Contents

1 About this manual ........................................6
   1.1 Legal and copyright information ........................................6
   1.2 Purpose of the document ........................................7
   1.3 Symbols and styles used ........................................7
   1.4 Overview: documentation modules for CR721S .........................8
   1.5 Overview: documentation for CODESYS 3.n .......................8
   1.6 How is this documentation structured? .............................9
   1.7 History of the document CR0721 ....................................9

2 Functions and features ..................................10

3 Safety instructions .....................................11
   3.1 Please note! ..................................................11
   3.2 What previous knowledge is required? ..............................12
   3.3 Important standards ..........................................13
   3.4 Organise the creation of safe machinery with the V model ....14
   3.5 Start-up behaviour of the controller ................................16
   3.6 Warnings used .............................................16
   3.7 Notes: serial number ........................................17

4 Installation .............................................18
   4.1 System requirements .........................................18
      4.1.1 Hardware ..............................................18
      4.1.2 Software ..............................................18
      4.1.3 Licensing ..............................................18
   4.2 Carry out installation .........................................19
      4.2.1 CODESYS programming software ................................19
      4.2.2 Complete package for ecomatController CR721S .............20
      4.2.3 Check the operating system version of the device ..........22
      4.2.4 Update the operating system of the device ..................24

5 System description .....................................26
   5.1 Hardware description .........................................26
      5.1.1 Hardware structure ....................................26
      5.1.2 Device supply (technology) ................................33
      5.1.3 Monitoring concept .....................................36
      5.1.4 Inputs (technology) ....................................38
      5.1.5 Outputs (technology) ..................................44
      5.1.6 Feedback in case of externally supplied outputs ..........52
   5.2 Interfaces ..................................................53
      5.2.1 Serial interface .........................................53
      5.2.2 Ethernet interface .......................................53
      5.2.3 CAN: Interfaces and protocols ..............................54
   5.3 Software description .........................................55
      5.3.1 Overview: Software .....................................55
      5.3.2 Software module for the device .............................57

6 Getting started .........................................59
   6.1 Start CODESYS .............................................59
   6.2 Create CODESYS project ....................................59
      6.2.1 Create new project with CR721S ............................60
7 System configuration

7.1 Configure PLC

7.1.1 Allocate memory partition

7.1.2 Allocate inputs/outputs

7.1.3 Manage files

7.1.4 Manage users and groups

7.2 Configure inputs and outputs

7.2.1 via system configuration

7.2.2 via function block

7.3 Configure interfaces

7.3.1 Configure serial interface

7.3.2 Configure Ethernet interface

7.3.3 Configure CAN interfaces

7.3.4 Interface configuration file comconf.cfg

8 Programming

8.1 Objects of a PLC application

8.2 Create PLC application

8.2.1 Supported programming languages

8.2.2 Supported variable types

8.2.3 Options to access input and output data

8.2.4 Configure task processing

8.3 Use ifm function libraries

8.3.1 Access to inputs

8.3.2 Access to outputs

8.3.3 Control device

8.3.4 Read device information

8.4 Use IO mapping

8.4.1 Access inputs

8.4.2 Access outputs

8.4.3 Read diagnostic data of the device

8.5 Use RawCAN (CAN Layer 2)

8.5.1 RawCAN: Control CAN network nodes

8.5.2 RawCAN: Send and receive CAN messages

8.5.3 RawCAN: Request and send remote CAN messages

8.6 Use CANopen

8.6.1 CANopen: Send and receive SDO

8.6.2 CANopen: Network Management (NMT)

8.7 Use SAE J1939

9 Operation

9.1 Transfer CODESYS project to device

9.1.1 Load the application to the device

9.1.2 Delete application from CR721S

9.2 Operating states

9.3 Status LEDs

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

9.3.2 Status LED: system PLC (SYS0, SYS1)

9.3.3 Status LED: System bootloader (SYS0)

9.3.4 Status LED: Ethernet interfaces (ETH0, ETH1)

9.3.5 Controlling LEDs in the applications
9.4 Reset
9.4.1 Supported reset variants ........................................ 99
9.4.2 Reset application (warm) ........................................ 99
9.4.3 Reset application (cold) ......................................... 100
9.4.4 Reset application (origin) ..................................... 100
9.5 Data transmission for series production ....................... 101
9.5.1 Transmission of the files with CODESYS .................. 101
9.5.2 Data transmission with TFTP ................................. 102
9.5.3 Files for series production .................................. 103
9.6 Display system information .................................. 104

10 ifm function libraries ........................................ 105
10.1 General .......................................................... 105
10.2 Library ifmCANopenManager.library ......................... 105
  10.2.1 COP_GetNodeState .......................................... 106
  10.2.2 COP_SDOread ................................................ 108
  10.2.3 COP_SDOwrite ............................................... 110
  10.2.4 COP_SendNMT .............................................. 112
  10.2.5 NMT_SERVICE (ENUM) ..................................... 114
  10.2.6 NMT_STATES (ENUM) ...................................... 114
10.3 Library ifmDeviceCR0721.library .......................... 115
  10.3.1 CAN_BAUDRATE (ENUM) .................................... 116
  10.3.2 CAN_CHANNEL (ENUM) ..................................... 116
  10.3.3 CANconstants (GVL) ....................................... 116
  10.3.4 SysInfo (GVL) ............................................. 116
  10.3.5 SysInfoStruct (STRUCT) ................................ 117
  10.3.6 LED_COLOUR (ENUM) ...................................... 117
  10.3.7 LED_FLASH_FREQ (ENUM) .................................. 117
10.4 Library ifmFastInput.library ................................ 118
  10.4.1 FastCount .................................................. 119
  10.4.2 IncEncoder .................................................. 121
  10.4.3 Period ...................................................... 123
  10.4.4 COUNT_DIRECTION (ENUM) ................................. 125
  10.4.5 ENCODER_RESOLUTION (ENUM) ............................ 125
  10.4.6 FREQ_SENSE_PERIODS (ENUM) ............................ 125
  10.4.7 MODE_FAST_COUNT (ENUM) ................................. 125
  10.4.8 MODE_INC_ENCODER (ENUM) ............................... 126
  10.4.9 MODE_PERIOD (ENUM) ..................................... 126
10.5 Library ifmIOcommon.library ................................ 127
  10.5.1 Input ....................................................... 128
  10.5.2 Output ..................................................... 131
  10.5.3 SetLED ..................................................... 134
  10.5.4 SupplySwitch ............................................... 136
  10.5.5 SystemSupply .............................................. 138
  10.5.6 Temperature ................................................ 140
  10.5.7 FILTER_INPUT (ENUM) ..................................... 142
  10.5.8 FILTER_OUTPUT (ENUM) ................................... 142
  10.5.9 MODE_INPUT (ENUM) ....................................... 143
  10.5.10 MODE_OUTPUT (ENUM) ..................................... 144
  10.5.11 SYS_VOLTAGE_CHANNEL (ENUM) ........................ 145
10.6 Library ifmOutGroup .......................................... 146
  10.6.1 OutputGroup ................................................ 147
  10.6.2 FILTER_OUTPUT_GROUP (ENUM) .......................... 150
  10.6.3 MODE_OUTPUT_GROUP (ENUM) ............................. 150
10.7 Library ifmOutHBridge ....................................... 151
  10.7.1 HBridge .................................................... 152
  10.7.2 MODE_BRAKE (ENUM) ...................................... 155
10.8 Library ifmOutPWM ........................................... 156
  10.8.1 CurrentControl ............................................... 157
  10.8.2 PWM1000 ................................................... 160
  10.8.3 MODE_CURRENT_CONTROL (ENUM) ........................ 164
  10.8.4 MODE_PWM (ENUM) .......................................... 164
Contents

10.9 Library ifmRawCAN.library ...................................................... 165
10.9.1 CAN_Enable ........................................................................ 166
10.9.2 CAN_Recover ................................................................. 168
10.9.3 CAN_RemoteRequest ....................................................... 170
10.9.4 CAN_RemoteResponse ..................................................... 172
10.9.5 CAN_Rx ........................................................................... 174
10.9.6 CAN_RxMask ................................................................. 176
10.9.7 CAN_RxRange ............................................................... 178
10.9.8 CAN_Tx ........................................................................... 180
10.9.9 CAN_Info (GVL) ............................................................. 182
10.9.10 CAN_BUS_STATE (STRUCT) ........................................ 182

11 Troubleshooting ................................................................. 183
11.1 Error classes ................................................................. 183
11.2 Error messages ............................................................... 184
11.3 Messages / diagnostic codes of the function blocks ................. 184

12 Appendix ........................................................................... 185
12.1 Directory structure and file overview ...................................... 185
12.2 ifm behaviour models for function blocks ................................ 186
  12.2.1 General ....................................................................... 186
  12.2.2 Behaviour model ENABLE ............................................ 186
  12.2.3 Behaviour model EXECUTE .......................................... 187

13 Glossary of Terms .............................................................. 188

14 Index .................................................................................. 202

15 Notizen • Notes • Notes .......................................................... 206

16 ifm weltweit • ifm worldwide • ifm à l’échelle internationale .......... 209
1 About this manual

1.1 Legal and copyright information

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

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

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

This manual describes the *ecomatmobile* family for mobile machines of *ifm electronic gmbh*:
- ecomatController <eco100-Bez> (Art.-Nr.: CR721S) firmware version V2.5.0.n and higher

The CODESYS programming system is required to program this device: version V3.5 SP11 or higher

These instructions describe the following topics:
- Configuration of the device in the setup mode
- Firmware update of the device in the recovery mode
- Configuration of the device using CODESYS
- Programming of the device-internal PLC of the CR721S using the CODESYS programming system.
- Description of the device-specific CODESYS function libraries

### 1.3 Symbols and styles used

**WARNING**
Death or serious irreversible injuries may result.

**CAUTION**
Slight reversible injuries may result.

**NOTICE**
Property damage is to be expected or may result.

Important note
Non-compliance may result in malfunction or interference.

Information
Supplementary note.

► ... Instructions
> ... Reaction, result
→ ... Cross-reference or internet link
123 Decimal number
0x123 Hexadecimal number
0b010 Binary number
[...] Designation of pushbuttons, buttons or indications
1.4 Overview: documentation modules for CR721S

The documentation for this devices consists of the following modules:
(Downloads from ifm's website → ifm weltweit • ifm worldwide • ifm à l’échelle internationale (→ p. 209) )

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

1.5 Overview: documentation for CODESYS 3.n

The following user documentation is provided by 3S GmbH for programming the CR721S with CODESYS:

<table>
<thead>
<tr>
<th>Document</th>
<th>Content / Description</th>
</tr>
</thead>
</table>
| Online help                     | • Context-sensitive help  
• Description of the CODESYS programming system  
After the installation of the programming system store and accessible on the hard disk of the PC/laptop: \Programme (x86)\3S CODESYS\CODESYS\Online Help |
| CODESYS installation and first steps | • Remarks about the installing of the programming system CODESYS  
• First steps for handling the programming system CODESYS  
After the installation of the programming system store and accessible on the hard disk of the PC/laptop: \Programme (x86)\3S CODESYS\CODESYS\Documentation |
This document is for programmers who program safety-related controllers. Download at: \www.codesys.com |
1.6 How is this documentation structured?

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

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

In case of malfunctions or uncertainties please contact the manufacturer at:
Contact → ifm weltweit • ifm worldwide • ifm à l’échelle internationale (→ p. 209)

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

We reserve the right to make alterations which can result in a change of contents of the documentation. You can find the current version on ifm’s website:
→ ifm weltweit • ifm worldwide • ifm à l’échelle internationale (→ p. 209)

1.7 History of the document CR0721

What has been changed in this manual? An overview:

<table>
<thead>
<tr>
<th>Date</th>
<th>State</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-06-13</td>
<td>01</td>
<td>new document</td>
</tr>
<tr>
<td>2017-12-19</td>
<td>02</td>
<td>Article number, operating system version, CODESYS version</td>
</tr>
</tbody>
</table>
2 Functions and features

This device is used to control processes in applications. For this, device 2 contains 2 PLCs that can be programmed independently of each other. In the CODESYS software platform, these PLCs are called:
• standard PLC
• safety PLC

⚠️ WARNING

The safety functionality is in preparation.
At present the device has NO safety functionality!
▶ Do NOT use the device for safety-related functions!
3 Safety instructions

3.1 Please note!

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

► The manufacturer of the machine/equipment is responsible for ensuring the safety of the machine/equipment.

► Follow the national and international regulations of the country in which the machine/installation is to be placed on the market!

⚠️ WARNING

Non-observance of these instructions can lead to property damage or bodily injury!

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

► The acting person must have read and understood the safety instructions and the corresponding chapters in this manual before working on and with this device.

► The acting person must be authorised to work on the machine/equipment.

► The acting person must have the qualifications and training required to perform this work.

► Adhere to the technical data of the devices!

You can find the current data sheet on ifm’s homepage.

► Observe the installation and wiring information as well as the functions and features of the devices!

→ supplied installation instructions or on ifm’s homepage

► Please note the corrections and notes in the release notes for the existing hardware, software and documentation, available on the ifm website

Website → ifm weltweit • ifm worldwide • ifm à l’échelle internationale (→ p. 209)

⚠️ WARNING

The user is responsible for the reliable function of the application programs he designed. If necessary, he must additionally carry out an approval test by corresponding supervisory and test organisations according to the national regulations.
WARNING
The safety functionality is in preparation. At present the device has NO safety functionality!
► Do NOT use the device for safety-related functions!

3.2 What previous knowledge is required?

This document is intended for people with knowledge of control technology and PLC programming with IEC 61131-3.
To program the PLC, the people should also be familiar with the CODESYS software.
The document is intended for specialists. These specialists are people who are qualified by their training and their experience to see risks and to avoid possible hazards that may be caused during operation or maintenance of a product. The document contains information about the correct handling of the product.
Read this document before use to familiarise yourself with operating conditions, installation and operation. Keep the document during the entire duration of use of the device.
Adhere to the safety instructions.
3.3 Important standards

⚠️ WARNING

The safety functionality is in preparation.
At present the device has NO safety functionality!
► Do NOT use the device for safety-related functions!

Among other things, the programmer of safety-related controllers should also know and observe the content of the following standards:

<table>
<thead>
<tr>
<th>Standard</th>
<th>Title, content</th>
</tr>
</thead>
</table>
| ISO 13849 | Standard: Safety of machinery, safety-related parts of control systems  
  • Part 1: General design principles  
  • Part 2: Validation  
  here: The device can be used up to PL d |
| IEC 62061 | Standard: Machine safety – functional safety of electric, electronic and programmable machine controllers  
  • Specification of the functional requirements  
  • Specification of the safety requirements  
  here: The device can be used up to SIL CL 2 |
3.4 Organise the creation of safe machinery with the V model

<table>
<thead>
<tr>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Define, observe, check and document the workflow steps in the V model</td>
</tr>
<tr>
<td>▶ Define responsibilities for tasks</td>
</tr>
<tr>
<td>▶ Define a person responsible for safety technology (Functional Safety Manager)</td>
</tr>
</tbody>
</table>

Every machine manufacturer should define the responsibilities for the respective tasks within his company through organisational measures. This is independent of the work steps...

• for machine design
• for the creation of the application program for the SafetyController

A staff member responsible for the safety technology (FSM = Functional Safety Manager) should also be appointed. In larger companies this role is assigned to staff members who have the primary responsibility.

▶ Tasks:
• prepare specifications
• prepare safety concept and machine specifications
• prepare functional specifications
• determine or calculate the reliability of the safety function
• document all work steps
• archive the documentation and keep it during the life cycle of the machine

▶ The execution and verification of a task must not be performed by the same staff member!
• create and verify specifications
• create and verify machine specifications
• create and verify the different safety-related function units
• verify the interaction of the function units

The 4-eye principle is typically applied here.

To illustrate this structure in graphic form the V model can be applied. The machine manufacturer adds the corresponding details and responsibilities to each work step of a particular application. The machine manufacturer could also – depending on the work packages – create several organisational units based on the V model.

It is important that this organisational structure is documented and archived.
Organise the creation of safe machinery with the V model

In accordance with the applicable directives and standards:
- adhere to the mechanical and electrical layout of the safety functions when designing the mobile machine
- document the respective steps of the specification and implementation process in a clearly structured way
- make a statement regarding their reliability and the likelihood of a dangerous error

► Archive these documents during the entire life cycle of the mobile machine or machine series!
3.5 Start-up behaviour of the controller

**WARNING**

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

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

A restart can, for example, be caused by:

- Voltage restoration after power failure
- Reset after the watchdog responded because the cycle time was too long
- Error elimination after an E-stop

To ensure safe controller behaviour:

- Monitor the voltage supply in the application program.
- In case of an error switch off all relevant outputs in the application program.
- Additionally monitor actuators which can cause hazardous movements in the application program (feedback).

3.6 Warnings used

**WARNING**

Death or serious irreversible injuries may result.

**CAUTION**

Slight reversible injuries may result.

**NOTICE**

Property damage is to be expected or may result.

Important note
Non-compliance may result in malfunction or interference.

Information
Supplementary note.
3.7 Notes: serial number

► In the user's production facility, draw a diagram of the controller network in the machine. Enter the serial number of each controller installed into the network diagram.

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

This chapter describes the installation of the software components that are necessary to program the CR721S.

4.1  System requirements

4.1.1  Hardware

- Device from the ifm product family ecomatController CR721S
- PC/laptop for CODESYS programming system ([→ Chapter Software > System requirements CODESYS Development System V3.5)
- Ethernet connection between CODESYS PC/laptop and Ethernet interface of the CR721S ([→ installation instructions)

4.1.2  Software

To program the device-internal PLC of the CR721S, the following software components are required:

<table>
<thead>
<tr>
<th>Component of Description</th>
<th>of</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODESYS Development System</td>
<td>3S</td>
<td>V3.5 SP11</td>
</tr>
<tr>
<td>[CR721S]_V2.5.0.n.zip ifm Complete package for ecomatController CR721S, consisting of: Components of the complete package ([→ p. 20)</td>
<td>V2.5.0.n</td>
<td></td>
</tr>
</tbody>
</table>

The features and functions warranted in this manual can only be obtained by using the software components in the versions that are mentioned here. On their website, ifm electronic provide the software components for download: [→ ifm weltweit • ifm worldwide • ifm à l’échelle internationale ([→ p. 209)

4.1.3  Licensing

By buying a controller CR721S, the buyer also purchases a licence that is valid for the use of the CODESYS 3.5 programming system.
4.2 Carry out installation

Contents
CODESYS programming software ................................................................. 19
Complete package for ecomatController CR721S ........................................... 20
Check the operating system version of the device ........................................... 22
Update the operating system of the device .................................................. 24

4.2.1 CODESYS programming software

The CODESYS Development System (short: CODESYS) serves as platform for the creation of PLC applications according to the standard IEC 61131-3.

Install CODESYS Development System

To install the software “CODESYS Development System”:

- For installation on the PC/laptop, administrator rights are required.

- Install the programming system CODESYS V3.5 SP11.
  → CODESYS installation and first steps
- CODESYS V3.5 SP11 is installed on the PC/laptop.
4.2.2 Complete package for ecomatController CR721S

Components of the complete package

To program the device-internal PLC, ifm provides a complete package "CODESYS for ifm R360III Products". The overall package is structured as follows and includes the following components:

<table>
<thead>
<tr>
<th>Data name / path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;CODESYS for ifm R360III Products&quot;.zip</td>
<td>Complete package</td>
</tr>
<tr>
<td>+ CODESYS_Package</td>
<td>Folder</td>
</tr>
<tr>
<td>+ Plugin</td>
<td></td>
</tr>
<tr>
<td>+ CODESYS Safety SIL2 xyz.package</td>
<td>Package &quot;CODESYS Safety SIL2 Plugin&quot;</td>
</tr>
<tr>
<td>+ ifm Safety SIL2 Extensions Vn.n.n.n.package</td>
<td>Package &quot;ifm Safety SIL2 Plugin&quot;</td>
</tr>
<tr>
<td>+ ifm_ecomatController_Vn.n.n.n.package</td>
<td>Package &quot;ecomatController (device description, libraries, etc.)&quot;</td>
</tr>
<tr>
<td>+ Device</td>
<td>Folder</td>
</tr>
<tr>
<td>+ boot</td>
<td></td>
</tr>
<tr>
<td>+ boot.ifm</td>
<td>Bootloader</td>
</tr>
<tr>
<td>+ os</td>
<td>Folder</td>
</tr>
<tr>
<td>+ ifmOS.ifm</td>
<td>Runtime system</td>
</tr>
</tbody>
</table>

The following components must be installed:
- CODESYS Safety SIL2 Plugin
- ifm Safety SIL2 Plugin
- ecomatController package (device description, libraries, etc.)
Installation

Install package (PC/laptop)

To install a package

Requirements

- CODESYS V3.5 SP11 is installed on the PC/laptop.
- ifm package "CODESYS for ifm R360III Products" is stored on the PC/laptop.

1 Start CODESYS
   - Start CODESYS as administrator.
   - CODESYS user interface appears.

2 Start Package Manager
   - Select [Tools] > [Package Manager] to start the Package Manager.
   - Package manager appears.
   - Window shows installed packages.

3 Install package
   - Click on [Install...].
   - The file explorer appears.
   - select the required file*.package and carry out a full installation.
   - The [Package Manager] window shows the installed package.
   - Click on [Close] to quit the Package Manager.
   - Save the project.
   - Close CoDeSys
   - Start CODESYS
   - The installed package is now available.

To install another package, proceed again as described.

Update package (PC/laptop)

To update a package:

1 Uninstall the old version of the package
   - Uninstall package (PC/laptop) (→ p. 22)

2 Install the new version of the package
   - Install package (PC/laptop) (→ p. 21)

3 Update device
   - In the device tree: Mark node [Device (CR721S)].
   - Select [Project] > [Update Device...].
   - 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.
Uninstall package (PC/laptop)

To uninstall a package:

1. **Start package manager**
   - Select [Tools] > [Package Manager] to start the Package Manager.
   - Window [Package Manager] shows installed packages.

2. **Uninstall package**
   - Activate checkbox [Display versions].
   - The window shows the version numbers of the installed packages.
   - Select the package version to be uninstalled and uninstall it with [Uninstall...].
   - Selected package version is uninstalled.
   - Click on [Close] to quit the Package Manager.

4.2.3 **Check the operating system version of the device**

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** (→ p. 63))
- Copy the file \info\swinfo.txt by clicking on [<<] to a local PC drive (→ **Manage files** (→ p. 68)).
- 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.
Check the hardware version of the device

To check the hardware version of the device:

- Connect to PLC (→ Set communication path of PLC (→ p. 63))
- Copy the file \info\devinfo.txt by clicking on [<<] to a local PC drive (→ Manage files (→ p. 68)).
- Open and check devinfo.txt in an editor, e.g. notepad

> devinfo.txt contains the following information:

<table>
<thead>
<tr>
<th>Section</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>Number</td>
<td>Hardware article number</td>
</tr>
<tr>
<td>HARDWARE</td>
<td>Name</td>
<td>Hardware article designation</td>
</tr>
<tr>
<td>MANUFACTURE</td>
<td>LPKnum</td>
<td>Production order number</td>
</tr>
<tr>
<td>MANUFACTURE</td>
<td>LPKerp</td>
<td>ERP material number</td>
</tr>
<tr>
<td>MANUFACTURE</td>
<td>Fabnum</td>
<td>Production order number of the final unit</td>
</tr>
</tbody>
</table>
4.2.4 Update the operating system of the device

**ATTENTION**

Important: During the update process, the electric voltage supply of the device must be assured. The device may only be disconnected from the voltage supply after the data transfer/update process is finished. An incomplete operating system update may destroy the device!

**ATTENTION**

Important: Field test units cannot be updated!

- Before the update process, please ensure:
  - Device is connected to the voltage supply and switched on
  - The Ethernet interface is connected to the same network as the PC

Update the operating system of the device with the batch file

**ATTENTION**

- Strictly follow the instructions on the screen during the entire update process!
  - Otherwise the controller may be destroyed!

