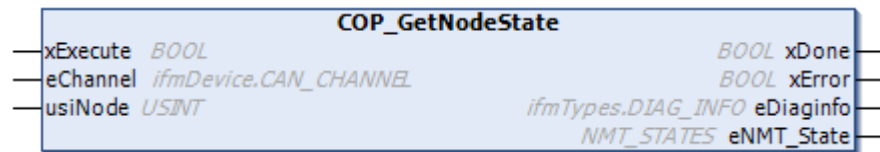
Name	Description	Possible values		Data type	Value
BUS_STATE	State of the CAN interface	UNDEFINED	State not available or not configured	INT	0
		ERROR_ACTIVE	Error counter Tx and Rx <= 96	INT	1
		ERROR_PASSIVE	Error counter Tx or Rx > 127 and error counter Tx or Rx < 255	INT	2
		ERROR_WARNING	Error counter Rx or Tx > 96 and error counter Rx or Tx <= 127	INT	3
		BUS_OFF	Error counter Tx > 255	INT	65535

11.3.2 Library ifmCANopenManager.library

The library contains program blocks (POU) and data structures for the programming of the functionality of a CANopen Manager.

11.3.2.1 COP_GetNodeState

Function block type: Function block (FB)
Behaviour model: EXECUTE
Library: ifmCANopenManager.library
Symbol in CODESYS:



Description

The FB indicates the current state of a CANopen node.

Input parameters

Parameter	Data type	Meaning	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE	do not execute FB
			TRUE	Execute FB. Execute the FB (xExecute = TRUE) until the function block execution is successfully terminated (xDone = TRUE). If xDone = TRUE, then reset the input xExecute to FALSE.
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ □ 122)	
usiNode	USINT	ID of the CANopen node	0	Local device
			1 ... 127	ID of the CANopen node

Output parameters

Parameter	Data type	Meaning	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	• FB successfully executed • FB can be called again
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	• An error has occurred • Action could not be executed • Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
eNMT_State	NMT_STATES	State of the CANopen node	NMT_STATES (ENUM) (→ □ 157)	

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_BUSY	Status: FB/function is currently executed.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!

• ERR_INVALID_VALUE	Error: at least 1 invalid input parameter or invalid combination of input parameters; Function call has been stopped.
• ERR_DEVICE_NOT_AVAILABLE	Error: Selected device unknown / not configured
• ERR_INVALID_CHANNEL	Error: Selected communication channel unknown / not configured

11.3.2.2 COP_SDOread

Function block type: Function block (FB)
Behaviour model: EXECUTE
Library: ifmCANopenManager.library
Symbol in CODESYS:



Description

The FB reads the contents of a Service Data Object (SDO) and writes them into a buffer storage. The SDO is selected via the CAN interface, the ID of the CANopen node, as well as index and subindex of the object directory.

The CANopen node has to reply to the request of the FB within a period of time defined by the user.

Input parameters

Parameter	Data type	Meaning	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE	do not execute FB
			TRUE	Execute FB. Execute the FB (xExecute = TRUE) until the function block execution is successfully terminated (xDone = TRUE). If xDone = TRUE, then reset the input xExecute to FALSE.
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ 122)	
usiNode	USINT	ID of the CANopen node	0	Local device
			1 ... 127	ID of the CANopen node
uiIndex	UINT	Index in the object directory		
usiSubIndex	USINT	Subindex of the index in the object directory		
pData	Pointer to USINT	Pointer on buffer storage		
udiBuffLen	UDINT	Size of the buffer storage (in byte)		
tTimeout	TIME	Max. response time	For example T#25ms	

Output parameters

Parameter	Data type	Meaning	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	- FB successfully executed - FB can be called again
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed

Parameter	Data type	Meaning	Possible values
xError	BOOL	Indication if an error has occurred during the FB execution	TRUE - An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)
udiLen	UDINT	Number of received bytes	

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_INVALID_CHANNEL	Error: Selected communication channel unknown / not configured
• ERR_INVALID_VALUE	Error: At least one input parameter is invalid or outside the value range.
• ERR_BUFFER_OVERFLOW	Error: Transmission buffer full; CAN message cannot write to buffer storage and is not transmitted
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_DEVICE_NOT_AVAILABLE	Error: Selected device unknown / not configured
• ERR_SDO_IDX_NOT_EXIST	Error: Object to be read/written does not exist
• ERR_SDO_SUBIDX_NOT_EXIST	Error: Subobject to be read/written does not exist
• ERR_SDO_UNSUPPORTED_ACCESS	Error: Read/write access to the selected object is not allowed
• ERR_SDO_DATA_TYPE	Error: Data type of the data to be written does not match the object or is outside the value range

11.3.2.3 COP_SDOWrite

Function block type: Function block (FB)
Behaviour model: EXECUTE
Library: ifmCANopenManager.library
Symbol in CODESYS:



Description

The FB writes the contents of a Service Data Object (SDO). The SDO is selected via the CAN interface, the ID of the CANopen node, as well as index and subindex of the object directory.

Input parameters

Parameter	Data type	Meaning	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE	do not execute FB
			TRUE	Execute FB. Execute the FB (xExecute = TRUE) until the function block execution is successfully terminated (xDone = TRUE). If xDone = TRUE, then reset the input xExecute to FALSE.
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ 122)	
usiNode	USINT	ID of the CANopen node	0	Local device
			1 ... 127	ID of the CANopen node
uiIndex	UINT	Index in the object directory		
usiSubIndex	USINT	Subindex of the index in the object directory		
pData	Pointer to USINT	Pointer on buffer storage		
udiLen	UDINT	Number of received bytes		
tTimeout	TIME	Max. response time	For example T#25ms	

Output parameters

Parameter	Data type	Meaning	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	- FB successfully executed - FB can be called again
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_INVALID_CHANNEL	Error: Selected communication channel unknown / not configured
• ERR_INVALID_VALUE	Error: At least one input parameter is invalid or outside the value range.
• ERR_BUFFER_OVERFLOW	Error: Transmission buffer full; CAN message cannot write to buffer storage and is not transmitted
• ERR_TIMEOUT	Error: The maximum permissible execution time was exceeded. The action was not finished.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_DEVICE_NOT_AVAILABLE	Error: Selected device unknown / not configured
• ERR_SDO_IDX_NOT_EXIST	Error: Object to be read/written does not exist
• ERR_SDO_SUBIDX_NOT_EXIST	Error: Subobject to be read/written does not exist
• ERR_SDO_UNSUPPORTED_ACCESS	Error: Read/write access to the selected object is not allowed
• ERR_SDO_DATA_TYPE	Error: Data type of the data to be written does not match the object or is outside the value range

11.3.2.4 COP_SendNMT

Function block type: Function block (FB)
Behaviour model: EXECUTE
Library: ifmCANopenManager.library
Symbol in CODESYS:



Description

The FB sends a command for the control of a CANopen node.

Input parameters

Parameter	Data type	Meaning	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE	do not execute FB
			TRUE	Execute FB. Execute the FB (xExecute = TRUE) until the function block execution is successfully terminated (xDone = TRUE). If xDone = TRUE, then reset the input xExecute to FALSE.
eChannel	CAN_CHANNEL	Identifier of the CAN Interface	CAN_CHANNEL (ENUM) (→ 122)	
usiNode	USINT	ID of the CANopen node	0	Local device
			1 ... 127	ID of the CANopen node
usiNMTservice	NMT_SERVICE	Command for the control of a CAN-open node	NMT_SERVICE (ENUM) (→ 157)	

Output parameters

Parameter	Data type	Meaning	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	- FB successfully executed - FB can be called again
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_INVALID_CHANNEL	Error: Selected communication channel unknown / not configured
• ERR_INVALID_VALUE	Error: at least 1 invalid input parameter or invalid combination of input parameters; Function call has been stopped.

• ERR_INTERNAL	Error: Internal system error ▶ Contact the ifm Service Center!
----------------	---

11.3.2.5 NMT_SERVICE (ENUM)

Name	Description	Possible values		Data type	Value
NMT_SERVICE	Command for the control of a CANopen node	SET_PRE_OPERATIONAL	Set preoperational state	INT	1
		SET_OPERATIONAL	Set operational state	INT	2
		RESET_NODE	Reset CAN node	INT	3
		RESET_COMM	Reset communication	INT	4
		STOP_NODE	Stop CAN node	INT	5

11.3.2.6 NMT_STATES (ENUM)

Name	Description	Possible values		Data type	Value
NMT_STATES	State of the CAN network	INIT	initialisation	INT	1
		PREOP	Preoperational	INT	2
		OPERATIONAL	Operational	INT	3
		STOP	STOP	INT	4
		NOT_AVAILABLE	Not available	INT	5
		UNKNOWN	unknown	INT	6

11.4 Input and output libraries

11.4.1 Library ifmConfigSwThreshold.library

The library features program blocks (POU) to configure the switching thresholds of inputs.

11.4.1.1 ConfigSwThreshold

Function block type: Function block (FB)

Library: ifmIOcommon.library

Symbol in CODESYS:



Description

The function block is used to set the switching thresholds for the values TRUE and FALSE of digital input channels in the operating modes IN_DIGITAL_CSI and IN_DIGITAL_CSO.

The following applies: $0 < \text{Switching threshold}_L < \text{Switching threshold}_H < 100\%$

Threshold	Default value for IN_DIGITAL_CSI	Default value for IN_DIGITAL_CSO
Switching threshold_L	30 % of VBB30	25 % of VBB30
Switching threshold_H	70 % of VBB30	65 % of VBB30

Tab. 2: Default switching thresholds IN_DIGITAL_CSI and IN_DIGITAL_CSO



Information about the function block behaviour:

The signal on the input `xExecute` must remain set to TRUE until the value at the output becomes `xDone` = TRUE. Only then will the module function be executed properly.

If the signal at `xExecute` = FALSE before `xDone` = becomes TRUE, the execution of the function block will be stopped.

- If `xDone` becomes = TRUE, then reset `xExecute` to FALSE.
- ▷ The execution of the function block is stopped and must only be restarted in the event of a fault.

Programming example:

```
VAR
  xExecute: BOOL:=TRUE;
  ConfigSwThreshold : ifmConfigSwThreshold.ConfigSwThreshold;
END_VAR
ConfigSwThreshold(xExecute:= xExecute)
If ConfigSwThreshold.xDone THEN
  xExecute:=FALSE;
END_IF
```



The diagnostic level of the input will be reset to the standard values when the switching threshold settings are changed. Then, call the function block `ConfigDiagLevel`.

Input parameters

Parameter	Data type	Meaning	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE	do not execute FB

Parameter	Data type	Meaning	Possible values	
xExecute	BOOL	Control execution of the FB	TRUE	Execute FB. Execute the FB (xExecute = TRUE) until the function block execution is successfully terminated (xDone = TRUE). After that, reset the input xExecute to FALSE with xDone = TRUE.
uiChannel	UINT	Input channel	Group + Channel → Data sheet → Chapter:: Note on wiring	
			Examples:	
			403	Group 4 + channel 3
			502	Group 5 + channel 2
uiLowThreshold	UINT	Low switching threshold for digital input in [%] of VBB30.	0...1000	
uiHighThreshold	UINT	Upper switching threshold for digital input in [%] of VBB30.	0...1000	



uiLowThreshold and uiHighThreshold are indicated in per mill [%] of VBB30 in the operating modes IN_DIGITAL_CSI and IN_DIGITAL_CSO.

Output parameters

Parameter	Data type	Meaning	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	- FB successfully executed - FB can be called again
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ 214):

• STAT_BUSY	Status: FB/function is currently executed.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_INACTIVE	Status: FB/function is inactive.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.

• DIAG_ACCESS	FB/function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.
---------------	---

11.4.2 Library ifmFastInput.library

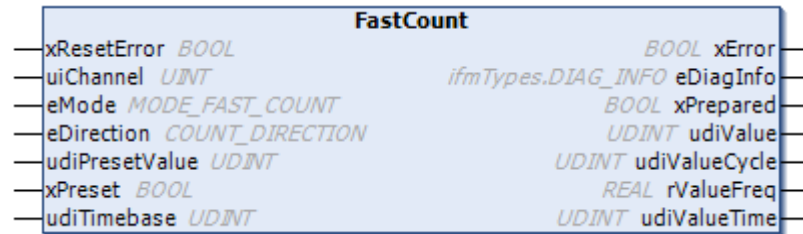
The library contains program blocks (POU) and enumeration types to control the quick inputs of the device.

11.4.2.1 FastCount

Function block type: Function block (FB)

Library: ifmIFastInput.library

Symbol in CODESYS:



Description

The FB functions as a counter block for pulses on fast input channels.

Input parameters

Parameter	Data type	Meaning	Possible values	
xResetError	BOOL	Reset request for an occurring error	FALSE => TRUE	Reset request to the lower level system
			otherwise	no action
uiChannel	UINT	Input channel	Group + Channel → Data sheet → Chapter:: Note on wiring	
			Examples:	
			403	Group 4 + channel 3
			502	Group 5 + channel 2
eMode	MODE_FAST_COUNT	Operating mode of the input channel	MODE_FAST_COUNT (ENUM) (→ 170)	
eDirection	COUNT_DIRECTION	Count direction	COUNT_DIRECTION (ENUM) (→ 170)	
udiPresetValue	UDINT	Preset counter value	permissible = 0...4 294 967 295	
xPreset	BOOL	Changeover switch: counter function active / adopt preset counter value	FALSE	counter active; the number of counted pulses is issued at udiValue.
			TRUE	The preset counter value is adopted; udiValue = udiPresetValue
udiTimebase	UDINT	Time base for frequency calculation in [ms] Only for eMode = • IN_COUNT_CSI • IN_COUNT_CSO	Value > 0 Example for determining the time base: $udiTimebase = 2 * 1 / frequency_{min}$ Frequency _{min} : lowest frequency to be detected	

Output parameters

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
xPrepared	BOOL	State of the FB outputs	FALSE	FB outputs invalid; FB is still processed
			TRUE	FB outputs valid; FB has been processed
udiValue	UDINT	Counter value; number of detected pulses	permissible = 0...4 294 967 295	
udiValueCycle	UDINT	Cycle time of the input signal in [μs]		
rValueFreq	REAL	frequency of the input signal in [Hz]		
udiValueTime	UDINT	Elapsed time since the last blanking pulse was detected in [μs]	0...4 294 967 295	

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

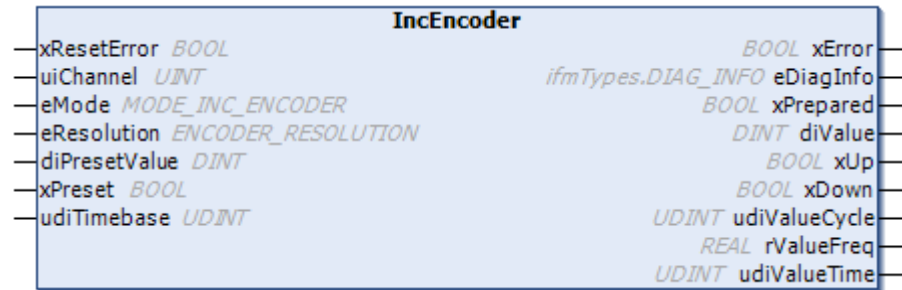
• STAT_PREPARING	Status: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_INACTIVE	Status: FB/function is inactive.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.
• DIAG_ACCESS	FB/function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.

11.4.2.2 IncEncoder

Function block type: Function block (FB)

Library: ifmIFastInput.library

Symbol in CODESYS:



Description

The FB is used to configure and to operate a digital input pair to record and count incremental encoder pulses.

Two frequency inputs constitute the input pair (channel A and channel B) that is configured and evaluated via the FB.

Behaviour at the counter limits

If the applicable value range is exceeded, the output switches to the minimum value of the applicable area. (= overflow)

If the applicable value range is not reached, the output switches to the maximum value of the applicable area. (= underflow)

Input parameters

Parameter	Data type	Meaning	Possible values	
xResetError	BOOL	Reset request for an occurring error	FALSE => TRUE	Reset request to the lower level system
			otherwise	no action
uiChannel	UINT	1. FRQ input channel (channel A) of the input channel pair.	Group + channel → Data sheet → Chapter: Note on wiring	
			Examples:	
			0	Channel A = IN0000 Channel B = IN0001 Channel IN0000 and channel IN0001 form the input channel pair
eMode	MODE_INC_ENCODER	Operating mode of the input channel	MODE_INC_ENCODER (ENUM) (→ 170)	
eResolution	ENCODER_RESOLUTION	Resolution / encoder mode	ENCODER_RESOLUTION (ENUM) (→ 170)	
diPresetValue	DINT	Preset counter value	-2 147 483 648...2 147 483 647	
xPreset	BOOL	Changeover switch: counter function active / adopt preset counter value	FALSE	counter active; the number of counted pulses is issued at udiValue.
			TRUE	The preset counter value is adopted; udiValue = udiPresetValue

Parameter	Data type	Meaning	Possible values
udiTimebase	UDINT	Time base for frequency calculation in [ms] Only for eMode = • IN_INC_ENCODER_CSI • IN_INC_ENCODER_CSO	Value > 0 Example for determining the time base: $\text{udiTimebase} = 2 * 1 / \text{frequency}_{\min}$ Frequency _{min} : lowest frequency to be detected

Output parameters

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
xPrepared	BOOL	State of the FB outputs	FALSE	FB outputs invalid; FB is still processed
			TRUE	FB outputs valid; FB has been processed
diValue	DINT	Counter value; number of detected pulses	permissible = -2 147 483 648...2 147 483 647	
xUp	BOOL	Code sequence upwards	FALSE	No count-up since the last call up
			TRUE	Count-up or overflow since that last call-up
xDown	BOOL	Code sequence downwards	FALSE	No count-down since the last call-up
			TRUE	Count-up or underflow since the last call-up
udiValueCycle	UDINT	Cycle time of the input signal in [μs]		
rValueFreq	REAL	frequency of the input signal in [Hz]		
udiValueTime	UDINT	Elapsed time since the last blanking pulse was detected in [μs]	0...4 294 967 295	

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_PREPARING	Status: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_INACTIVE	Status: FB/function is inactive.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.
• DIAG_ACCESS	FB/function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.

11.4.2.3 Period

Function block type: Function block (FB)

Library: ifmIFastInput.library

Symbol in CODESYS:



Description

The FB is used to configure and to operate an input channel or a pair of input channels to detect and count pulses.

In the operating modes IN_PHASE_CSI and IN_PHASE_CSO (to be set at the eMode function block input), a phase measurement is carried out on one input channel pair. The input channel pair is defined by indicating the channel with the even number of the input channel pair (channel A) at the input uiChannel.

In the other operating modes, a signal evaluation is carried out at the input channel defined at the uiChannel input.