To update the operating system of the device with the batch file update.bat:

- Unpack the ZIP file with the batch file update.bat and the corresponding files in a local memory location.
  - Important: The data path in which the batch file is unpacked may not contain any blanks!

- Execute the batch file update.bat
- Follow the instructions on the screen.
  - The [ecomatController Update Start Menu] appears.

  If the device is not set to the standard IP address 192.168.82.247:
  - Use the [i] key to select the menu item [Set device ip-address].
  - Enter the IP address set in the device and confirm with [RETURN].
  - The IP address setting in the batch program has been changed.

  - Use the [P] key to call up the menu item [Ping device]

  The ping command is executed. The device must answer to the ping request to enable an update process.
If ping is successful:

- Use the [0] key to call up the menu item [Continue update process].
- Order number, software and hardware version are read from the device.

  If an update is possible using the read data:
  - The [ecomatController Update Menu] appears.
  - Otherwise: Error message and return to the [ecomatController Update Start Menu].

- Follow the instructions on the screen.
- The updated process is executed.

**ATTENTION**

- After loading the first file cmd.im and after the instruction on the screen, execute a power-on reset of the controller.
- Otherwise the controller may be destroyed!

- After the power-on reset, continue the update process following the instructions on the screen.
- The user is informed about the success of the update process.

  If the update process is finished successfully:
  - Disconnect the voltage supply.
  - Re-connect the voltage supply after the waiting time.
  - The PLC boots with a new operating system.
5 System description

5.1 Hardware description

5.1.1 Hardware structure
Overview: Hardware

System context of the controller

**WARNING**

The safety functionality is in preparation. At present the device has NO safety functionality!

► Do NOT use the device for safety-related functions!

All devices of this controller family can execute both security levels simultaneously:
- PLC for safety-related application (in the figure: yellow areas)
- PLC for standard applications (in the figure: blue areas)

Elements that can belong to both security levels are grey.

---

**Figure: System context of the controller**
System overview

Details:

<table>
<thead>
<tr>
<th>Module</th>
<th>see</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power supply</td>
<td>Device supply (technology) (→ p. 33)</td>
</tr>
<tr>
<td>Inputs</td>
<td>Inputs (technology) (→ p. 38)</td>
</tr>
<tr>
<td>Interfaces</td>
<td>Interfaces</td>
</tr>
<tr>
<td>Memory, ports</td>
<td>Available memory (→ p. 31)</td>
</tr>
<tr>
<td>Processing</td>
<td>Standard PLC and safety PLC</td>
</tr>
<tr>
<td>HMI</td>
<td>Status LEDs</td>
</tr>
<tr>
<td>Outputs</td>
<td>Outputs (technology) (→ p. 44)</td>
</tr>
</tbody>
</table>
Block diagram of the supply and of the output deactivation

Figure: Block diagram of the supply

Note on wiring

Group designations

Inputs and outputs are assigned in groups.
Input groups are required for 2-channel safety-related inputs.
The identifier of an input or output results from the following principle:
1. Type (IN, OUT)
2. Group number (00...18)
3. Channel number (00...08)
Examples:
- IN0002 = input | group 00 | channel 02 in this group
- OUT0507 = output | group 05 | channel 07 in this group

In case of ready-to-use multipoint connectors, both the corresponding identifier and the corresponding pin number is indicated on each individual wire.
Standard PLC and safety PLC

⚠️ WARNING
The safety functionality is in preparation. At present the device has NO safety functionality!
▶ Do NOT use the device for safety-related functions!

The device features separate controllers:
• for standard functions
• for safety-related functions (safety)

⚠️ Before the programming of the application may even begin:
▶ distribute the resources to both PLCs (Chapter Configure PLC (→ p. 66)). (system, inputs, outputs, user LEDs)
Available memory

Memory allocation

IEC61131-1 divides the memory for the storage of the user data into:
- memory of the applications (parts are configurable)
  - IEC code non-safe
  - IEC code safe
- 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 in one data format)
- IEC memory bytes (permanent storage of application-specific data at application-specific addresses)

⚠️ WARNING
The safety functionality is in preparation.
At present the device has NO safety functionality!
► Do NOT use the device for safety-related functions!

<table>
<thead>
<tr>
<th>FLASH memory</th>
<th>RAM</th>
<th>remanent memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>physical: 9.0 Mbytes available: 6.0 Mbytes</td>
<td>physical: 10 Kbytes available: 10 Kbytes</td>
</tr>
<tr>
<td>IEC code (safe)</td>
<td>IEC RAM (configurable)</td>
<td>IEC retain (safe) (2.5 Kbytes)</td>
</tr>
<tr>
<td>(configurable)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IEC code (non-safe)</td>
<td>IEC RAM (configurable)</td>
<td>IEC memory bytes (safe) (2.5 Kbytes)</td>
</tr>
<tr>
<td>(configurable)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>user files</td>
<td>IEC retain (configurable)</td>
<td>IEC memory bytes (non-safe) (2.5 Kbytes)</td>
</tr>
<tr>
<td>(1.0 Mbytes)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table: memory areas
Memory allocation variants

⚠️ WARNING
The safety functionality is in preparation. At present the device has NO safety functionality!
► Do NOT use the device for safety-related functions!

The user can select from the pre-defined configurations of the memory partitioning. The configuration enables optimum separation between safety-relevant application and standard application.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Memory allocation (in bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IEC code safe</td>
</tr>
<tr>
<td>Configuration 1</td>
<td>1.0 Mbytes</td>
</tr>
<tr>
<td>Configuration 2 (preset)</td>
<td>2.0 Mbytes</td>
</tr>
<tr>
<td>Configuration 3</td>
<td>3.0 Mbytes</td>
</tr>
<tr>
<td>Configuration 4</td>
<td>4.0 Mbytes</td>
</tr>
</tbody>
</table>

Table: configurable memory partitioning

Description of the allocation of the memory partitions in CODESYS: → Allocate memory partition
(→ p. 66)
5.1.2 Device supply (technology)

Contents

Voltage ranges of the on-board system ................................................................. 33
Start conditions .......................................................................................................... 33
Switch on/off via main switch ...................................................................................... 34
Switch on/off via ignition lock (Terminal 15) .............................................................. 35

Voltage ranges of the on-board system

The system monitors the voltage ranges of the on-board system. The voltages mentioned here apply to the specified range ± 1 % (at 36 V).

<table>
<thead>
<tr>
<th>Voltage [V]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td>to</td>
</tr>
<tr>
<td>&lt; +5.5</td>
<td>Undervoltage VBB15, VBB30: If the controller was in the OPERATING mode, the controller shuts down.</td>
</tr>
<tr>
<td>+5.5</td>
<td>&lt; 8.0</td>
</tr>
<tr>
<td>+8.0</td>
<td>+32.0</td>
</tr>
<tr>
<td>&gt; +32.0</td>
<td>+36.0</td>
</tr>
<tr>
<td>&gt; +36.0</td>
<td>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!</td>
</tr>
</tbody>
</table>

Start conditions

The device only boots when sufficient voltage is applied to the power supply connection VBB30 and to VBB15 (= terminal 15). In vehicles clamp 15 is the plus cable switched by the ignition lock. This voltage must be provided by the on-board system of the mobile machine. ➔ chapter Monitoring concept (➔ p. 36)
Switch on/off via main switch

To do so: VBB15 is connected with VBB30

Procedure when switching on the main switch

> The system recognises the applied voltage (VBB15 > 5 V AND VBB30 > 8 V) and activates the connection of the controller to the VBB30 potential via solid-state switch.

> The controller boots and starts.

Procedure when switching off the main switch:

> Running tasks will continue till the end.
  IMPORTANT: The maximum length of tasks is 50 ms!
  Tasks that run for longer will be aborted before the end of the task.

> the system automatically stores the retain data
  the input signals are no longer read
  the outputs are switched off.

> the system switches off completely

If this behaviour is not wanted, use the circuit via ignition lock:
Switch on/off via ignition lock (Terminal 15)

To do so:
- Connect the VBB15 via the ignition lock (= vehicle terminal 15 *) with the vehicle plus pole.
- Connect VBB30 directly with the vehicle plus pole (= vehicle terminal 30).

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

Procedure as with switch-on via ignition lock:
- The ignition lock applies voltage to VBB15 (= vehicle terminal 15 *).
- The system recognises the applied voltage (VBB15 > 5 V AND VBB30 > 8 V) and activates the connection of the controller to the VBB30 potential via solid-state switch.
- The ignition lock is bypassed. Latching of the control voltage is established.
- The controller boots and starts.

Procedure for switching off via ignition lock:
- Evaluate VBB15 in the application via FB SystemSupply (→ p. 138) if VBB15 < 5 V:
  - execute necessary actions (e.g.):
    - stop machine gently
    - transmit required data
    - save required data and close
  - set the the input xSwitchOff of the FB SupplySwitch (→ p. 136) to TRUE
  - running tasks continue till the end.
  - stop the application
  - the system automatically stores the retain data the input signals are no longer read the outputs are switched off.
  - the system lifts the latching via VBB30:
  - the system switches off completely
5.1.3 Monitoring concept

The controller monitors the supply voltage for overvoltage and undervoltage. In case of undervoltage, the controller switches off.

Monitoring and securing mechanisms

Switch off outputs via solid-state switch

⚠️ WARNING
Danger due to unintentional deactivation of all outputs!
If monitoring routines detect a system error:
> The device switches off the energy for all outputs of the affected output groups.
→ chapter Output groups
→ chapter List of outputs (→ p. 50)

During the program process, the output switches are under the user's full software control. For further safety, the corresponding applicable national regulations must be complied with.

If an error occurs during the program sequence, it is possible to disconnect the output switches from voltage via the FB OutputGroup in order to separate critical plant sections.

⚠️ WARNING
Danger due to unintentional and dangerous start of machine or plant sections!
► When creating the program, the programmer must ensure that no unintentional and dangerous start of machines or plant sections after a fault (e.g. e-stop) and the following fault elimination can occur!
⇒ Realise restart inhibit.

► In case of an error, set the outputs concerned to FALSE in the program!
Hardware description

Watchdog

The watchdog has multiple levels:

- IEC task-related watchdog
  This watchdog works in the ifm operating system and is executed in each CPU core. Each 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 change to logic "0".
  Error class = A

→ chapter Error classes

To eliminate the fault:

► Rebooting the PLC is necessary via voltage on/off.

Configure IEC watchdog

![Exclamation mark]

- Familiarise yourself with the following CODESYS functions!
  - Watchdog:
    → Online help > CODESYS Development System > Programming Applications > Task Configuration > Creating a Task Configuration > Tab 'Configuration'
  - Task configuration:
    → Online help > CODESYS Development System > Programming Applications > Task Configuration

To configure the IEC watchdog of a task:

► Open task configuration (→ Configure task processing (→ p. 82))
► Activate watchdog with option field [Enable]
► Enter watchdog [Time]
► Set [Sensitivity]
  > Watchdog is configured

![Exclamation mark]

The watchdog time must be shorter than the interval time.
The watchdog time must be longer than the runtime of the task.
5.1.4 Inputs (technology)

Contents

Types of inputs .......................................................................................................................... 38
List of inputs ................................................................................................................................. 42

Types of inputs

Contents

Binary input block diagram plus/minus-switching ................................................................. 38
Input type IN MULTIFUNCTION-A ............................................................................................ 39
Input type IN FREQUENCY-A/B ............................................................................................... 39
Input type IN RESISTOR-A ......................................................................................................... 40
Input type IN DIGITAL-A ............................................................................................................ 40
Input type IN DIGITAL-B ............................................................................................................. 41

We differentiate between the following input types:

Binary input block diagram plus/minus-switching

Binary input block diagram, plus-switching (B+) for positive sensor signal:
Input = open ⇒ Signal = Low (GND)

CSI = Current Sinking
In = connection of binary input n
(S) = sensor

Binary input block diagram, minus-switching (B−) for negative sensor signal:
Input = open ⇒ Signal = High (Supply)

CSO = current sourcing
In = connection of binary input n
(S) = sensor
Input type IN MULTIFUNCTION-A

Binary and analogue inputs

Block diagram
IN MULTIFUNCTION-A

IN#### = Connection of multi-function input no. ####
(CR) = device
(1) = input filter
(2) = analogue current measurement
   \( R_i = 300 \, \Omega \)
(3) = binary input, plus-switching
   \( R_E = 10 \, k\Omega \)
(4) = analogue voltage measurement
   \( 0...10 \, V / 0...32 \, V \)
(5) = measured value
(6) = reference voltage for ratio

Configure input → Chapter System configuration

Possible operating modes:
- Binary input CSI (BL) \((R_E = 10 \, k\Omega)\) or Namur
- Analogue current measurement 0...20 mA
- Analogue voltage measurement 0...10 V
- Analogue voltage measurement 0...32 V
- Analogue voltage measurement, ratiometric to the reference voltage

Input type IN FREQUENCY-A/B

Binary and fast inputs

Block diagram
IN FREQUENCY-A/B

IN#### = Connection of frequency / counting input no. ####
(CR) = device
(1) = input filter
(2) = CSI binary input
   plus-switching
(3) = CSO binary input
   minus-switching
(4) = measured value (analogue)
(5) = measured value (binary)

Configure input → Chapter System configuration

Possible operating modes:
- Binary input CSI (BL) \((R_E = 10 \, k\Omega)\) or Namur
- Binary input CSO (BH)
- Analogue voltage measurement 0...10 V (only for input type IN FREQUENCY-B)
- Pulse measurement CSI (BL) (frequency measurement, ratio measurement, pulse counter)
- Pulse measurement CSO (BH) (frequency measurement, ratio measurement, pulse counter)
Input type IN RESISTOR-A

Binary inputs and resistance measurement

Possible operating modes:
- Binary input CSI (B_L) (R_E = 10 kΩ) or Namur
- Resistance measurement 0...30 kΩ

Input type IN DIGITAL-A

Binary inputs

Possible operating modes:
- Binary input CSI (B_L) (R_E = 10 kΩ) or Namur
Input type IN DIGITAL-B

Binary inputs

Block diagram
IN DIGITAL-B
IN#### = Connection of binary input no. ####
(CR) = device
(1) = input filter
(2) = measured value (analogue)

Input not configurable

Possible operating modes:
- Binary input CSI (BL) (R_E = 3.3 kΩ) or Namur
## List of inputs

<table>
<thead>
<tr>
<th>IEC identifier</th>
<th>Input type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN0000</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN0001</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN0002</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN0003</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN0100</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0101</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0102</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0103</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0200</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0201</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0202</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0203</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0300</td>
<td>IN Digital-B 3.2k</td>
</tr>
<tr>
<td>IN0301</td>
<td>IN Digital-B 3.2k</td>
</tr>
<tr>
<td>IN0400</td>
<td>IN Resistor-A</td>
</tr>
<tr>
<td>IN0401</td>
<td>IN Resistor-A</td>
</tr>
<tr>
<td>IN0500</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN0501</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN0502</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN0503</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN0600</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0601</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0602</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0603</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0700</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0701</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0702</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0703</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN0800</td>
<td>IN Digital-B 3.2k</td>
</tr>
<tr>
<td>IN0801</td>
<td>IN Digital-B 3.2k</td>
</tr>
<tr>
<td>IN0900</td>
<td>IN Resistor-A</td>
</tr>
<tr>
<td>IN0901</td>
<td>IN Resistor-A</td>
</tr>
<tr>
<td>IN1000</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN1001</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN1002</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN1003</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN1100</td>
<td>IN Multifunction-A</td>
</tr>
</tbody>
</table>
### System description

#### Hardware description

<table>
<thead>
<tr>
<th>IEC identifier</th>
<th>Input type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN1101</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN1102</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN1103</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN1200</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1201</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1202</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1203</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1300</td>
<td>IN Digital-B</td>
</tr>
<tr>
<td>IN1301</td>
<td>IN Digital-B</td>
</tr>
<tr>
<td>IN1302</td>
<td>IN Digital-B</td>
</tr>
<tr>
<td>IN1303</td>
<td>IN Digital-B</td>
</tr>
<tr>
<td>IN1400</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1401</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1402</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1403</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1500</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN1501</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN1502</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN1503</td>
<td>IN Frequency-A</td>
</tr>
<tr>
<td>IN1600</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN1601</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN1602</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN1603</td>
<td>IN Multifunction-A</td>
</tr>
<tr>
<td>IN1700</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1701</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1702</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1703</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1800</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1801</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1802</td>
<td>IN Digital-A</td>
</tr>
<tr>
<td>IN1803</td>
<td>IN Digital-A</td>
</tr>
</tbody>
</table>
5.1.5 Outputs (technology)

**Contents**

Output types .......................................................... 44
List of outputs ....................................................... 50

**Output types**

**Contents**

Binary output block diagram plus/minus-switching ............................................. 44
Output type OUT PWM-n-A ............................................ 45
Output type OUT PWM-n-B ............................................ 46
Output type OUT PWM-n-BRIDGE-A .................................. 47
Output type OUT Supply-A ........................................... 48
Output type OUT Voltage-A .......................................... 49

We differentiate between the following output types:

**Binary output block diagram plus/minus-switching**

![Diagram](image_url)

Output block diagram plus-switching (B+) for positive output signal

Output block diagram, minus-switching (B-) for negative output signal

CSO = current sourcing  
Qn = connection of output n  
(L) = load

CSI = current sinking  
Qn = connection of output n  
(L) = load
Output type OUT PWM-n-A

n = current rating
Example: n = 25 \(\Rightarrow\) \(I_{\text{max}} = 2.5\) A

Binary 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) = device
(1) = measured value (analogue)
(2) = measured value (binary)

Configure input → Chapter System configuration

Possible operating modes:
- Binary output CSO (BH)
- analogue output CSO with pulse width modulation (PWM_h)
- analogue output CSO with pulse width modulation, current-controlled (PWM_i)

Setting and measurement via:
- FB Output (→ p. 131) for binary output
- FB PWM1000 (→ p. 160) for PWM
- FB CurrentControl (→ p. 157) for current control (PWM_i)
Output type OUT PWM-n-B

n = current rating
Example: n = 25  \( \Rightarrow \)  \( I_{\text{max}} = 2.5 \) A

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

Configure input → Chapter System configuration

Possible operating modes:
- Binary output CSO (B\(_{\text{H}}\)) with restricted current measurement
- analogue output CSO with pulse width modulation (PWM\(_{\text{H}}\)), without current measurement

Setting and measurement via:
- Function block **Output** (→ p. 131) for binary output
- Function block **PWM1000** (→ p. 160) for PWM
Output type OUT PWM-n-BRIDGE-A

n = current rating
Example: n = 25 ⇒ Imax = 2.5 A

Binary output or analogue output with pulse width modulation (PWM), optionally current-controlled (PWM_i) or bridge output (via PWM)

Possible operating modes:
- Binary output CSO (B_H)
- Binary output CSI (B_L)
- analogue output CSO with pulse width modulation (PWM_H), current controlled (PWM_i)
- analogue output CSI with pulse width modulation (PWM_L)
- analogue output CSO with pulse width modulation, current-controlled (PWM_i)
- Pair of outputs as bridge with pulse width modulation (PWM)

Setting and measurement via:
- FB Output (→ p. 131) for binary output
- FB PWM1000 (→ p. 160) for PWM
- FB CurrentControl (→ p. 157) for current control (PWM_i)
- FB HBridge (→ p. 152) for bridge output
Output type OUT Supply-A

The output OUT3000 is used to supply sensors with a stable voltage (5 V or 10 V) that is not affected by fluctuations of the supply voltage.

<table>
<thead>
<tr>
<th>NOTICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference voltage output can get damaged!</td>
</tr>
<tr>
<td>► Do NOT apply any external voltage!</td>
</tr>
</tbody>
</table>

Setting and measurement via FB Output (→ p. 131) or via system configuration:

Setting / measurement via system configuration

Setting the reference voltage:
► In the device tree, select [Local_IO] > [Outputs] > tab [Parameter] > [OUT3000]
► Activate the required list element in the column [Value]:
  for 5 V: [OUT_SENSOR_05] or
  for 10 V: [OUT_SENSOR_10]

Monitoring of the values at the reference voltage output:
► In the device tree, select [Local_IO] > [Outputs] > tab [IO-Mapping] > [OUT3000_I]
  > [OutVoltageDiag] indicates the measured voltage in [mV]
  [OutCurrentDiag] indicates the measured voltage in [mA]

Setting / measurement via FB Output

Setting the reference voltage:
► Use the inputs in the FB output as follows:
  [uiChannel] = 3000
  [eMode] = [OUT_SENSOR_05] (for 5 V) or
  [eMode] = [OUT_SENSOR_10] (for 10 V)

Monitoring of the values at the reference voltage output:
► Read the outputs in the FB output as follows:
  > [uiOutVoltage] indicates the measured voltage in [mV]
  [uiOutCurrent] indicates the measured current in [mA]

Details → FB Output (→ p. 131).
Output type OUT Voltage-A

The output provides 0...10 V e.g. for further controllers or actuators.
M3071n / CR071n: only OUT3001
M3072n / CR072n: OUT3001 and OUT3002
The output is protected against overload and automatically switches off if overloaded.

NOTICE
Reference voltage output can get damaged!
► Do NOT apply any external voltage!

Setting and measurement via FB Output or via system configuration:

Setting / measurement via system configuration

Setting the reference voltage:
► Select [Local_IO] > [Outputs] > tab [Parameter] > [OUT3001 / OUT3002] in the device tree
► Enter the required value [V] in the column [Value]
  permissible = 0...10

Monitoring of the values at the reference voltage output:
► Select [Local_IO] > [Outputs] > tab [IO-Mapping] > [OUT3001_I / OUT3002_I] in the device tree
  > [OutVoltageDiag] indicates the measured voltage in [mV]

Setting / measurement via FB Output

Setting the reference voltage:
► Use the inputs in the FB output as follows:
  [uiChannel] = 3001 / 3002
  [uiValue] = required voltage in [mV]
  permissible = 0...10000

Monitoring of the values at the reference voltage output:
► Read the outputs in the FB output as follows:
  > [uiOutVoltage] indicates the measured voltage in [mV]
  [uiOutCurrent] indicates the measured current in [mA]

Details → FB Output (→ p. 131).
## List of outputs

<table>
<thead>
<tr>
<th>IEC identifier</th>
<th>Output type</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT0000</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0001</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0002</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0003</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0004</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0005</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0006</td>
<td>OUT PWM-40-Bridge-A</td>
</tr>
<tr>
<td>OUT0007</td>
<td>OUT PWM-40-Bridge-A</td>
</tr>
<tr>
<td>OUT0008</td>
<td>OUT PWM-40-A</td>
</tr>
<tr>
<td>OUT0100</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0101</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0102</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0103</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0104</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0105</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0106</td>
<td>OUT PWM-40-Bridge-A</td>
</tr>
<tr>
<td>OUT0107</td>
<td>OUT PWM-40-Bridge-A</td>
</tr>
<tr>
<td>OUT0108</td>
<td>OUT PWM-40-A</td>
</tr>
<tr>
<td>OUT0200</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0201</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0202</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0203</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0204</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0205</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0206</td>
<td>OUT PWM-40-Bridge-A</td>
</tr>
<tr>
<td>OUT0207</td>
<td>OUT PWM-40-Bridge-A</td>
</tr>
<tr>
<td>OUT0208</td>
<td>OUT PWM-40-A</td>
</tr>
<tr>
<td>OUT0300</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0301</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0302</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0303</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0304</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0305</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0306</td>
<td>OUT PWM-40-Bridge-A</td>
</tr>
<tr>
<td>OUT0307</td>
<td>OUT PWM-40-Bridge-A</td>
</tr>
<tr>
<td>OUT0308</td>
<td>OUT PWM-40-A</td>
</tr>
<tr>
<td>OUT0400</td>
<td>OUT PWM-25-A</td>
</tr>
</tbody>
</table>
### System description

<table>
<thead>
<tr>
<th>IEC identifier</th>
<th>Output type</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT0401</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0402</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0403</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0404</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0405</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0406</td>
<td>OUT PWM-40-Bridge-A</td>
</tr>
<tr>
<td>OUT0407</td>
<td>OUT PWM-40-Bridge-A</td>
</tr>
<tr>
<td>OUT0408</td>
<td>OUT PWM-40-A</td>
</tr>
<tr>
<td>OUT0500</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0501</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0502</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0503</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0504</td>
<td>OUT PWM-25-A</td>
</tr>
<tr>
<td>OUT0505</td>
<td>OUT PWM-25-B</td>
</tr>
<tr>
<td>OUT0506</td>
<td>OUT PWM-40-Bridge-A</td>
</tr>
<tr>
<td>OUT0507</td>
<td>OUT PWM-40-Bridge-A</td>
</tr>
<tr>
<td>OUT0508</td>
<td>OUT PWM-40-A</td>
</tr>
<tr>
<td>OUT3000</td>
<td>OUT Supply-A</td>
</tr>
<tr>
<td>OUT3001</td>
<td>OUT Voltage-A</td>
</tr>
<tr>
<td>OUT3002</td>
<td>OUT Voltage-A</td>
</tr>
</tbody>
</table>
5.1.6 Feedback in case of externally supplied outputs

⚠️ Do not apply any external voltage to the outputs!

> As soon as output group switch \( \text{VBBn}\_\text{SW}\_\text{Q} = \text{FALSE} \):
   The internal device monitoring checks the voltage on the contact bar after the output group switch. If then a voltage of \( > 0.4 \text{ VBBn} \) is measured:
   • the controller reports error class C,
   • the controller switches the group to the safe state.

Safe state of the group = all outputs are switched off
• all outputs will be switched off
• the controller reports the error to the IEC application

To reboot the device:
► remove the error cause
► do a power-on reset.

OR error handling in the IEC application:
► remove the error cause
► Remove error of the group via \text{xResetError}.

Figure: Example of wiring with blocking diodes due to the risk of feedback

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

Successful:
If \( \text{VBBn}\_\text{SW}\_\text{Q} = \text{FALSE} \), the controller does not go to error class C when the S1 contact is closed.

NOTE
Help for externally connected outputs
► Decouple the externally connected outputs by means of diodes so that no external voltage can be connected to the output terminal of the controller!
5.2 Interfaces

The device includes the interfaces described in the following.

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

5.2.1 Serial interface

This device features a serial interface. The serial interface can generally be used in combination with the following functions:
• program download
• debugging
Connections and data → data sheet

5.2.2 Ethernet interface

NOTICE
Only use the Ethernet interface in a secure network environment (e.g. separate network or VPN)!
Otherwise, unauthorised persons can read or manipulate data or tamper with the functions of the device.

This device features an Ethernet interface with 2 ports via an internal switch. This enables line wiring between several devices.

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 → data sheet
5.2.3 CAN: Interfaces and protocols

⚠️ Familiarise yourself with the following CODESYS functions!
- CAN-based fieldbuses
  → Online help > Fieldbus support > CAN-based fieldbuses

The device has 4 CAN interfaces. Each CAN interface supports the following protocols:
- RawCAN (CAN Layer 2)
- CANopen Manager
- CANopen Device
- CANopen Safety Manager
- CANopen Safety Device
- J1939 Manager

⚠️ WARNING
The safety functionality is in preparation.
At present the device has NO safety functionality!
⚠️ Do NOT use the device for safety-related functions!
5.3 Software description

Overview: Software ................................................................. 55
Software module for the device ............................................. 57

5.3.1 Overview: Software

We differentiate between the following software components:

Software on the PC/notebook.

The programming environment CODESYS Development System is installed in the PC/notebook to create and debug both applications. The controller supports service and maintenance via CODESYS or via other tools.

The CODESYS functions are extended with Config plugins. Thereby, additional setting options for memory and inputs/outputs become available. ifm electronic provides adequate device descriptions for the CODESYS Development System for each derivative. IEC libraries for the safe and non-safe applications provide CODESYS and the programmer with access options to the functions of the controller.
Software in the controller

The controller processes the applications by means of several software components. The ifm operating system with the CODESYS CTRL-SIL2 and the CODESYS CTRL constitutes, among other things, the runtime environment that executes both applications. The real-time operating system enables separate execution of the safe and the non-safe software components in the controller.

⚠️ WARNING

The safety functionality is in preparation. At present the device has NO safety functionality!
► Do NOT use the device for safety-related functions!

Using the configuration file comconf.cfg, the programmer can control the interface. The programmer can store files and directories on the controller and use them in the application. Or the applications themselves create files and store them in the controller. The bootloader is a fallback level for cases in which the ifm operating system is no (longer) available or corrupt.
5.3.2 Software module for the device

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

<table>
<thead>
<tr>
<th>Software module</th>
<th>Can the user change the module?</th>
<th>Using what?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>yes</td>
<td>CODESYS, Tools for service and maintenance</td>
</tr>
<tr>
<td>with libraries</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a) for standard PLC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b) for safety PLC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ifm operating system</td>
<td>Upgrade yes Downgrade no</td>
<td>CODESYS, Tools for service and maintenance</td>
</tr>
<tr>
<td>Bootloader</td>
<td>no</td>
<td>---</td>
</tr>
<tr>
<td>(Hardware)</td>
<td>no</td>
<td>---</td>
</tr>
</tbody>
</table>

We describe this software module in the following:

**Bootloader**

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

⚠️ Only execute the bootloader update when explicitly requested by ifm!

**Operating system**

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

→ Chapter Software module for the device

The device is supplied with the installed operating system. Verifying and changing the operating system version → Chapter Check the operating system version of the device (→ p. 22)

The operating system only needs to be downloaded once - if at all. The application can then be loaded (also several times) in a PLC without affecting the operating system.

The operating system can be downloaded from ifm electronic gmbh’s website:

→ ifm weltweit • ifm worldwide • ifm à l’échelle internationale (→ p. 209)
Application

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

⚠️ WARNING

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

Libraries

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifmCANopenManager</td>
<td>Functions for use of the CAN interfaces as CANopen Manager</td>
</tr>
<tr>
<td>ifmDeviceCR721S</td>
<td>Data structures, enumeration types and global variables</td>
</tr>
<tr>
<td>ifmFastInput</td>
<td>Functions to access the fast inputs of the device</td>
</tr>
<tr>
<td>ifmIOcommon</td>
<td>Functions for access to the inputs and outputs of the device</td>
</tr>
<tr>
<td>ifmIOconfigDiagProt</td>
<td>Functions to configure the I/O-related diagnostic and protective functions</td>
</tr>
<tr>
<td>ifmOutGroup</td>
<td>Functions to control output group switches</td>
</tr>
<tr>
<td>ifmOutHBridge</td>
<td>Functions to access H-bridge outputs</td>
</tr>
<tr>
<td>ifmOutPWM</td>
<td>Functions to access PWM outputs</td>
</tr>
<tr>
<td>ifmRawCAN</td>
<td>Functions for use of the CAN interfaces as CAN Layer 2</td>
</tr>
<tr>
<td>ifmSysInfo</td>
<td>Functions to set / read system information</td>
</tr>
<tr>
<td>ifmTypes</td>
<td>Global types and interfaces for other ifm libraries</td>
</tr>
</tbody>
</table>

Detailed information about the ifm function libraries: → ifm function libraries (→ p. 105)
This chapter contains information about the first steps to program the device with CODESYS.

6.1 Start CODESYS

Requirements

> Software components are correctly installed (→ Installation).

Start CODESYS

➤ Double-click on [V3.5 SP11] symbol
➤ CODESYS starts.
➤ CODESYS user interface appears.

6.2 Create CODESYS project

⚠️ Familiarise yourself with the following CODESYS functions!

- Create a project
  → Online help > CODESYS Development System > Creating and Configuring a Project
- Manage a project
  → Online help > CODESYS Development System > Protecting and Saving the Project

ifm electronic provides a special template for each model of the device family. The user can select the corresponding template when the project is created.
6.2.1 Create new project with CR721S

Requirements

1. ifm package "CODESYS for ifm R360III Products" has been correctly installed (→ Installation).

1. Create new CR721S project
   - Select [File] > [New Project...].
   - The window [New Project] appears.
   - Set the following values:
     1. [Templates]: Select the device project template, e.g. [CR0721 Project]
     2. [Name]: Enter the project name
     3. [Location]: Select storage location for the project file
   - Click on [OK] to adopt the selected values.
   - CODESYS creates a new CR721S project.
   - The window [Devices] shows the device tree of the project (→ Overview: Project structure with CR721S).

2. Save the project
   - Select [File] > [Save Project].
   - CODESYS saves the project.
6.2.2 Overview: Project structure with CR721S

A CODESYS project contains all components for configuration, management and programming of the CR721S. All components of a project are shown in the window [Devices] in a hierarchic tree view.

CODESYS projects with a CR721S have the following structure:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifm_CR721S_Root (ifm CR721S Root)</td>
<td>provides access to the settings of the CR721S</td>
</tr>
<tr>
<td></td>
<td>→ Configure PLC</td>
</tr>
<tr>
<td>SafetyPLC (ifm CR721S SIL2)</td>
<td>Content of the PLC called &quot;SafetyPLC&quot;</td>
</tr>
<tr>
<td>StandardPLC (ifm CR721S Standard)</td>
<td>Content of the PLC called &quot;StandardPLC&quot;</td>
</tr>
<tr>
<td>PLC logic</td>
<td>contains the applications of the CR721S</td>
</tr>
<tr>
<td></td>
<td>→ Objects of a PLC application (→ p. 79)</td>
</tr>
<tr>
<td>System info</td>
<td>provides access to the device information</td>
</tr>
<tr>
<td></td>
<td>→ Display system information (→ p. 104)</td>
</tr>
<tr>
<td>Local_IO</td>
<td>provides access to the configuration options of the inputs and outputs</td>
</tr>
<tr>
<td></td>
<td>→ Configure inputs and outputs (→ p. 69)</td>
</tr>
<tr>
<td>HMI</td>
<td>provides access to the configuration options of the operating and display elements</td>
</tr>
<tr>
<td>Communication</td>
<td>provides access to the configuration options of the communication interfaces</td>
</tr>
<tr>
<td></td>
<td>→ Configure CAN interfaces (→ p. 72)</td>
</tr>
</tbody>
</table>

The programmer can adjust the terms in the structure before the expression in brackets:

► Right mouse click on the term > [Properties...]

> The window [Properties] appears > tab [Common]

► enter a term

► confirm with [OK]
6.3 Use CODESYS user manual

This manual only describes the integration, configuration and the programming of the CR721S using the CODESYS development system.

For the description of user actions and user interface elements the CODESYS terminology will be used.

Standard functions and methods of CODESYS will not be described. At the beginning of each section there will be a reference to the corresponding chapters of the CODESYS online help.

To access the online help of the CODESYS development system:

► Start CODESYS.
► The CODESYS user interface appears.
► Press [F1].
► Online help of the CODESYS development system appears.

⚠ Familiarise yourself with the CODESYS development system! In particular with the following topics:
  - Names and functions of the user interface elements
  - Basic menu functions
  - Programming techniques and methods for data retention
6.4 Configure programming interface

Programming of the device-internal PLC is made via the Ethernet interface of the device (position of the connections: → Installation instructions).

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! (→ Installation instructions).
- For connection in the network, an experienced user or system administrator should set up the network addresses and do the configuration.

NOTICE

Only use the Ethernet interface in a secure network environment (e.g. separate network or VPN)! Otherwise, unauthorised persons can read or manipulate data or tamper with the functions of the device.

6.4.1 Set communication path of PLC

To configure the communication path between the programming system CODESYS and the device-internal PLC:

Preparations
> CODESYS PC/laptop and Ethernet interface of the device are connected.
> Optional: Adjust IP settings of the Ethernet interface.

1 Select communication settings
> In the device tree: Double-click on symbol [Device (CR721S)]
> In the editor window: Select tab [Communication].
> Editor window shows communication settings.

2 Select gateway
> Select the requested gateway in the list [Gateway].
> List shows selected gateway.

3 Set communication path
> Activate [Scan Network …].
> Window [Select Device] appears.
> Select gateway node and start scan process with [Scan network].
> CODESYS scans network for devices.
> Window shows network path and detected devices.
> Select node of the device and activate [OK] to set the communication path to the device-internal PLC.
> CODESYS can transfer data to the device-internal PLC.
6.5 Add ifm function libraries to the application

Familiarise yourself with the following CODESYS functions!
- Library manager
  → Online help > CODESYS Development System > Managing Libraries > Adding a library to the application

The ifm package includes function libraries for the programming of the device under CODESYS. The libraries are installed in CODESYS together with the ifm package. The user can add the libraries individually to an application he/she needs for the programming.

By means of the container library ifmR360-3.library, the user can add all functions available for the device to the project.

To integrate a library into a project:

Requirements:
- ifm package is correctly installed (→ Install package (PC/laptop) (→ p. 21)).

Load container library
- In the device tree: Double-click on [PLC Logic] > [Application] > [Library Manager]
- Editor window shows table of added libraries.
- Click on [Add library].
- Select requested library and click on [OK] to add the selected library to the application.
- CODESYS adds the selected library to the project.
- Editor window shows the library.
6.6 Activate the access protection for a project

► Familiarise yourself with the following CODESYS functions!
  ▪ Protect and save project
    → Online help > CODESYS Development System > Protect and save project

The user can use a password to protect the device from unauthorised access.
► Select [Project] > [Project Settings...].
  > Window [Project Settings] appears.
  ► Select [Security].
  ► Activate checkbox [Enable project file encryption].
  ► 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  Configure PLC

**WARNING**
The safety functionality is in preparation.
At present the device has NO safety functionality!
► Do NOT use the device for safety-related functions!

7.1.1  Allocate memory partition

**WARNING**
The safety functionality is in preparation.
At present the device has NO safety functionality!
► Do NOT use the device for safety-related functions!

Further information: → *Memory allocation variants* (→ p. 32)

To allocate the memory partitions to the PLCs:

1  Select memory partition
   ► In the device tree: Double-click on symbol [Device (CR721S)]
   ► In the editor window: Select [Memory Layout] tab.
   > The editor window shows the partitioning of the memory:

<table>
<thead>
<tr>
<th>Memory layout</th>
<th>Partitioning Safety PLC / PLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemoryLayout_4s_1</td>
<td>4 MB / 1 MB</td>
</tr>
<tr>
<td>MemoryLayout_3s_2</td>
<td>3 MB / 2 MB</td>
</tr>
<tr>
<td>MemoryLayout_2s_3</td>
<td>2 MB / 3 MB</td>
</tr>
<tr>
<td>MemoryLayout_1s_4</td>
<td>1 MB / 4 MB</td>
</tr>
</tbody>
</table>

2  Set memory partitioning
   ► Highlight the required memory partition
   ► Click on the [Update Devices] button
   > Memory partitioning is adopted in CODESYS

3  Load memory partition into the device
   ► Click on the [Download Configuration] button
   > Memory partition is downloaded to the device
7.1.2 Allocate inputs/outputs

**WARNING**
The safety functionality is in preparation. At present the device has NO safety functionality!

- Do NOT use the device for safety-related functions!

To allocate the I/Os to the PLCs:

⚠️ Before the programming of the application may even begin:

1. **Select I/O allocation**
   - In the CODESYS device tree: Double-click on symbol [Device (CR721S)]
   - In the editor window: Select [I/O Assignment] tab.
   - Editor window shows the PLC allocation of the inputs/outputs (excerpt):

<table>
<thead>
<tr>
<th>Section</th>
<th>Element</th>
<th>Parameter</th>
<th>Standard PLC</th>
<th>Safety PLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>System info</td>
<td>IP Settings</td>
<td>--</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Local_IO</td>
<td>Inputs</td>
<td>IN0000</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN0001</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN0002</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>Outputs</td>
<td>OUT0000</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT0001</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT0002</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>System_Outputs</td>
<td>VBB0_SW</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>VBB1_SW</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>Supply_Switch</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>HMI</td>
<td>User_LEDs</td>
<td>User LED 0</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User LED 1</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User LED 2</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User LED 3</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

2. **Set I/O allocation**
   - Highlight all I/Os in the column [StandardPLC] to allocate it to the standard PLC
   - Highlight all I/Os in the column [SafetyPLC] to allocate it to the safety PLC
   - The I/Os are allocated
7.1.3 Manage files

To transfer files between PC and device:

1 Select file view
   - In the device tree: Double-click on symbol [Device (CR721S)]
   - In the editor window: Select the [Files] tab.
   - The editor window shows the file structure on the PC on the left and on the device on the right

2 Transfer file from PC to device
   - Highlight the file on the left
   - Select device target directory on the right
   - Start transfer using the [>>] button
   - The file is transferred to the device

3 Transfer the file from the device to the PC
   - Highlight the file on the right
   - Select PC target directory on the left
   - Start the transfer using the [<<] button
   - The file is transferred to the PC

7.1.4 Manage users and groups

This function has not yet been implemented.
7.2 Configure inputs and outputs

The inputs and outputs can be configured applying two methods:

7.2.1 via system configuration

This method is useful if the configuration is not supposed to be changed again during the runtime of the application.

Note: Only the I/Os allocated to the PLC can be configured!

Procedure using the example of the operating mode of an input / output:

► In the CODESYS device tree: Extend required PLC > Element [Local_IO]
► Double-click on [Inputs] / [Outputs]
► Click on [Parameters] tab
► The parameter setting view of the inputs / outputs appears
► Select input / output from the list
► Double-click in the column [Value] of the parameter [Mode]
► Click on arrow symbol
► List of possible modes appears
► Click on the required mode
► The mode for the input / output is set
► If needed, set further parameters as described, e.g. filters, periods, frequency, etc.

7.2.2 via function block

This method is useful if the configuration is supposed to be changed during the runtime of the application.

The operating type of the inputs and outputs is set via the block input eMode of the following FBs. Examples:

- FB Input (→ p. 128) > Input eMode
- FB Output (→ p. 131) > Input eMode
- FB OutputGroup > Input eMode
7.3 Configure interfaces

7.3.1 Configure serial interface

The CODESYS service communication via RS232 only works with the preset baud rate. For other purposes, the device supports the following baud rates:
- 9600 baud
- 19200 baud
- 28800 baud
- 38400 baud
- 57600 baud
- 115200 baud (preset)

Setting the interface: → Interface configuration file comconf.cfg (→ p. 78)

7.3.2 Configure Ethernet interface

Setting the interface: → Interface configuration file comconf.cfg (→ p. 78)

Factory setting:
- IP address = 192.168.82.247
- Subnet mask = 255.255.255.0
- Gateway address = 192.168.82.21
- UDP port = 12345

NOTICE
Only use the Ethernet interface in a secure network environment (e.g. separate network or VPN)! Otherwise, unauthorised persons can read or manipulate data or tamper with the functions of the device.

The IP parameter of the Ethernet interface

In order to update the runtime system of the CR721S via a network, the device must be connected to the corresponding network. For the configuration of the Ethernet interface, the following options are available:

- Manual
  The user defines the parameters of the Ethernet interface manually:
  - IP address
  - Subnet mask
  - Gateway address
  ▶ Observe the Address assignment in Ethernet networks (→ p. 71) in Ethernet networks!

- Automatic
  The interface parameters are set via the Dynamic Host Configuration Protocol (DHCP).
  (the development of this function is still in progress)

Setting the interface: → Interface configuration file comconf.cfg (→ p. 78)
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, 256x254 = 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:

<table>
<thead>
<tr>
<th>Participant A IP address</th>
<th>Participant A Subnet mask</th>
<th>Participant B IP address</th>
<th>Participant B Subnet mask</th>
<th>Communication of participants possible?</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.82.247</td>
<td>255.255.255.0</td>
<td>192.168.82.10</td>
<td>255.255.255.0</td>
<td>Yes, 254 participants possible</td>
</tr>
<tr>
<td>192.168.82.247</td>
<td>255.255.255.0</td>
<td>192.168.82.247</td>
<td>255.255.255.0</td>
<td>No (same IP address)</td>
</tr>
<tr>
<td>192.168.82.247</td>
<td>255.255.255.0</td>
<td>192.168.82.10</td>
<td>255.255.0.0</td>
<td>No (different subnet mask)</td>
</tr>
<tr>
<td>192.168.82.247</td>
<td>255.255.255.0</td>
<td>192.168.116.10</td>
<td>255.255.255.0</td>
<td>No (different IP address range: 82 vs. 116)</td>
</tr>
<tr>
<td>192.168.222.213</td>
<td>255.255.0.0</td>
<td>192.168.222.123</td>
<td>255.255.0.0</td>
<td>Yes, 65 024 participants possible</td>
</tr>
<tr>
<td>192.168.111.213</td>
<td>255.255.0.0</td>
<td>192.168.222.123</td>
<td>255.255.0.0</td>
<td>Yes, 65 024 participants possible</td>
</tr>
<tr>
<td>192.168.82.247</td>
<td>255.255.255.0</td>
<td>192.168.82.0</td>
<td>255.255.255.0</td>
<td>No; the whole network is disturbed because the IP address xxx.xxx.xxx.0 is not allowed</td>
</tr>
</tbody>
</table>
### 7.3.3 Configure CAN interfaces

The CAN interfaces are configurable as follows:
- via system configuration:
  - CANopen
  - SAE J1939
- via function block:
  - RAW-CAN

Under Vendor = 3S, you will find, among others, the following entries:

**CIA CANopen**
- CIA CANopenManager
  - CANopen_Manager
  - CANopen_Manager_SIL2
- CIA Local Device
  - CANopen Device
  - CANopen Device SIL2

**SAE J1939**
- SAE J1939 Manager
  - J1939_Manager

---

⚠️ **WARNING**

The safety functionality is in preparation.

At present the device has NO safety functionality!

► Do NOT use the device for safety-related functions!

The functions of the following protocols are available, but NOT yet suitable for safety applications:
- CANopen_Manager_SIL2
- CANopen_Device_SIL2
System configuration

Configure interfaces

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!

- **Attach CAN bus**
  - In the CODESYS device tree: Right mouse click on [Communication] > [CAN].
  - Select [Add Device...].
  - Window [Add Device] appears.
  - Select [Vendor:] [ifm electronic].
  - In the list below: Select [ifmCANbus].
  - Confirm the selection with [Add Device].
  - Close the window [Add Device] with the [Close] button.

- **Assign CAN interface**
  - In the CODESYS device tree: Double-click on [Communication] > [CAN] > [ifmCANBus].
  - Tab [General] > [General] > [Network]: assign this setting with ▲/▼ to a CAN interface. permissible = 0...3
  - Select the required value for baud rate [Baudrate (bit/s)] from the list field.

- **Attach CANopen manager**
  - In the CODESYS device tree: Right mouse click on [Communication] > [CAN] > [ifmCANBus].
  - Select [Add Device...].
  - Window [Add Device] appears.
  - Select [Vendor:] [<All vendors>].
  - In the list below: Select [Fieldbusses] > [CiA CANopen] > [CiA CANopenManager] > [CANopenManager].
  - Confirm the selection with [Add Device].
  - Close the window [Add Device] with the [Close] button.
Set CANopen manager parameters

► In the CODESYS device tree: Double-click on [Communication] > [CAN] > [CiA CANopenManager] > [CANopenManager].

► Tab [General] > [General] > [Node ID]: assign a node ID to this interface using ▲/▼. permissible = 1...127

► Select the further parameters according to the requirements, e.g.:
  ▪ Configure the heartbeat protocol in the section [Nodeguarding]:
  ▪ Click on the field of options in order to activate [Heartbeat Producing]
  ▪ Set the parameters [Node ID] and [Producer Time (ms)]
  ▪ Configure the sync protocol in the section [Sync]:
  ▪ Click on the field of options to activate [Enable Sync Producing], if necessary.
  ▪ Set the parameters [COB-ID (Hex)], [Cycle Period (μs)] and [Window Length (μs)]
  ▪ In the section [Time]: The time protocol is not supported.

> With the menu [File] > [Save Project], 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).
**via system configuration: CANopen device**

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

These entries are equivalent.