Input parameters

Parameter	Data type	Meaning	Possible values	
xResetError	BOOL	Reset request for an occurring error	FALSE => TRUE	Reset request to the lower level system
			Otherwise	no action
uiChannel	UINT	The following applies to IN_PHASE_CSI and IN_PHASE_CSO: 1. FRQ input channel (channel A) of the input channel pair. The following applies to operating modes that only use one input channel: input channel (even)	Group + channel → Data sheet → Chapter:: Note on wiring	
			Example of the operating modes • IN_PHASE_CSI • IN_PHASE_CSO	
			0	Channel A = IN0000 Channel B = IN0001 Channel IN0000 and channel IN0001 form the input channel pair
			Example of operating modes with one input channel	
			402	Group 4 + channel 2 = input channel IN0402
eMode	MODE_PERIOD	Operating mode of the input channel	MODE_PERIOD (ENUM) (→ □ 171)	
ePeriod	FREQ_SENSE_PERIODS	Number of pulse periods for averaging Not for eMode = • IN_FREQUENCY_CSI • IN_FREQUENCY_CSO	FREQ_SENSE_PERIODS (ENUM) (→ □ 170)	

Parameter	Data type	Meaning	Possible values
udiTimebase	UDINT	Time base for frequency calculation in [ms] Only for eMode = • IN_FREQUENCY_CSI • IN_FREQUENCY_CSO	Value > 0 Example for determining the time base: $\text{udiTimebase} = 2 * 1 / \text{frequency}_{\min}$ Frequency _{min} : lowest frequency to be detected

Output parameters

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
xPrepared	BOOL	State of the FB outputs	FALSE	FB outputs invalid; FB is still processed
			TRUE	FB outputs valid; FB has been processed
udiValueCycle	UDINT	Cycle time of the input signal in [μs]		
rValueFreq	REAL	Frequency of the input signal in [Hz]		
udiValueTime	UDINT	Elapsed time since the last blanking pulse was detected in [μs]	0...4 294 967 295	
uiValueRatio	UINT	Depends on the mode that is set in the eMode input. Pulse/pause ratio of the input signal in [%] at: • IN_PERIOD_RATIO_CSI • IN_PERIOD_RATIO_CSO • IN_PERIOD_RATIO_US_CSI • IN_PERIOD_RATIO_US_CSO		
		Phase shift of the input signal at the B channel to the signal at the A channel in [°] • IN_PHASE_CSI • IN_PHASE_CSO		

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_PREPARING	Status: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_INACTIVE	Status: FB/function is inactive.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.
• DIAG_ACCESS	FB/function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.

• DIAG_SLOW_SIGNAL	No valid measurement at the input for too long.
--------------------	---

11.4.2.4 COUNT_DIRECTION (ENUM)

Name	Description	Possible values	
COUNT_DIRECTION	Count direction	COUNT_OFF	Counting function off
		COUNT_UP	Counting function up
		COUNT_DOWN	Counting function down

11.4.2.5 ENCODER_RESOLUTION (ENUM)

Name	Description	Possible values	
ENCODER_RESOLUTION	resolution	FULL_PERIOD	Counts each rising edge on one channel (A)
		HALF_PERIOD	Counts each rising and falling edge on one channel (A)
		EVERY_EDGE	Counts each rising and falling edge on all channels (A and B)

11.4.2.6 FREQ_SENSE_PERIODS (ENUM)

Name	Description	Possible values	
FREQ_SENSE_PERIODS	Number of clock periods for the averaging	PERIODS_n with n = 1	No averaging
		PERIODS_n with n = 2...16	Averaging via n periods After the first full period, the measured value is already provided. After the second full period, averaged values are provided.

11.4.2.7 MODE_FAST_COUNT (ENUM)

Name	Description	Possible values	
MODE_FAST_COUNT	Operating mode of the inputs	UNCHANGED	Setting remains unchanged
		IN_COUNT_CSI	Input to count fast signal edges; CSI
		IN_COUNT_CSO	Input to count fast signal edges; CSO
		MONITOR	- Only output data will be updated. - Values, configurations and process data will not be written. - For applications that are not owners of the resource.
		OFF	Block disabled.

11.4.2.8 MODE_INC_ENCODER (ENUM)

Name	Description	Possible values	
MODE_INC_ENCODER	Operating mode of the input	UNCHANGED	Setting remains unchanged
		IN_INC_ENCODER_CSI	Input for the evaluation of an incremental encoder, channel A; CSI
		IN_INC_ENCODER_CSO	Input for the evaluation of an incremental encoder, channel A; CSO

Name	Description	Possible values	
MODE_INC_ENCODER	Operating mode of the input	MONITOR	- Only output data will be updated. - Values, configurations and process data will not be written. - For applications that are not owners of the resource.
		OFF	Block disabled.

11.4.2.9 MODE_PERIOD (ENUM)

Name	Description	Possible values	
MODE_PERIOD	Operating mode of the period input	UNCHANGED	Setting remains unchanged
		IN_FREQUENCY_CSI	Input for frequency measurement; CSI
		IN_FREQUENCY_CSO	Input for frequency measurement; CSO
		IN_PERIOD_RATIO_CSI	Input for absolute and ratiometric period measurement; Measuring resolution 6.25 μ s; maximum period duration 104 s; CSI
		IN_PERIOD_RATIO_CSO	Input for absolute and ratiometric period measurement; Measuring resolution 6.25 μ s; maximum period duration 104 s; CSO
		IN_PHASE_CSI	Input pair for phase measurement, CSI
		IN_PHASE_CSO	Input pair for phase measurement, CSO
		OFF	Block disabled.
		IN_PERIOD_RATIO_US_CSI	Input for absolute and ratiometric period measurement; Measuring resolution 1 μ s; maximum period duration 16 s; CSI
		IN_PERIOD_RATIO_US_CSO	Input for absolute and ratiometric period measurement; Measuring resolution 1 μ s; maximum period duration 16 s; CSO
		MONITOR	- Only output data will be updated. - Values, configurations and process data will not be written. - For applications that are not owners of the resource.

11.4.3 Library ifmIOcommon.library

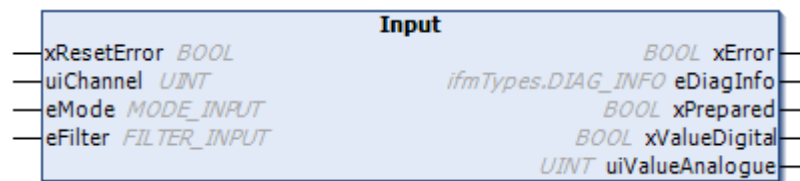
The library contains program blocks (POU) and enumeration types for the control of the inputs and outputs of the device.

11.4.3.1 Input

Function block type: Function block (FB)

Library: ifmIOcommon.library

Symbol in CODESYS:



Description

The FB is used to configure and read a digital or analogue input channel.

Operating mode:

The operating mode of the input channel can be set via the eMode input. The measurement and the output value result from the set operating mode.

Filter:

The input signal can be changed with a digital low-pass filter. Configure the filter via the input eFilter.

Input parameters

Parameter	Data type	Meaning	Possible values	
xResetError	BOOL	Reset request for an occurring error	FALSE => TRUE	Reset request to the lower level system
			otherwise	no action
uiChannel	UINT	Input channel	Group + Channel → Data sheet → Chapter:: Note on wiring	
			Examples:	
			403	Group 4 + channel 3
			502	Group 5 + channel 2
eMode	MODE_INPUT	Operating mode of the input channel	MODE_INPUT (ENUM) (→ 187)	
eFilter	FILTER_INPUT	Filter definition of the input channel	FILTER_INPUT (ENUM) (→ 186)	

Output parameters

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
xPrepared	BOOL	State of the FB outputs	FALSE	FB outputs invalid; FB is still processed
			TRUE	FB outputs valid; FB has been processed

Parameter	Data type	Meaning	Possible values	
xValueDigital	BOOL	Logical state of the input channel	FALSE	Low level or operating mode is "Analogue"
			TRUE	High Level
uiValueAnalogue	UINT	Measured input value in analogue operating mode. The interpretation of the input value depends on the setting at the eMode input. MODE_INPUT (ENUM) (→ □ 187)	0...65535 Examples with different operating modes at the eMode input (MODE_INPUT): • IN_VOLTAGE_10: 0...10 000 [mV] • IN_VOLTAGE_32: 0...32 000 [mV] • IN_VOLTAGE_RATIO: 0...1 000 [%] • IN_CURRENT_CSI: 0...20 000 [μA] • IN_RESISTOR: 16...30 000 [Ω]	

Diagnostic codes

Messages / diagnostic codes of the function blocks ([→ □ 214](#)):

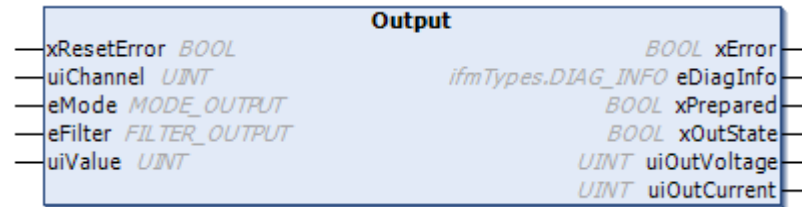
• STAT_PREPARING	Status: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_INACTIVE	Status: FB/function is inactive.
• ERR_INTERNAL	Error: internal system error. ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• ERR_COMPARE_MISMATCH	Error: Deviation of measured value and monitoring value too high
• ERR_OVERLOAD_CURRENT	Error: Maximum current exceeded.
• ERR_STUCK_AT_HIGH	Error: Signal frozen, signal state high.
• ERR_STUCK_AT_LOW	Error: Signal frozen, signal state low.
• ERR_RANGE_OVERRUN	Error: The measured value exceeds the set upper limit value.
• ERR_RANGE_UNDERRUN	Error: The measured value does not reach the set lower limit value.
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.
• DIAG_NO_CALIB	The selected resource has no valid calibration. The displayed values are maybe faulty.
• DIAG_ACCESS	FB/Function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.
• DIAG_OVERVOLTAGE_VBBX	• For inputs: Error: Reference voltage exceeded. • For outputs. Error: Voltage of the corresponding output groups supply or at VBB30 / VBB15 not reached.
• DIAG_OVERVOLTAGE	Maximum signal voltage exceeded.
• DIAG_RANGE_OVERRUN	The measured value exceeds the set upper limit value.
• DIAG_RANGE_UNDERRUN	The measured value does not reach the set lower limit value.
• DIAG_INVALID_FILTER	One or several filter settings are invalid.
• DIAG_COMPARE_MISMATCH	A discrepancy was detected between the mission values and the monitor values of the selected resource.

11.4.3.2 Output

Function block type: Function block (FB)

Library: ifmIOcommon.library

Symbol in CODESYS:



Description

The FB is used to configure and control a digital or analogue output channel.

Input parameters

Parameter	Data type	Meaning	Possible values	
xResetError	BOOL	Reset request for an occurring error	FALSE => TRUE	Reset request to the lower level system
			otherwise	no action
uiChannel	UINT	Output channel	Group + Channel → Data sheet → Chapter:: Note on wiring	
			Examples:	
			103	Group 1 + channel 3
			301	Group 3 + channel 1
eMode	MODE_OUTPUT	Operating type of the output channel	MODE_OUTPUT (ENUM) (→ 187)	
eFilter	FILTER_OUTPUT	Filter definition of the output channel	FILTER_OUTPUT (ENUM) (→ 186)	
uiValue	UINT	Value that is to be written to the output		
		In the digital mode or sensor supply mode; if setting at the eMode input =	0	Output deactivated
		• OUT_DIGITAL_CSI • OUT_DIGITAL_CSO • OUT_SENSOR_05 • OUT_SENSOR_10	1	Output activated
		In analogue mode; if setting at the eMode input =	0...10 000	
		• OUT_ANALOGUE_10 Values indicated in [mV]		

Output parameters



The state of the output `xOutState` may deviate from the required output state if e.g. an output group has been deactivated due to an error.

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication of whether an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication of whether an error has occurred during the FB execution	TRUE	- An error has occurred - The action could not be executed - Observe the diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
xPrepared	BOOL	State of the FB outputs	FALSE	FB outputs invalid; FB is still being processed
			TRUE	FB outputs valid; FB has been processed
xOutState	BOOL	Return value activation state of the selected output The state may deviate from the required output state if e.g. an output group has been deactivated due to an error.	FALSE	Output is deactivated
			TRUE	Output is activated
uiOutVoltage	UINT	Current output voltage in (value in mV) Only available for the operating modes "analogue" and "sensor"	0	Operating mode neither "analogue" nor "sensor"
			≠ 0	Output voltage
uiOutCurrent	UINT	Present output current in (value in mA) Not available in the following operating modes: - OUT_DIGITAL_CSI - OUT_ANALOGUE_10 - OUT_SENSOR_05 - OUT_SENSOR_10	0...final value of the measuring range	

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_PREPARING	Status: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_INACTIVE	Status: FB/function is inactive.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• ERR_STUCK_AT_HIGH	Error: Signal frozen, signal state high.
• ERR_STUCK_AT_LOW	Error: Signal frozen, signal state low.
• ERR_OVERLOAD_CURRENT	Error: Maximum current exceeded.
• ERR_LOW_CURRENT	Error: Minimum current not reached.
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.
• DIAG_NO_CALIB	The selected resource has no valid calibration. The displayed values are maybe faulty.
• DIAG_ACCESS	FB/function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.
• DIAG_STUCK_AT_HIGH	Error: Signal frozen, signal state high.
• DIAG_STUCK_AT_LOW	Error: Signal frozen, signal state low.
• DIAG_OVERLOAD_CURRENT	Maximum current exceeded.
• DIAG_LOW_CURRENT	Minimum current not reached.

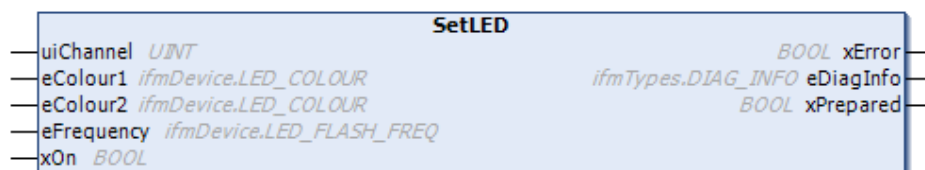
• DIAG_INVALID_FILTER	One or several filter settings are invalid.
• DIAG_UNDERVOLTAGE_VBBX	• For inputs. Error: Reference voltage not reached. • For outputs. Error: The voltage of the corresponding output group supply or at VBB30 / VBB15 is not reached.

11.4.3.3 SetLED

Function block type: Function block (FB)

Library: ifmIOcommon.library

Symbol in CODESYS:



Description

The FB is used to configure and control an LED.

Input parameters

Parameter	Data type	Meaning	Possible values	
uiChannel	UINT	Output channel of the ED	0...3	Device LED APP 0...3
eColour1	LED_COLOUR (ENUM)	LED colour status 1	LED_COLOUR (ENUM) (→ 122)	
eColour2	LED_COLOUR (ENUM)	LED colour status 0	LED_COLOUR (ENUM) (→ 122)	
eFrequency	ENUM	LED flashing frequency	LED_FLASH_FREQ (ENUM) (→ 123)	
xOn	BOOL	Switch LED	FALSE	LED off
			TRUE	LED on

Output parameters

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
xPrepared	BOOL	State of the FB outputs	FALSE	FB outputs invalid; FB is still processed
			TRUE	FB outputs valid; FB has been processed

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_PREPARING	Status: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
• ERR_INVALID_FREQUENCY	Error: Unsupported frequency.
• ERR_INVALID_COLOUR	Error: Unsupported colour.

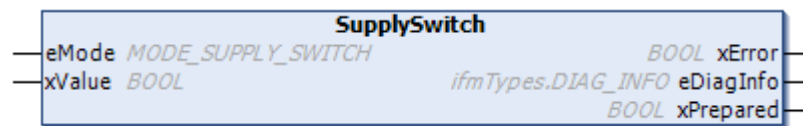
• ERR_INVALID_VALUE	Error: At least one input parameter is invalid or outside the value range.
• ERR_INSTANCE	Error: Instance is ZERO or invalid.
• ERR_ACCESS	Error: FB/function cannot access the required resource; Resource is blocked by another task.
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_NOT_SUPPORTED	Error: Invalid function calls; Function is not supported.

11.4.3.4 SupplySwitch

Function block type: Function block (FB)

Library: ifmIOcommon.library

Symbol in CODESYS:



Description



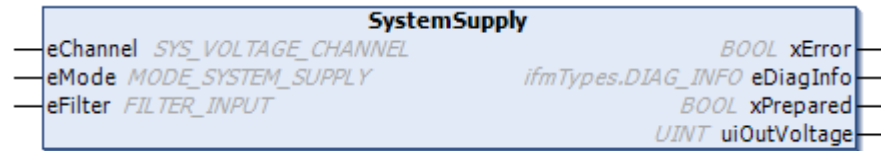
The FB is not supported by the device.

11.4.3.5 SystemSupply

Function block type: Function block (FB)

Library: ifmIOcommon.library

Symbol in CODESYS:



Description

The FB indicates the value of the system voltage.

Input parameters

Parameter	Data type	Meaning	Possible values
eChannel	SYS_VOLTAGE_CHANNEL (ENUM)	System voltage channel	SYS_VOLTAGE_CHANNEL (ENUM) (→ 188)
eMode	MODE_SYSTEM_SUPPLY	Operating mode	MODE_SYSTEM_SUPPLY (ENUM) (→ 188)
eFilter	FILTER_INPUT	Filter definition of the input channel	FILTER_INPUT (ENUM) (→ 186)

Output parameters

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
xPrepared	BOOL	State of the FB outputs	FALSE	FB outputs invalid; FB is still processed
			TRUE	FB outputs valid; FB has been processed
uiOutVoltage	UINT	Current output voltage of the selected system voltage channel in (value in mV)	0...maximum operating voltage	

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_PREPARING	Status: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
• STAT_INACTIVE	Status: FB/function is inactive.
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!

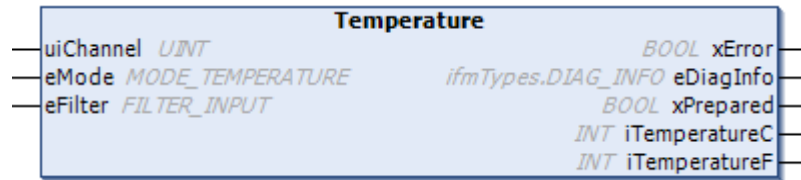
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• DIAG_ACCESS	FB/function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.
• DIAG_NO_CALIB	The selected resource has no valid calibration. The displayed values are maybe faulty.
• DIAG_OVERVOLTAGE_VBBX	• For inputs: Error: Reference voltage exceeded. • For outputs: Error: Voltage of the corresponding output groups supply or at VBB30 / VBB15 not reached.
• DIAG_INVALID_FILTER	One or several filter settings are invalid.

11.4.3.6 Temperature

Function block type: Function block (FB)

Library: ifmIOcommon.library

Symbol in CODESYS:



Description

The function block indicates the temperature depending on the set channel:

Channel 0 = not used

Channel 1 = processor temperature

Input parameters

Parameter	Data type	Meaning	Possible values
uiChannel	UINT	Input channel	0001: Channel 1 = processor temperature
eMode	MODE_TEMPERATURE	Operating mode	MODE_TEMPERATURE (ENUM) (→ 188)
eFilter	FILTER_INPUT	Filter definition of the input channel	FILTER_INPUT (ENUM) (→ 186)

Output parameters

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
xPrepared	BOOL	State of the FB outputs	FALSE	FB outputs invalid; FB is still processed
			TRUE	FB outputs valid; FB has been processed
iTemperatureC	INT	Measured temperature (value in °C)	e.g. 35	
iTemperatureF	INT	Measured temperature in (value in °F)	e.g. 95	

Diagnostic codes

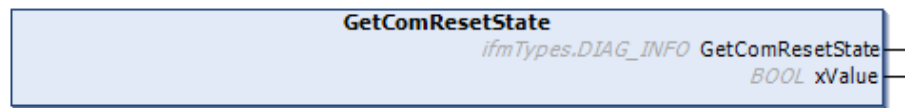
Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_PREPARING	Status: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_INACTIVE	Status: FB/function is inactive.

• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.
• DIAG_ACCESS	FB/Function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.
• DIAG_LOW_TEMPERATURE	Temperature in the device at the set measuring channel falls below the lower warning limit value.
• DIAG_HIGH_TEMPERATURE	Temperature in the device at the set measuring channel exceeds the upper warning limit value.
• DIAG_INVALID_FILTER	One or several filter settings are invalid.

11.4.3.7 GetComResetState

Function block type: Function (FUN)
Library: ifmIOcommon.library
Symbol in CODESYS:



Description

The FUN GetComResetState shows the activation status of the COM_RESET input of the device at the output xValue.

- FALSE = User-defined settings of the `comconf.cfg` file are used for the communication interfaces.
- TRUE = Factory settings of the file `comconf.cfg` are used for the communication interfaces.



The activation status of the input may not correspond to the actual status of the device's communication settings.

The device determines when the actual evaluation and configuration of the communication settings has taken place.

Normally, the status of this input is evaluated once at startup.

The status can then change without affecting the actual configuration any further.

Output parameters

Parameter	Data type	Meaning	Possible values
GetComResetState	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)
xValue	BOOL	Current status of the connection RESET-COM (COM_RESET) of the device.	• FALSE: Low signal, user-defined settings of the file <code>comconf.cfg</code> are used. • TRUE: High signal, factory settings of the file <code>comconf.cfg</code> are used.

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_BUSY	Status: FUN is currently executed.
• STAT_DONE	Status: FUN has been successfully executed and completed. There are valid results on the outputs.
• ERR_INTERNAL	Error: Unknown error ► Contact the ifm Service Center!
• ERR_NOT_SUPPORTED	Error: Invalid function call; function is not supported.
• ERR_INVALID_MODULE	Error: The specified module is not available for the device.

11.4.3.8 FILTER_INPUT (ENUM)

The input signal can be changed with a digital low-pass filter.

For the output signal of the function bloc, the delay time is changed to the input signal change by the filter. This applies both to the switch-on and the switch-off pulse.

Name	Description	Possible values	Digital signal delay	Analogue signal delay
FILTER_INPUT	Valid filters for inputs of the function blocks	UNCHANGED	No change of settings	
		FILTER_0 (default)	0.6 ms (no digital low-pass filter is set)	1.7 ms (no digital low-pass filter is set)
		FILTER_1	0.9 ms	3.3 ms
		FILTER_2	2.1 ms	7.0 ms
		FILTER_3	4.0 ms	14.1 ms
		FILTER_4	7.6 ms	28.9 ms
		FILTER_5	15.2 ms	58.4 ms
		FILTER_6	30.8 ms	117.2 ms
		FILTER_7	61.6 ms	235.2 ms
		FILTER_8	123.2 ms	470.8 ms
		FILTER_9	246.4 ms	942.4 ms
		FILTER_10	493.2 ms	1885.6 ms
		FILTER_11	986.4 ms	3772.0 ms
		FILTER_12	1972.4 ms	7544.4 ms

More information: Filter ([→ Filter](#)  221)

11.4.3.9 FILTER_OUTPUT (ENUM)

Filter setting for current measurement of an output.

The signal of the current measurement is damped via a first-order low-pass filter.

Name	Description	Possible values	
FILTER_OUTPUT	Valid filters for outputs of the function blocks	UNCHANGED	No change of settings
		FILTER_0 (default)	1.7 ms
		FILTER_1	1.8 ms
		FILTER_2	2.4 ms
		FILTER_3	3.9 ms
		FILTER_4	7.4 ms
		FILTER_5	14.7 ms
		FILTER_6	29.3 ms
		FILTER_7	58.8 ms
		FILTER_8	117.7 ms
		FILTER_9	235.6 ms
		FILTER_10	471.4 ms
		FILTER_11	943.0 ms
		FILTER_12	1886.1 ms

More information: Filter ([→ Filter](#)  221)

11.4.3.10 MODE_INPUT (ENUM)

Name	Description	Possible values	
MODE_INPUT	Operating mode of the inputs	UNCHANGED	Preset mode is maintained
		IN_DIGITAL_CSI (default)	Input for analogue value measurement and digital evaluation without diagnostics; CSI
		IN_DIGITAL_CSI_NAMUR	Input for analogue value measurement and digital evaluation with NAMUR-capable diagnostics; CSI
		IN_VOLTAGE_10	Input for analogue current measurement 0...10 V; CSI
		IN_VOLTAGE_32	Input for analogue current measurement 0...32 V; CSI
		IN_VOLTAGE_RATIO	Input for ratiometric current measurement in relation to VBB30; 0...1 000 ‰; CSI
		IN_CURRENT_CSI	Input for current measurement 0...20 mA; CSI
		IN_RESISTOR	Input for resistance measurement; 16...30 000 Ω; CSO
		IN_DIGITAL_CSO	Input for analogue value measurement and digital evaluation without diagnostics; CSO
		MONITOR	No parameters or process data are written. Only the FB output data is updated. For use in a PLC application to which the resource does not belong.
		OFF	POU disabled.

11.4.3.11 MODE_OUTPUT (ENUM)

Name	Description	Possible values	
MODE_OUTPUT	Operating mode of the outputs	UNCHANGED	Preset mode is maintained
		OUT_DIGITAL_CSI	Digital output without diagnostics; CSI
		OUT_DIGITAL_CSO (default for standard outputs)	Digital output without diagnostics; CSO
		OUT_ANALOGUE_10	Analogue output to generate a selectable voltage 0...10 V without diagnostics. Generated with the help of a filtered PWM signal. CSO
		OUT_SENSOR_05	Output with fixed output voltage 5 V for the sensor supply without diagnostics and without protection. CSO
		OUT_SENSOR_10	Output with fixed output voltage 10 V for the sensor supply without diagnostics and without protection. CSO
		MONITOR	No parameters or process data are written. Only the FB output data is updated. For use in a PLC application to which the resource does not belong.
		OFF (default for analogue outputs and sensor supply)	Block disabled.

11.4.3.12 MODE_SYSTEM_SUPPLY (ENUM)

Name	Description	Possible values	
MODE_SYSTEM_SUPPLY	Operating mode	UNCHANGED	Preset mode is maintained
		SYS_SUPPLY	POU enabled.
		MONITOR	No parameters or process data are written. Only the FB output data is updated. For use in a PLC application to which the resource does not belong.
		OFF	Block disabled.

11.4.3.13 MODE_TEMPERATURE (ENUM)

Name	Description	Possible values	
MODE_TEMPERATURE	Operating mode	UNCHANGED	Preset mode is maintained
		SYS_TEMPERATURE	POU enabled.
		MONITOR	No parameters or process data are written. Only the FB output data is updated. For use in a PLC application to which the resource does not belong.
		OFF	Block disabled.

11.4.3.14 SYS_VOLTAGE_CHANNEL (ENUM)

Name	Description	Possible values	
SYS_VOLTAGE_CHANNEL	List of all available system voltages.	VBB30	Terminal 30 system voltage
		VBB15	Terminal 15 system voltage of the ignition switch

11.4.4 Library ifmIOconfigDiagProt.library

The library contains program blocks (POU) and enumeration types to configure the I/O-related diagnostics and protective functions. The behaviour in case of errors and for diagnostic purposes can be set with it.

11.4.4.1 ConfigDiagLevel

Function block type: Function block (FB)
Behaviour model: Execute
Library: ifmIOconfigDiagProt.library
Symbol in CODESYS:



Description

The FB is used to set the diagnostic level of the system-internal error detection of input and output channels and output channel groups. You can set, for example, the time delay and the upper and lower limit value of the error detection.

ATTENTION

Faulty configuration of the following parameters:

- uiDetectionTime
- uiDiagLimMin
- uiDiagLimMax

When setting a time behaviour configuration of the lower / upper limit value of input channels/output channels that is different from the factory setting.

- ▷ Risk of material damage.
- ▷ Failure or delay of error detection is possible.
- ▶ Set upper and lower limit value and verify with a function test.
- ▶ Set uiDetectionTime and verify with a function test. Errors will not be detected before the uiDetectionTime has elapsed.



To prevent incorrect diagnostics on outputs in the operating mode OUT_PWM_CS0:

- ▶ Set the DetectionTime as follows: $uiDetectionTime \geq 2 / uiFrequency$
- ▶ Note: PWM1000 (→ 205) and SF_PWM1000



Information about the function block behaviour:

The signal on the input xExecute must remain set to TRUE until the value at the output becomes xDone = TRUE. Only then will the module function be executed properly.

If the signal at xExecute = FALSE before xDone = becomes TRUE, the execution of the function block will be stopped.

- ▶ If xDone becomes = TRUE, then reset xExecute to FALSE.
- ▷ The execution of the function block is stopped and must only be restarted in the event of a fault.

Programming example:

```

VAR
  xExecute: BOOL:=TRUE;
  ConfigDiagLevel : ifmIOconfigDiagProt.ConfigDiagLevel;
END_VAR
  ConfigDiagLevel(xExecute:= xExecute )
  If ConfigDiagLevel.xDone THEN
    xExecute:=FALSE;
  END_IF

```

Input parameters

Parameter	Data type	Meaning	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE	do not execute FB
			TRUE	Execute FB. Execute the FB (xExecute = TRUE) until the function block execution is successfully terminated (xDone = TRUE). After that, reset the input xExecute to FALSE with xDone = TRUE.
uiChannel	UINT	Input/output channel or output channel group to be set	Group + Channel → Data sheet → Chapter:: Note on wiring	
			Examples:	
			403	Group 4 + channel 3
			502	Group 5 + channel 2
eDestination	DEST	Type of channel / output channel group	DEST (ENUM) (→ 195)	
uiDetection-Time	UINT	Time delay in [ms] until an error issued after the user-defined upper/lower limit has been exceeded or not reached.		
uiDiagLimMin	UINT	Lower limit value to diagnose an error. For inputs: Indication in [μA], [mV], [Ω] or [%], depends on the operating mode that is set. For outputs: Indication in [mA] or [mV], depending on the operating mode that is set.		
uiDiagLimMax	UINT	Upper limit value to diagnose an error. For inputs: Indication in [μA], [mV], [Ω] or [%], depends on the operating mode that is set. For outputs: Indication in [mA] or [mV], depending on the operating mode that is set.		



uiDiagLimitMin and uiDiagLimitMax are indicated in per mill [%] of VBB30 in the operating modes IN_DIGITAL_CSI and IN_DIGITAL_CSO.

Output parameters

Parameter	Data type	Meaning	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	• FB successfully executed • FB can be called again
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	• An error has occurred • Action could not be executed • Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_BUSY	Status: FB/function is currently executed.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_INACTIVE	Status: FB/function is inactive.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.
• DIAG_ACCESS	FB/function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.
• DIAG_WINDOW	Invalid value for user diagnostics window.

11.4.4.2 ConfigDiagProt

Function block type: Function block (FB)
Behaviour model: Execute
Library: ifmIOconfigDiagProt.library
Symbol in CODESYS:



Description

The FB is used to configure the diagnostic and protective behaviour of input and output channels and output channel groups.

ATTENTION

When configuring the diagnostic and protective behaviour of input channels / output channels in a way that is different from the factory setting (factory setting: `eDIAG_PROT_MODE = DIAG_PROT`).

- ▷ Risk of material damage.
- ▷ Risk of failure or incorrect diagnostic and protective behaviour.
- ▶ Ensure fail-safe diagnostic and protective behaviour of the input channels and output channels.
- ▶ Program the evaluation of the diagnostic messages / measured values in the IEC application.



Information about the function block behaviour:

The signal on the input `xExecute` must remain set to TRUE until the value at the output becomes `xDone = TRUE`. Only then will the module function be executed properly.

If the signal at `xExecute = FALSE` before `xDone =` becomes TRUE, the execution of the function block will be stopped.

- ▶ If `xDone` becomes = TRUE, then reset `xExecute` to FALSE.
- ▷ The execution of the function block is stopped and must only be restarted in the event of a fault.

Programming example:

```
VAR
  xExecute: BOOL:=TRUE;
  ConfigDiagProt : ifmIOconfigDiagProt.ConfigDiagProt;
END_VAR
ConfigDiagProt(xExecute:= xExecute )
If ConfigDiagProt.xDone THEN
  xExecute:=FALSE;
END_IF
```

Input parameters

Parameter	Data type	Meaning	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE	do not execute FB

Parameter	Data type	Meaning	Possible values	
xExecute	BOOL	Control execution of the FB	TRUE	Execute FB. Execute the FB (xExecute = TRUE) until the function block execution is successfully terminated (xDone = TRUE). After that, reset the input xExecute to FALSE with xDone = TRUE.
uiChannel	UINT	Input/output channel or output channel group to be set	Group + Channel → Data sheet → Chapter:: Note on wiring	
			Examples:	
			403	Group 4 + channel 3
			502	Group 5 + channel 2
eDestination	DEST	Type of channel / output channel group	DEST (ENUM) (→ 195)	
eDiagMode	eDIAG_PROT_MODE	Setting of the error detection type	eDIAG_PROT_MODE (ENUM) (→ 195)	

Output parameters

Parameter	Data type	Meaning	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	• FB successfully executed • FB can be called again
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	• An error has occurred • Action could not be executed • Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_BUSY	Status: FB/function is currently executed.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_INACTIVE	Status: FB/function is inactive.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.
• DIAG_ACCESS	FB/function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.
• DIAG_DIAG_PROT_MODE	Invalid mode for diagnostic/protective function.

11.4.4.3 DEST (ENUM)

Name	Description	Possible values	
DEST	Type of channel	INPUT	Input
		OUTPUT	Output
		OUT_GROUP	Output group

11.4.4.4 eDIAG_PROT_MODE (ENUM)

Name	Description	Possible values	
eDIAG_PROT_MODE	Setting of the I/O error diagnostics mode	DIAG_PROT	Initial value Diagnostics and protection activated, in case of error: - ERR message - Set input / output values = 0
		DIAG	Diagnostics activated, protection deactivated, in case of an error: - DIAG message - no error response of the input / output values
		NO_DIAG	Diagnostics and protection deactivated, in case of an error: - no message - no error response of the input / output values

11.4.5 Library ifmOutGroup

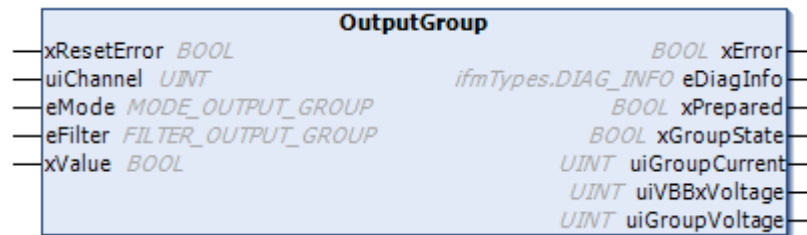
The library contains function blocks (POU) to control extended output functions.

11.4.5.1 OutputGroup

Function block type: Function block (FB)

Library: ifmOutGroup.library

Symbol in CODESYS:



Description

The function block controls the activation status of an output group and provides diagnostic information about the group and the connected outputs. Using the FB, an output group including the corresponding outputs can be switched on or off.

Input parameters

Parameter	Data type	Meaning	Possible values	
xResetError	BOOL	Reset request for an occurring error	FALSE => TRUE	Reset request to the lower level system
			otherwise	no action
uiChannel	UINT	Output channel group	→ Data sheet → Chapter: Note on wiring	
eMode	MODE_OUTPUT_GROUP	Operating type of the output channel group	MODE_OUTPUT_GROUP (ENUM)	
eFilter	FILTER_OUTPUT_GROUP	Defines the limit frequency of the output filter	FILTER_OUTPUT_GROUP (ENUM) (→ □ 200)	
xValue	BOOL	Activation requirement for the output group	FALSE	Deactivate output group
			TRUE	Activate output group



The error of an output group will only be reset if all corresponding outputs are error-free.

Output parameters



The state of the output `xOutState` may deviate from the required output state if e.g. an output group has been deactivated due to an error.

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication of whether an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - The action could not be executed - Observe the diagnostic information

Parameter	Data type	Meaning	Possible values	
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
xPrepared	BOOL	State of the FB outputs	FALSE	FB outputs invalid; FB is still being processed
			TRUE	FB outputs valid; FB has been processed
xGroupState	BOOL	Return value activation state of the selected output group The state may deviate from the required output state if e.g. an output group has been deactivated due to an error.	FALSE	Output group is deactivated
			TRUE	Output value is activated
uiGroupCurrent	UINT	Measured output current of the entire group in [mA]	Permissible = 0...final value of the measuring range	
uiVBBxVoltage	UINT	Measured voltage before the group switch in [mV]	Permissible = 0...final value of the measuring range	
uiGroupVoltage	UINT	Measured voltage after the group switch in [mV]	Permissible = 0...final value of the measuring range	

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_PREPARING	Status: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_INACTIVE	Status: FB/function is inactive.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• ERR_OVERVOLTAGE_VBBX	- For inputs: Error: Reference voltage exceeded. - For outputs. Error: Voltage of the corresponding output groups supply or at VBB30 / VBB15 not reached.
• ERR_OVERLOAD_CURRENT	Error: Maximum current exceeded.
• ERR_GROUP_SW_TEST	Error: Start-up test of the group switch has failed. Possible causes: - VBBx is not supplied. - At least one output of the group is supplied externally. - At least one output detects STUCK_AT_HIGH. - Defect at group switch or output
• ERR_AT_GROUP_OUTPUT	Error: At least one output of the output group is in an error state.
• ERR_STUCK_AT_HIGH	Error: Signal frozen, signal state high.
• ERR_STUCK_AT_LOW	Error: Signal frozen, signal state low.
• ERR_UNDERVOLTAGE_VBBX	- For inputs: Error: Reference voltage not reached. - For outputs. Error: The voltage of the corresponding output group supply or at VBB30 / VBB15 is not reached.
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.
• DIAG_NO_CALIB	The selected resource has no valid calibration. The displayed values are maybe faulty.
• DIAG_ACCESS	FB/function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.
• DIAG_OVERLOAD_CURRENT	Maximum current exceeded.
• DIAG_AT_GROUP_OUTPUT	At least one output of the output group is in the error state.

• DIAG_INVALID_FILTER	One or several filter settings are invalid.
• DIAG_STUCK_AT_HIGH	Error: Signal frozen, signal state high.
• DIAG_STUCK_AT_LOW	Error: Signal frozen, signal state low.

11.4.5.2 FILTER_OUTPUT_GROUP (ENUM)

Filter setting for voltage measurement in an output group.

The signal of the voltage measurement is damped via a first-order low-pass filter.