- **Attach CAN bus**
  - In the CODESYS device tree: Right mouse click on [Communication] > [CAN] > [ifmCANBus].
  - Select [Add Device...].
  - Window [Add Device] appears.
  - Select [Vendor:] [ifm electronic].
  - In the list below: Select [ifmCANbus].
  - Confirm the selection with [Add device].
  - Close the window [Add device] with the [Close] button.

- **Assign CAN interface**
  - In the CODESYS device tree: Double-click on [Communication] > [CAN] > [ifmCANBus].
  - Tab [General] > [General] > [Network]: assign this setting with ▲/▼ to a CAN interface. permissible = 0...3
  - Select the required value for baud rate [Baudrate (bit/s)] from the list field.

- **Attach CANopen device**
  - In the CODESYS device tree: Right mouse click on [Communication] > [CAN] > [ifmCANBus].
  - [Add Device...] Select.
  - Window [Add Device] appears.
  - Select [Vendor:] <All vendors>.
  - In the list below: Select [Fieldbusses] > [CiA CANopen] > [Local Device] > [CANopenDevice].
  - Confirm the selection with [Add Device].
  - Close the window [Add Device] with the [Close] button.

- **Set CANopen device parameters**
  - In the CODESYS device tree: Double-click on [Communication] > [CAN] > [CANopenDevice].
  - Tab [General] > [General] > [Node ID]: assign a node ID to this interface using ▲/▼. permissible = 1...127
  - Select the further parameters according to the requirements.
  - With the menu [File] > [Save Project], the values become valid.

When activating the option field [Werkseinstellungen]: 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 CANopen device.

In the CANopen Device, the correct baud rate and the node ID must be set, so that the CANopen master will recognise the device.
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.

- **Attach CAN bus**
  - In the CODESYS device tree: Right mouse click on [Communication] > [CAN].
  - Select [Add Device...].
  - Window [Add Device] appears.
  - Select [Vendor:] [ifm electronic].
  - In the list below: Select [ifmCANbus].
  - Confirm the selection with [Add Device].
  - Close the window [Add Device] with the [Close] button.

- **Assign CAN interface**
  - In the CODESYS device tree: Double-click on [Communication] > [CAN] > [ifmCANBus].
  - Tab [General] > [General] > [Network]: assign this setting with ▲/▼ to a CAN interface.
  - permissible = 0...3
  - Select the required value for baud rate [Baudrate (bit/s)] from the list field.

- **Attach J1939 manager**
  - In the CODESYS device tree: Right mouse click on [Communication] > [CAN] > [ifmCANBus].
  - Select [Add Device...].
  - Window [Add Device] appears.
  - Select [Vendor:] <All Vendors>.
  - In the list below: Select [Fieldbusses] > [SAE J1939] > [J1939 Manager] > [J1939_Manager].
  - Confirm the selection with [Add Device].
  - Close the window [Add Device] with the [Close] button.

- **Set J1939 manager parameters**
  - In the CODESYS device tree: Double-click [Communication] > [CAN] > [J1939_Manager].
  - Tab [General] > [Database] > [Database]: select the list from the required database.
  - default = J1939Default

Users can use their own databases.

They must be at the following storage location: 
C:\ProgramData\CODESYS\J1939 Databases
The directory ProgramData is hidden by default.

> With the menu [File] > [Save Project], the values become valid.
- **Attach J1939-ECU**
  - In the CODESYS device tree: Right mouse click on [Communication] > [CAN] > [ifmCANBus] > [J1939_Manager].
  - Select [Add Device...].
  - Window [Add Device] appears.
  - In the area [Device]: [Vendor:] select <All vendors>.
  - In the list below: Select [Fieldbusses] > [J1939] > [J1939_ECU] > [J1939_ECU].
  - Confirm the selection with [Add Device].
  - Close the window [Add Device] with the [Close] button.

- **Set J1939-ECU parameters**
  - In the CODESYS device tree: Double-click on [Communication] > [CAN] > [J1939_Manager] > [J1939_ECU].
  - Make the following settings in the tab [General] in the section [General] according to the specific application:

<table>
<thead>
<tr>
<th>user case</th>
<th>[Local Device]</th>
<th>Significance [Preffered Address]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receiving broadcast data of the ECU</td>
<td>[ ]</td>
<td>deactivated Address of the ECU from which the data is to be received</td>
</tr>
<tr>
<td>No transmission</td>
<td>[ ]</td>
<td>No transmission</td>
</tr>
<tr>
<td>Sending data (broadcast and P2P)</td>
<td>[ ]</td>
<td>activated Address of the ifm controller</td>
</tr>
<tr>
<td>Receiving P2P data</td>
<td>[ ]</td>
<td>Receiving P2P data</td>
</tr>
</tbody>
</table>

  - Add parameter groups in the tab [TX-Signals] by clicking on [Add PG].
  - The settings become valid with menu [File] > [Save Project].

via function block: **RAW-CAN**

The Library ifmRawCAN.library (→ p. 165) features several function blocks for this application.
7.3.4 Interface configuration file comconf.cfg

The file directory /com of the device contains the file comconf.cfg. To change the configuration data of the following interfaces, this file must be written into the device with the corresponding changes:
- Serial interface
- Ethernet interface
- CAN interfaces

Factory setting of the content:

```plaintext
[ETHERNET]
Number=1
IpV4Address=192.168.82.247
IpV4SubnetMask=255.255.255.0
IpV4Gateway=192.168.82.21
UDPPort=12345

[ETHERNET0]
IpV4Address=192.168.82.247
IpV4SubnetMask=255.255.255.0
IpV4Gateway=192.168.82.21
UDPPort=12345

[CAN]
Number=4

[CAN0]
Baud rate=250000
NodeId=127

[CAN1]
Baud rate=250000
NodeId=126

[CAN2]
Baud rate=250000
NodeId=125

[CAN3]
Baud rate=250000
NodeId=124

[COM]
Number=1

[COM0]
Baud rate=115200
Bits=8
Parity=0
Stop=1
```

► To start the (deactivated) device with these default settings:
   (the file comconf.cfg is not taken into consideration)
   TRUE on connection RESET-COM (Pin 72) simultaneous with POWER-ON
   After the start-up: FALSE on RESET-COM

► To start the (deactivated) device with the content of the (new) file comconf.cfg:
   FALSE on connection RESET-COM (Pin 72) simultaneously with POWER-ON
8 Programming

8.1 Objects of a PLC application

All objects of a PLC application are listed as sub-elements of the node [Application] in the device tree. In the basic configuration, a PLC application contains the following objects:

- [Library manager] Provides access to the standard and device-specific function libraries:
  → Use ifm function libraries

- [PLC_PRG (PRG)] Provides access to the editor of the PLC application
  → Create PLC application (→ p. 80)

- [Task configuration] Provides access to the settings of the task processing:
  → Configure task processing (→ p. 82)

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

Familiarise yourself with the following CODESYS functions!

- Online help > CODESYS Development System > Programming Applications

CODESYS automatically generates the function block PLC_PRG (PRG) during project creation. 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)]
- Editor window shows input mask of the selected programming language.
- Enter program code.

8.2.1 Supported programming languages

The following table shows which ifm function libraries support which programming languages according to IEC 61131:

<table>
<thead>
<tr>
<th>Library</th>
<th>function block diagram (FBD)</th>
<th>sequential function chart (SFC)</th>
<th>instruction list (IL)</th>
<th>continuous function chart (CFC)</th>
<th>ladder diagram (LD)</th>
<th>structured text (ST)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifmDeviceCR721S.library</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ifmCANopenManager.library</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ifmRawCAN.library</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ifmFastInput.library</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ifmIOcommon.library</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ifmOutHBridge.library</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ifmOutGroup.library</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ifmOutPWM.library</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Legend:
- X = is supported
- – = is not supported
8.2.2 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 (currently not supported)
  → Online help > CODESYS Development System > Exchanging Data on the Network > Network Variables

The device supports the following variable types:

<table>
<thead>
<tr>
<th>Variable type</th>
<th>Declaration</th>
<th>Scope of validity</th>
<th>Memory behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td>local</td>
<td>In the declaration part of the POU</td>
<td>Applies only to the POU in which it has been declared</td>
<td>volatile</td>
</tr>
<tr>
<td>local retain</td>
<td></td>
<td></td>
<td>non volatile</td>
</tr>
<tr>
<td>global</td>
<td>in the global variable list (GVL)</td>
<td>applies to all POUs of the project</td>
<td>volatile</td>
</tr>
<tr>
<td>global retain</td>
<td></td>
<td></td>
<td>non volatile</td>
</tr>
<tr>
<td>network</td>
<td>In network variable lists</td>
<td>Values are available to all projects in the whole network if the variable is contained in their network variables lists</td>
<td>Volatile</td>
</tr>
<tr>
<td>Network retain</td>
<td></td>
<td></td>
<td>non volatile</td>
</tr>
</tbody>
</table>

For performance reasons, do not overuse 64 bit variables!

CAN network variables are not supported!

Retain variables

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

Typical applications for retain variables are for example:
- operating hours which are counted up and retained while the machine is in operation,
- position values of incremental encoders,
- preset values entered in the screen device,
- machine parameters,
  i.e. all variables whose values must not get lost when the device is switched off.
8.2.3 Options to access input and output data

In a CODESYS project, each input and output has a physical address according to the IEC standard (e.g. %IW5). CODESYS offers the following options to access this address from a PLC application and thereby to access the input and outputs data of the device:

- Access to IEC address via AT declaration
- Definition of an ALIAS for an IEC address
- Link a program variable to an IEC address (mapping)

8.2.4 Configure task processing

Familiarise yourself with the following CODESYS functions!

- Task configuration:
  - Online help > CODESYS Development System > Programming Applications > Task Configuration

The processing of the tasks is controlled by parameters. The user can set the parameters for each task.

CODESYS automatically creates the following task when the project is created:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>Task for the processing of the main program [PLC_PRG (PRG)]</td>
</tr>
</tbody>
</table>

For subprograms with POUs that are to be executed several times per PLC cycle:

- Create new task.
- Configure task properties:
  1. [Priority]: permissible = 0 (high) ... 3 (low) (select a priority for each task)
  2. [Type]: Cyclic
  3. [Interval]: Interval of the task call-ups in [ms]

The interval time must be longer than the runtime of the task.

- Recommended: Activate watchdog: → Configure IEC watchdog (→ p. 37)

The watchdog time must be shorter than the interval time.

If the CAN buses are heavily utilised:

- Configure task properties:
  1. [Priority]: high
  2. [Type]: Cyclic
  3. [Interval]: requested cycle time (=transmission interval)

- Assign subprograms with the POUs for CAN communication to the CAN tasks.
8.3 Use ifm function libraries

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifmCANopenManager</td>
<td>Functions for use of the CAN interfaces as CANopen Manager</td>
</tr>
<tr>
<td>ifmDeviceCR721S</td>
<td>Data structures, enumeration types and global variables</td>
</tr>
<tr>
<td>ifmFastInput</td>
<td>Functions to access the fast inputs of the device</td>
</tr>
<tr>
<td>ifmIOcommon</td>
<td>Functions for access to the inputs and outputs of the device</td>
</tr>
<tr>
<td>ifmIOconfigDiagProt</td>
<td>Functions to configure the I/O-related diagnostic and protective functions</td>
</tr>
<tr>
<td>ifmOutGroup</td>
<td>Functions to control output group switches</td>
</tr>
<tr>
<td>ifmOutHBridge</td>
<td>Functions to access H-bridge outputs</td>
</tr>
<tr>
<td>ifmOutPWM</td>
<td>Functions to access PWM outputs</td>
</tr>
<tr>
<td>ifmRawCAN</td>
<td>Functions for use of the CAN interfaces as CAN Layer 2</td>
</tr>
<tr>
<td>ifmSysInfo</td>
<td>Functions to set / read system information</td>
</tr>
<tr>
<td>ifmTypes</td>
<td>Global types and interfaces for other ifm libraries</td>
</tr>
</tbody>
</table>

Detailed information about the ifm function libraries: → ifm function libraries (→ p. 105)

8.3.1 Access to inputs

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

<table>
<thead>
<tr>
<th>Function element</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input (→ p. 128)</td>
<td>Assigns an operating mode to an input channel</td>
</tr>
<tr>
<td>FastCount (→ p. 119)</td>
<td>Provides the current state of the selected channel</td>
</tr>
<tr>
<td>IncEncoder (→ p. 121)</td>
<td>Up/down counter function to evaluate encoders</td>
</tr>
</tbody>
</table>
| Period (→ p. 123) | • measures at the indicated channel:  
                       • the frequency and the period length (cycle time) in [μs],  
                       • measures at the indicated channel pair:  
                       • the phase shift in [°] between channel A and channel B |
8.3.2 Access to outputs

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

<table>
<thead>
<tr>
<th>Function element</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output (→ p. 131)</td>
<td>Assigns an operating mode to an output channel. Provides the current state of the selected channel.</td>
</tr>
<tr>
<td>OutputGroup</td>
<td>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.</td>
</tr>
<tr>
<td>HBridge (→ p. 152)</td>
<td>H bridge on a PWM channel pair.</td>
</tr>
<tr>
<td>PWM1000 (→ p. 160)</td>
<td>Initialises and configures a PWM-capable output channel. The mark-to-space ratio can be indicated in steps of 1‰.</td>
</tr>
<tr>
<td>CurrentControl (→ p. 157)</td>
<td>Current controller for a PWMi output channel.</td>
</tr>
</tbody>
</table>

8.3.3 Control device

The following function elements are available to control the device:

<table>
<thead>
<tr>
<th>Function element</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SupplySwitch (→ p. 136)</td>
<td>Switch off the unit.</td>
</tr>
<tr>
<td>SetLED (→ p. 134)</td>
<td>Change the frequency and the colour of the status LED in the application program.</td>
</tr>
</tbody>
</table>

8.3.4 Read device information

To read information from the device the following functional elements are available:

<table>
<thead>
<tr>
<th>Function element</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemSupply (→ p. 138)</td>
<td>Indicates the value of the system voltage.</td>
</tr>
<tr>
<td>Temperature (→ p. 140)</td>
<td>Indicates the value of the system temperature.</td>
</tr>
</tbody>
</table>
8.4 Use IO mapping

During the IO mapping (I/O image), global variables are coupled to the IEC addresses (%Ixx, %Qxx). Via symbol names, the user has access to the following elements from the application:

- inputs and outputs
- Functions of the operating elements
- Functions of the display elements
- States of system components and characteristic values

⚠️ The addresses of the system flags can change if the PLC configuration is extended.

► While programming only use the symbol names of the system flags!
# 8.4.1 Access inputs

The user can use the following global variables to access the operating modes and the values of the inputs of the device.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Data type</th>
<th>Access</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INnnnnn_I.</td>
<td></td>
<td>r</td>
<td>Value of the analogue input</td>
<td>0 ... 65535</td>
</tr>
<tr>
<td>ValueAnalogue</td>
<td>UINT</td>
<td>r</td>
<td>Value of the digital input</td>
<td>FALSE: Input deactivated</td>
</tr>
<tr>
<td>ValueDigital</td>
<td>BIT</td>
<td>r</td>
<td>Value of the digital input</td>
<td>TRUE: Input activated</td>
</tr>
<tr>
<td>ValueCount</td>
<td>UDINT</td>
<td>r</td>
<td>Value of the counting input</td>
<td>0 ... 4294967295</td>
</tr>
<tr>
<td>ValueLastCountWasUp</td>
<td>BIT</td>
<td>r</td>
<td>Counting direction upwards</td>
<td>FALSE: not active</td>
</tr>
<tr>
<td>ValueLastCountWasDown</td>
<td>BIT</td>
<td>r</td>
<td>Counting direction downwards</td>
<td>TRUE: active</td>
</tr>
<tr>
<td>ValueCycle</td>
<td>UDINT</td>
<td>r</td>
<td>Cycle time</td>
<td>0 ... 4294967295</td>
</tr>
<tr>
<td>ValueFreq</td>
<td>REAL</td>
<td>r</td>
<td>Frequency</td>
<td>1.401e-45 ... 3.403e+38</td>
</tr>
<tr>
<td>ValueTime</td>
<td>UDINT</td>
<td>r</td>
<td>Elapsed time since the last edge evaluation</td>
<td>0 ... 4294967295</td>
</tr>
<tr>
<td>ValueRatio</td>
<td>UINT</td>
<td>r</td>
<td>Pulse/pause ratio</td>
<td>0 ... 65535</td>
</tr>
<tr>
<td>Error</td>
<td>BIT</td>
<td>r</td>
<td>Error</td>
<td>FALSE: no error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TRUE: Error</td>
</tr>
</tbody>
</table>

| INnnnnn_Q.            | ENUM of INT | r/w    | Read/set counting direction                      | COUNT_OFF: Counting off           |
| CountDirection        |             |        |                                                  | COUNT_UP: Counting upward         |
|                      |             |        |                                                  | COUNT_DOWN: Zählen abwärts        |
| Counting downward     | BIT         | w      | Set preset value                                  | FALSE: no action                  |
|                      |             |        |                                                  | TRUE: Set preset value            |

Legend:
- r = read only
- r/w = read and write

The valid value ranges and the type and number of the variables of the input depend on the active operating mode of the input.

- Observe configuration of the inputs! → **Configure inputs and outputs** (→ p. 69)
## 8.4.2 Access outputs

The user can use the following global variables to access the operating modes and the values of the outputs of the device.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Data type</th>
<th>Access</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTnnnn_I</td>
<td>UINT</td>
<td>r/w</td>
<td>Current value of the analogue output</td>
<td>0 ... 65535</td>
</tr>
<tr>
<td>OutCurrent</td>
<td>UINT</td>
<td>r/w</td>
<td>PWM Ratio</td>
<td>0 ... 65535</td>
</tr>
<tr>
<td>Ratio</td>
<td>UINT</td>
<td>r/w</td>
<td>PWM Ratio</td>
<td>0 ... 65535</td>
</tr>
<tr>
<td>OutVoltageDiag</td>
<td>UINT</td>
<td>r</td>
<td>Measured voltage value of the analogue output in mV</td>
<td>0 ... 65535</td>
</tr>
<tr>
<td>OutCurrentDiag</td>
<td>UINT</td>
<td>r</td>
<td>Measured current value of the analogue output in mA</td>
<td>0 ... 65535</td>
</tr>
<tr>
<td>OutState</td>
<td>BIT</td>
<td>r/w</td>
<td>Output status</td>
<td>0 ... 4294967295</td>
</tr>
<tr>
<td>Error</td>
<td>BIT</td>
<td>r/w</td>
<td>Error</td>
<td>FALSE no error</td>
</tr>
<tr>
<td>OUTnnnn_Q</td>
<td>UINT</td>
<td>r/w</td>
<td>Analogue output value</td>
<td>0 ... 65535</td>
</tr>
<tr>
<td>ValueAnalogue</td>
<td>UINT</td>
<td>r/w</td>
<td>Digital output value</td>
<td>0 ... 65535</td>
</tr>
<tr>
<td>ValueDigital</td>
<td>UINT</td>
<td>r/w</td>
<td>Digital output value</td>
<td>0 ... 65535</td>
</tr>
<tr>
<td>OutVoltage</td>
<td>UINT</td>
<td>r/w</td>
<td>Output voltage</td>
<td>0 ... 65535</td>
</tr>
<tr>
<td>Error</td>
<td>BIT</td>
<td>r/w</td>
<td>Error</td>
<td>FALSE no error</td>
</tr>
</tbody>
</table>

Legend:
- r = read only
- r/w = read and write

The valid value ranges and the type and number of the variables of the output depend on the active operating mode of the output.

► Observe configuration of the outputs! Configure inputs and outputs (→ p. 69)
# 8.4.3 Read diagnostic data of the device

The user can use the following global variables to access the current diagnostic data of the device:

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Access</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>iTemperature0</td>
<td>INT</td>
<td>r</td>
<td>Temperature on the system board (value in °C)</td>
<td>-32768 ... 32767</td>
</tr>
<tr>
<td>iTemperature1</td>
<td>INT</td>
<td>r</td>
<td>Temperature on the system board (value in °C)</td>
<td>-32768 ... 32767</td>
</tr>
<tr>
<td>uiVoltageVBB15</td>
<td>UINT</td>
<td>r</td>
<td>Voltage at power input VBB15 (value in mV)</td>
<td>0 ... 65535 mV</td>
</tr>
<tr>
<td>uiVoltageVBB30</td>
<td>UINT</td>
<td>r</td>
<td>Voltage at power input VBB30 (value in mV)</td>
<td>0 ... 65535 mV</td>
</tr>
</tbody>
</table>

Legend:
- \( r \) = read only
8.5 Use RawCAN (CAN Layer 2)

In order to access one of the CAN interfaces configured for CANopen operation, the following POUs are available.

Requirements:
- The CAN interface is configured for operation as RawCAN (CAN Layer 2) (→ Configure CAN interfaces (→ p. 72)).

8.5.1 RawCAN: Control CAN network nodes

The following POUs are available to control a node in a CAN network:

<table>
<thead>
<tr>
<th>Function element</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN_Enable (→ p. 166)</td>
<td>initialises the specified CAN interface</td>
</tr>
<tr>
<td></td>
<td>configures the CAN baud rate</td>
</tr>
<tr>
<td>CAN_Recover (→ p. 168)</td>
<td>controls the processing of a failure of the specified CAN channel</td>
</tr>
<tr>
<td></td>
<td>If the CAN channel fails, reset the CAN interface and reboot</td>
</tr>
</tbody>
</table>

8.5.2 RawCAN: Send and receive CAN messages

The following POUs are available to send or receive messages in a CAN network:

<table>
<thead>
<tr>
<th>Function element</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN_Rx (→ p. 174)</td>
<td>configures a data receive object and reads the receive buffer of the data object</td>
</tr>
<tr>
<td>CAN_RxMask (→ p. 176)</td>
<td>receives CAN messages of a non-coherent area</td>
</tr>
<tr>
<td></td>
<td>The area is defined via a bit pattern and a bit mask</td>
</tr>
<tr>
<td>CAN_RxRange (→ p. 178)</td>
<td>receives CAN messages of a coherent area</td>
</tr>
<tr>
<td></td>
<td>The area is defined via an upper and lower limit</td>
</tr>
<tr>
<td>CAN_Tx (→ p. 180)</td>
<td>asynchronous transmission of CAN messages</td>
</tr>
</tbody>
</table>
8.5.3 RawCAN: Request and send remote CAN messages

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

<table>
<thead>
<tr>
<th>Function element</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN_RemoteRequest (→ p. 170)</td>
<td>Send a request for a remote message</td>
</tr>
<tr>
<td>CAN_RemoteResponse (→ p. 172)</td>
<td>reply to the request of a remote message</td>
</tr>
</tbody>
</table>
8.6 Use CANopen

Contents

CANopen: Send and receive SDO ................................................................. 91
CANopen: Network Management (NMT) ...................................................... 91

⚠️ Observe the notes on task configuration! (→ Configure task processing (→ p. 82))
⚠️ Observe the notes about CANopen! (→ System manual)

In order to access one of the CAN interfaces configured for CANopen operation, the following POUs are available in ifm libraries.
Further POUs are available in CODESYS libraries from 3S.

Requirements
- The device is configured as CANopen manager (master) (→ via system configuration: CANopen Manager (→ p. 73)).

8.6.1 CANopen: Send and receive SDO

The following POUs are available to send or receive Service Data Objects (SDO):

<table>
<thead>
<tr>
<th>Function element</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COP_SDOread (→ p. 106)</td>
<td>Read Service Data Object (SDO)</td>
</tr>
<tr>
<td>COP_SDOwrite (→ p. 110)</td>
<td>Write Service Data Object (SDO)</td>
</tr>
</tbody>
</table>

8.6.2 CANopen: Network Management (NMT)

The following POUs are available for the management of the CANopen network:

<table>
<thead>
<tr>
<th>Function element</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COP_GetNodeState (→ p. 106)</td>
<td>Request state of one or several CANopen devices</td>
</tr>
<tr>
<td>COP_SendNMT (→ p. 112)</td>
<td>Send an NMT control command to a CANopen device</td>
</tr>
</tbody>
</table>
8.7 Use SAE J1939

To use the SAE J1939 network protocol, 3S provides the library IoDrvJ1939.
9 Operation

9.1 Transfer CODESYS project to device

To save the CODESYS project on the device, transfer the following component:
- Application (→ Load the application to the device (→ p. 93))

Observe notes on the operating modes of the PLC of the device!
→ Operating states of CR721S

9.1.1 Load the application to the device

To transfer the created application as boot project to the device:

Requirements:
- Communication path is set (→ Set communication path of PLC (→ p. 63)).
- Project tested.