Name	Description	Possible values	
FILTER_OUTPUT_GROUP		UNCHANGED	No change of settings
		FILTER_0	1.7 ms
		FILTER_1	1.8 ms
		FILTER_2	2.4 ms
		FILTER_3	3.9 ms
		FILTER_4	7.4 ms
		FILTER_5	14.7 ms
		FILTER_6	29.3 ms
		FILTER_7	58.8 ms
		FILTER_8	117.7 ms
		FILTER_9	235.6 ms
		FILTER_10	471.4 ms
		FILTER_11	943.0 ms
		FILTER_12	1886.1 ms

More information: Filter ([→ Filter](#)  221)

11.4.5.3 MODE_OUTPUT_GROUP (ENUM)

Name	Description	Possible values	
MODE_OUTPUT_GROUP	Operating type of the output group	UNCHANGE	Setting remains unchanged
		OUT_DIGITAL	Digital output without diagnostics and without protection
		OUT_DIGITAL_VIRT	For output groups of type Output-Group-B with virtual group switch and without hardware group switch. The group outputs are activated/de-activated via the software. NO secondary switch-off option.
		MONITOR	No parameters or process data are written. Only the FB output data is updated. For use in a PLC application to which the resource does not belong.
		OFF	POU disabled.

11.4.6 Library ifmOutPWM

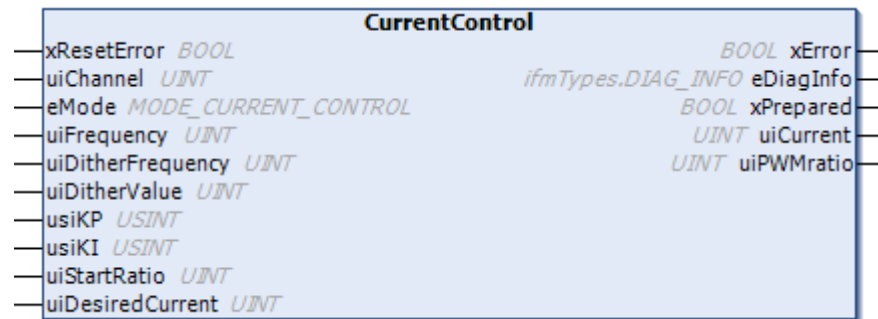
The library function blocks (POU) and enumeration types for pulse width modulation and current control of output channels.

11.4.6.1 CurrentControl

Function block type: Function block (FB)

Library: ifmIOOutPWM.library

Symbol in CODESYS:



Description

The FB is used to configure and operate a current controlled output. The current control is provided by means of pulse width modulation (PWM). The configuration of PWM frequency and dither is also done with this FB.

The controller operates in dependence of the period duration of the PWM signal. The two adjustment parameters `usiKI` and `usiKP` represent the integral and the proportional part of the controller.

- It is recommended to set `KI=50` and `KP=50` as start values so as to determine the best setting of the controller. Depending on the requested controller behaviour the values can gradually be incremented (controller is stronger / faster) or decremented (controller is weaker / slower).

- ▷ With target value `uiDesiredCurrent=0`, the output will be switched off immediately.

The controller has a fast compensation mechanism for voltage drops of the supply voltage. In addition to the controller behaviour of the controller and on the basis of the voltage drop, the ratio of the PWM is increased such that the controller reaches as quickly as possible the desired value.



- When defining the parameter `uiDitherValue`, make sure that the resulting PWM ratio in the operating range of the loop control remains between 0...1000 ‰:

- $uiPWMratio + uiDitherValue < 1000 \text{ ‰}$ and
- $uiPWMratio - uiDitherValue > 0 \text{ ‰}$.

With PWM frequencies below 100 Hz and additional dither, the current control can no longer reach the specified accuracy (→ Data sheet).

Input parameters

Parameter	Data type	Meaning	Possible values	
xResetError	BOOL	Reset request for an occurring error	FALSE => TRUE	Reset request to the lower level system
			otherwise	no action
uiChannel	UINT	Output channel	Group + Channel → Data sheet → Chapter:: Note on wiring	
			Examples:	
			103	Group 1 + channel 3
			301	Group 3 + channel 1
eMode	MODE_CURRENT_CONTROL	Operating type of the output channel	MODE_CURRENT_CONTROL (ENUM) (→ □ 207)	

Parameter	Data type	Meaning	Possible values
uiFrequency	UINT	PWM frequency of the output signal in [Hz]	→ Data sheet
uiDitherFrequency	UNIT	Frequency for the dither signal at the PWM output in [Hz]	permissible = $0 \dots \text{uiFrequency} / 2$ The value at uiDitherFrequency must be an integer part of the value indicated to uiFrequency. Examples: uiFrequency = 300 Hz uiDitherFrequency = 50 Hz => $300 / 50 = 6$ => even divisor, valid uiDitherFrequency = 100 Hz => $300 / 100 = 3$ => odd divisor, invalid
uiDitherValue	UNIT	Peak-to-peak value of the dither signal which overlays with the PWM signal, in [%]	permissible = $0 \dots 1000$ If the resulting PWM ratio value is outside the $0 \dots 1000$ % range, the dither value will be temporarily internally reduced to the minimum/maximum value that is possible, so that the mean value of the PWM ratio corresponds with the required value.
usiKP	USINT	Proportional component of the output signal	$0 \dots 255$ If usiKP = 0 then no control.
usiKI	USINT	Integral component of the output signal	$0 \dots 255$ If usiKI = 0 then no control.
uiStartRatio	UNIT	Mark-to-space ratio of the PWM output signal after start of the controller in [%]	permissible = $0 \dots 1000$ Recommended standard start value: 500
uiDesiredCurrent	UINT	Preset value at the output channel in [mA]. When 0 is set, the output is immediately deactivated.	$0 \dots 65535$

Output parameters

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
xPrepared	BOOL	State of the FB outputs	FALSE	FB outputs invalid; FB is still processed
			TRUE	FB outputs valid; FB has been processed
uiCurrent	UINT	Output current signal in [mA]	available = $0 \dots$ final value of the measuring range	
uiPWMRatio	UINT	PWM pulse ration calculated by the PI controller in [%]		

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_PREPARING	Status: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_INACTIVE	Status: FB/function is inactive.

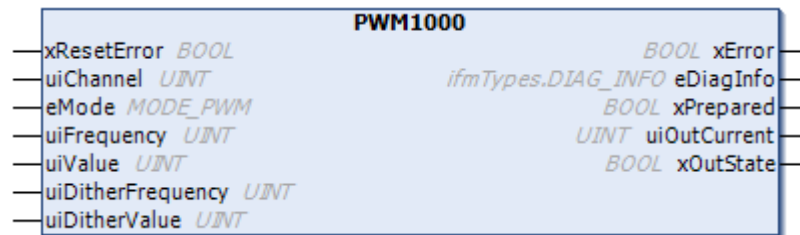
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• ERR_STUCK_AT_HIGH	Error: Signal frozen, signal state high.
• ERR_OVERLOAD_CURRENT	Error: Maximum current exceeded.
• ERR_LOW_CURRENT	Error: Minimum current not reached.
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.
• DIAG_NO_CALIB	The selected resource has no valid calibration. The displayed values are maybe faulty.
• DIAG_ACCESS	FB/function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.
• DIAG_STUCK_AT_HIGH	Error: Signal frozen, signal state high.
• DIAG_OVERLOAD_CURRENT	Maximum current exceeded.
• DIAG_LOW_CURRENT	Minimum current not reached.
• DIAG_UNDERVOLTAGE_VBBX	• For inputs: Error: Reference voltage not reached. • For outputs. Error: The voltage of the corresponding output group supply or at VBB30 / VBB15 is not reached.
• DIAG_INVALID_FREQUENCY	The frequency is not supported in the active mode.
• DIAG_INVALID_DITHER	At least one dither parameter is invalid.

11.4.6.2 PWM1000

Function block type: Function block (FB)

Library: ifmOutPWM.library

Symbol in CODESYS:



Description

The FB is used to configure and to operate an output with pulse width modulation.



If the function block `ConfigDiagLevel` is used:

► Please observe the formula to calculate the `DetectionTime`. [ConfigDiagLevel](#) (→ [190](#))

Input parameters

Parameter	Data type	Meaning	Possible values
xResetError	BOOL	Reset request for an occurring error	FALSE => TRUE Reset request to the lower level system
			otherwise no action
uiChannel	UINT	Input channel	Group + Channel → Data sheet → Chapter:: Note on wiring
			Examples:
			403 Group 4 + channel 3
			502 Group 5 + channel 2
eMode	MODE_PWM	Operating type of the output channel	MODE_PWM (ENUM) (→ 207)
uiFrequency	UINT	PWM frequency of the output signal in [Hz]	→ Data sheet
uiValue	UNIT	Pulse/pause ration of the PWM output signal in [%]	permissible = 0...1000
uiDitherFrequency	UNIT	Frequency for the dither signal at the PWM output in [Hz]	permissible = 0...uiFrequency / 2 The value at uiDitherFrequency must be an integer part of the value indicated to ui-Frequency. Examples: uiFrequency = 300 Hz uiDitherFrequency = 50 Hz => 300 / 50 = 6 => even divisor, valid uiDitherFrequency = 100 Hz => 300 / 100 = 3 => odd divisor, invalid
uiDitherValue	UNIT	Peak-to-peak value of the dither signal which overlays with the PWM signal, in [%]	permissible = 0...1000 If the resulting PWM ratio value is outside the 0...1000 % range, the dither value will be temporarily internally reduced to the minimum/maximum value that is possible, so that the mean value of the PWM ratio corresponds with the required value.

Output parameters



The state of the output `xOutState` may deviate from the required output state if e.g. an output group has been deactivated due to an error.

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication of whether an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	- An error has occurred - The action could not be executed - Observe the diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
xPrepared	BOOL	State of the FB outputs	FALSE	FB outputs invalid; FB is still being processed
			TRUE	FB outputs valid; FB has been processed
uiOutCurrent	UINT	Present output current in [mA]	Permissible = 0...final value of the measuring range	
xOutState	BOOL	Return value activation state of the selected output The state may deviate from the required output state if e.g. an output group has been deactivated due to an error.	FALSE	Output is deactivated
			TRUE	Output is activated

Diagnostic codes

Messages / diagnostic codes of the function blocks (→ [214](#)):

• STAT_PREPARING	Status: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• STAT_INACTIVE	Status: FB/function is inactive.
• ERR_INTERNAL	Error: Internal system error ► Contact the ifm Service Center!
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!
• ERR_INTERNAL_COMMUNICATION	Error: Internal communication error
• ERR_STUCK_AT_HIGH	Error: Signal frozen, signal state high.
• ERR_OVERLOAD_CURRENT	Error: Maximum current exceeded.
• ERR_LOW_CURRENT	Error: Minimum current not reached.
• DIAG_INVALID_VALUE	At least one input parameter is invalid or exceeds the permissible area.
• DIAG_NO_CALIB	The selected resource has no valid calibration. The displayed values are maybe faulty.
• DIAG_ACCESS	FB/function cannot access the required resource; The resource is not assigned to the accessing PLC in the IO mapping.
• DIAG_STUCK_AT_HIGH	Error: Signal frozen, signal state high.
• DIAG_OVERLOAD_CURRENT	Maximum current exceeded.
• DIAG_LOW_CURRENT	Minimum current not reached.
• DIAG_UNDERVOLTAGE_VBBX	- For inputs: Error: Reference voltage not reached. - For outputs: Error: The voltage of the corresponding output group supply or at VBB30 / VBB15 is not reached.
• DIAG_INVALID_FREQUENCY	The frequency is not supported in the active mode.
• DIAG_INVALID_DITHER	At least one dither parameter is invalid.

11.4.6.3 MODE_CURRENT_CONTROL (ENUM)

Name	Description	Possible values	
MODE_CURRENT_CONTROL	Operating mode of the output	UNCHANGED	Setting is maintained
		OUT_CURRENT_CSO	Output for current control without diagnostics and without protection; CSO
		MONITOR	No parameters or process data are written. Only the FB output data is updated. For use in a PLC application to which the resource does not belong.
		OFF	Block disabled.

11.4.6.4 MODE_PWM (ENUM)

Name	Description	Possible values	
MODE_PWM	Operating mode of the output	UNCHANGED	Setting is maintained
		OUT_PWM_CSI	PWM output without diagnostics; CSI
		OUT_PWM_CSO	PWM output without diagnostics; CSO
		MONITOR	No parameters or process data are written. Only the FB output data is updated. For use in a PLC application to which the resource does not belong.
		OFF	POU disabled.

11.5 Help function libraries

11.5.1 Library ifmSysInfo.library

The library contains function block (FB) and data structures (STRUCT, ENUM) for the provision of the device information.

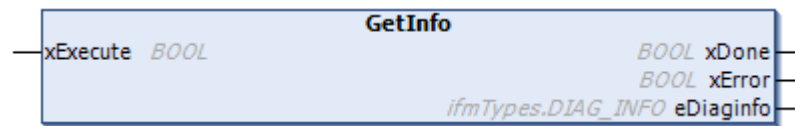
11.5.1.1 GetInfo

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifmSysInfo.library

Symbol in CODESYS:



Description

The FB reads the system information of the device and the IEC application and writes the read values to the global variable [aSysInfoList \(GVL\)](#) (→ [□ 211](#)), e.g:

- Hardware revision
- Serial number of the device
- Date of production
- Description text of the IEC application (maximum 80 characters)

Only the system information provided by the device is read out.

Input parameters

Parameter	Data type	Description	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE => TRUE	FB is executed once
			Other	No impact on FB processing

Output parameters

Parameter	Data type	Description	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	• FB successfully executed • FB can be called again
xError	BOOL	Indication if an error has occurred during the FB execution	FALSE	No error has occurred or the FB is still being executed
			TRUE	• An error has occurred • Action could not be executed • Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	

Diagnostic codes

• STAT_INACTIVE	Status: FB/function is inactive.
• STAT_BUSY	Status: FB/function is currently executed.
• STAT_DONE	Status: FB/function has been successfully executed and completed. There are valid results on the outputs.
• ERR_EXCEEDED_RANGE	The description text of the application is too long.

• ERR_NOT_SUPPORTED	Error: invalid function call; Function is not supported.
• ERR_INTERNAL	Error: internal system error. ► Contact the ifm Service Center!
• ERR_INVALID_VALUE	Error: At least one information type to be read is not supported by the device
• ERR_UNDEFINED	Error: Unknown error ► Contact the ifm Service Center!

11.5.1.2 aSysInfoList (GVL)

Name	Description	Data type	Initial value
OS_VERSION		SYS_INFO (STRUCT) (→ □ 212)	0
BOOTLOADER_VERSION			1
HW_VERSION_SW_REL	Software-related hardware revision		2
HW_VERSION	Hardware version, e.g. "AD"		3
SIS_SYS_VERSION			4
SIS_DEV1_VERSION			5
SIS_DEV2_VERSION			6
DEVICE_SERIAL_NUMBER	Serial number		7
DEVICE_ORDER_NUMBER	Order number, e.g. "CR721S"		8
WORK_ORDER_NUM_DEVICE			9
PCB_ERP_NUMBER			10
MANUFACTURER_DATE	Date of production		11
DEVICE_NAME	Device name, e.g. "eco-matController/124-1"		12
IEC_APP_SAFE	Combined description and version of the PLC_1 application (source: Error in the IEC application) Maximum 80 characters.		13
IEC_APP_NONSAFE	Combined description and version of the PLC_2 application (source: Properties of the IEC application). Maximum 80 characters.		14
MAC_ADDRESS_0			15
MAC_ADDRESS_1			16
MAC_ADDRESS_2			17
MAC_ADDRESS_3			18
MAC_ADDRESS_4			19
MAC_ADDRESS_5			20
MAC_ADDRESS_6			21
MAC_ADDRESS_7			22
IEC_APP_DESCRIPTION_SAFE	Description of the PLC_1 application (Source: Properties of the IEC application).		23
IEC_APP_DESCRIPTION_NONSAFE	Description of the PLC_2 application (Source: Properties of the IEC application).		24
IEC_APP_VERSION_SAFE	Version of the PLC_1 application from the properties of the IEC application		25
IEC_APP_VERSION_NONSAFE	Version of the PLC_2 application from the properties of the IEC application		26

11.5.1.3 SYS_INFO (STRUCT)

Designation	Data type	Meaning	Possible values
sName	STRING (32)	Name of the system component	E.g. firmware version
sValue	STRING (255)	Value of the system component	e.g. 3.1

12 Troubleshooting

12.1	Error classes	213
12.2	Error messages	213
12.3	Messages / diagnostic codes of the function blocks	214
12.4	Removing a faulty IEC application	214
12.5	Finding errors in the IEC application	214
12.6	Exporting the log file of the IEC application	215

12.1 Error classes

An error is classified according to its possible impact. The error class determines how the system reacts when a specific error occurs.

Error class		Description	Reaction
A	Fatal Error	The overall integrity of the controller is no longer guaranteed. Errors in the central components of the controller that affect the behaviour of other components.	• The controller deactivates components allocated to both PLCs. • The controller switches off both PLCs. • The controller saves the information in the error log.
B	Serious Error	One or several PLCs can no longer be executed.	• The controller deactivates components allocated to the PLC. • The controller deactivates the concerned PLC. • The controller saves the information in the error log.
C	Component Error	Error in a controller component; The function of one or several components of the controller is no longer guaranteed.	• The controller puts the affected function into a defined state. • The controller reports the error to the application. • The controller saves the information in the error log.
D	Periphery Error	Errors on or in the periphery; a function can no longer be executed.	• The controller puts the affected function into a defined state. • The controller reports the error to the application. • The controller saves the information in the error log. • The error can be reset in the application (as often as required).

12.2 Error messages

(Most) FBs provide, among others, the following signals at their outputs.

► Evaluate these signals in the application!

Parameter	Data type	Meaning	Possible values	
xError	BOOL	Indication if an error occurred during the FB execution	FALSE	No error occurred or the FB is still being executed
			TRUE	• An error has occurred • Action could not be executed • Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	



Lists of diagnostic codes are part of the function block descriptions ([→ ifm function libraries](#) [□ 120](#)).

12.3 Messages / diagnostic codes of the function blocks

Status/diagnostic/error messages of the function blocs are defined in the global Enum DIAG_INFO.

They have one of the following prefixes depending on the type of message:

Prefix	Type of message	Description
STAT	Status message	Status messages contain information about the condition of the function block during the normal procedure.
DIAG	Diagnostic message	Diagnostic messages contain information about a failure event. They reset themselves after the failure event has disappeared and can optionally be evaluated by the application.
ERR	Error message	Error messages contain information about a failure event. They must be reset in the application after the failure event has disappeared.

Examples for messages / diagnostic codes:

- STAT_INACTIVE
- DIAG_OPEN_CIRCUIT
- ERR_OVERVOLTAGE



Lists of diagnostic codes are part of the function block descriptions ([→ ifm function libraries](#) [□ 120](#)).