1 Translate application
   - In the device tree: highlight application as active application.
   - Use [Build] > [Rebuild] to translate the active application.
   - CODESYS generates program code.

2 Load application to the device
   - Only for safety PLC: Change to debug mode with [SIL2] > [Enter debug mode...].
   - Use [Online] > [Login] connect with the device.
   - Active application is loaded to the device (download).
   - Application on the device is in the STOP state.

3 Start application
   - Use [Debug] > [Start] to start the application.
   - Application goes to the RUN state.
9.1.2 Delete application from CR721S

To delete an application stored on the device:

1. Connect with the device
   - In the device tree: highlight application as active application.
   - Use [Online] > [Login] to establish connection to the device.
   - CODESYS is in the online mode.

2. Delete application
   - In the editor window: Select [Device] > [Applications] tab.
   - Press [Refresh List] to refresh the view.
   - List shows the applications that are stored on the device.
   - Delete all applications in the device with [Remove All].
     OR:
     Highlight requested application and press [Remove] to delete it from the device.
   - Selected application will be deleted.
9.2 Operating states

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

- **SYSTEM_STOP**
  - Application: STOP
  - Process Communication: OFF
  - Service Communication: ON

- **POWER_OFF (Init)**
  - Application: STOP
  - Process Communication: OFF
  - Service Communication: OFF

- **INIT**
  - Application: STOP
  - Process Communication: OFF
  - Service Communication: OFF

- **Safety PLC: OPERATING**

- **Standard PLC: OPERATING**

- **RUNTIME_STOP**
  - Application: STOP
  - Process Communication: OFF
  - Service Communication: ON

- **RUNTIME_OPERATING**
  - Application: OPERATING
  - Process Communication: IEC-Controlled
  - Service Communication: ON

- **RUNTIME_DEBUG_STOP**
  - Application: STOP
  - Process Communication: OFF
  - Service Communication: ON

- **RUNTIME_DEBUG_RUN**
  - Application: OPERATING
  - Process Communication: IEC-Controlled
  - Service Communication: ON

- **UPDATE**
  - Application: STOP
  - Process Communication: OFF
  - Service Communication: ON

- **SHUTDOWN**
  - Application: SHUTDOWN
  - Process Communication: OFF
  - Service Communication: OFF

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
9.3 Status LEDs

The device has the following LEDs:

<table>
<thead>
<tr>
<th>LED</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS0</td>
<td>Status of the standard PLC&lt;br&gt; Status of the ifm operating system&lt;br&gt; Status of the bootloader</td>
</tr>
<tr>
<td>SYS1</td>
<td>Status of the safety PLC&lt;br&gt; Status of the ifm operating system</td>
</tr>
<tr>
<td>ETH0</td>
<td>Status of the Ethernet interface 0</td>
</tr>
<tr>
<td>ETH1</td>
<td>Status of the Ethernet interface 1</td>
</tr>
<tr>
<td>APPL0</td>
<td>LEDs for free use in the application</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>LED colour</th>
<th>Display</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>off</td>
<td>permanently off</td>
<td>ifm operating system on the unit: POWER_OFF</td>
</tr>
<tr>
<td>Green</td>
<td>permanently on</td>
<td>ifm operating system on the unit: INIT</td>
</tr>
<tr>
<td>Red</td>
<td>permanently on</td>
<td>ifm operating system on the unit: SYSTEM_STOP&lt;br&gt; Error class = A</td>
</tr>
<tr>
<td>no change</td>
<td>no change</td>
<td>ifm operating system on the unit: SHUTDOWN&lt;br&gt; no change</td>
</tr>
<tr>
<td>green-yellow</td>
<td>Flashing with 2 Hz</td>
<td>ifm operating system on the unit: UPDATE&lt;br&gt; (time frame = 200 ms)</td>
</tr>
</tbody>
</table>
9.3.2 Status LED: system PLC (SYS0, SYS1)

The SYS0 LED is for the "standard PLC".
The SYS1 LED is for the "safety PLC".
The status of one of the PLCs has no influence on the display of the other PLC.

<table>
<thead>
<tr>
<th>LED colour</th>
<th>Display</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>permanently on</td>
<td>RUNTIME_OPERATING no application loaded</td>
</tr>
<tr>
<td>Green</td>
<td>Flashing with 2 Hz</td>
<td>RUNTIME_OPERATING Application = RUN (time frame = 200 ms)</td>
</tr>
<tr>
<td>Yellow</td>
<td>Flashing with 2 Hz</td>
<td>RUNTIME_DEBUG_RUN Application = RUN (time frame = 200 ms)</td>
</tr>
<tr>
<td>Yellow</td>
<td>permanently on</td>
<td>RUNTIME_DEBUG_STOP Application = STOP</td>
</tr>
<tr>
<td>Red</td>
<td>flashes with 10 Hz</td>
<td>RUNTIME_STOP Error class = B (time frame = 200 ms)</td>
</tr>
</tbody>
</table>

9.3.3 Status LED: System bootloader (SYS0)

The SYS0 LED is for the bootloader status only.
The SYS1 LED is switched off in these cases.

⚠️ Only execute the bootloader update when explicitly requested by ifm!

<table>
<thead>
<tr>
<th>LED colour</th>
<th>Display</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Flashing with 5 Hz</td>
<td>no runtime system loaded (time frame = 200 ms)</td>
</tr>
<tr>
<td>green-yellow</td>
<td>flashes with 5 Hz</td>
<td>Bootloader update process active (time frame = 200 ms)</td>
</tr>
</tbody>
</table>
### 9.3.4 Status LED: Ethernet interfaces (ETH0, ETH1)

The two Ethernet interfaces indicate their status as follows:

<table>
<thead>
<tr>
<th>LED colour</th>
<th>Display</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>permanently on</td>
<td>Ethernet connection is established non data traffic</td>
</tr>
<tr>
<td>Green</td>
<td>blinks</td>
<td>Ethernet connection is established with data traffic</td>
</tr>
</tbody>
</table>

### 9.3.5 Controlling LEDs in the applications

The LEDs APPL0 to APPL3 are for free use in the applications. This is the function of the FB `SetLED` (→ p. 134).

Possible colours: → `LED_COLOUR (ENUM)` (→ p. 117)
Possible frequencies: → `LED_FLASH_FREQ (ENUM)` (→ p. 117)
9.4 Reset

9.4.1 Supported reset variants

The following table shows the reset variants supported by the device-internal CODESYS PLC and the resulting system behaviour:

<table>
<thead>
<tr>
<th>Type of reset</th>
<th>System behaviour</th>
<th>Triggering actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset (warm)</td>
<td>• application goes to STOP state.</td>
<td>→ Reset application (warm) (→ p. 99)</td>
</tr>
<tr>
<td></td>
<td>• Standard variables (VAR) of the application are initialised.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Remanent variables (VAR RETAIN) of the application keep their current values.</td>
<td></td>
</tr>
<tr>
<td>Reset (cold)</td>
<td>• application changes to the STOP state.</td>
<td>→ Reset application (cold) (→ p. 100)</td>
</tr>
<tr>
<td></td>
<td>• All variables (VAR, VAR RETAIN) of the application are initialised.</td>
<td></td>
</tr>
<tr>
<td>Reset (default)</td>
<td>• application goes to STOP state.</td>
<td>→ Reset application (origin) (→ p. 100)</td>
</tr>
<tr>
<td></td>
<td>• The application on the PLC is deleted.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• All variables (VAR, VAR RETAIN) of the application are initialised.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PLC is reset to the default state.</td>
<td></td>
</tr>
</tbody>
</table>

A variable that has been declared without an initialisation value is initialised with the variable-specific standard value (e.g. INT = 0).

9.4.2 Reset application (warm)

To reset the application:
► In the device tree: Select [Application] and select
► [Online] > [Login] as active application.
► 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.
9.4.3  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

9.4.4  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
  > PLC is reset to the original state.
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

9.5.1 Transmission of the files with CODESYS

To transfer files between PC and device:
1. Select file view
   ► In the device tree: Double-click on symbol [Device (CR721S)]
   ► In the editor window: Select the [Files] tab.
   > The editor window shows the file structure on the PC on the left and on the device on the right
2. Transfer file from PC to device
   ► Highlight the file on the left
   ► Select device target directory on the right
   ► Star transfer using the [>] button
   > The file is transferred to the device
3. Transfer the file from the device to the PC
   ► Highlight the file on the right
   ► Select PC target directory on the left
   ► Start the transfer using the [<<] button
   > The file is transferred to the PC
9.5.2 Data transmission with TFTP

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

**Transfer file from device to PC:**

```
tftp -i IP-Adresse GET source target
```

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-Adresse 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
```
9.5.3 Files for series production

The following files must be transferred:

<table>
<thead>
<tr>
<th>Data name / path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apps</td>
<td>Folder</td>
</tr>
<tr>
<td>• standard.app</td>
<td>Application non-safe</td>
</tr>
<tr>
<td>• safe.app</td>
<td>Application safe</td>
</tr>
<tr>
<td>os</td>
<td>Folder</td>
</tr>
<tr>
<td>• ifmOS.ifm</td>
<td>ifmOS</td>
</tr>
<tr>
<td>cfg</td>
<td>Folder</td>
</tr>
<tr>
<td>• comconf.cfg</td>
<td>Communication config</td>
</tr>
<tr>
<td>• memconf.ifm</td>
<td>Memory config</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Data name / path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>retain</td>
<td>Folder</td>
</tr>
<tr>
<td>• standard.ret</td>
<td>Application retain</td>
</tr>
<tr>
<td>• standard.mb</td>
<td>Application memory</td>
</tr>
<tr>
<td>• safe.ret</td>
<td>Application retain</td>
</tr>
<tr>
<td>• safe.mb</td>
<td>Application memory</td>
</tr>
<tr>
<td>data</td>
<td>Folder</td>
</tr>
<tr>
<td>• <em>.</em></td>
<td>Memory space for</td>
</tr>
<tr>
<td></td>
<td>user-defined data</td>
</tr>
</tbody>
</table>
9.6 Display system information

In the online mode the device tree displays the current values of the following system parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Settings</td>
<td>IP settings</td>
<td>--</td>
</tr>
<tr>
<td>IP Address</td>
<td>IP address of the device</td>
<td>E.g. 192.168.0.100</td>
</tr>
<tr>
<td>IP Mask</td>
<td>Subnet mask of the network</td>
<td>E.g. 255.255.255.0</td>
</tr>
<tr>
<td>Gateway Address</td>
<td>IP address of the network gateway</td>
<td>E.g. 192.168.0.2</td>
</tr>
<tr>
<td>Version Firmware</td>
<td>Version of the installed firmware</td>
<td>E.g. V1.4.0</td>
</tr>
<tr>
<td>Serial Number Device</td>
<td>Serial number of the device</td>
<td>E.g. 1511AB019</td>
</tr>
</tbody>
</table>

To display the system information of the device:

► Establish connection between CODESYS and CR721S.
► Select [Online] > [Login].
► CODESYS changes to the online mode.
► In the device tree: Double-click on [System_Info]
► In the editor window: Select tab [Parameter].
► In the editor window: Table shows current values of the system parameters.
10 ifm function libraries

General information about:

- Messages / diagnostic codes of the function blocks (→ p. 184)
- ifm behaviour models for function blocks (→ p. 186)

10.2 Library ifmCANopenManager.library

The library contains program blocks (POU) and data structures for the programming of the functionality of a CANopen Manager.
10.2.1 **COP_GetNodeState**

**Function block type:** Function block (FB)  
**Behaviour model:** EXECUTE  
**Library:** ifmCANopenManager.library  
**Symbol in CODESYS:**

The FB indicates the current state of a CANopen node.

### Input parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xExecute</td>
<td>BOOL</td>
<td>Control execution of the FB</td>
<td>FALSE, TRUE, Other</td>
</tr>
<tr>
<td>eChannel</td>
<td>CAN_CHANNEL</td>
<td>Identifier of the CAN interface</td>
<td>CAN_CHANNEL (ENUM) (→ p. 116)</td>
</tr>
<tr>
<td>usiNode</td>
<td>USINT</td>
<td>ID of the CANopen node</td>
<td>0, 1 ... 127</td>
</tr>
</tbody>
</table>

**Description**

The FB indicates the current state of a CANopen node.
### Output parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| 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 occurred during the FB execution | FALSE (No error occurred or the FB is still being executed)  
TRUE (Error 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) (→ p. 114) |

#### Diagnostic codes:

- **STAT_INACTIVE**: State: FB/Function is inactive.
- **STAT_BUSY**: State: FB/Function is currently executed.
- **STAT_DONE**: State: 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_BUS_OFF**: Error: CAN interface is in the “BUS OFF” state
- **ERR_COMMUNICATION**: Error: no Connection to the network node or network node not available
10.2.2 COP_SDOread

**Function block type:** Function block (FB)  
**Behaviour model:** EXECUTE

**Library:** ifmCANopenManager.library  
**Symbol in CODESYS:**

```plaintext
Function block (FB)  
Behaviour model: EXECUTE
Library: ifmCANopenManager.library
```

**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**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xExecute</td>
<td>BOOL</td>
<td>Control execution of the FB</td>
<td>FALSE → TRUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Other: No impact on FB processing</td>
</tr>
<tr>
<td>eChannel</td>
<td>CAN_CHANNEL</td>
<td>Identifier of the CAN interface</td>
<td>→ CAN_CHANNEL (ENUM) (→ p. 116)</td>
</tr>
<tr>
<td>usiNode</td>
<td>USINT</td>
<td>ID of the CANopen node</td>
<td>0: Local device</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 ... 127: ID of the CANopen node</td>
</tr>
<tr>
<td>uiIndex</td>
<td>UINT</td>
<td>Index in the object directory</td>
<td></td>
</tr>
<tr>
<td>usiSubIndex</td>
<td>USINT</td>
<td>Subindex of the index in the object directory</td>
<td></td>
</tr>
<tr>
<td>pData</td>
<td>Pointer to USINT</td>
<td>Pointer on buffer storage</td>
<td></td>
</tr>
<tr>
<td>udiBuffLen</td>
<td>UDINT</td>
<td>Size of the buffer storage (in byte)</td>
<td></td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td>Max. response time</td>
<td>E.g. T#25ms</td>
</tr>
</tbody>
</table>
### Output parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xDone</td>
<td>BOOL</td>
<td>Indication of whether execution of the FB has been successfully completed</td>
<td>FALSE: FB is executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: • FB successfully executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• FB can be called again</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: • Error occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Action could not be executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>udiLen</td>
<td>UDINT</td>
<td>Number of received bytes</td>
<td></td>
</tr>
</tbody>
</table>

#### Diagnostic codes:

- **STAT_INACTIVE**
  - State: FB/Function is inactive.
- **STAT_DONE**
  - State: FB/Function has been successfully executed and completed. There are valid results on the outputs.
- **ERR_CHAN_UNKNOWN**
  - 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_INVALID_OBJ_ENTRY**
  - Error: Object directory entry is invalid.
- **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!
10.2.3 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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xExecute</td>
<td>BOOL</td>
<td>Control execution of the FB</td>
<td>FALSE ↔ TRUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FB is executed once</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Other</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No impact on FB processing</td>
</tr>
<tr>
<td>eChannel</td>
<td>CAN_CHANNEL</td>
<td>Identifier of the CAN Interface</td>
<td>→ CAN_CHANNEL (ENUM) (→ p. 116)</td>
</tr>
<tr>
<td>usINode</td>
<td>USINT</td>
<td>ID of the CANopen node</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Local device</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 ... 127</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ID of the CANopen node</td>
</tr>
<tr>
<td>uiIndex</td>
<td>UINT</td>
<td>Index in the object directory</td>
<td></td>
</tr>
<tr>
<td>usISubIndex</td>
<td>USINT</td>
<td>Subindex of the index in the object directory</td>
<td></td>
</tr>
<tr>
<td>pData</td>
<td>POINTER TO USINT</td>
<td>Pointer on buffer storage</td>
<td></td>
</tr>
<tr>
<td>udILen</td>
<td>UDINT</td>
<td>Number of received bytes</td>
<td></td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td>Max. response time</td>
<td></td>
</tr>
</tbody>
</table>
### Output parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xDone</td>
<td>BOOL</td>
<td>Indication of whether execution of the FB has been successfully completed</td>
<td>FALSE: FB is executed&lt;br&gt;TRUE: FB successfully executed&lt;br&gt;FB can be called again</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed&lt;br&gt;TRUE: Error occurred&lt;br&gt;Action could not be executed&lt;br&gt;Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
</tbody>
</table>

#### Diagnostic codes:

- **STAT_INACTIVE**  
  State: FB/Function is inactive.
- **STAT_DONE**      
  State: FB/Function has been successfully executed and completed. There are valid results on the outputs.
- **ERR_CHAN_UNKNOWN**  
  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_INVALID_OBJ_ENTRY**  
  Error: Object directory entry is invalid.
- **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!
10.2.4 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 parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xExecute</td>
<td>BOOL</td>
<td>Control execution of the FB</td>
<td>FALSE (\rightarrow) TRUE (\rightarrow) FB is executed once (\rightarrow) Other (\rightarrow) No impact on FB processing</td>
</tr>
<tr>
<td>eChannel</td>
<td>CAN_CHANNEL</td>
<td>Identifier of the CAN Interface</td>
<td>(\rightarrow) CAN_CHANNEL (ENUM) (\rightarrow) p. 116</td>
</tr>
<tr>
<td>usINode</td>
<td>USINT</td>
<td>ID of the CANopen node</td>
<td>0 (\rightarrow) Local device (\rightarrow) 1 ... 127 (\rightarrow) ID of the CANopen node</td>
</tr>
<tr>
<td>usINMTservice</td>
<td>NMT_SERVICE</td>
<td>Command for the control of a CANopen node</td>
<td>(\rightarrow) NMT_SERVICE (ENUM) (\rightarrow) p. 114</td>
</tr>
</tbody>
</table>
## Output parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| 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 occurred during the FB execution | FALSE (No error occurred or the FB is still being executed)  
TRUE (Error occurred  
Action could not be executed  
Note diagnostic information) |
| eDiagInfo  | DIAG_INFO | Diagnostic information | → List below (diagnostic codes:) |

### Diagnostic codes:
- **STAT_INACTIVE**  
  State: FB/Function is inactive.
- **STAT_DONE**  
  State: FB/Function has been successfully executed and completed. There are valid results on the outputs.
- **ERR_CHAN_UNKNOWN**  
  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!
### 10.2.5 NMT_SERVICE (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMT_SERVICE</td>
<td>Command for the control of a CANopen node</td>
<td>INIT_NODE</td>
<td>INT</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_PRE_OPERATIONAL</td>
<td>INT</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_OPERATIONAL</td>
<td>INT</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESET_NODE</td>
<td>INT</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESET_COMM</td>
<td>INT</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STOP_NODE</td>
<td>INT</td>
<td>5</td>
</tr>
</tbody>
</table>

### 10.2.6 NMT_STATES (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMT_STATES</td>
<td>State of the CAN network</td>
<td>INIT</td>
<td>INT</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PREOP</td>
<td>INT</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OPERATIONAL</td>
<td>INT</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STOP</td>
<td>INT</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOT_AVAILABLE</td>
<td>INT</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UNKNOWN</td>
<td>INT</td>
<td>5</td>
</tr>
</tbody>
</table>
10.3 Library ifmDeviceCR0721.library

The library contains all device-specific data structures, enumeration types, global variables and constants.
10.3.1 CAN_BAUDRATE (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN baud rate</td>
<td>Data transmission rate of the CAN interface</td>
<td>KBAUD_20 20 kilobaud</td>
<td>INT</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KBAUD_33 33.3 kilobaud</td>
<td>INT</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KBAUD_50 50 kilobaud</td>
<td>INT</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KBAUD_83 83.3 kilobaud</td>
<td>INT</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KBAUD_100 100 kilobaud</td>
<td>INT</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KBAUD_125 125 kilobaud</td>
<td>INT</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KBAUD_250 250 kilobaud</td>
<td>INT</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KBAUD_500 500 kilobaud</td>
<td>INT</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KBAUD_800 800 kilobaud</td>
<td>INT</td>
<td>800</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KBAUD_1000 1000 kilobaud</td>
<td>INT</td>
<td>1000</td>
</tr>
</tbody>
</table>

10.3.2 CAN_CHANNEL (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN_CHANNEL</td>
<td>Identifier of the CAN Interface</td>
<td>CHAN_0 CAN interface 0</td>
<td>INT</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAN_1 CAN interface 1</td>
<td>INT</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAN_2 CAN interface 2</td>
<td>INT</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAN_3 CAN interface 3</td>
<td>INT</td>
<td>3</td>
</tr>
</tbody>
</table>

10.3.3 CANconstants (GVL)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>usiNumberCANitf</td>
<td>Number of the CAN interfaces of the devices</td>
<td>UINT</td>
<td>4</td>
</tr>
</tbody>
</table>

10.3.4 SysInfo (GVL)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>usiNumberOfSysInfo</td>
<td>Number of system components of the device</td>
<td>USINT</td>
<td>8</td>
</tr>
<tr>
<td>aSysInfoList</td>
<td>Variable with list of the system components</td>
<td>ARRAY[0..8] OF SysInfoStruct (STRUCT) (→ p. 117)</td>
<td></td>
</tr>
</tbody>
</table>
## 10.3.5 SysInfoStruct (STRUCT)

<table>
<thead>
<tr>
<th>Designation</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>eInfoType</td>
<td>INFO_TYPE</td>
<td>System component</td>
<td>E.g. FIRMWARE_DEVICE</td>
</tr>
<tr>
<td>sValue</td>
<td>STRING (255)</td>
<td>Value of the system component</td>
<td>E.g. 3.1</td>
</tr>
<tr>
<td>sName</td>
<td>STRING (32)</td>
<td>Name of the system component</td>
<td>E.g. FW Device</td>
</tr>
</tbody>
</table>

## 10.3.6 LED_COLOUR (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED_COLOUR</td>
<td>Colour of the LED (RGB code)</td>
<td>BLACK (OFF) Off</td>
<td>UINT 0x00 0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WHITE White</td>
<td>UINT 0xFF FFFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RED Red</td>
<td>UINT 0xFF 0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GREEN Green</td>
<td>UINT 0x00 FF00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLUE Blue</td>
<td>UINT 0x00 FF00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>YELLOW Yellow</td>
<td>UINT 0x00 FF00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAGENTA Magenta</td>
<td>UINT 0x00 FF00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CYAN Cyan</td>
<td>UINT 0x00 FF00</td>
<td></td>
</tr>
</tbody>
</table>

## 10.3.7 LED_FLASH_FREQ (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED_FLASH_FREQ</td>
<td>Flashing frequency of the status LED</td>
<td>FRQ_0Hz off</td>
<td>INT 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FRQ_05Hz 0.5 Hz</td>
<td>INT 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FRQ_1Hz 1 Hz</td>
<td>INT 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FRQ_2Hz 2Hz</td>
<td>INT 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FRQ_5Hz 5 Hz</td>
<td>INT 7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FRQ_10Hz 10Hz</td>
<td>INT 8</td>
<td></td>
</tr>
</tbody>
</table>
# 10.4 ifmFastInput.library

<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>FastCount</td>
</tr>
<tr>
<td>IncEncoder</td>
</tr>
<tr>
<td>Period</td>
</tr>
<tr>
<td>COUNT_DIRECTION (ENUM)</td>
</tr>
<tr>
<td>ENCODER_RESOLUTION (ENUM)</td>
</tr>
<tr>
<td>FREQ_SENSE_PERIODS (ENUM)</td>
</tr>
<tr>
<td>MODE_FAST_COUNT (ENUM)</td>
</tr>
<tr>
<td>MODE_INC_ENCODER (ENUM)</td>
</tr>
<tr>
<td>MODE_PERIOD (ENUM)</td>
</tr>
</tbody>
</table>

The library contains function blocks (POU) and enumeration types to control the quick inputs of the device.
10.4.1 FastCount

Function block type: Function block (FB)

Library: ifmFastInput.library
Symbol in CODESYS:

Description

The FB functions as a counter block for pulses on fast input channels.

Input parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xResetError</td>
<td>BOOL</td>
<td>Reset request for an occurring error</td>
<td>FALSE → TRUE: Reset request to the lower level system</td>
</tr>
<tr>
<td>uiChannel</td>
<td>UINT</td>
<td>Input channel</td>
<td>Group + channel → Data sheet → Note on wiring (→ p. 29)</td>
</tr>
<tr>
<td>eMode</td>
<td>MODE_FAST_COUNT</td>
<td>Operating mode of the input channel</td>
<td>→ MODE_FAST_COUNT (ENUM) (→ p. 125)</td>
</tr>
<tr>
<td>eDirection</td>
<td>COUNT_DIRECTION</td>
<td>Counting direction</td>
<td>→ COUNT_DIRECTION (ENUM) (→ p. 125)</td>
</tr>
<tr>
<td>udiPresetValue</td>
<td>UDINT</td>
<td>Preset counter value</td>
<td>permissible = 0...4 294 967 295</td>
</tr>
<tr>
<td>xPreset</td>
<td>BOOL</td>
<td>Changeover switch: counter function active / adopt preset counter value</td>
<td>FALSE: counter active; the number of counted pulses is issued to udiValue.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: The preset counter value is adopted; udiValue = udiPresetValue</td>
</tr>
</tbody>
</table>
Output parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Error occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Action could not be executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>xPrepared</td>
<td>BOOL</td>
<td>State of the FB outputs</td>
<td>FALSE: FB outputs still invalid; FB is still processed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: FB outputs valid; FB has been processed</td>
</tr>
<tr>
<td>udiValue</td>
<td>UDINT</td>
<td>Counter value; number of detected pulses</td>
<td>permissible = 0...4 294 967 295</td>
</tr>
<tr>
<td>udiValueCycle</td>
<td>UDINT</td>
<td>Cycle time of the input signal in [µs]</td>
<td></td>
</tr>
<tr>
<td>rValueFreq</td>
<td>REAL</td>
<td>Frequency of the input signal in [Hz]</td>
<td></td>
</tr>
<tr>
<td>udiValueTime</td>
<td>UDINT</td>
<td>Time elapsed since the last edge evaluation in [µs]</td>
<td>0...4 294 967 295</td>
</tr>
</tbody>
</table>

Diagnostic codes (→ Messages / diagnostic codes of the function blocks (→ p. 184)):
- ERR_INVALID_VALUE Error: At least one input parameter is invalid or outside the value range.
- ERR_INTERNAL Error: Internal system error
  ◆ Contact the ifm Service Center!
- ERR_UNDEFINED Error: Unknown error
  ◆ Contact the ifm Service Center!
- ERR_TIMING reserved
- DIAG_INVALID_VALUE At least one input parameter is invalid or exceeds the permissible area.
- DIAG_INTERNAL Internal system error.
- DIAG_ACCESS FB/Function cannot access the required resource; Resource is blocked by another task.
- DIAG_CHANGEOVER_TIME Minimum changeover time for the highside-lowside selection of the drivers has not yet expired.
- DIAG_SLOW_SIGNAL Input signal is too slow for the measurement.
## 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, output switches to the maximum value of the applicable area. (= outside range)

### Input parameter

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xResetError</td>
<td>BOOL</td>
<td>Reset request for an occurring error</td>
<td>FALSE → TRUE: Reset request to the lower level system</td>
</tr>
<tr>
<td>uiChannel</td>
<td>UINT</td>
<td>1. Input channel (channel A) of the pair of input channels</td>
<td>Group + channel → Data sheet → Note on wiring (= p. 29)</td>
</tr>
<tr>
<td>eMode</td>
<td>MODE_INC_ENCODER</td>
<td>Operating mode of the input channel</td>
<td>→ MODE_INC_ENCODER (ENUM) (= p. 126)</td>
</tr>
<tr>
<td>eResolution</td>
<td>ENCODER_RESOLUTION</td>
<td>Resolution / encoder mode</td>
<td>→ ENCODER_RESOLUTION (ENUM) (= p. 125)</td>
</tr>
<tr>
<td>diPresetValue</td>
<td>DINT</td>
<td>Preset counter value</td>
<td>-2 147 483 648...2 147 483 647</td>
</tr>
<tr>
<td>xPreset</td>
<td>BOOL</td>
<td>Changeover switch: counter function active / adopt preset counter value</td>
<td>FALSE: counter active; the number of counted pulses is issued to udiValue.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE: The preset counter value is adopted; udiValue = diPresetValue</td>
<td></td>
</tr>
</tbody>
</table>
Output parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed; TRUE: Error occurred, Action could not be executed, Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>xPrepared</td>
<td>BOOL</td>
<td>State of the FB outputs</td>
<td>FALSE: FB outputs still invalid; FB is still processed; TRUE: FB outputs valid; FB has been processed</td>
</tr>
<tr>
<td>dValue</td>
<td>DINT</td>
<td>Counter value; number of detected pulses</td>
<td>permissible = -2 147 483 648...2 147 483 647</td>
</tr>
<tr>
<td>xUp</td>
<td>BOOL</td>
<td>Code sequence upwards</td>
<td>FALSE: No count-up since the last call up; TRUE: Count-up or overflow since that last call up</td>
</tr>
<tr>
<td>xDown</td>
<td>BOOL</td>
<td>Code sequence downwards</td>
<td>FALSE: No count-down since the last call-up; TRUE: Count-up or underflow since the last call-up</td>
</tr>
<tr>
<td>udiValueCycle</td>
<td>UDINT</td>
<td>Cycle time of the input signal in [µs]</td>
<td></td>
</tr>
<tr>
<td>rValueFreq</td>
<td>REAL</td>
<td>Frequency of the input signal in [Hz]</td>
<td></td>
</tr>
<tr>
<td>udiValueTime</td>
<td>UDINT</td>
<td>Time elapsed since the last edge evaluation in [µs]</td>
<td>0...4 294 967 295</td>
</tr>
</tbody>
</table>

Diagnostic codes (→ Messages / diagnostic codes of the function blocks (→ p. 184)):

- **ERR_INVALID_VALUE**
  Error: At least one input parameter is invalid or outside the value range.

- **ERR_INTERNAL**
  Error: Internal system error
  - Contact the ifm Service Center!

- **ERR_UNDEFINED**
  Error: Unknown error
  - Contact the ifm Service Center!

- **ERR_TIMING**
  reserved

- **DIAG_INVALID_VALUE**
  At least one input parameter is invalid or exceeds the permissible area.

- **DIAG_INTERNAL**
  Internal system error.

- **DIAG_ACCESS**
  FB/Function cannot access the required resource; Resource is blocked by another task.

- **DIAG_CHANGEOVER_TIME**
  Minimum changeover time for the highside-lowside selection of the drivers has not yet expired.

- **DIAG_SLOW_SIGNAL**
  Input signal is too slow for the measurement.
10.4.3 Period

Function block type: Function block (FB)

Library: ifmFastInput.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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xResetError</td>
<td>BOOL</td>
<td>Reset request for an occurring error</td>
<td>FALSE, TRUE: Reset request to the lower level system</td>
</tr>
<tr>
<td>uiChannel</td>
<td>UINT</td>
<td>Input channel</td>
<td>Group + channel → Data sheet → Note on wiring (→ p. 29)</td>
</tr>
<tr>
<td>eMode</td>
<td>MODE_PERIOD</td>
<td>Operating mode of the input channel</td>
<td>→ MODE_PERIOD (ENUM) (→ p. 126)</td>
</tr>
<tr>
<td>ePeriod</td>
<td>FREQ_SENSE_PERIODS</td>
<td>Number of pulse periods for averaging</td>
<td>→ FREQ_SENSE_PERIODS (ENUM) (→ p. 125)</td>
</tr>
<tr>
<td>udiTimebase</td>
<td>UDINT</td>
<td>Time base for frequency calculation in [ms] Only used in eMode:</td>
<td>→ MODE_PERIOD (ENUM) (→ p. 126)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• IN_FREQUENCY_CSI</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• IN_FREQUENCY_CSO</td>
<td></td>
</tr>
</tbody>
</table>
Output parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>* Error occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>* Action could not be executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>* Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>xPrepared</td>
<td>BOOL</td>
<td>State of the FB outputs</td>
<td>FALSE: FB outputs still invalid; FB is still processed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: FB outputs valid; FB has been processed</td>
</tr>
<tr>
<td>udiValueCycle</td>
<td>UDINT</td>
<td>Cycle time of the input signal in [µs]</td>
<td></td>
</tr>
<tr>
<td>rValueFreq</td>
<td>REAL</td>
<td>Frequency of the input signal in [Hz]</td>
<td></td>
</tr>
<tr>
<td>udiValueTime</td>
<td>UDINT</td>
<td>Time elapsed since the last edge evaluation in [µs]</td>
<td>0...4 294 967 295</td>
</tr>
<tr>
<td>uiValueRatio</td>
<td>UINT</td>
<td>Depends on the mode that is set in the eMode input. Pulse/pause ratio of the input signal in [%] at:</td>
<td>Phase shift of the input signal at the B channel to the signal at the A channel in [°]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>* IN_PERIOD_RATIO_CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>* IN_PERIOD_RATIO_CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>* IN_PHASE_CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>* IN_PHASE_CSO</td>
</tr>
</tbody>
</table>

Diagnostic codes (→ Messages / diagnostic codes of the function blocks (→ p. 184)):

- **STAT_PREPARING**  
  State: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.

- **STAT_DONE**  
  State: FB/Function has been successfully executed and completed. There are valid results on the outputs.

- **ERRINVALIDVALUE**  
  Error: At least one input parameter is invalid or outside the value range.

- **ERRINTERNAL**  
  Error: Internal system error  
  → Contact the ifm Service Center!

- **ERREXCEEDED_RANGE**  
  Error: The value exceeds the value range of its data type.

- **ERRUNDEFINED**  
  Error: Unknown error  
  → Contact the ifm Service Center!

- **ERR_TIMING**  
  reserved

- **DIAGINVALIDVALUE**  
  At least one input parameter is invalid or exceeds the permissible area.

- **DIAGINTERNAL**  
  Internal system error.

- **DIAGACCESS**  
  FB/Function cannot access the required resource; Resource is blocked by another task.

- **DIAG_CHANGEOVER_TIME**  
  Minimum changeover time for the highside-lowside selection of the drivers has not yet expired.

- **DIAG SLOW SIGNAL**  
  Input signal is too slow for the measurement.
## 10.4.4 COUNT_DIRECTION (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT_DIRECTION</td>
<td>Counting direction</td>
<td></td>
</tr>
<tr>
<td>COUNT_OFF</td>
<td>Counting function off</td>
<td></td>
</tr>
<tr>
<td>COUNT_UP</td>
<td>Counting function up</td>
<td></td>
</tr>
<tr>
<td>COUNT_DOWN</td>
<td>Counting function down</td>
<td></td>
</tr>
</tbody>
</table>

## 10.4.5 ENCODER_RESOLUTION (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENCODER_RESOLUTION</td>
<td>Resolution</td>
<td></td>
</tr>
<tr>
<td>FULL_PERIOD</td>
<td>Counts each rising edge on one channel (A)</td>
<td></td>
</tr>
<tr>
<td>HALF_PERIOD</td>
<td>Counts each rising and falling edge on one channel (A)</td>
<td></td>
</tr>
<tr>
<td>EVERY_EDGE</td>
<td>Counts each rising and falling edge on all channels (A and B)</td>
<td></td>
</tr>
</tbody>
</table>

## 10.4.6 FREQ_SENSE_PERIODS (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREQ_SENSE_PERIODS</td>
<td>Number of clock periods for the averaging</td>
<td></td>
</tr>
<tr>
<td>PERIODS_n mit n = 1</td>
<td>No averaging</td>
<td></td>
</tr>
<tr>
<td>PERIODS_n mit n = 2...16</td>
<td>Averaging via n periods</td>
<td></td>
</tr>
</tbody>
</table>

## 10.4.7 MODE_FAST_COUNT (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE_FAST_COUNT</td>
<td>Operating mode of the inputs</td>
<td></td>
</tr>
<tr>
<td>UNCHANGED</td>
<td>Setting remains unchanged</td>
<td></td>
</tr>
<tr>
<td>IN_COUNT_CSI</td>
<td>Input to count fast signal edges; CSI</td>
<td></td>
</tr>
<tr>
<td>IN_COUNT_CSO</td>
<td>Input to count fast signal edges; CSO</td>
<td></td>
</tr>
<tr>
<td>MONITOR</td>
<td>Only output data will be updated.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Values, configurations and process data are not written.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For applications that are not owners of the resource.</td>
<td></td>
</tr>
</tbody>
</table>
### 10.4.8 MODE_INC_ENCODER (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOTE_INC_ENCODER</td>
<td>Operating mode of the input</td>
<td>UNCHANGED: Setting remains unchanged</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_INC_ENCODER_CSI: Input for the evaluation of an incremental encoder, channel A; CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_INC_ENCODER_CSO: Input for the evaluation of an incremental encoder, channel A; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONITOR: Only output data will be updated. Values, configurations ans process data are not written. For applications that are not owners of the resource.</td>
</tr>
</tbody>
</table>

### 10.4.9 MODE_PERIOD (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE_PERIOD</td>
<td>Operating mode of the period input</td>
<td>UNCHANGED: Setting remains unchanged</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_FREQUENCY_CSI: Input for frequency measurement; CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_FREQUENCY_CSO: Input for frequency measurement; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_PERIOD_RATIO_CSI: Input for absolute and ratiometric period measurement; CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_PERIOD_RATIO_CSO: Input for absolute and ratiometric period measurement; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_PHASE_CSI: Input pair for phase measurement, CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_PHASE_CSO: Input pair for phase measurement, CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONITOR: Only output data will be updated. Values, configurations ans process data are not written. For applications that are not owners of the resource.</td>
</tr>
</tbody>
</table>
10.5 Library ifmIOcommon.library

The library contains program blocks (POU) and enumeration types for the control of the inputs and outputs of the device.
10.5.1 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.

Filter:
The input signal can be changed with a digital low-pass filter. Configure the filter via the input eFilter.

Input parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xResetError</td>
<td>BOOL</td>
<td>Reset request for an occurring error</td>
<td>FALSE when switching from FALSE to TRUE: Reset request to the lower level system</td>
</tr>
<tr>
<td>uiChannel</td>
<td>UINT</td>
<td>Input channel</td>
<td></td>
</tr>
<tr>
<td>eMode</td>
<td>MODE_INPUT</td>
<td>Operating mode of the input channel</td>
<td></td>
</tr>
<tr>
<td>eFilter</td>
<td>FILTER_INPUT</td>
<td>Filter definition of the input channel</td>
<td></td>
</tr>
</tbody>
</table>

Examples:
- 403 Group 4 + channel 3
- 502 Group 5 + channel 2

Note on wiring (→ p. 29)
### Output parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Error occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Action could not be executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>xPrepared</td>
<td>BOOL</td>
<td>State of the FB outputs</td>
<td>FALSE: FB outputs still invalid; FB is still processed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: FB outputs valid; FB has been processed</td>
</tr>
<tr>
<td>xValueDigital</td>
<td>BOOL</td>
<td>Logical state of the input in digital operating mode</td>
<td>FALSE: Low Level</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In case of analogue operation, the output is FALSE</td>
<td>TRUE: High Level</td>
</tr>
<tr>
<td>uiValueAnalogue</td>
<td>UINT</td>
<td>Measured input value in analogue operating mode. The interpretation of the input value depends on the setting at the eMode input. MODE_INPUT (ENUM) (→ p. 143)</td>
<td>permissible = 0...65 535</td>
</tr>
</tbody>
</table>
Diagnostic codes (→ Messages / diagnostic codes of the function blocks (→ p. 184)):

- **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_SHORT_CIRCUIT**
  Error: Short circuit with GND or VBBx.

- **ERR_OPEN_CIRCUIT**
  Error: Open circuit detected. Possible cause: Wire break.

- **ERR_OVERLOAD_CURRENT**
  Error: Maximum current exceeded.

- **ERR_STUCK_AT**
  Error: Signal is frozen.

- **ERR_STUCK_AT_HIGH**
  Error: Signal frozen, signal state high.

- **ERR_STUCK_AT_LOW**
  Error: Signal frozen, signal state low.

- **ERR_OVERVOLTAGE**
  Error: Maximum signal voltage exceeded.

- **ERR_UNDERVOLTAGE**
  Error: Minimum signal voltage not reached.

- **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.

- **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.

- **DIAG_INVALID_VALUE**
  At least one input parameter is invalid or exceeds the permissible area.

- **DIAG_INTERNAL**
  Internal system error.

- **DIAG_OPEN_CIRCUIT**
  Open circuit detected. Possible cause: Wire break.

- **DIAG_SHORT_CIRCUIT**
  Error: Short circuit with GND or VBBx.

- **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_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_OVERLOAD_CURRENT**
  Maximum current exceeded.

- **DIAG_STUCK_AT**
  Error: Signal is frozen.

- **DIAG_STUCK_AT_HIGH**
  Error: Signal frozen, signal state high.

- **DIAG_STUCK_AT_LOW**
  Error: Signal frozen, signal state low.

- **DIAG_OVERVOLTAGE**
  Maximum signal voltage exceeded.

- **DIAG_UNDERVOLTAGE**
  Error: Minimum signal voltage not reached.

- **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; Resource is blocked by another task.

- **DIAG_CHANGEOVER_TIME**
  Minimum changeover time for the highside-lowside selection of the drivers has not yet expired.
10.5.2 Output

Function block type: Function block (FB)

Library: ifmIOcommon.library

Symbol in CODESYS:

```
Output
  □ xResetError  BOOL    xError
  □ uiChannel   UINT    eDiagInfo
  □ eMode      MODE_OUTPUT  xPrepared
  □ eFilter    FILTER_OUTPUT  xOutState
  □ uiValue    UINT    uiOutVoltage
```

Description

The FB is used to configure and control a digital or analogue output channel.

Input parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xResetError</td>
<td>BOOL</td>
<td>Reset request for an occurring error</td>
<td>FALSE When switching from FALSE TRUE: Reset request to the lower level system</td>
</tr>
<tr>
<td>uiChannel</td>
<td>UINT</td>
<td>Output channel</td>
<td>Group + channel → Data sheet → Note on wiring (→ p. 29) Examples: 703 Group 7 + channel 3 1203 Group 12 + channel 3</td>
</tr>
<tr>
<td>eMode</td>
<td>MODE_OUTPUT</td>
<td>Operating type of the output channel</td>
<td>→ MODE_OUTPUT (ENUM) (→ p. 144)</td>
</tr>
<tr>
<td>eFilter</td>
<td>FILTER_OUTPUT</td>
<td>Filter definition of the output channel</td>
<td>→ FILTER_OUTPUT (ENUM) (→ p. 142)</td>
</tr>
<tr>
<td>uiValue</td>
<td>UINT</td>
<td>Value that is to be written to the output</td>
<td>FALSE Output deactivated TRUE Output activated</td>
</tr>
</tbody>
</table>

In the digital mode or sensor supply mode; if setting at the eMode input =
- OUT_DIGITAL_CSI
- OUT_DIGITAL_CSO
- OUT_SENSOR_05
- OUT_SENSOR_10

In analogue mode; if setting at the eMode input =
- OUT_ANALOGUE_10
Values indicated in [mV] 0...10 000
# Output parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed</td>
</tr>
</tbody>
</table>
|              |           |                                                                              | TRUE: • Error 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 still invalid; FB is still processed |
|              |           |                                                                              | TRUE: FB outputs valid; FB has been processed         |
| xOutState    | BOOL      | Return value activation state of the selected output                        | FALSE: Output is deactivated                           |
|              |           | (The state may deviate from the required output state if e.g. a safety function has deactivated an output group due to an error.) |
|              |           |                                                                              | TRUE: Output is activated                             |
| uiOutVoltage | UINT      | Current output voltage in [mV]                                              | 0: Operating mode neither "analogue" nor "sensor"     |
|              |           | Only available for the operating modes "analogue" and "sensor"               | ≠ 0: Operating mode "analogue" or "sensor"             |
|              |           |                                                                              | available = 0...final value of the measuring range    |
| uiOutCurrent | UINT      | Present output current in [mA]                                              | available = 0...final value of the measuring range    |
Diagnostic codes (→ Messages / diagnostic codes of the function blocks (→ p. 184)):

- **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 SHORT_CIRCUIT**
  Error: Short circuit with GND or VBBx.

- **ERR STUCK_AT**
  Error: Signal is frozen.

- **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.

- **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.

- **DIAG INVALID_VALUE**
  At least one input parameter is invalid or exceeds the permissible area.

- **DIAG INTERNAL**
  Internal system error.

- **DIAG ACCESS**
  FB/Function cannot access the required resource; Resource is blocked by another task.

- **DIAG CHANGEOVER TIME**
  Minimum changeover time for the highside-lowside selection of the drivers has not yet expired.

- **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 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 STUCK_AT**
  Error: Signal is frozen.

- **DIAG STUCK_AT_HIGH**
  Error: Signal frozen, signal state high.

- **DIAG STUCK_AT_LOW**
  Error: Signal frozen, signal state low.

- **DIAG SHORT_CIRCUIT**
  Error: Short circuit with GND or VBBx.

- **DIAG NO_CALIB**
  The selected resource has no valid calibration.
  The displayed values are maybe faulty.
10.5.3 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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>uiChannel</td>
<td>UINT</td>
<td>Output channel of the ED</td>
<td>0...3 Device LED APP 0...3</td>
</tr>
<tr>
<td>eColour1</td>
<td>ENUM</td>
<td>LED colour status 1</td>
<td>→ LED_COLOUR (ENUM)</td>
</tr>
<tr>
<td>eColour2</td>
<td>ENUM</td>
<td>LED colour status 0</td>
<td>→ LED_COLOUR (ENUM)</td>
</tr>
<tr>
<td>eFrequency</td>
<td>ENUM</td>
<td>Flashing frequency of the</td>
<td>→ LED_FLASH_FREQ (ENUM)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>status LED</td>
<td></td>
</tr>
</tbody>
</table>

Output parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error</td>
<td>FALSE No error occurred or the FB is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>occurred during the FB</td>
<td>still being executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>execution</td>
<td>TRUE Error occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Error occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Action could not be executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>xPrepared</td>
<td>BOOL</td>
<td>State of the FB outputs</td>
<td>FALSE FB outputs still invalid; FB is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>still processed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE FB outputs valid; FB has been</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>processed</td>
</tr>
</tbody>
</table>
Diagnostic codes (→ Messages / diagnostic codes of the function blocks (→ p. 184)):

- **STAT_INACTIVE**
  State: FB/Function is inactive.

- **STAT_BUSY**
  State: FB/Function is currently executed.

- **STAT_DONE**
  State: FB/Function has been successfully executed and completed. There are valid results on the outputs.

- **STAT_PREPARING**
  State: 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/Funktion 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.
10.5.4  SupplySwitch

Function block type: Function block (FB)

Library: ifmIOcommon.library
Symbol in CODESYS:

Description

The FB stops all running applications and switches off the voltage supply latching (terminal 30) in order to shut down the device safely. The voltage supply latching is only deactivated if the following conditions are met:
- Voltage VBB15 < 5.5 V (undervoltage)

⚠️ The separation from the VBB30 takes place when all IEC tasks are finished.

Input parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xSwitchOff</td>
<td>BOOL</td>
<td>Deactivate latching switch of the device</td>
<td>FALSE: No action</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Request deactivation of the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>latching switch</td>
</tr>
</tbody>
</table>

Output parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB</td>
<td>FALSE: No error occurred or the FB is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>execution</td>
<td>still being executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Error occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Action could not be executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>xPrepared</td>
<td>BOOL</td>
<td>State of the FB outputs</td>
<td>FALSE: FB outputs still invalid; FB is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>still processed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: FB outputs valid; FB has been</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>processed</td>
</tr>
</tbody>
</table>
Diagnostic codes (→ Messages / diagnostic codes of the function blocks (→ p. 184)):

- **STAT_DONE**
  State: 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_UNDEFINED**
  Error: Unknown error
  ► Contact the ifm Service Center!

- **DIAG_INVALID_VALUE**
  At least one input parameter is invalid or exceeds the permissible area.

- **DIAG_INTERNAL**
  Internal system error.

- **DIAG_ACCESS**
  FB/Function cannot access the required resource; Resource is blocked by another task.
10.5.5 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>eChannel</td>
<td>ENUM</td>
<td>System voltage channel</td>
<td>→ SYS_VOLTAGE_CHANNEL (ENUM) (→ p. 145)</td>
</tr>
</tbody>
</table>

Output parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed&lt;br&gt;TRUE: Error occurred&lt;br&gt;Action could not be executed&lt;br&gt;Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>xPrepared</td>
<td>BOOL</td>
<td>State of the FB outputs</td>
<td>FALSE: FB outputs still invalid; FB is still processed&lt;br&gt;TRUE: FB outputs valid; FB has been processed</td>
</tr>
<tr>
<td>uiOutVoltage</td>
<td>UINT</td>
<td>Current output voltage of the selected system voltage channel in [mV]</td>
<td>permissible = 0...maximum operating voltage</td>
</tr>
</tbody>
</table>
Diagnostic codes ( Messages / diagnostic codes of the function blocks (→ p. 184)):

- **STAT_DONE**  
  State: FB/Function has been successfully executed and completed. There are valid results on the outputs.

- **STAT_PREPARING**  
  State: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.

- **ERR_INVALID_VALUE**  
  Error: At least one input parameter is invalid or outside the value range.

- **ERR_INTERNAL**  
  Error: Internal system error
  - Contact the ifm Service Center!

- **ERR_UNDEFINED**  
  Error: Unknown error
  - Contact the ifm Service Center!

- **ERR_SHORT_CIRCUIT**  
  Error: Short circuit with GND or VBBx.

- **ERR_OPEN_CIRCUIT**  
  Error: Open circuit detected. Possible cause: Wire break.

- **ERR_OVERLOAD_CURRENT**  
  Error: Maximum current exceeded.

- **ERR_STUCK_AT**  
  Error: Signal is frozen.

- **ERR_STUCK_AT_HIGH**  
  Error: Signal frozen, signal state high.

- **ERR_STUCK_AT_LOW**  
  Error: Signal frozen, signal state low.

- **ERR_OVERVOLTAGE**  
  Error: Maximum signal voltage exceeded.

- **ERR_UNDervoltage**  
  Error: Minimum signal voltage not reached.

- **ERR_SHORT_CIRCUIT**  
  Error: Short circuit with GND or VBBx.

- **ERR_OPEN_CIRCUIT**  
  Error: Open circuit detected. Possible cause: Wire break.

- **DIAG_INVALID_VALUE**  
  At least one input parameter is invalid or exceeds the permissible area.

- **DIAG_INTERNAL**  
  Internal system error.

- **DIAG_OPEN_CIRCUIT**  
  Open circuit detected. Possible cause: Wire break.

- **DIAG_SHORT_CIRCUIT**  
  Error: Short circuit with GND or VBBx.

- **DIAG_UNDervoltage_VBBx**  
  For inputs: Error: Reference voltage not reached.
  For outputs. Error: Voltage of the corresponding output groups supply or at VBB30 / VBB15 is not reached.

- **DIAG_OVERVOLTAGE_VBBx**  
  For inputs: Error: Reference voltage exceeded.
  For outputs. Error: Voltage of the corresponding output groups supply or at VBB30 / VBB15 is not reached.

- **DIAG_OVERLOAD_CURRENT**  
  Maximum current exceeded.

- **DIAG_STUCK_AT**  
  Error: Signal is frozen.

- **DIAG_STUCK_AT_HIGH**  
  Error: Signal frozen, signal state high.

- **DIAG_STUCK_AT_LOW**  
  Error: Signal frozen, signal state low.

- **DIAG_OVERVOLTAGE**  
  Maximum signal voltage exceeded.

- **DIAG_UNDervoltage**  
  Error: Minimum signal voltage not reached.

- **DIAG_NO_CALIB**  
  The selected resource has no valid calibration. The displayed values are maybe faulty.
### 10.5.6 Temperature

**Function block type:** Function block (FB)

**Library:** ifmIOcommon.library

**Symbol in CODESYS:**

![Symbol in CODESYS]

**Description**

The FB indicates the value of the system temperature.

**Input parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>uiChannel</td>
<td>UINT</td>
<td>Input channel</td>
<td>Group + channel → Data sheet → Note on wiring (→ p. 29)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Examples:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>403 Group 4 + channel 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>502 Group 5 + channel 2</td>
</tr>
</tbody>
</table>

**Output parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| xError    | BOOL      | Indication if an error occurred during the FB execution | FALSE No error occurred or the FB is still being executed
|           |           |             | TRUE • Error 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 still invalid; FB is still processed
|           |           |             | TRUE FB outputs valid; FB has been processed |
| iTemperatureC | INT | Measured temperature in [°C] | e.g. 35 |
| iTemperatureF | INT | Measured temperature in [°F] | e.g. 95 |
Diagnostic codes (→ *Messages / diagnostic codes of the function blocks* (→ p. 184)):

- **STAT_PREPARING**  
  State: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.

- **STAT_DONE**  
  State: FB/Function has been successfully executed and completed. There are valid results on the outputs.

- **ERR_INVALID_VALUE**  
  Error: At least one input parameter is invalid or outside the value range.

- **ERR_INTERNAL**  
  Error: Internal system error
  ▶ Contact the ifm Service Center!

- **ERR_UNDEFINED**  
  Error: Unknown error
  ▶ Contact the ifm Service Center!

- **DIAG_INTERNAL**  
  Internal system error.

- **DIAG_INVALID_VALUE**  
  At least one input parameter is invalid or exceeds the permissible area.
### 10.5.7 **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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
<th>Digital signal delay</th>
<th>Analogue signal delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILTER_INPUT</td>
<td>Valid filters for inputs of the FBs</td>
<td>UNCHANGED</td>
<td>No change of settings</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_0</td>
<td>0.6 ms</td>
<td>1.7 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(no digital low-pass filter is set)</td>
<td>(no digital low-pass filter is set)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_1</td>
<td>0.9 ms</td>
<td>3.3 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_2</td>
<td>2.1 ms</td>
<td>7.0 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_3</td>
<td>4.0 ms</td>
<td>14.1 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_4</td>
<td>7.6 ms</td>
<td>28.9 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_5</td>
<td>15.2 ms</td>
<td>58.4 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_6</td>
<td>30.8 ms</td>
<td>117.2 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_7</td>
<td>61.6 ms</td>
<td>235.2 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_8</td>
<td>123.2 ms</td>
<td>470.8 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_9</td>
<td>246.4 ms</td>
<td>942.4 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_10</td>
<td>493.2 ms</td>
<td>1885.6 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_11</td>
<td>986.4 ms</td>
<td>3772.0 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_12</td>
<td>1972.4 ms</td>
<td>7544.4 ms</td>
</tr>
</tbody>
</table>

### 10.5.8 **FILTER_OUTPUT (ENUM)**

Filter setting for the current measurement of an output. The signal of the current measurement is damped via a first-order low-pass filter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILTER_OUTPUT</td>
<td>Valid filter for the outputs of the FBs</td>
<td>UNCHANGED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No change of settings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_12</td>
</tr>
</tbody>
</table>
## 10.5.9 MODE_INPUT (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE_INPUT</td>
<td>Operating mode of the inputs</td>
<td>UNCHANGED: Preset mode is maintained</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_DIGITAL_CSI: Input for analogue value measurement and digital evaluation without diagnostics; CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_DIGITAL_CSI_NAMUR: Input for analogue value measurement and digital evaluation with NAMUR-capable diagnostics; CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_VOLTAGE_10: Input for analogue current measurement 0...10 V; CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_VOLTAGE_32: Input for analogue current measurement 0...32 V; CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_VOLTAGE_RATIO: Input for ratiometric current measurement in relation to VBB30; CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_CURRENT_CSI: Input for current measurement 0...20 mA; CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_RESISTOR: Input for resistance measurement; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_DIGITAL_CSO: Input for analogue value measurement and digital evaluation without diagnostics; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IN_DIGITAL_CSO_DIAG: Input for analogue value measurement and digital evaluation with diagnostics similar to NAMUR; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>


### 10.5.10 MODE_OUTPUT (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE_OUTPUT</td>
<td>Operating mode of the outputs</td>
<td>UNCHANGED: Preset mode is maintained</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT_DIGITAL_CSI: Digital output without diagnostics; CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT_DIGITAL_CSO: Digital output without diagnostics; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT_SENSOR_05: Output with fixed output voltage 5 V for the sensor supply without diagnostics and without protection. CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT_SENSOR_10: Output with fixed output voltage 10 V for the sensor supply without diagnostics and without protection. CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>
### 10.5.11 SYS_VOLTAGE_CHANNEL (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS_VOLTAGE_CHANNEL</td>
<td>List of all available system voltages.</td>
<td>VBB30 Terminal 30 system voltage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VBB15 Terminal 15 system voltage of the ignition switch</td>
</tr>
</tbody>
</table>
10.6 Library ifmOutGroup

The library contains function blocks (POU) to control extended output functions.
### 10.6.1 OutputGroup

**Function block type:** Function block (FB)

**Library:** ifmOutGroup.library

**Symbol in CODESYS:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xResetError</td>
<td>BOOL</td>
<td>Reset request for an occurring error</td>
<td>FALSE → TRUE: Reset request to the lower level system</td>
</tr>
<tr>
<td>uiChannel</td>
<td>UINT</td>
<td>Output channel group</td>
<td>→ Data sheet → Note on wiring (→ p. 29)</td>
</tr>
<tr>
<td>eMode</td>
<td>MODE_OUTPUT_GROUP</td>
<td>Operating type of the output channel group</td>
<td>→ MODE_OUTPUT_GROUP (ENUM) (→ p. 150)</td>
</tr>
<tr>
<td>eFilter</td>
<td>FILTER_OUTPUT_GROUP</td>
<td>Defines the limit frequency of the output filter</td>
<td>→ FILTER_OUTPUT_GROUP (ENUM) (→ p. 150)</td>
</tr>
<tr>
<td>xValue</td>
<td>BOOL</td>
<td>Activation requirement for the output group</td>
<td>FALSE → deactivate output group</td>
</tr>
</tbody>
</table>

The error of an output group will only be reset if all corresponding outputs are error-free.
### Output parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed</td>
</tr>
</tbody>
</table>
|                   |             |                                                                             | TRUE:  
|                   |             | - Error 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 still invalid; FB is still processed                           |
|                   |             |                                                                             | TRUE: FB outputs valid; FB has been processed                                    |
| xGroupState       | BOOL        | Return value activation state of the selected output group                  | FALSE: Output group is deactivated                                               |
|                   |             |                                                                             | TRUE: Output value is activated                                                  |
|                   |             | ![Note](https://example.com) The state may deviate from the required output state if e.g. a safety function has deactivated an output group due to an error. |                                                                                |
| uiGroupCurrent    | UINT        | Measured output current of the entire group in [mA]                         | available = 0...final value of the measuring range                               |
| uiVBBxVoltage     | UINT        | Measured voltage before the group switch in [mV]                           | available = 0...final value of the measuring range                               |
| uiGroupVoltage    | UINT        | Measured voltage after the group switch in [mV]                            | available = 0...final value of the measuring range                               |
Diagnostic codes (→ Messages / diagnostic codes of the function blocks (→ p. 184)):

- **ERR_INVALID_VALUE**: Error: At least one input parameter is invalid or outside the value range.
- **ERR_INTERNAL**: Error: Internal system error
  - Contact the ifm Service Center!
- **ERR_UNDEFINED**: Error: Unknown error
  - Contact the ifm Service Center!
- **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.
- **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.
- **DIAG_INVALID_VALUE**: At least one input parameter is invalid or exceeds the permissible area.
- **DIAG_INTERNAL**: Internal system error.
- **DIAG_ACCESS**: FB/Function cannot access the required resource; Resource is blocked by another task.
- **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_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_STUCK_AT_HIGH**: Error: Signal frozen, signal state high.
- **DIAG_STUCK_AT_LOW**: Error: Signal frozen, signal state low.
- **DIAG_NO_CALIB**: The selected resource has no valid calibration.
  - The displayed values are maybe faulty.
- **DIAG_OVERLOAD_CURRENT**: Maximum current exceeded.
- **ERR_OVERLOAD_CURRENT**: Error: Maximum current exceeded.
- **DIAG_AT_GROUP_OUTPUT**: At least one output of the output group is in an error state.
- **ERR_AT_GROUP_OUTPUT**: Error: At least one output of the output group is in an error state.
### 10.6.2 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILTER_OUTPUT_GROUP</td>
<td>No change of settings</td>
<td>UNCHANGED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILTER_12</td>
</tr>
</tbody>
</table>

### 10.6.3 MODE_OUTPUT_GROUP (ENUM)

Operating type of the output group

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE_OUTPUT_GROUP</td>
<td>Operating type of the output group</td>
<td>UNCHANGED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT_DIGITAL_CS0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONITOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Setting remains unchanged</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Digital output without</td>
</tr>
<tr>
<td></td>
<td></td>
<td>diagnostics and without</td>
</tr>
<tr>
<td></td>
<td></td>
<td>protection; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No parameters or process</td>
</tr>
<tr>
<td></td>
<td></td>
<td>data are written. Only the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FB output data is updated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For use in a PLC application to which the resource does not belong.</td>
</tr>
</tbody>
</table>

© ifm electronic gmbh www.ifm.com
10.7 Library ifmOutHBridge

The library contains function blocks (POU) to control extended output functions via an HBridge.
10.7.1 HBridge

Function block type: Function block (FB)

Library: ifmOutHBridge.library
Symbol in CODESYS:

![HBridge diagram]

Description

The FB configures and controls a pair of output channels in the "HBridge" operating type to control a motor.

Input parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xResetError</td>
<td>BOOL</td>
<td>Reset request for an occurring error</td>
<td>FALSE: When switching from FALSE to TRUE: Reset request to the lower level system</td>
</tr>
<tr>
<td>uiChannel</td>
<td>UINT</td>
<td>1. Output channel (channel A) of the output channel pair</td>
<td>Group + channel → Data sheet → Note on wiring (→ p. 29) Examples: 703 Group 7 + channel 3 1203 Group 12 + channel 3</td>
</tr>
<tr>
<td>uiFrequency</td>
<td>UINT</td>
<td>PWM frequency of the output signal in [Hz]</td>
<td>→ Data sheet</td>
</tr>
<tr>
<td>xDirection</td>
<td>BOOL</td>
<td>The direction in which the current flows via the bridge connections. Determines the direction of rotation of the connected motor.</td>
<td>FALSE: PWM Current Sourcing (CSO) is on channel A TRUE: PWM Current Sourcing (CSO) is on channel B</td>
</tr>
<tr>
<td>eBrakeMode</td>
<td>MODE_BRAKE</td>
<td>Brake mode that applies when the direction of rotation is changed or when stopping</td>
<td>→ MODE_BRAKE (ENUM) (→ p. 155)</td>
</tr>
<tr>
<td>eBrakeValue</td>
<td>UINT</td>
<td>Pulse/pause ratio of the PWM output signal at the corresponding current sinking output of the bridge in [%] The input is only relevant in the eBrakeModes that end with &quot;_DYNAMIC&quot; (= dynamic brake). permissible = 0...1</td>
<td></td>
</tr>
<tr>
<td>tBrakeTime</td>
<td>TIME</td>
<td>Indicates the braking time for the current sinking side of the bridge. The input is only relevant in eBrakeModes ending with &quot;_BTIME&quot;. permissible = 0...1 h</td>
<td></td>
</tr>
<tr>
<td>uiValue</td>
<td>UNIT</td>
<td>Pulse/pause ratio of the PWM output signal in [%]</td>
<td>permissible = 0...1</td>
</tr>
</tbody>
</table>
Output parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed</td>
</tr>
</tbody>
</table>
|             |             |                                                                            | TRUE:  
|             |             | • Error 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 still invalid; FB is still processed  |
|             |             |                                                                            | TRUE: FB outputs valid; FB has been processed        |
| uiOutCurrent| UINT        | Measured current at the PWM output during normal operation in [mA]           | available = 0...final value of the measuring range   |
|             |             | When braking, uiOutCurrent is = 0 because no regular current exists in the lowside path. |                                                   |
Diagnostic codes (→ Messages / diagnostic codes of the function blocks (→ p. 184)):

- **STAT_PREPARING**
  State: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.

- **STAT_DONE**
  State: FB/Function has been successfully executed and completed. There are valid results on the outputs.

- **ERR_INVALID_VALUE**
  Error: At least one input parameter is invalid or outside the value range.

- **ERR_INTERNAL**
  Error: Internal system error
  - Contact the ifm Service Center!

- **ERR_UNDEFINED**
  Error: Unknown error
  - Contact the ifm Service Center!

- **ERR_SHORT_CIRCUIT**
  Error: Short circuit with GND or VBBx.

- **ERR_STUCK_AT**
  Error: Signal is frozen.

- **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.

- **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.

- **DIAG_INVALID_VALUE**
  At least one input parameter is invalid or exceeds the permissible area.

- **DIAG_INTERNAL**
  Internal system error.

- **DIAG_ACCESS**
  FB/Function cannot access the required resource; Resource is blocked by another task.

- **DIAG_CHANGEOVER_TIME**
  Minimum changeover time for the highside-lowside selection of the drivers has not yet expired.

- **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_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_STUCK_AT**
  Error: Signal is frozen.

- **DIAG_STUCK_AT_HIGH**
  Error: Signal frozen, signal state high.

- **DIAG_STUCK_AT_LOW**
  Error: Signal frozen, signal state low.

- **DIAG_SHORT_CIRCUIT**
  Error: Short circuit with GND or VBBx.

- **DIAG_NO_CALIB**
  The selected resource has no valid calibration. The displayed values are maybe faulty.

- **DIAG_CONTROL_DITHER**
  The requested dither value cannot be set because the calculation from dither and PWM value is higher than 1000 per mill.
## 10.7.2 MODE_BRAKE (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE_BRAKE</td>
<td>Braking mode that is applied when changing the direction (xDirection) or when stopping (uiValue = 0).</td>
<td></td>
</tr>
<tr>
<td>UNCHANGED</td>
<td>Setting remains unchanged</td>
<td></td>
</tr>
<tr>
<td>BRAKE_OFF</td>
<td>No braking. The voltage direction is changed immediately.</td>
<td></td>
</tr>
<tr>
<td>BRAKE_EMCY</td>
<td>Emergency brakes:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• In case of change of direction: Braking only during iBrakeTime.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When stopping: Braking during and after the iBrakeTime is elapsed.</td>
<td></td>
</tr>
<tr>
<td>BRAKE_EMCY_BTIME</td>
<td>Emergency brakes, but only during iBrakeTime.</td>
<td></td>
</tr>
<tr>
<td>BRAKE_DYNAMIC</td>
<td>Like BRAKE_EMCY mode, but dynamic braking with the uiBrakeValue.</td>
<td></td>
</tr>
<tr>
<td>BRAKE_DYNAMIC_BTIME</td>
<td>Like BRAKE_EMCY_BTIME mode, but dynamic braking with the uiBrakeValue.</td>
<td></td>
</tr>
</tbody>
</table>
10.8 Library ifmOutPWM

## Contents

<table>
<thead>
<tr>
<th>Block/Type</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentControl</td>
<td>157</td>
</tr>
<tr>
<td>PWM1000</td>
<td>160</td>
</tr>
<tr>
<td>MODE_CURRENT_CONTROL (ENUM)</td>
<td>164</td>
</tr>
<tr>
<td>MODE_PWM (ENUM)</td>
<td>164</td>
</tr>
</tbody>
</table>

The library function blocks (POU) and enumeration types for pulse width modulation and current control of output channels.
### 10.8.1 CurrentControl

**Function block type:** Function block (FB)

**Library:** ifmIOutPWM.library

**Symbol in CODESYS:**

#### Description

The FB is used to configure and operate a current controlled output. The current control is supported by pulse width modulation (PWM). The configuration of PWM frequency and dither is also done with this FB.

#### Input parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xResetError</td>
<td>BOOL</td>
<td>Reset request for an occurring error</td>