12.4 Removing a faulty IEC application

If a **SERIOUS_ERROR** (error class B) occurs during the development of an internal PLC due to a faulty IEC application, the IEC application can be deleted as follows:

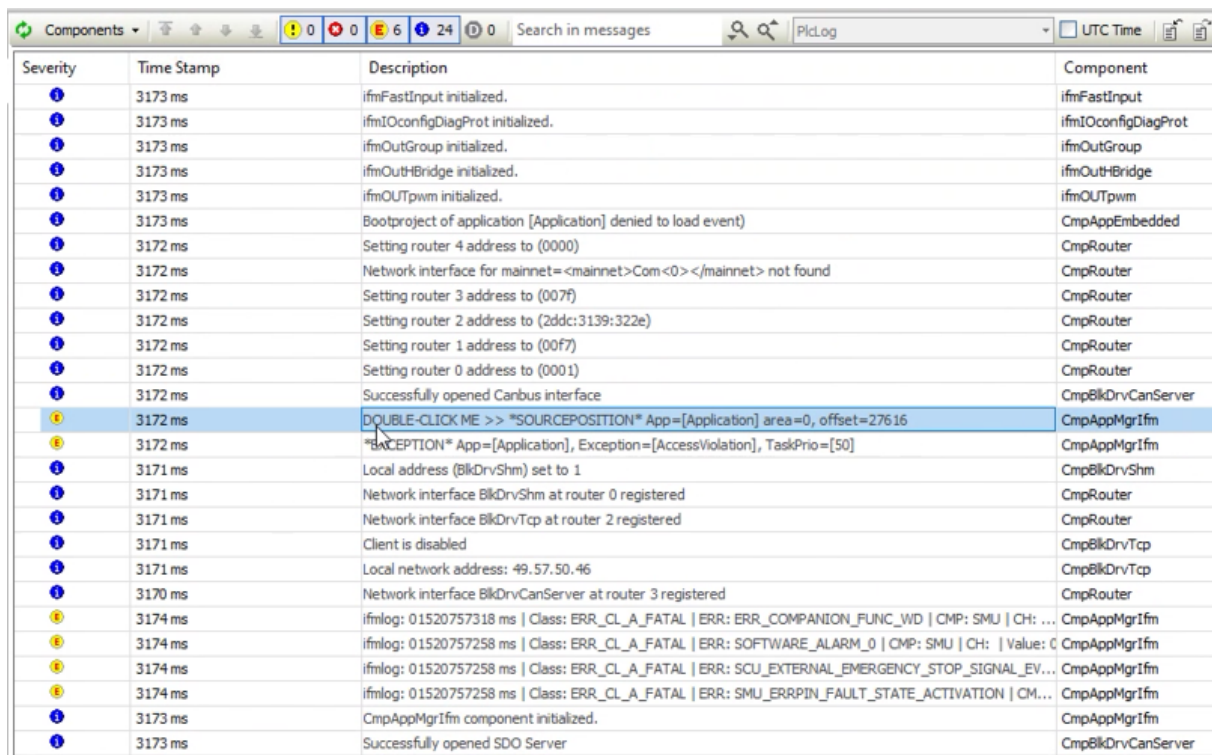
- ▶ Disconnect the appliance from the regular operating voltage.
- ▶ Switch on and operate the device with a voltage between 7 V and 8 V.
 - ▷ The device starts. The SYS0 and SYS1 LEDs are static green.
 - ▷ The device switches to the **SYSTEM_STOP** state after 10 seconds. The SYS0 and SYS2 LEDs are static red.
- ▶ Delete the IEC application of the relevant device-internal PLC, e.g. delete the relevant APP file with CODESYS [Generic Device Editor > Files] in the `apps` directory on the device.
- ▶ Perform a power-on reset and start the device with regular operating voltage → (Technical data). [Voltage ranges](#) ([→ □ 23](#))
 - ▷ The device is ready for operation.
 - ▷ The **SERIOUS_ERROR** has been eliminated.
 - ▷ The faulty IEC application can be corrected and reloaded onto the device.

12.5 Finding errors in the IEC application

To be able to find errors in the IEC application during development:

- ✓ The IEC application opened in CODESYS is identical to the IEC application on the device.
- ✓ CODESYS is connected to the device and the relevant device-internal PLC.

- Call up the PLC Log of the relevant device-internal PLC at: Double-click on the PLC >[Log].
- Click on the  icon.
- ▷ The current content of PLC Log appears.



Severity	Time Stamp	Description	Component
Info	3173 ms	ifmFastInput initialized.	ifmFastInput
Info	3173 ms	ifmIOconfigDiagProt initialized.	ifmIOconfigDiagProt
Info	3173 ms	ifmOutGroup initialized.	ifmOutGroup
Info	3173 ms	ifmOutHBridge initialized.	ifmOutHBridge
Info	3173 ms	ifmOUTpwm initialized.	ifmOUTpwm
Info	3173 ms	Bootproject of application [Application] denied to load event)	CmpAppEmbedded
Info	3172 ms	Setting router 4 address to (0000)	CmpRouter
Info	3172 ms	Network interface for mainnet=<mainnet>Com<0></mainnet> not found	CmpRouter
Info	3172 ms	Setting router 3 address to (007f)	CmpRouter
Info	3172 ms	Setting router 2 address to (2ddc:3139:322e)	CmpRouter
Info	3172 ms	Setting router 1 address to (00f7)	CmpRouter
Info	3172 ms	Setting router 0 address to (0001)	CmpRouter
Info	3172 ms	Successfully opened Canbus interface	CmpBlkDrvCanServer
Warning	3172 ms	DOUBLE-CLICK ME >> *SOURCEPOSITION* App=[Application] area=0, offset=27616	CmpAppMgrIfm
Warning	3172 ms	*EXCEPTION* App=[Application], Exception=[AccessViolation], TaskPrio=[50]	CmpAppMgrIfm
Info	3171 ms	Local address (BlkDrvShm) set to 1	CmpBlkDrvShm
Info	3171 ms	Network interface BlkDrvShm at router 0 registered	CmpRouter
Info	3171 ms	Network interface BlkDrvTcp at router 2 registered	CmpRouter
Info	3171 ms	Client is disabled	CmpBlkDrvTcp
Info	3171 ms	Local network address: 49.57.50.46	CmpBlkDrvTcp
Info	3170 ms	Network interface BlkDrvCanServer at router 3 registered	CmpRouter
Warning	3174 ms	ifmlog: 01520757318 ms Class: ERR_CL_A_FATAL ERR: ERR_COMPANION_FUNC_WD CMP: SMU CH: ...	CmpAppMgrIfm
Warning	3174 ms	ifmlog: 01520757258 ms Class: ERR_CL_A_FATAL ERR: SOFTWARE_ALARM_0 CMP: SMU CH: Value: 0	CmpAppMgrIfm
Warning	3174 ms	ifmlog: 01520757258 ms Class: ERR_CL_A_FATAL ERR: SCU_EXTERNAL_EMERGENCY_STOP_SIGNAL_EV...	CmpAppMgrIfm
Warning	3174 ms	ifmlog: 01520757258 ms Class: ERR_CL_A_FATAL ERR: SMU_ERRPIN_FAULT_STATE_ACTIVATION CM...	CmpAppMgrIfm
Info	3173 ms	CmpAppMgrIfm component initialized.	CmpAppMgrIfm
Info	3173 ms	Successfully opened SDO Server	CmpBlkDrvCanServer

Fig. 17: CODESYS PLC Log

- Double-click on one SOURCEPOSITION entry in the list.
- Click on the menu item [Show source code in editor].
- ▷ CODESYS jumps to the error location in the program code.

12.6 Exporting the log file of the IEC application

For service purposes, the content of PLC Log can be exported and sent together with the project. The fault can then be analysed without a connection to the controller:

Export

- ✓ CODESYS is connected to the device and the relevant device-internal PLC.
- Call up the PLC Log of the relevant device-internal PLC at: Double-click on the PLC >[Log].
- Click on the  icon.
- ▷ The current content of PLC Log appears.



- Click on the  icon to export the content of PLC Log as an XML file.
- Pass on the CODESYS project and the XML log file for error analysis.

Import for error analysis

- Open the project from which the log file was exported in CODESYS and generate the code for the PLC concerned. No connected target device is required, just a correct device description and the correct project.

- ▶ Call up the PLC Log of the relevant device-internal PLC at: Double-click on the PLC >[Log].
- ▶ Click on the  icon to import the content of the XML file.
 - ▷ The content of PLC Log is loaded from the XML file and displayed.
- ▶ Double-click on one SOURCEPOSITION entry in the list.
- ▶ Click on the menu item [Show source code in editor].
- ▷ CODESYS jumps to the error location in the program code.

13 Appendix

13.1	Wiring	218
13.2	Filter	221
13.2.1	Filter times of the inputs	221
13.2.2	Filter times of the outputs	222
13.3	Using the operators	223
13.4	Behaviour in case of floating point operations	226
13.4.1	Behaviour in case of specific arguments	226
13.4.2	Behaviour in case of conversions	226
13.5	Behaviour in case of integer operations	227
13.6	Supported SysSocket functions	228
13.7	Directory structure and file overview	230
13.8	Overview of user rights	231
13.8.1	ifm rights	232
13.8.2	CODESYS rights	232
13.9	Task configuration example	234
13.9.1	Initial situation	234
13.9.2	Setting the task configuration	235
13.9.3	Assigning the CAN manager to the bus task	236
13.9.4	Mapping of the CAN variables	237
13.9.5	Calling CAN POU's	239
13.9.6	Task monitoring	239
13.10	ifm behaviour models for function blocks	240
13.10.1	General	240
13.10.2	Behaviour model ENABLE	240
13.10.3	Behaviour model EXECUTE	240

13.1 Wiring

Further information on the inputs and outputs and the electrical connection of the device: → Operating instructions and → Technical data

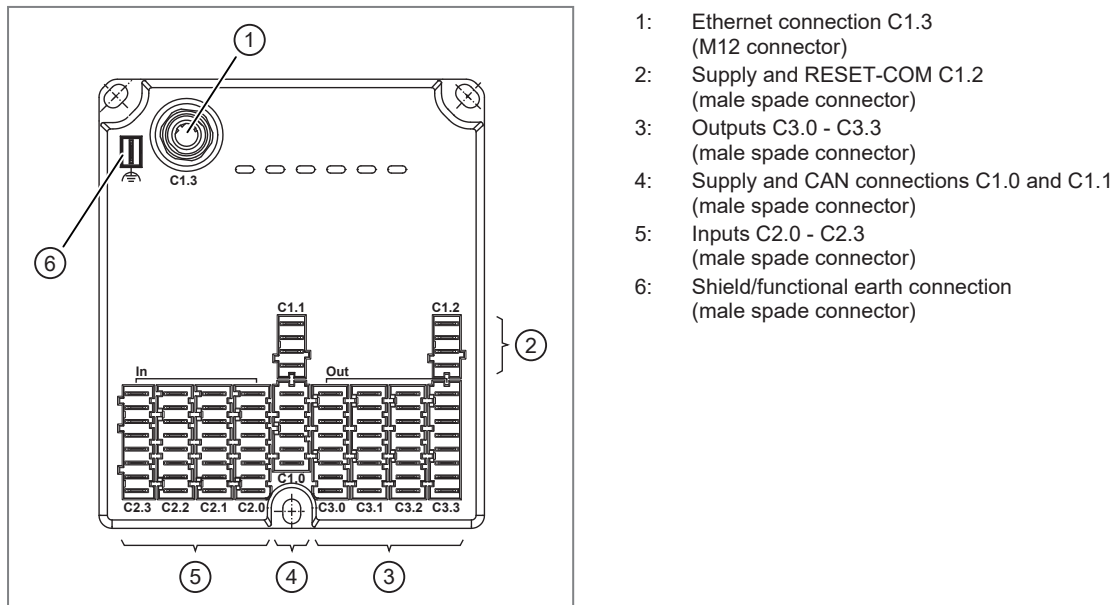


Fig. 18: Connectors

Assignment of the terminal blocks:

All VBB₃₀ connections of the terminal blocks are connected to each other in the device.

All GND connections are connected to each other in the device.

Terminal block	Connection	Function
C1.0	VBB ₃₀	Controller supply
	VBB ₀	Output group 0 supply
	VBB ₁	Output group 1 supply
	GND ₀	Controller GND
	CAN0_H	CAN interface 0 (high)
	CAN0_L	CAN interface 0 (low)
C1.1	VBB ₃₀	Controller supply
	GND ₁	Controller GND
	CAN1_H	CAN interface 1 (high)
	CAN1_L	CAN interface 1 (low)
C1.2	VBB ₂	Output group 2 supply
	GND ₂	Controller GND
	NC	not used
	RESET-COM	Service input of the controller

Tab. 3: Terminal blocks C1.0, C1.1, C1.2

Terminal block	Connection	Function
C2.0	VBB ₃₀	Sensor supply (fed from VBB ₃₀ of C1.0 / C1.1)
	IN0400	Frequency input (IN FREQUENCY-B)
	IN0401	Frequency input (IN FREQUENCY-B)
	GND	Sensor GND
	GND	Sensor GND
	IN0500	Frequency input (IN FREQUENCY-B)

Terminal block	Connection	Function
C2.0	IN0501	Frequency input (IN FREQUENCY-B)
	VBB ₃₀	Sensor supply (fed from VBB ₃₀ of C1.0 / C1.1)
C2.1	VBB ₃₀	Sensor supply (fed from VBB ₃₀ of C1.0 / C1.1)
	IN0200	Resistor input (IN RESISTOR-A)
	IN0201	Resistor input (IN RESISTOR-A)
	GND	Sensor GND
	GND	Sensor GND
	IN0300	Resistor input (IN RESISTOR-A)
	IN0301	Resistor input (IN RESISTOR-A)
	VBB ₃₀	Sensor supply (fed from VBB ₃₀ of C1.0 / C1.1)
C2.2	VBB ₃₀	Sensor supply (fed from VBB ₃₀ of C1.0 / C1.1)
	IN0000	Multi-function input (IN MULTIFUNCTION-A)
	IN0001	Multi-function input (IN MULTIFUNCTION-A)
	GND	Sensor GND
	GND	Sensor GND
	IN0100	Multi-function input (IN MULTIFUNCTION-A)
	IN0101	Multi-function input (IN MULTIFUNCTION-A)
	VBB ₃₀	Sensor supply (fed from VBB ₃₀ of C1.0 / C1.1)
C2.3	VBB ₃₀	Sensor supply (fed from VBB ₃₀ of C1.0 / C1.1)
	IN0600	Multi-function input (IN MULTIFUNCTION-A)
	IN0601	Multi-function input (IN MULTIFUNCTION-A)
	GND	Sensor GND
	GND	Sensor GND
	IN0700	Multi-function input (IN MULTIFUNCTION-A)
	IN0701	Multi-function input (IN MULTIFUNCTION-A)
	VBB ₃₀	Sensor supply (fed from VBB ₃₀ of C1.0 / C1.1)

Tab. 4: Terminal blocks C2.0, C2.1, C2.2, C2.3

Terminal block	Connection	Function
C3.0	OUT0000	Digital / PWM / PWM-I output 2.5 A (OUT PWM-25-A)
	GND	Actuator GND
	OUT0001	Digital / PWM / PWM-I output 2.5 A (OUT PWM-25-A)
	GND	Actuator GND
	OUT0002	Digital / PWM output 2.5 A (OUT PWM-25-B)
	GND	Actuator GND
	OUT0003	Digital / PWM output 2.5 A (OUT PWM-25-B)
	GND	Actuator GND
C3.1	OUT0004	Digital / PWM output 1 A (OUT PWM-10-D)
	GND	Actuator GND
	OUT0005	Digital / PWM output 1 A (OUT PWM-10-D)
	GND	Actuator GND
	OUT0006	Digital / PWM output 1 A (OUT PWM-10-D)
	GND	Actuator GND
	OUT0007	Digital / PWM output 1 A (OUT PWM-10-D)
	GND	Actuator GND

Terminal block	Connection	Function
C3.2	OUT0100	Digital / PWM output 2.5 A (OUT PWM-25-B)
	GND	Actuator GND
	OUT0101	Digital / PWM output 2.5 A (OUT PWM-25-B)
	GND	Actuator GND
	OUT0102	Digital / PWM output 4.0 A (OUT PWM-40-B)
	GND	Actuator GND
	OUT0103	Digital / PWM output 4.0 A (OUT PWM-40-B)
	GND	Actuator GND
C3.3	OUT0200	Digital / PWM / PWM-I output 2.5 A (OUT PWM-25-A)
	GND	Actuator GND
	OUT0201	Digital / PWM / PWM-I output 2.5 A (OUT PWM-25-A)
	GND	Actuator GND
	OUT0202	Digital / PWM output 2.5 A (OUT PWM-25-B)
	GND	Actuator GND
	OUT0203	Digital / PWM output 2.5 A (OUT PWM-25-B)
	GND	Actuator GND

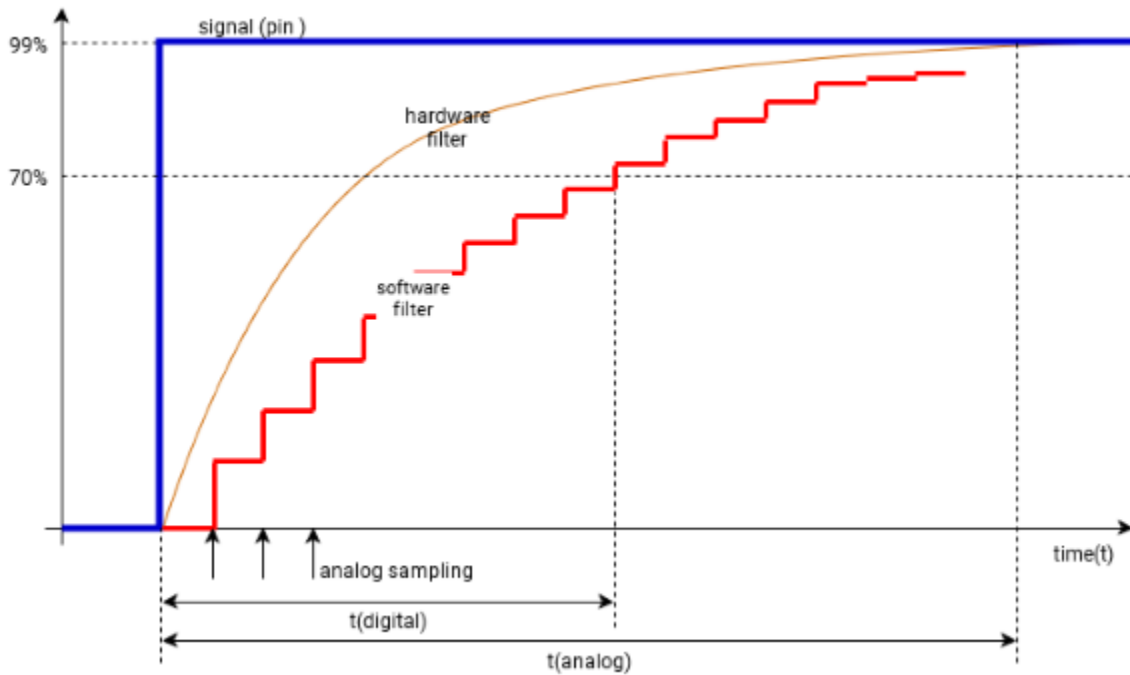
Tab. 5: Terminal blocks C3.0, C3.1, C3.2, C3.3

13.2 Filter

Signals on analogue and digital inputs as well as the current measurements of the outputs are filtered by the controller. The filtering is done with the hardware filter (not configurable) and the first-order software filter (configurable in the IEC application).

The switching thresholds for digital signals are 30%/25% and 70%/65% of V_{BB30} as standard. The filter time $t(\text{digital})$ for digital inputs is therefore shorter than the filter time $t(\text{analog})$ for analogue inputs.

The filters for analogue inputs and current measurements on the outputs are calculated with 99% of the signal change.



13.2.1 Filter times of the inputs

Filters of the inputs				
Filter no.	$t_{\text{max}}(\text{digital})$	$t_{\text{nom}}(\text{digital})$	$t_{\text{max}}(\text{analog})$	$t_{\text{nom}}(\text{analog})$
0	0.7ms	0.6 ms	1.8ms	1.7ms
1	1.1ms	0.9ms	3.9ms	3.3ms
2	2.5ms	2.0ms	8.7ms	7.0ms
3	5.0ms	4.0ms	17.6ms	14.1ms
4	9.5ms	7.6ms	36.0ms	28.9ms
5	19.0ms	15.2ms	73.0ms	58.4ms
6	38.5ms	30.8ms	146.5ms	117.2ms
7	77.0ms	61.6ms	294.0ms	235.2ms
8	154.0ms	123.2ms	588.5ms	470.8ms
9	308.0ms	246.4ms	1178.0ms	942.4ms
10	616.5ms	493.2ms	2357.0ms	1885.6ms
11	1233.0ms	986.4ms	4715.0ms	3772.0ms
12	2465.5ms	1972.4ms	9430.5ms	7544.4ms

13.2.2 Filter times of the outputs

Filter for current measurement of the output in digital operation		
Filter no.	$t_{\max}(\text{analog})$	$t_{\text{nom}}(\text{analog})$
0	1.7ms	1.7ms
1	2.2ms	1.8ms
2	3.8ms	2.4ms
3	7.2ms	3.9ms
4	14.5ms	7.4ms
5	29.2ms	14.7ms
6	58.6ms	29.3ms
7	117.6ms	58.8ms
8	235.4ms	117.7ms
9	471.2ms	235.6ms
10	942.8ms	471.4ms
11	1886.0ms	943.0ms
12	3772.2ms	1886.1ms

13.3 Using the operators



Observe the CODESYS online help:

→ Online help > CODESYS Development System > Reference, Programming > Operators

ATTENTION

Unpredictable result when using operators (AND, OR, XOR, ...) with input values of different data types, for example BOOL and BYTE, etc.

- ▷ Risk of material damage.
- ▷ Failure of the function.
- ▶ Only use operators with input values of the same data types.

The following table shows in which PLC and in which PRG CODESYS standard operators may be used:

Operator group	Operator	PLC_2 Standard PRG	PLC_1 Standard PRG	Restriction
Arithmetic	ADD	X	X	
	SUB	X	X	
	MUL	X	X	
	DIV	X	X	
	MOD	X	X	
	MOVE	X	X	
	INDEXOF	X	X	
	SIZEOF	X	X	
Bit string	AND	X	X	
	OR	X	X	
	XOR	X	X	
	NOT	X	X	
	AND_THEN	X	X	
	OR_ELSE	X	X	
Bitshift	SHL	X	X	
	SHR	X	X	
	ROL	X	X	
	ROR	X	X	
Selection	SEL	X	X	
	MAX	X	X	
	MIN	X	X	
	LIMIT	X	X	
	MUX	X	X	
Comparison	GT	X	X	
	LT	X	X	
	LE	X	X	
	GE	X	X	
	EQ	X	X	
	NE	X	X	
Address	ADR	X	X	
	Content Operator	X	X	

Operator group	Operator	PLC_2 Standard PRG	PLC_1 Standard PRG	Restriction
Address	BITADDR	X	X	
Call	CAL	---	---	
Type conversion	BOOL_TO	X	X	
	TO_BOOL	X	X	
	<INT_Type>_TO_ <INT_Type>	X	X	
	REAL_TO- / LRE- AL_TO	X	X	
	TIME_TO / TIME_OF_DAY_TO	X	X	
	DATE_TO / DT_TO	X	X	
	STRING_TO	X	X	
	TRUNC	X	X	
	TRUNC_INT	X	X	
	TO_<xxx>	X	X	
Numeric	ABS	X	X	For minimum of signed integer data types, the return value is undefined (e.g. ABS(-128) for data type SINT.).
	SQRT	X	X	Definition range ≥ 0 , for definition range < 0 result = NaN
	LN	X	X	Definition range > 0 , for definition range ≤ 0 result = NaN
	LOG	X	X	Definition range > 0 , for definition range ≤ 0 result = NaN
	EXP	X	X	
	EXPT	X	X	
	SIN	X	X	For accuracy up to the 4th decimal place: Definition range (REAL) = -6.2678940e+01...1.3169797e+02
	ASIN	X	X	Definition range = -1.0...1.0
	COS	X	X	For accuracy up to the 4th decimal place: Definition range (REAL) = -1.1476853e+02...1.1476853e+02
	TAN	X	X	For accuracy up to the 4th decimal place: REAL definition range = -4.5447968e+01...4.7023037e+01 Excepted are singularities around $\pi/2 + k \cdot \pi \pm 0.1$
	ACOS	X	X	Definition range = -1.0...1.0
	ATAN	X	X	
Namespace	Namespace for Global Variable Lists	X	X	
	Enumeration Namespace	X	X	
	Library Namespace	X	X	
Other	__DELETE	---	---	

Operator group	Operator	PLC_2 Standard PRG	PLC_1 Standard PRG	Restriction
Other	__ISVALIDREF	---	---	
	__NEW	---	---	
	__QUERYINTERFACE	---	---	
	__QUERYPOINTER	---	---	
	__TRY, __CATCH, __FINALLY, __ENDTRY	---	---	
	__VARINFO	---	---	
	INI	---	---	

Legend:

X = may be used

--- = may not be used

13.4 Behaviour in case of floating point operations

REAL operations are calculated in the CPU with the Floating Point Unit (FPU). This leads to a fast execution of these arithmetic operations.

LREAL operations are calculated via software library. This leads to a longer processing time for the arithmetic operations.

The FPU supports IEEE-754 REAL values, except for denormalised REAL values.

► If denormalised values cannot be excluded in calculations: Use LREAL values.

13.4.1 Behaviour in case of specific arguments

When calculating operations and functions with specific REAL/LREAL arguments, the controller shows the following behaviour:

- REAL-/LREAL-division by +/- 0.0 leads to a `SERIOUS_ERROR` and to the state `RUNTIME_STOP`.
- REAL/LREAL operations with the argument NaN (not a number) have the result NaN or lead to a `SERIOUS_ERROR` and the state `RUNTIME_STOP`. Exemptions: `exp(1.0,NaN) = 1.0`; `exp(NaN,0.0) = 1.0`
- REAL/LREAL operations with the argument +/- Inf (Infinity) lead to undefined results or to a `SERIOUS_ERROR` and to the state `RUNTIME_STOP`.
- REAL operations with the arguments +/- denormal lead to undefined results.
- REAL operations `<=` and `>=` with at least one argument NaN lead to the result `TRUE` (IEEE Intel Issue) .

ATTENTION

Unpredictable control behaviour when calling up mathematical operations and functions with specific REAL/LREAL arguments.

▷ Risk of material damage.

► Only use REAL/LREAL arguments within the normal definition range for mathematical operations/functions.

► Avoid special REAL/LREAL arguments in calculations.

► Check REAL/LREAL arguments before calculations.

13.4.2 Behaviour in case of conversions

The following behaviour occurs with the conversion functions `REAL_TO_[TYPE]` and `LREAL_TO_[TYPE]`:

Function	Argument	Result
<code>REAL_TO_BYTE</code> , <code>REAL_TO_USINT</code> , <code>REAL_TO_WORD</code> , <code>REAL_TO_UINT</code> , <code>LREAL_TO_BYTE</code> , <code>LREAL_TO_USINT</code> , <code>LREAL_TO_WORD</code> , <code>LREAL_TO_UINT</code>	<code>< 0</code>	<code>SERIOUS_ERROR</code>
<code>REAL_TO_BYTE</code> , <code>REAL_TO_USINT</code> , <code>REAL_TO_WORD</code> , <code>REAL_TO_UINT</code> , <code>LREAL_TO_BYTE</code> , <code>LREAL_TO_USINT</code> , <code>LREAL_TO_WORD</code> , <code>LREAL_TO_UINT</code>	<code>> upper value range limit [TYPE]</code>	Undefined value or <code>SERIOUS_ERROR</code>

Function	Argument	Result
REAL_TO_SINT, REAL_TO_INT, LREAL_TO_SINT, LREAL_TO_INT	< lower value range limit [TYPE] > upper value range limit [TYPE]	Undefined value or SERIOUS_ERROR
REAL_TO_DWORD, REAL_TO_DINT, REAL_TO_UDINT, REAL_TO_LWORD, REAL_TO_ULINT, REAL_TO_LINT, LREAL_TO_DWORD, LREAL_TO_DINT, LREAL_TO_UDINT, LREAL_TO_LWORD, LREAL_TO_ULINT, LREAL_TO_LINT	< lower value range limit [TYPE] > upper value range limit [TYPE]	SERIOUS_ERROR

ATTENTION

The result of the conversion functions REAL_TO_[TYPE] and LREAL_TO_[TYPE] does not fit into the value range of the target data type (e.g. BYTE, USINT, ...).

- ▷ The corresponding PLC continues to run with incorrect values.
- ▷ The corresponding PLC goes into the error state.
- ▶ Ensure that no invalid conversions are carried out.
- ▶ Before conversion, check for valid arguments within the value range of the target data type, see tables.

13.5 Behaviour in case of integer operations

The controller shows the behaviour described in the table and leads to the described results if invalid integer arguments are used for calculation:

Operator	Arguments	Result
ABS	Minimum signed variable e.g. ABS(-128) (minimum of data type SINT)	Minimum signed variable -128
MOD	0	0

13.6 Supported SysSocket functions

The following table shows in which PLC and in which PRG SysSocket functions may be used:

SysSocket function	PLC 1 Standard PRG	PLC 2 Standard PRG	
# SysSockAccept	X	X	
# SysSockBind	X	X	
# SysSockClose	X	X	
# SysSockCloseUdp	X	X	
# SysSockConnect	X	X	
# SysSockCreate	partly	partly	Family: SOCKET_AF_INET ; Protocol: SOCKET_IPPROTO_IP, SOCKET_IPPROTO_TCP, SOCKET_IPPROTO_UDP ; Type: SOCKET_STREAM, SOCKET_DGRAM
# SysSockCreateUdp	partly	partly	Family: SOCKET_AF_INET ; Protocol: SOCKET_IPPROTO_IP, SOCKET_IPPROTO_TCP, SOCKET_IPPROTO_UDP ; Type: SOCKET_STREAM, SOCKET_DGRAM
# SysSockFdInit	X	X	
# SysSockFdIsset	X	X	
# SysSockFdZero	X	X	
# SysSockGetFirstAdapterInfo	X	X	
# SysSockGetHostByName	X	X	
# SysSockGetHostName	X	X	
# SysSockGetNextAdapterInfo	X	X	
# SysSockGetOption	partly	partly	SOCKET_SO_KEEPALIVE, SOCKET_TCP_NODELAY, SOCKET_SO_REUSEADDR, SOCKET_SO_BROADCAST
# SysSockGetOSHandle	X	X	
# SysSockGetPeerName	---	---	
# SysSockGetRecvSizeUdp	---	---	
# SysSockGetSockName	---	---	
# SysSockGetSubnetMask	X	X	
# SysSockHtonl	X	X	
# SysSockHtons	X	X	
# SysSockInetAddr	X	X	
# SysSockInetNtoa	X	X	
# SysSockIoctl	partly	partly	working commands: IFM_SOCKET_FIONREAD, IFM_SOCKET_FIONBIO
# SysSockLists	X	X	
# SysSockNtohl	X	X	
# SysSockNtohs	X	X	
# SysSockPing	X	X	
# SysSockRecv	partly	partly	Flags are unused
# SysSockRecvFrom	X	X	Flag: SOCKET_MSG_NONE not allowed, but value is still unregarded
# SysSockREcvFromUdp	X	X	Flag: SOCKET_MSG_NONE not allowed, but value is still unregarded

SysSocket function	PLC 1 Standard PRG	PLC 2 Standard PRG	
# SysSockREcvFromUdp2	X	X	Flag: SOCKET_MSG_NONE not allowed, but value is still unregarded
# SysSockSelect	X	X	
# SysSockSend	partly	partly	Flags are unused
# SysSockSendTo	X	X	Flag: SOCKET_MSG_NONE not allowed, but value is still unregarded
# SysSockSendToUdp	X	X	Flag: SOCKET_MSG_NONE not allowed, but value is still unregarded
# SysSockSetIPAddress	X	X	will break the connection with code-sys
# SysSockSetIpAddressAndNetMask	X	X	will break the connection with code-sys
# SysSockSetOption	partly	partly	SOCKET_SO_KEEPALIVE, SOCKET_TCP_NODELAY, SOCKET_SO_REUSEADDR, SOCKET_SO_BROADCAST
# SysSockSetSubnetMask	X	X	may break the connection with code-sys
# SysSockShutdown	---	---	

Legend:

X = may be used

--- = may not be used

13.7 Directory structure and file overview

The following directories and files are stored in the device:

Data name / path	Description
apps - standard.app - safe.app	Folder PLC_2 application PLC_1 application
os ifmOS.ifm	Folder ifmOS
boot boot.ifm	Folder Bootloader
sis - sisys.ifm - sisdev1.ifm - sisdev2.ifm	Folder SIS-SYS SIS-DEV1 SIS-DEV2
cfg - comconf.cfg - memconf.ifm - iomapping.cfg	Folder Communication configuration Memory configuration Input / output resources
calib - calib-sup.ifm - calib-in.ifm - calib-group.ifm - calib-out.ifm	Folder Calibration files Calibration files Calibration files Calibration files
retain - standard.ret - standard.mb - safe.ret - safe.mb	Folder Application Retain PLC_2 Application Memory Bytes PLC_2 Application Retain PLC_1 Application Memory Bytes PLC_1
log - logging.ifm - statistics.ifm	Folder Log file Statistics file
data - symsapp.xml - symnapp.xml	Folder - Symbol-File PLC_1 - Symbol-File PLC_2
auth auth.txt	Folder Authentication file
compat compat.ifm	Folder Compatibility File
info - appinfo.txt - devinfo.txt - swinfo.txt - comlog.txt - filelist.txt	Folder Application information Device information Software information Log file communication Listing of files in the /data folder

13.8 Overview of user rights



- Execute the `SysFileOpen` function exclusively with the `ACCESS_MODE = AM_READ` or `AM_READ_PLUS` (function parameter `am`) in case of read-only file system objects.

Object	Both applications	Application PLC_1		Application PLC_2		IEC application SysFile	User login required
User	admin	s-admin	s-service	n-admin	n-service	iec	
Groups	Administrator	s-admin	s-service	n-admin	n-service	iec	
Hidden	No	No	No	No	No	Yes	
Data system objects							
" "	ro	ro	ro	ro	ro	na	Yes
/	ro	ro	ro	ro	ro	na	Yes
/os	rw	ro	ro	ro	ro	na	Yes
/boot	rw	ro	ro	ro	ro	na	Yes
/sis/sissys.ifm	rw	ro	ro	ro	ro	na	Yes
/sis/sisdev1.ifm	ro	ro	ro	ro	ro	na	Yes
/sis/sisdev2.ifm	ro	ro	ro	ro	ro	na	Yes
/cfg	rw	ro	ro	ro	ro	na	Yes
/calib	na	na	na	na	na	na	Yes
/security/passwd.ifm	rw	na	na	na	na	na	Yes
/apps/safe.app	rw	rw	ro	na	na	na	Yes
/apps/standard.app	rw	na	na	rw	ro	na	Yes
/retain/safe.ret	rw	rw	ro	na	na	na	Yes
/retain/safe.mb	rw	rw	ro	na	na	na	Yes
/retain/standard.ret	rw	na	na	rw	ro	na	Yes
/retain/standard.mb	rw	na	na	rw	ro	na	Yes
/log/logging.ifm	ro	ro	ro	ro	ro	na	Yes
/log/statistics.ifm	na	na	na	na	na	na	Yes
/data	rw	rw	ro	rw	ro	ro	Yes
/compat	rw	rw	ro	rw	ro	na	Yes
/cmd	rw	na	na	na	na	na	Yes
/info/*	ro	ro	ro	ro	ro	ro	No
/auth	rw	rw	rw	rw	rw	na	No
/security/setpasswd.txt	rw	na	na	na	na	na	Yes
/fit	na	na	na	na	na	na	Yes
IoT request							
/iot/getidentity	ro	ro	ro	ro	ro	ro	No
/iot/deviceinfo	ro	ro	ro	ro	ro	na	Yes
/iot/gettree	ro	ro	ro	ro	ro	na	Yes
/iot/querytree	ro	ro	ro	ro	ro	na	Yes
/iot/getdatamulti	ro	ro	ro	ro	ro	na	Yes
/iot/getelementinfo	ro	ro	ro	ro	ro	na	Yes
/iot/software/firmware	rw	na	na	na	na	na	Yes
/iot/software/memconfig	rw	na	na	na	na	na	Yes

Object	Both applications	Application PLC_1		Application PLC_2		IEC application SysFile	User login required
User	admin	s-admin	s-service	n-admin	n-service	iec	
Groups	Administrator	s-admin	s-service	n-admin	n-service	iec	
Hidden	No	No	No	No	No	Yes	
/iot/software/customer	rw	rw	rw	rw	rw	na	Yes
/iot/software/swinfo	ro	ro	ro	ro	ro	na	Yes
/iot/software/appinfo	ro	ro	ro	ro	ro	na	Yes
/iot/software/fileaccess	rw	rw	rw	rw	rw	na	Yes
/iot/devicecontrol	wo	na	na	na	na	na	Yes
/iot/network	rw	ro	ro	ro	ro	na	Yes
/iot/application	rw	ro	ro	ro	ro	na	Yes
/iot/devicestatus	ro	ro	ro	ro	ro	na	Yes
CODESYS runtime objects							
UserManagement	AECV	V	V	V	V	na	Yes
PLC_1 Each sub-node	AECV	AECV	EV	na	na	na	Yes
PLC_2 Each sub-node	AECV	na	na	AECV	EV	na	Yes

13.8.1 ifm rights

ifm rights		authorised access
ro	read only	read only
rw	read write	Read and write/overwrite
wo	write only	write only
na	not available	not allowed

13.8.2 CODESYS rights

CODESYS rights				Permissions			
(source: CODESYS online help)				A	E	C	V
Objects			Action	Add/ re- move chil- dren	Execute	Change	View:
Device			Login	-	-	-	x
	Logger		Read entries	-	-	-	x
	PlcLogic						
		Application	Login	-	-	-	x
			Create	-	-	x	-
			Create child object	x	-	x	-
			Delete	-	-	x	-
			Load / online change	-	-	x	-
			Create boot project	-	-	x	-
			Read variable	-	-	-	x
			Write variable	-	-	x	x

CODESYS rights				Permissions			
(source: CODESYS online help)				A	E	C	V
Objects			Action	Add/ re- move chil- dren	Execute	Change	View:
			Force variable	-	-	x	x
			Set and delete breakpoint	-	x	x	-
			Set next statement	-	x	x	-
			Read call stack	-	-	-	x
			Single cycle	-	x	-	-
			Switch on flow control	-	x	x	-
			Read flow control	-	-	-	x
			Start / Stop	-	x	-	-
			Reset	-	-	x	-
	PlcShell		Execute PLC shell command	-	x	-	-
	Settings		Read settings	-	-	-	x
			Write settings	-	-	x	-
	UserManagement		Read configuration	-	-	-	x
			Write configuration	-	-	x	-

Legend:

x = authorisation is required

- = authorisation is not relevant

13.9 Task configuration example

The following example describes an option to configure task and bus settings of a CR413S CODESYS project.

The example does not claim to be exhaustive and to apply to all cases, but merely illustrate the technical aspects.

- Whether the presented exemplary configuration can be transferred to the respective application must be evaluated and verified by the system manufacturer.

13.9.1 Initial situation

- CODESYS project with CR413S
- The device tree of `PLC_2` is open
- The CAN interface 0 is configured as ifmCANbus / CANopen
- ifm CR2033 CompactModule is configured as a node on CAN interface 0
- CAN interface 1 is configured as ifmCANbus / J1939
- `J1939_ECU_Tx` configured as local device on CAN interface 1
- `J1939_ECU_Rx` configured as node on CAN interface 1
- In the application of `PLC_2`, the following PRGs are available
 - `CANopen` (CANopen processing)
 - `J1939` (J1939 processing)
 - `PLC_PRG` (main program)

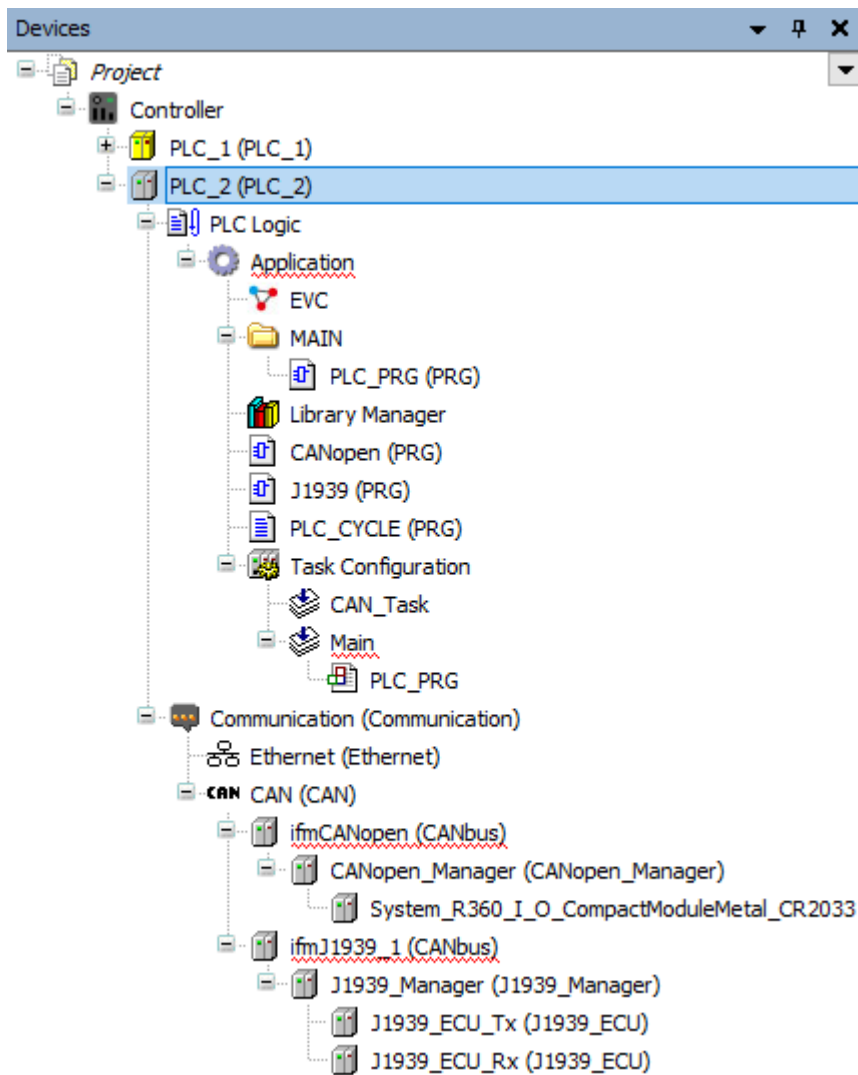


Fig. 19: CODESYS device tree

13.9.2 Setting the task configuration

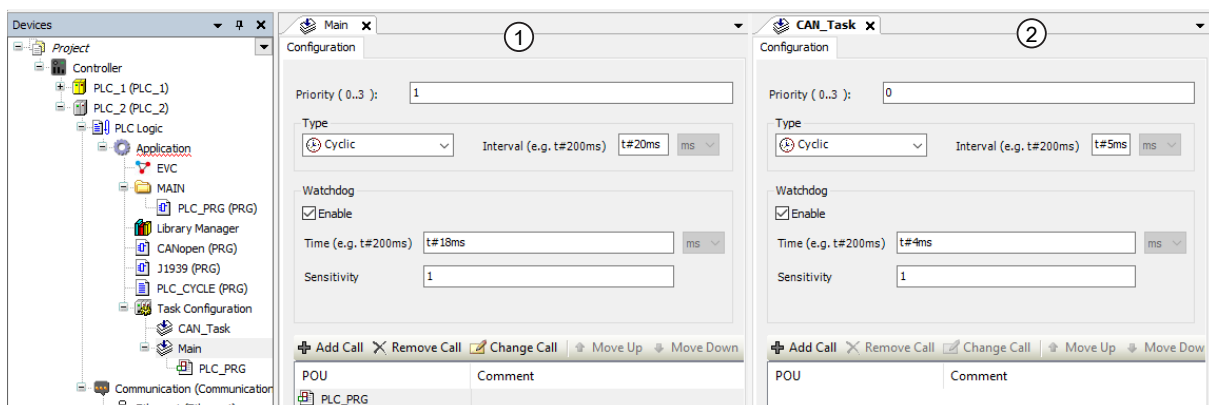


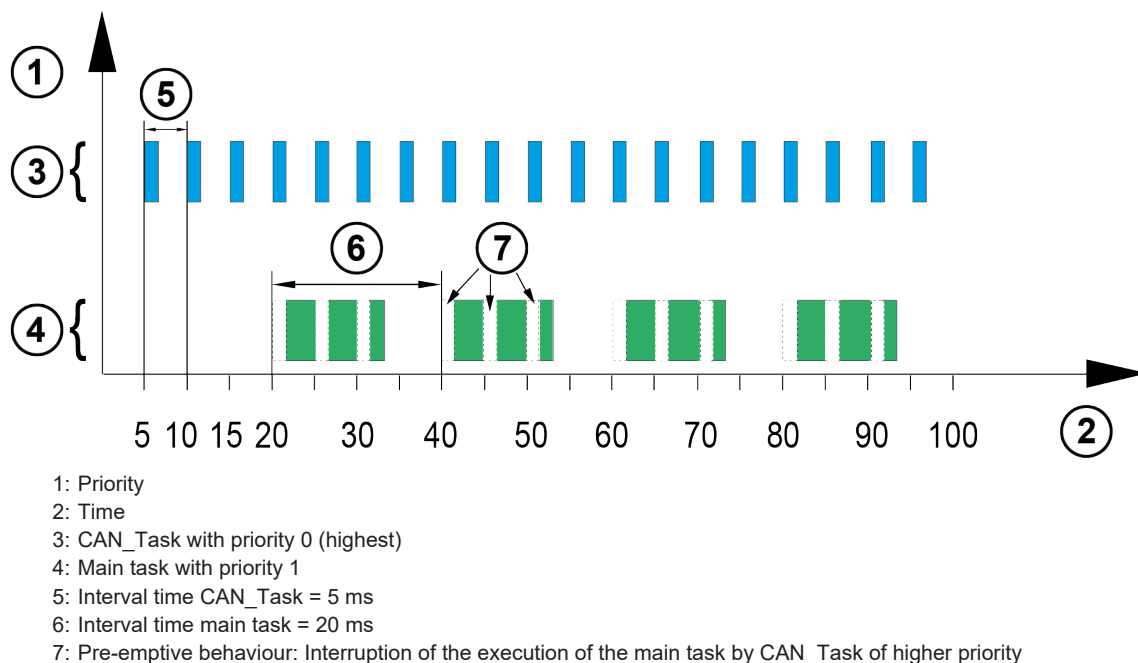
Fig. 20: CODESYS task configuration

► Create the following tasks in the task configuration of PLC_2 and configure them according to the table:

- 1: Main for the main program (PLC_PRG)
- 2: CAN_Task for data transmission via CAN bus

Task	Priority	Type	Interval	Watchdog enabled	Watchdog time	Watchdog sensitivity	POU call
Main	1	Cyclic	t#20ms	Activated	t#18ms	1	PLC_PRG
CAN_Task	0 (highest)	Cyclic	t#5ms	Activated	t#4ms	1	--

Time response of the tasks



! Information of task settings: The task behaviour of the controller is generally pre-emptive. This means that the task with the higher priority interrupts the task with the lower priority if the task with the lower priority is not yet processed in case of a renewed interval start of the task with the higher priority.

Recommendation:

- ▶ A maximum of 4 tasks is possible. However, ideally only use 2 tasks since the complexity of the time response and the necessary task settings increases with the number of tasks due to the pre-emptive behaviour.
- ▶ Division into 2 tasks: Task 1 for CAN communication, task 2 to process the main program. Possible settings, see table.

For error-free CAN communication:

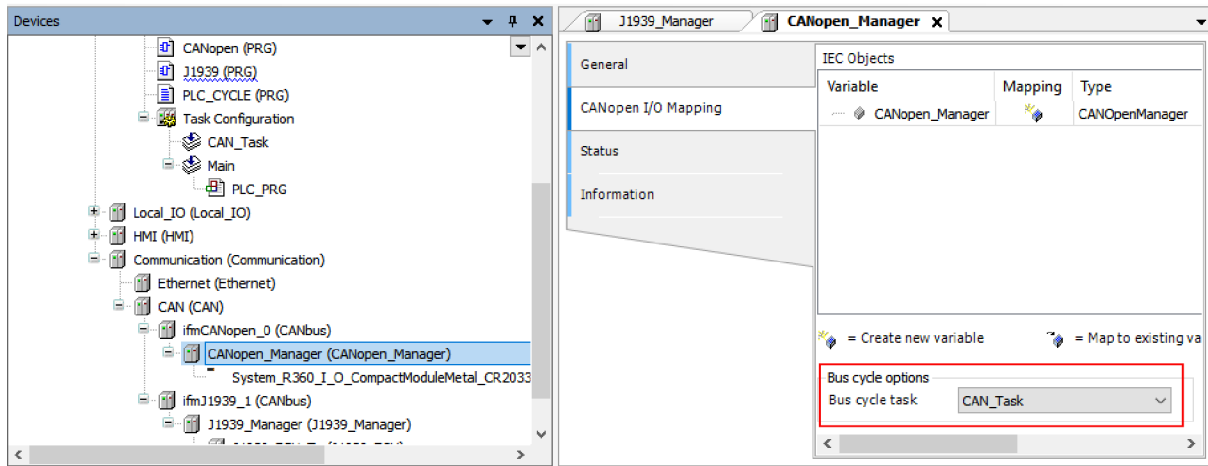
- ▶ Using 2 tasks: Process the CAN communication in an own task.
- ▶ Interval time of the CAN task: According to the J1939 specification, the Transmission Repetition Rate for TSC1 must be 10 ms. The interval for the CAN task is set to 5 ms. Thanks to this, despite of the fluctuation (Jitter = approx. 1 ms), it is assured that the Transmission Repetition Rate of 10 ms will not be exceeded and no messages will be lost.
- ▶ CAN task priority: Assign the highest priority to the CAN task to assure cyclic execution.

13.9.3 Assigning the CAN manager to the bus task

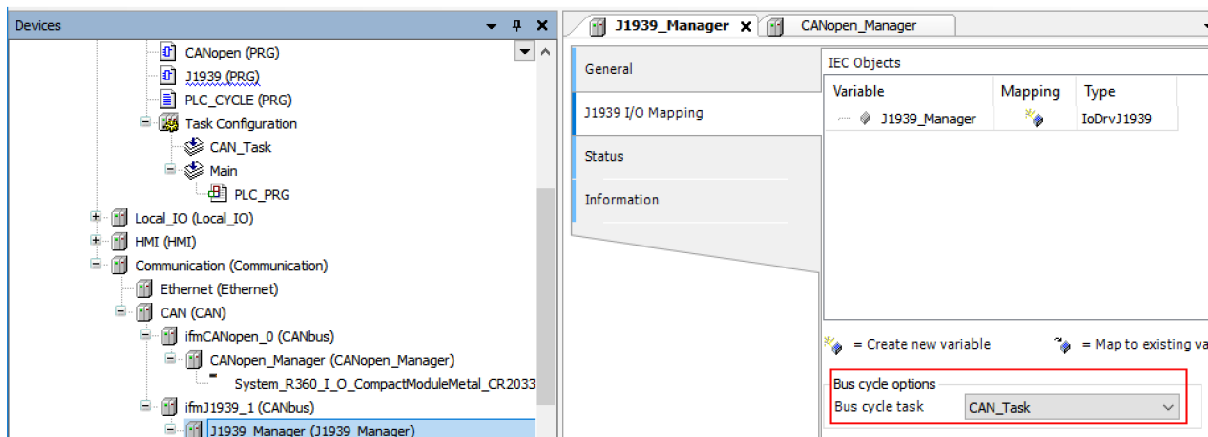
In the properties of the CANopen / J1939 manager, the CANopen / J1939 I/O image is assigned to the `CAN_Task`.

Only if the assignment is correct, the CAN communication in will run in the CAN task interval.

Assignment of the CANopen I/O image to the CAN_Task



Assignment of the J1939 I/O image to the CAN_Task



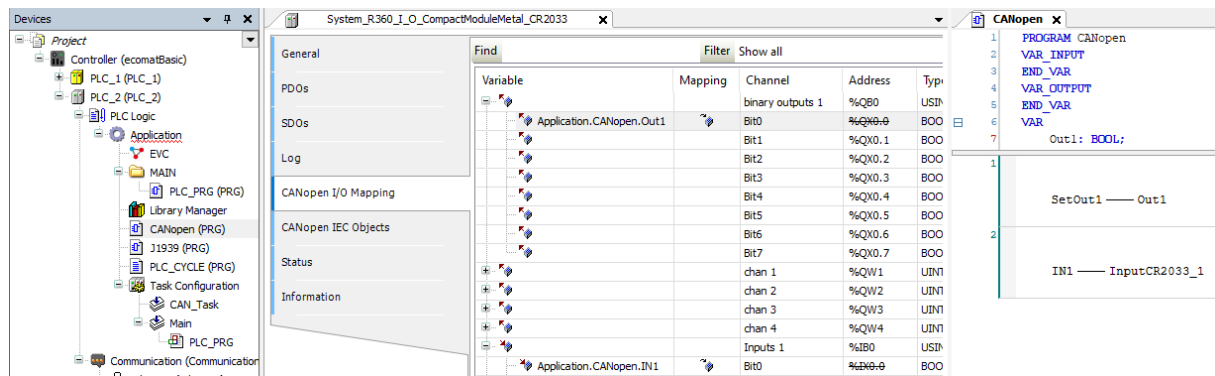
13.9.4 Mapping of the CAN variables

The CAN variables are mapped on PRG variables.

The values of the CAN variables are accessed from the PRG and the PRG variables.

Mapping in the I/O image CANopen

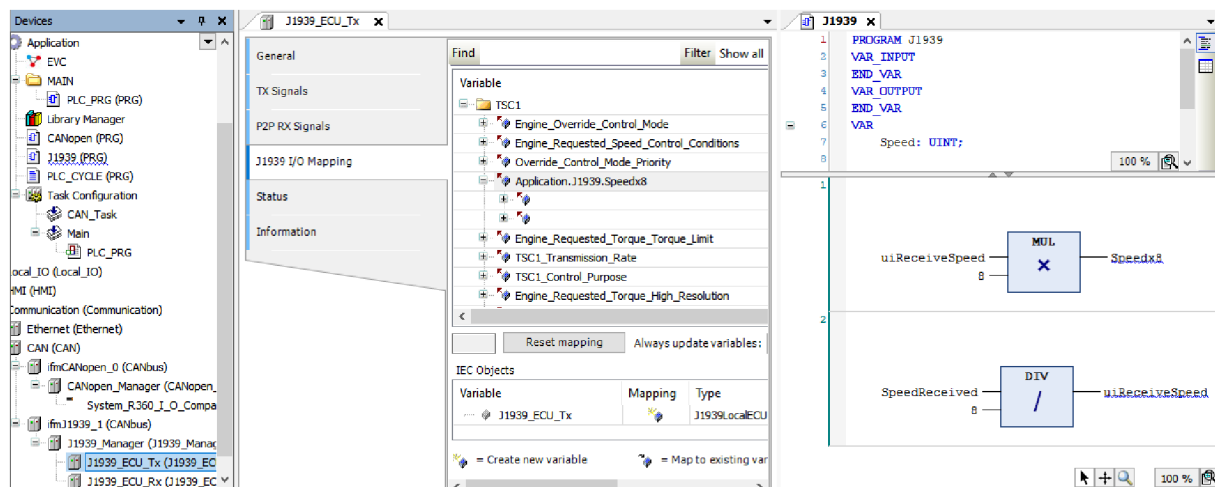
CAN variable channel	PRG variable	Application	PRG name	Variable name in the PRG
Outputs1.Bit0	Application.CAN-open.Out1	Application	CANopen	Out1
Inputs1.Bit0	Application.CAN-open.IN1	Application	CANopen	IN1



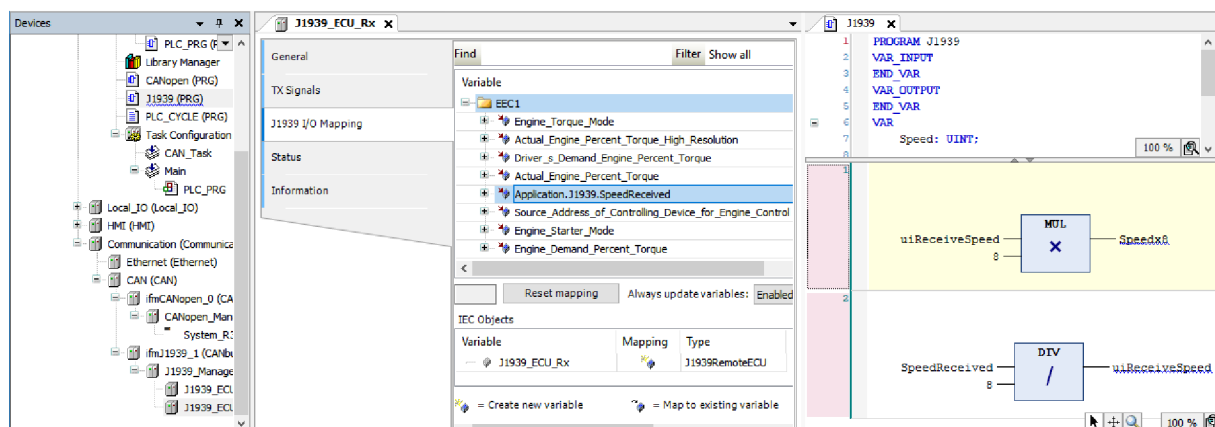
Mapping in the I/O image J1939

Device	CAN variable channel	PRG variable	Application	PRG name	Variable name in the PRG
J1939_ECU_Tx	TSC1.Engine_Requested_Speed_Speed_Limit	Application.J1939.Speed8x	Application	J1939	Speed8x
J1939_ECU_Rx	EEC1.Engine_Speed	Application.J1939.SpeedReceived	Application	J1939	SpeedReceived

J1939_ECU_Tx



J1939_ECU_Rx



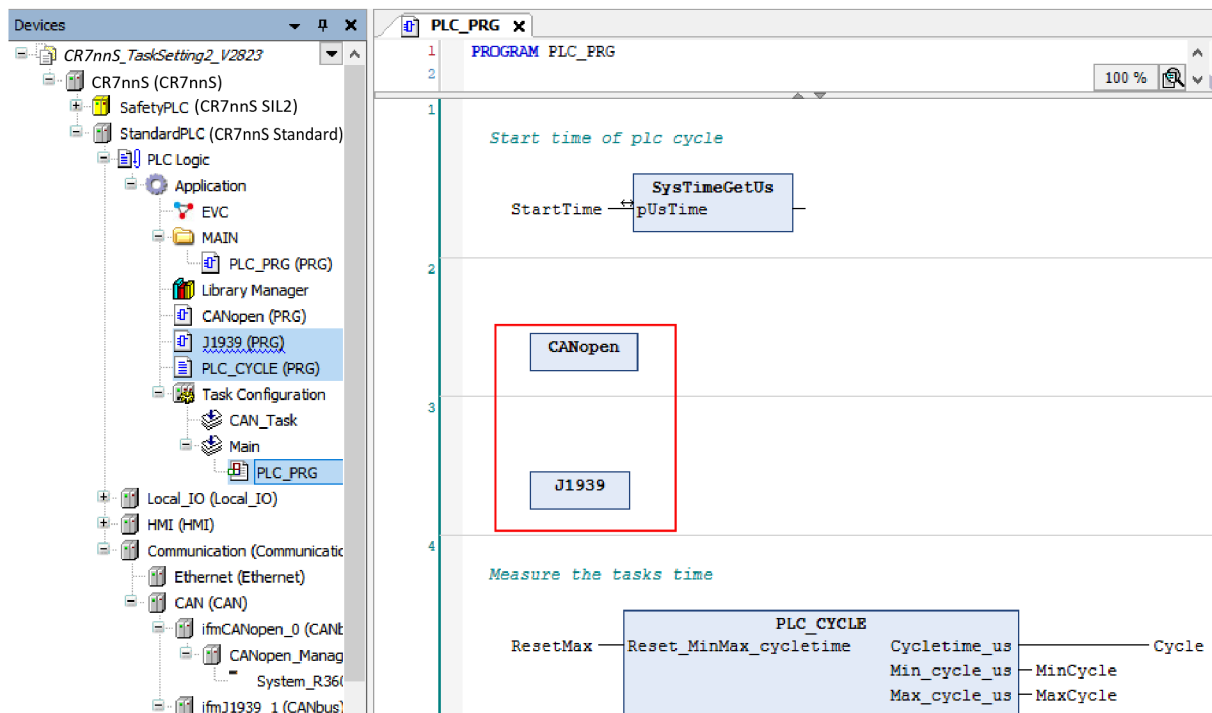
13.9.5 Calling CAN POU's

The CAN variables are further processed in the PRGs `CANopen` and `J1939`. Content of the PRGs (programming example) → [Mapping of the CAN variables \(→ 237\)](#)

The main program `PLC_PRG` calls the PRGs.

All PRGs are processed in the `Main` task.

Call up of the PRGs



13.9.6 Task monitoring

Online monitoring of the bus cycle time

Project.project* - CODESYS

File Edit View Project Build Online Debug Tools Window Help SIL2

Application [PLC_2: PLC Logic]

Task Configuration

Task	Status	IEC-Cycle Count	Cycle Count	Configure...	Last Cycle Time (µs)	Average Cycle Time (µs)	Max. Cycle Time (µs)	Min. Cycle Time (µs)	Jitter (µs)	Min. Jitter (µs)	Max. Jitter (µs)
CAN_Task	Valid	8158	8158	5 ms	356	327	1162	266	-	-	-
Main	Valid	2036	2036	20 ms	15	15	376	12	-	-	-

13.10 ifm behaviour models for function blocks

This chapter describes the behaviour models for ifm function blocks.

13.10.1 General

ifm function blocks feature the following outputs to return status and error information:

Output	Description	
xError	TRUE	An error has occurred.
	FALSE	No error has occurred.
eDiagInfo	Diagnostic/error information: Messages / diagnostic codes of the function blocks (→ 214)	

All inputs and outputs in the function block that belong to the ifm behaviour model are featured at the top.

13.10.2 Behaviour model ENABLE

Function blocks that use the behaviour model ENABLE are cyclically processed as long as the status at the input is xEnable = TRUE.

If xEnable = FALSE, the function block will not be executed. All function block outputs are reset to their preset default values and will not be updated. In this case the following applies: xError = FALSE and eDiagInfo = STAT_INACTIVE.

Function blocks that have no xEnable input are processed cyclically when the application is started. The processing is only terminated when the application is stopped. The behaviour corresponds with the behaviour of a function block with a permanent TRUE at the xEnable input.

13.10.2.1 Response to errors

In case of an error, xError is set to TRUE and eDiagInfo indicates the diagnostic code as long as xEnable is = TRUE.

Irrespective of the data type, all other outputs of the function block will be reset to the following values:

Data type	Value
numerical	0 / 0.0
String	Empty string
BOOL/Bit	FALSE

13.10.3 Behaviour model EXECUTE

Function blocks that have the EXECUTE behaviour model are processed once after a rising edge at the xExecute input.

If the function block has executed its function successfully, the output is set xDone = TRUE.



For some function blocks, the signal to xExecute has to remain set to TRUE until xDone=TRUE. If the signal to xExecute becomes FALSE beforehand, the edit process of the function block is aborted without result.

The performance can be seen from the description of the input parameter xExecute of the corresponding function block.

13.10.3.1 Response to errors

In case of an error, xError is set to TRUE and eDiagInfo indicates the error status as long as xExecute is = TRUE.

The output xDone is set to FALSE since the execution could not be finished successfully.

Irrespective of the data type, all other outputs of the function block will be reset to the following values:

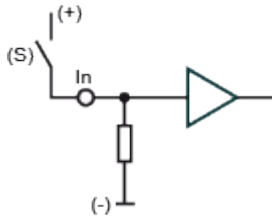
Data type	Value
numerical	0 / 0.0
String	Empty string
BOOL/Bit	FALSE

List of figures

Fig. 1	System context.....	17
Fig. 2	System overview	21
Fig. 3	Memory areas of the controller	22
Fig. 4	Block diagram of the supply	23
Fig. 5	Basic structure of the output cut-off for output groups type A.....	37
Fig. 6	Basic structure of the output deactivation for output groups type B.....	37
Fig. 7	Example: Wiring with blocking diodes to prevent power feedback with output group type OutputGroup-A.....	38
Fig. 8	Example: Wiring with blocking diodes to prevent feedback with output group type OutputGroup-B.....	39
Fig. 9	Software overview	43
Fig. 10	Software modules	44
Fig. 11	CODESYS project structure ecomatBasic	54
Fig. 12	Users and groups.....	61
Fig. 13	Interface plugin.....	72
Fig. 14	Objects of a standard PLC application.....	87
Fig. 15	Handling of non-volatile data.....	90
Fig. 16	Operating states.....	114
Fig. 17	CODESYS PLC Log.....	215
Fig. 18	Connectors.....	000
Fig. 19	CODESYS device tree	235
Fig. 20	CODESYS task configuration	235

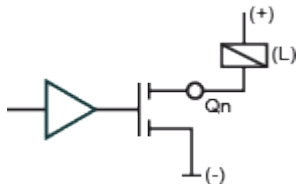
Glossary

CSI Input



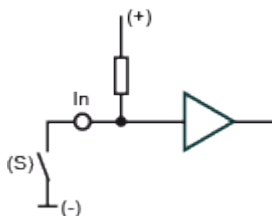
CSI Input = Current Sinking Input; In = connection of digital input n (S) = sensor; Block diagram, plus-switching digital input for positive sensor signal: input = open => signal = low (GND); Low-Side Input (BL)

CSI output



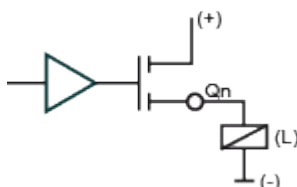
CSI Output: Current Sinking Output; Qn = connection of output n; (L) = load; Block diagram, minus-switching output (BL) for negative output signal; Low-Side Output (BL)

CSO Input



CSO Input = Current Sourcing Input; In = connection of digital input n (S) = sensor; Block diagram, minus-switching digital input for negative sensor signal: Input = open => signal = high (Supply); High-Side Input (BH)

CSO output



CSO Output: Current Sourcing Output; Qn = connection of output n; (L) = load; Block diagram, plus-switching digital output for positive sensor signal; High-Side Output (BH)

Dither

to dither = to sway / tremble. 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.

Operating system

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

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.

ratiometric

Ratio (lat.) = proportion Measurements can also be made ratiometrically = ratio measurement. If the output signal of a sensor is proportional to its supply voltage, ratiometric measurement (= measurement in relation to the supply) can reduce, ideally even eliminate, the influence of fluctuations in the supply.

Index

A

Access protection	13
Access to inputs	95
Access to outputs	95
Accessing inputs and outputs	91
Activate the access protection for a project	60
Add function libraries to the application	69
Application	44
Application information (appinfo.txt)	108
Assigning inputs/outputs – PLC_1 / PLC_2	64
Assigning the memory allocation - PLC_1 / PLC_2	64
Assign the CAN manager to the bus task	236
aSysInfoList (GVL)	211
Available memory	21

B

Behaviour in case of conversions	226
Behaviour in case of floating point operations	226
Behaviour in case of integer operations	227
Behaviour in case of specific arguments	226
Behaviour in case of voltage dip	24
Behaviour model ENABLE	240
Behaviour model EXECUTE	240
Behaviour model for ifm function blocks	240
Block diagram of the supply	23
Bootloader	44
BUS_STATE (ENUM)	147

C

Call CAN POU's	239
Call sequence for diagnostic function blocks for inputs and outputs	94
CAN	
Interfaces and protocols	40
CAN libraries	126
CAN_BAUDRATE (ENUM)	122
CAN_BUS_STATE (STRUCT)	147
CAN_CHANNEL (ENUM)	122
CAN_Enable	127
CAN_Info (GVL)	147
CAN_Recover	129
CAN_RemoteRequest	131
CAN_RemoteResponse	133
CAN_Rx	135
CAN_RxMask	137
CAN_RxRange	139
CAN_RxRangeExt	141
CAN_Status	143
CAN_Tx	145
CANconstants (GVL)	122
CANopen	
Network Management (NMT)	97
Send and receive SDO	97
Carrying out the installation	47

Change history	10
Check the communication path (blinking test)	59
Check the hardware version of the device	102
Check the operating system version of the device	101
Checking and updating installed add-ons	50
CODESYS	10
CODESYS Debugging	110
CODESYS reset requirements (PLC_1)	118
Components of the package	48
ConfigDiagLevel	190
ConfigDiagProt	193
ConfigSwThreshold	159
Configuration time	20
Configure CAN interfaces	76
Configure Ethernet interface	73
Configure IEC watchdog	71
Configure interfaces	71
Configure task processing	68
Configuring an interface in CODESYS	72
Configuring inputs and outputs	86
Configuring PLCs	68
Configuring the Ethernet interface	56
Connection via CAN	101
Connection via Ethernet	101
Controlling LEDs in the applications	116
Controlling the device	95
COP_GetNodeState	149
COP_SDOread	151
COP_SDOwrite	153
COP_SendNMT	155
COUNT_DIRECTION (ENUM)	170
Create a user in the CODESYS project	63
Creating a standard PLC application	88
Cross references inputs	25
Cross references outputs	31
CurrentControl	202
Cyber security	12
Cyclic diagnostics	18

D

Data exchange between PLC_1 and PLC_2;	69
Data transmission for series production	104
Data transmission with TFTP	106
Data types	88
Default behaviour of the inputs and outputs in case of an error	92
Delete the application program on the device	104
DEST (ENUM)	195
Device library	121
Device supply (technology)	23
Diagnostics	18
Digital input block diagram minus-switching	30
Digital input plus-switching principle circuit	30
Digital output minus-switching principle circuit	35

Digital output plus-switching principle circuit	35
Directory structure and file overview	230
Display of the controller searched for	59
Documentation of the overall application	107
Documenting the serial number and operating system version	107

E

eDIAG_PROT_MODE (ENUM)	195
ENCODER_RESOLUTION (ENUM)	170
Enter information about applications	63
Error classes	213
Error in the IEC application	99
Error messages	213
Establishing a connection to the device	101
Ethernet interface	41
Example\	
High stack consumption	70
Low stack consumption	71
Executing reset variants	118

F

FastCount	163
Feedback in case of externally supplied outputs	38
File management	67
Files for series production	107
Filter	221
Filter times of the inputs	221
Filter times of the outputs	222
FILTER_INPUT (ENUM)	186
FILTER_OUTPUT (ENUM)	186
FILTER_OUTPUT_GROUP (ENUM)	200
FREQ_SENSE_PERIODS (ENUM)	170
Function description of non-volatile data	90

G

GetInfo	209
Getting started	52

H

Hardware	47
Hardware description	21
Hardware information (devinfo.txt)	108
Hardware structure	21
Help function libraries	208

I

ifm ecologOne data transmission	105
ifm function libraries	120
IncEncoder	165
Influence of actuators on the output diagnostics	39
Initial situation	234
Input	173
Input and output libraries	158
Input type IN FREQUENCY-B	27

Input type IN MULTIFUNCTION-A	25
Input type IN RESISTOR-A	29
Inputs (technology)	25
Install package (PC/laptop)	49
Installation	46
Installing CODESYS Development System	48
Integrated programmable logic controllers	17
Intended use	15
Interface configuration file comconf.cfg	81
Interfaces	40

L

LED_COLOUR (ENUM)	122
LED_FLASH_FREQ (ENUM)	123
Libraries	45
Library ifmCANopenManager.library	148
Library ifmConfigSwThreshold.library	158
Library ifmDeviceCR07nn.library	121
Library ifmFastInput.library	162
Library ifmIOcommon.library	172
Library ifmIOconfigDiagProt.library	189
Library ifmOutGroup	196
Library ifmOutPWM	201
Library ifmRawCAN.library	126
Library ifmSysInfo.library	208
Licensing	43
Loading the PLC_1 application to the device	103
Loading the PLC_2 application to the device	103

M

Managing device users	61
Mapping inputs/outputs with the CODESYS plugin	65
Mapping of the CAN variables	237
Memory allocation	21
Memory allocation variants	22
Messages / diagnostic codes of the function blocks	214
Modbus	99
MODE_CURRENT_CONTROL (ENUM)	207
MODE_FAST_COUNT (ENUM)	170
MODE_INC_ENCODER (ENUM)	170
MODE_INPUT (ENUM)	187
MODE_OUTPUT (ENUM)	187
MODE_OUTPUT_GROUP (ENUM)	200
MODE_PERIOD (ENUM)	171
MODE_PWM (ENUM)	207
MODE_SYSTEM_SUPPLY (ENUM)	188
MODE_TEMPERATURE (ENUM)	188
Module slot detection	84

N

NMT_SERVICE (ENUM)	157
NMT_STATES (ENUM)	157
Non-volatile data	89
Note on wiring	21
Note!	11

O

Objects of a standard PLC application	87
Operating states	113
Operating system	44
Output	175
Output group type OutputGroup-A	36
Output group type OutputGroup-B	37
Output groups	36
Output type OUT PWM-n-A	31
Output type OUT PWM-n-B	33
Output type OUT PWM-n-D	34
Output types	31
OutputGroup	197
Outputs (technology)	31
Overview	
Hardware	21
ifm user documentation	9
Software	42
Overview of operating mode states	113
Overview of user rights	231

P

Period	167
Preparation of the PLC	61
Programming of the standard application	87
Purpose of the document	8
PWM1000	205

R

RawCAN	
Control CAN network nodes	96
Request and send remote CAN messages	96
Send and receive CAN messages	96
Read the device information	108
Reading measured system values	95
Reset	117, 124, 185
Reset application (cold)	118
Reset application (origin)	119
Reset application (warm)	118
Reset behaviour of the system	117
RESET_TYPE (ENUM)	125
Response to errors	240, 241

S

Self-test at restart (start-up test)	18
Service communication	41, 42
Set communication path of the PLC	56
Set the communication path of the PLC	57
SetLED	178
Setting the filter	92
Setting the operating mode	91
Setting the task configuration	235
Set-up and maintenance	100
Sign the user in to the CODESYS project	63
sleep mode	24
Software	47
Software description	42

Software in the controller	43
Software information (swinfo.txt)	109
Software modules for the device	44
Software on the PC	43
Stack	70
Standard diagnostic limit values of inputs	92
Standard diagnostic limit values of outputs	93
Start conditions	24
Start-up behaviour of the controller	12
Status LED	
Ethernet interfaces (ETH)	116
Sleep mode (SYS0)	116
System bootloader (SYS0)	116
system ifm operating system (SYS0+SYS1)	115
System PLC (SYS0, SYS1)	115
Status LEDs	114
SupplySwitch	180
Supported programming languages	88
Supported SysSocket functions	228
Supported variable types	88
Switching off output groups via IEC application	36
Switching on and off	24
SYS_INFO (STRUCT)	212
SYS_VOLTAGE_CHANNEL (ENUM)	188
SysInfo (GVL)	122
SysInfoStruct (STRUCT)	122
SysSocket interface	97
System (in general)	17
System configuration	61
System context of the controller	17
System description	17
System overview	21
System requirements	47
SystemSupply	181

T

Task configuration example	234
Task monitoring	239
Temperature	183
Time response	19
Time-related behaviour IEC application	19
Transferring a CODESYS project to the device	102
Transmission of the files with CODESYS	106
Troubleshooting	213
Types of inputs	25

U

Uninstall package (PC/laptop)	50
Update the operating system of the device	102
Update the operating system of the device with ifm ecolgOne	102
Updating a CODESYS project from SP11 to SP18	54
Updating a package (PC/laptop)	49
Use CANopen	96
Use RawCAN (CAN Layer 2)	95
Use SAE J1939	97

User-defined data	67
Using CAN	95
Using Ethernet	97
Using ifm function libraries	94
Using the operators	223

V

via function block	
RAW-CAN	81
Via system configuration	
CANopen device	79
CANopen Manager	77
J1939 Manager	79
Voltage ranges	23

W

Watchdog	18
----------	----