<td>FALSE → TRUE: Reset request to the lower level system</td>
</tr>
<tr>
<td>uiChannel</td>
<td>UINT</td>
<td>Output channel</td>
<td>Group + channel&lt;br&gt;→ Data sheet&lt;br&gt;→ Note on wiring (→ p. 29)</td>
</tr>
<tr>
<td>eMode</td>
<td>MODE_CURRENT_CONTROL</td>
<td>Operating type of the output channel</td>
<td>→ MODE_CURRENT_CONTROL (ENUM) (→ p. 164)</td>
</tr>
<tr>
<td>uiFrequency</td>
<td>UINT</td>
<td>PWM frequency of the output signal in [Hz]</td>
<td>permissible = 0...uiFrequency / 2&lt;br&gt;The value at uiDitherFrequency must be an integer part of the value indicated to uiFrequency. Examples:&lt;br&gt;uiFrequency = 300 Hz&lt;br&gt;uiDitherFrequency = 50 Hz ⇒ 300 / 50 = 6&lt;br&gt;⇒ even factor, valid&lt;br&gt;uiDitherFrequency = 100 Hz ⇒ 300 / 100 = 3&lt;br&gt;⇒ uneven factor, invalid&lt;br&gt;Invalid values are corrected to the value that matches the next lower integer factor.</td>
</tr>
<tr>
<td>uiDitherFrequency</td>
<td>UNIT</td>
<td>Frequency for the dither signal at the PWM output in [Hz]</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>Data type</td>
<td>Description</td>
<td>Possible values</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>--------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>uiDitherValue</td>
<td>UNIT</td>
<td>Peak-to-peak value of the dither signal which overlays with the PWM signal, in [%]</td>
<td>permissible = 0...1 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.</td>
</tr>
<tr>
<td>usiKP</td>
<td>USINT</td>
<td>Proportional component of the output signal</td>
<td>permissible = 0...255</td>
</tr>
<tr>
<td>usiKi</td>
<td>USINT</td>
<td>Integral component of the output signal</td>
<td>0 ... 255</td>
</tr>
<tr>
<td>uiDesiredCurrent</td>
<td>UINT</td>
<td>Default value at the output channel.</td>
<td>When 0 is set, the output is immediately deactivated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 ... 65535</td>
</tr>
</tbody>
</table>

**Output parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>xPrepared</td>
<td>BOOL</td>
<td>State of the FB outputs</td>
<td>FALSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE</td>
</tr>
<tr>
<td>uiCurrent</td>
<td>UINT</td>
<td>Output current signal in [mA]</td>
<td>available = 0...final value of the measuring range</td>
</tr>
<tr>
<td>uiPWMRatio</td>
<td>UINT</td>
<td>PWM pulse ration calculated by the PI controller in [%]</td>
<td></td>
</tr>
</tbody>
</table>
Diagnostic codes (→ Messages / diagnostic codes of the function blocks (→ p. 184)):

- **ERR_INVALID_VALUE**  Error: At least one input parameter is invalid or outside the value range.
- **ERR_INTERNAL**  Error: Internal system error
  - Contact the ifm Service Center!
- **ERR_UNDEFINED**  Error: Unknown error
  - Contact the ifm Service Center!
- **ERR_SHORT_CIRCUIT**  Error: Short circuit with GND or VBBx.
- **ERR_STUCK_AT**  Error: Signal is frozen.
- **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.
- **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.
- **DIAG_INVALID_VALUE**  At least one input parameter is invalid or exceeds the permissible area.
- **DIAG_INTERNAL**  Internal system error.
- **DIAG_ACCESS**  FB/Function cannot access the required resource; Resource is blocked by another task.
- **DIAG_CHANGEOVER_TIME**  Minimum changeover time for the highside-lowside selection of the drivers has not yet expired.
- **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_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_STUCK_AT**  Error: Signal is frozen.
- **DIAG_STUCK_AT_HIGH**  Error: Signal frozen, signal state high.
- **DIAG_STUCK_AT_LOW**  Error: Signal frozen, signal state low.
- **DIAG_SHORT_CIRCUIT**  Error: Short circuit with GND or VBBx.
- **DIAG_NO_CALIB**  The selected resource has no valid calibration.
  - The displayed values are maybe faulty.
- **DIAG_CONTROL_DITHER**  The requested dither value cannot be set because the calculation from dither and PWM value is higher than 1000 per mill.
10.8.2 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.
### Input parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xResetError</td>
<td>BOOL</td>
<td>Reset request for an occurring error</td>
<td>TRUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When switching from FALSE to TRUE, Reset request to the lower level system</td>
<td></td>
</tr>
<tr>
<td>uiChannel</td>
<td>UINT</td>
<td>Input channel</td>
<td>Group + channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Examples:</td>
<td>403</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>502</td>
</tr>
<tr>
<td>eMode</td>
<td>MODE_PWM</td>
<td>Operating type of the output channel</td>
<td>MODE_OUTPUT_GROUP (ENUM) (→ p. 150)</td>
</tr>
<tr>
<td>uiFrequency</td>
<td>UINT</td>
<td>PWM frequency of the output signal in [Hz]</td>
<td>Data sheet</td>
</tr>
<tr>
<td>uiValue</td>
<td>UNIT</td>
<td>Pulse/pause ration of the PWM output signal in [%]</td>
<td>permissible = 0...1</td>
</tr>
<tr>
<td>uiDitherFrequency</td>
<td>UNIT</td>
<td>Frequency for the dither signal at the PWM output in [Hz]</td>
<td>permissible = 0...uiFrequency / 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The value at uiDitherFrequency must be an integer part of the value indicated to uiFrequency. Examples:</td>
<td>even factor, valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>uiFrequency = 300 Hz</td>
<td>uiDitherFrequency = 50 Hz ⇒ 300 /50 = 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>uiDitherFrequency = 100 Hz</td>
<td>uiDitherFrequency = 100 Hz ⇒ 300 /100 = 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>invalid</td>
<td>Invalid values are corrected to the value that matches the next lower integer factor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Examples:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>uiDitherValue</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Peak-to-peak value of the dither signal which overlays with the PWM signal, in [%]</td>
<td>permissible = 0...1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>
## Output parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| xError       | BOOL       | Indication if an error occurred during the FB execution                      | FALSE: No error occurred or the FB is still being executed  
|              |            |                                                                             | TRUE:  
|              |            | - Error 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 still invalid; FB is still processed  
|              |            |                                                                             | TRUE: FB outputs valid; FB has been processed |
| uiGroupCurrent | UINT      | Measured output current of the entire group in [mA]                         | available = 0...final value of the measuring range |
| xGroupState  | BOOL       | Return value activation state of the selected output group                  | FALSE: Output group is deactivated            |
|              |            | ! The state may deviate from the required output state if e.g. a safety function has deactivated an output group due to an error. | TRUE: Output value is activated               |
Diagnostic codes (→ Messages / diagnostic codes of the function blocks (→ p. 184)):

- **ERR_INVALID_VALUE** Error: At least one input parameter is invalid or outside the value range.
- **ERR_INTERNAL** Error: Internal system error
  - Contact the ifm Service Center!
- **ERR_UNDEFINED** Error: Unknown error
  - Contact the ifm Service Center!
- **ERR_SHORT_CIRCUIT** Error: Short circuit with GND or VBBx.
- **ERR_STUCK_AT** Error: Signal is frozen.
- **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.
- **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.
- **DIAG_INVALID_VALUE** At least one input parameter is invalid or exceeds the permissible area.
- **DIAG_INTERNAL** Internal system error.
- **DIAG_ACCESS** FB/Function cannot access the required resource; Resource is blocked by another task.
- **DIAG_CHANGEover_TIME** Minimum changeover time for the highside-lowside selection of the drivers has not yet expired.
- **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_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_STUCK_AT** Error: Signal is frozen.
- **DIAG_STUCK_AT_HIGH** Error: Signal frozen, signal state high.
- **DIAG_STUCK_AT_LOW** Error: Signal frozen, signal state low.
- **DIAG_SHORT_CIRCUIT** Error: Short circuit with GND or VBBx.
- **DIAG_NO_CALIB** The selected resource has no valid calibration.
  - The displayed values are maybe faulty.
- **DIAG_CONTROL_DITHER** The requested dither value cannot be set because the calculation from dither and PWM value is higher than 1000 per mill.
### 10.8.3 MODE_CURRENT_CONTROL (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE_CURRENT_CONTROL</td>
<td>Operating mode of the output</td>
<td>UNCHANGED: Setting is maintained</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT_CURRENT_CSO: Output for current control without diagnostics and without protection; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT_CURRENT_CSO_DIAG: Output for current control with diagnostics and without protection; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT_CURRENT_CSO_DIAG_PROT: Output for current control with diagnostics and protection; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

### 10.8.4 MODE_PWM (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE_PWM</td>
<td>Operating mode of the output</td>
<td>UNCHANGED: Setting is maintained</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT_PWM_CSI: PWM output without diagnostics; CSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT_PWM_CSO: PWM output without diagnostics; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT_PWM_CSO_DIAG: PWM output with diagnostics and without protection; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT_PWM_CSO_DIAG_PROT: PWM output with diagnostics and with protection; CSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>
10.9 Library ifmRawCAN.library

<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN_Enable</td>
<td>166</td>
</tr>
<tr>
<td>CAN_Recover</td>
<td>168</td>
</tr>
<tr>
<td>CAN_RemoteRequest</td>
<td>170</td>
</tr>
<tr>
<td>CAN_RemoteResponse</td>
<td>172</td>
</tr>
<tr>
<td>CAN_Rx</td>
<td>174</td>
</tr>
<tr>
<td>CAN_RxMask</td>
<td>176</td>
</tr>
<tr>
<td>CAN_RxRange</td>
<td>178</td>
</tr>
<tr>
<td>CAN_Tx</td>
<td>180</td>
</tr>
<tr>
<td>CAN_Info (GVL)</td>
<td>182</td>
</tr>
<tr>
<td>CAN_BUS_STATE (STRUCT)</td>
<td>182</td>
</tr>
</tbody>
</table>

8722

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.
10.9.1 CAN_Enable

Function block type: Function block (FB)
Behaviour model: ENABLE
Library: ifmRawCAN.library
Symbol in CODESYS:

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 parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xEnable</td>
<td>BOOL</td>
<td>Control activity of the FB</td>
<td>FALSE: FB is deactivated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: FB is activated</td>
</tr>
<tr>
<td>eChannel</td>
<td>CAN_CHANNEL</td>
<td>Identifier of the CAN Interface</td>
<td>→ CAN_CHANNEL (ENUM) (→ p. 116)</td>
</tr>
<tr>
<td>eBaudrate</td>
<td>CAN_BAUD_RATE</td>
<td>Baud rate of the CAN channel</td>
<td>→ CAN_BAUDRATE (ENUM)</td>
</tr>
</tbody>
</table>
### Output parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed; TRUE: Error occurred, Action could not be executed, Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
</tbody>
</table>

#### Diagnostic codes:

- **STAT_INACTIVE**  
  State: FB/Function is inactive.
- **STAT_DONE**  
  State: 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_ALREADY_SET**  
  Error: Requested baud rate cannot be set because another baud rate has already been defined.
- **ERR_UNDEFINED**  
  Error: Unknown error  
  ► Contact the ifm Service Center!
10.9.2 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 parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xExecute</td>
<td>BOOL</td>
<td>Control execution of the FB</td>
<td>FALSE, TRUE, Other</td>
</tr>
<tr>
<td>eChannel</td>
<td>CAN_CHANNEL</td>
<td>Identifier of the CAN Interface</td>
<td>→ CAN_CHANNEL (ENUM) (→ p. 116)</td>
</tr>
<tr>
<td>usiNumberRetry</td>
<td>USINT</td>
<td>Max. number of retries</td>
<td>E.g. 4</td>
</tr>
<tr>
<td>tinhibitTime</td>
<td>TIME</td>
<td>Time until the CAN interface is started again</td>
<td>E.g. #2ms</td>
</tr>
</tbody>
</table>
**Output parameter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xDone</td>
<td>BOOL</td>
<td>Indication of whether execution of the FB has been successfully completed</td>
<td>FALSE FB is executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE • FB successfully executed • FB can be called again</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE No error occurred or the FB is still being executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE • Error occurred • Action could not be executed • Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>usiRetryCount</td>
<td>USINT</td>
<td>Counter for retries carried out since the last activation of the FB</td>
<td></td>
</tr>
</tbody>
</table>

**Diagnostic codes:**

- **STAT_INACTIVE** State: FB/Function is inactive.
- **STAT_DONE** State: 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!
### 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 parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xExecute</td>
<td>BOOL</td>
<td>Control execution of the FB</td>
<td>FALSE → TRUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FB is executed once</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Other → Other</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No impact on FB processing</td>
</tr>
<tr>
<td>eChannel</td>
<td>CAN_CHANNEL</td>
<td>Identifier of the CAN Interface</td>
<td>→ CAN_CHANNEL (ENUM) (→ p. 116)</td>
</tr>
</tbody>
</table>
| 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:                                 | FALSE → TRUE                          |
|             |           | - Standard Frame (11 bits identifier)                 | Standard Frame*                       |
|             |           | - Extended-Frame (29 bits identifier)                 | Extended Frame                       |
| usiDLC      | UINT      | Number of the data bytes in the CAN message (DLC = Data Length Count) | 0 ... 7 bytes*                         |

* ... preset value
## Output parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xDone</td>
<td>BOOL</td>
<td>Indication of whether execution of the FB has been successfully completed</td>
<td>FALSE: FB is executed&lt;br&gt;TRUE: ▪ FB successfully executed ▪ FB can be called again</td>
</tr>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed&lt;br&gt;TRUE: ▪ Error occurred ▪ Action could not be executed ▪ Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>aData</td>
<td>ARRAY [0...7] OF USINT</td>
<td>Array for storage of the data received</td>
<td></td>
</tr>
</tbody>
</table>

### Diagnostic data:

- **STAT_INACTIVE**
  State: FB/Function is inactive.

- **STAT_DONE**
  State: FB/Function has been successfully executed and completed. There are valid results on the outputs.

- **STAT_BUSY**
  State: 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.
10.9.4 CAN_RemoteResponse

Function block type: Function block (FB)
Behaviour model: ENABLE
Library: ifmRawCAN.library

Symbol in CODESYS:

CAN_RemoteResponse

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 parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xEnable</td>
<td>BOOL</td>
<td>Control activity of the FB</td>
<td>FALSE: FB is deactivated, TRUE: FB is activated</td>
</tr>
<tr>
<td>eChannel</td>
<td>CAN_CHANNEL</td>
<td>Identifier of the CAN Interface</td>
<td>→ CAN_CHANNEL (ENUM) (→ p. 116)</td>
</tr>
<tr>
<td>udlID</td>
<td>UDINT</td>
<td>Identifier of the CAN message</td>
<td>for Standard Frame (11 bits identifier): 0 ... 2047, for Extended-Frame (29 bits identifier): 0 ... 536.870.911</td>
</tr>
<tr>
<td>xExtended</td>
<td>BOOL</td>
<td>Requested frame type: - Standard Frame (11 bits identifier), - Extended-Frame (29 bits identifier)</td>
<td>FALSE: Standard Frame*, TRUE: Extended Frame</td>
</tr>
<tr>
<td>usiDLC</td>
<td>UINT</td>
<td>Number of the data bytes in the CAN message (DLC = Data Length Count)</td>
<td>0, 7 bytes</td>
</tr>
</tbody>
</table>
Output parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB</td>
<td>FALSE: No error occurred or the FB is still being</td>
</tr>
<tr>
<td></td>
<td></td>
<td>execution</td>
<td>executed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: • Error occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Action could not be executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>uiRTR_Cnt</td>
<td>UINT</td>
<td>Number of received remote requests after the last</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FB call</td>
<td></td>
</tr>
</tbody>
</table>

Diagnostic code:
- STAT_INACTIVE State: FB/Function is inactive.
- STAT_DONE State: 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!
10.9.5 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.

Input parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xEnable</td>
<td>BOOL</td>
<td>Control activity of the FB</td>
<td>FALSE - FB is deactivated, TRUE - FB is activated</td>
</tr>
<tr>
<td>eChannel</td>
<td>CAN_CHANNEL</td>
<td>Identifier of the CAN Interface</td>
<td>→ CAN_CHANNEL (ENUM) (→ p. 116)</td>
</tr>
<tr>
<td>xExtended</td>
<td>BOOL</td>
<td>Requested frame type: - Standard Frame (11 bits identifier), - Extended-Frame (29 bits identifier)</td>
<td>FALSE - Standard Frame*, TRUE - Extended Frame</td>
</tr>
<tr>
<td>udiID</td>
<td>UDINT</td>
<td>Identifier of the CAN message</td>
<td>for Standard Frame (11 bits identifier): 0 ... 2047, for Extended-Frame (29 bits identifier): 0 ... 536.870.911</td>
</tr>
</tbody>
</table>
### Output parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed</td>
</tr>
</tbody>
</table>
|                 |                    |                                                                            | TRUE:  
|                 |                    | • Error 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          | UINT               | Number of the data bytes in the CAN message (DLC = Data Length Count)      | 0 bytes*                                    |
|                 |                    |                                                                            | 7 bytes                                     |
| uiAvailable     | UINT               | ▪ Number of received CAN messages since the last FB call                    | 0: No CAN messages received between 2 FB calls |
|                 |                    | ▪ Current CAN message is taken into account                                |                                             |

#### Error codes:

- **STAT_INACTIVE**
  State: FB/Function is inactive.

- **STAT_DONE**
  State: 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!
10.9.6 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:
Pattern: 000 0010 0000
Mask: 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 (→ p. 174)

Input parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xEnable</td>
<td>BOOL</td>
<td>Control activity of the FB</td>
<td>FALSE FB is deactivated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE FB is activated</td>
</tr>
<tr>
<td>eChannel</td>
<td>CAN_CHANNEL</td>
<td>Identifier of the CAN Interface</td>
<td>→ CAN_CHANNEL (ENUM) (→ p. 116)</td>
</tr>
<tr>
<td>xExtended</td>
<td>BOOL</td>
<td>Requested frame type:</td>
<td>FALSE Standard Frame*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Standard Frame (11 bits identifier)</td>
<td>TRUE Extended Frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Extended-Frame (29 bits identifier)</td>
<td></td>
</tr>
<tr>
<td>udlIDSet</td>
<td>UDINT</td>
<td>Preset bit pattern for the masking of the</td>
<td>E.g. 000 0010 0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>identifier of the CAN message</td>
<td></td>
</tr>
<tr>
<td>udlIDMask</td>
<td>UDINT</td>
<td>Bit pattern of the required area</td>
<td>E.g. 000 1111 1111</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 ... bit relevant for selection</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ... bit not relevant for selection</td>
<td></td>
</tr>
</tbody>
</table>

* ... preset value
## Output parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
|           |           |             | TRUE | • Error 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    | UINT      | Number of the data bytes in the CAN message (DLC = Data Length Count) | 0 ... 7 | 0 bytes*   
|           |           |             |     | 7 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  
|           |           |             | 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**: State: FB/Function is inactive.
- **STAT_DONE**: State: 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!
10.9.7 CAN_RxRange

Function block type: Function block (FB)
Behaviour model: ENABLE
Library: ifmRawCAN.library
Symbol in CODESYS: CAN_RxRange

Description

The FB receives CAN messages of a coherent area. The area is defined by an upper and lower limit. The following rules apply to the definition of this area:

- Lower and upper limit:
  - Standard Frames: 0 ... 2047 (11-bit identifier)
  - Extended Frames: 0 ... 536 870 911 (29-bit identifier)
  - The value for the lower limit must be <= 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.

⚠ General behaviour of the FB: → CAN_Rx (→ p. 174)

Input parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xEnable</td>
<td>BOOL</td>
<td>Control activity of the FB</td>
<td>FALSE: FB is deactivated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: FB is activated</td>
</tr>
<tr>
<td>eChannel</td>
<td>CAN_CHANNEL</td>
<td>Identifier of the CAN Interface</td>
<td>→ CAN_CHANNEL (ENUM) (→ p. 116)</td>
</tr>
<tr>
<td>xExtended</td>
<td>BOOL</td>
<td>Requested frame type:</td>
<td>FALSE: Standard Frame*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Standard Frame (11 bits identifier)</td>
<td>TRUE: Extended Frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Extended-Frame (29 bits identifier)</td>
<td></td>
</tr>
<tr>
<td>udiIDStart</td>
<td>UDINT</td>
<td>Start of the required area</td>
<td>E.g. 000 0000 0010</td>
</tr>
<tr>
<td>udiIDStop</td>
<td>UDINT</td>
<td>End of the required area</td>
<td>E.g. 000 0000 1000</td>
</tr>
</tbody>
</table>

* ... preset value
Output parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE - No error occurred or the FB is still being executed, TRUE - Error occurred, Action could not be executed, Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
<tr>
<td>aData</td>
<td>ARRAY [0...7] OF USINT</td>
<td>Array for storage of the data received</td>
<td></td>
</tr>
<tr>
<td>usiDLC</td>
<td>UINT</td>
<td>Number of the data bytes in the CAN message (DLC = Data Length Count)</td>
<td>0 ... 7 bytes</td>
</tr>
<tr>
<td>uiAvailable</td>
<td>UINT</td>
<td>Number of received CAN messages since the last FB call, Current CAN message is taken into account</td>
<td>0 - No CAN messages received between 2 FB calls, n - n CAN messages received</td>
</tr>
<tr>
<td>udiID</td>
<td>UDINT</td>
<td>Identifier of the CAN message</td>
<td>• for Standard Frame (11 bits identifier): 0 ... 2047, • for Extended-Frame (29 bits identifier): 0 ... 536,870,911</td>
</tr>
</tbody>
</table>

Diagnostic codes:
- **STAT_INACTIVE** State: FB/Function is inactive.
- **STAT_DONE** State: 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!
## 10.9.8 CAN_Tx

**Function block type:** Function block (FB)  
**Behaviour model:** ENABLE  
**Library:** ifmRawCAN.library

### 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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| xEnable    | BOOL            | Control activity of the FB                            | FALSE  
|            |                 |                                                       | FB is deactivated                        |
|            |                 |                                                       | TRUE  
|            |                 |                                                       | FB is activated                          |
| eChannel   | CAN_CHANNEL     | Identifier of the CAN Interface                       | → CAN_CHANNEL (ENUM) (→ p. 116)          |
| udidID     | 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:                                 | FALSE  
|            |                 | - Standard Frame (11 bits identifier)                 | Standard Frame*                          |
|            |                 | - Extended-Frame (29 bits identifier)                 | TRUE  
|            |                 |                                                       | Extended Frame                          |
| usiDLC     | UINT            | Number of the data bytes in the CAN message (DLC = Data Length Count) | 0                  
|            |                 |                                                       | 0 bytes*                               |
|            |                 |                                                       | 7  
|            |                 |                                                       | 7 bytes                                |
| aData      | ARRAY [0...7] OF USINT | Array with the data to be sent                                                |

* ... preset value
Output parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| xError      | BOOL       | Indication if an error occurred during the FB execution | FALSE: No error occurred or the FB is still being executed  
              |            |                                                  | TRUE:  
              |            | • Error occurred  
              |            | • Action could not be executed  
              |            | • Note diagnostic information                          |
| eDiagInfo   | DIAG_INFO  | Diagnostic information                           | → List below (diagnostic codes:)                     |

Diagnostic codes:
- STAT_INACTIVE State: FB/Function is inactive.
- STAT_DONE State: 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!
## 10.9.9 CAN_Info (GVL)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data type</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>eBusState</td>
<td>Status of the CAN interface to CiA 11898</td>
<td>→ CAN_BUS_STATE (STRUCT) (→ p. 182)</td>
<td>Undefined</td>
</tr>
<tr>
<td>uiBaudRate</td>
<td>Current baud rate</td>
<td>UINT</td>
<td>0*...65535.</td>
</tr>
<tr>
<td>udiRxCount</td>
<td>Counter for all messages detected at the CAN interface</td>
<td>UINT</td>
<td>0*...65535.</td>
</tr>
<tr>
<td>uiErrorCntRx</td>
<td>Error counter Rx (receive)</td>
<td>UINT</td>
<td>0*...65535.</td>
</tr>
<tr>
<td>uiErrorCntTx</td>
<td>Error counter Tx (send)</td>
<td>UINT</td>
<td>0*...65535.</td>
</tr>
<tr>
<td>xWarningRx</td>
<td>Warning signal for error counter Rx</td>
<td>BOOL</td>
<td>FALSE* uiErrorCntRx &lt; 96</td>
</tr>
<tr>
<td>xWarningTx</td>
<td>Warning signal for error counter Tx</td>
<td>BOOL</td>
<td>TRUE uiErrorCntRx &gt;= 96</td>
</tr>
</tbody>
</table>

* = preset value

## 10.9.10 CAN_BUS_STATE (STRUCT)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible values</th>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN_BUS_STATE</td>
<td>State of the CAN interface</td>
<td>UNDEFINED Interface not available or not configured</td>
<td>INT</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ERROR_ACTIVE Error counter Tx/Rx &lt;= 127</td>
<td>INT</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ERROR_PASSIVE Error counter Tx/Rx &gt; 127 and Error counter Tx &gt; 255</td>
<td>INT</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BUS_OFF Error counter Tx &gt; 255</td>
<td>INT</td>
<td>65535</td>
</tr>
</tbody>
</table>
11 Troubleshooting

11.1 Error classes

An error is classified according to its possible impact. The error class determines how the system reacts when a specific error occurs.

<table>
<thead>
<tr>
<th>Error class</th>
<th>Description</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Fatal Error</td>
<td>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.</td>
<td>• The controller deactivates components allocated to the PLC. • The controller deactivates the concerned PLC. • The controller saves the information in the error log.</td>
</tr>
<tr>
<td>B Serious error</td>
<td>One or several PLCs can no longer be executed.</td>
<td>• The controller deactivates components allocated to the PLC. • The controller deactivates the concerned PLC. • The controller saves the information in the error log.</td>
</tr>
<tr>
<td>C Component errors</td>
<td>Error in a controller component; The function of one or several components of the controller is no longer guaranteed.</td>
<td>• 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.</td>
</tr>
<tr>
<td>D Periphery errors</td>
<td>Errors on or in the periphery; a function can no longer be executed.</td>
<td>• 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).</td>
</tr>
</tbody>
</table>
11.2 Error messages

(Most) FBs provide, among others, the following signals at their outputs.
► Evaluate these signals in the application!

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>BOOL</td>
<td>Indication if an error occurred during the FB execution</td>
<td>FALSE: No error occurred or the FB is still being executed; TRUE: Error occurred; Action could not be executed; Note diagnostic information</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>DIAG_INFO</td>
<td>Diagnostic information</td>
<td>→ List below (diagnostic codes:)</td>
</tr>
</tbody>
</table>

Lists of diagnostic codes are part of the function block descriptions → ifm function libraries (→ p. 105)

11.3 Messages / diagnostic codes of the function blocks

Status/diagnostic/error messages of the function blocks are defined in the global Enum DIAG_INFO. They have one of the following prefixes depending on the type of message:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Type of message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT</td>
<td>Status message</td>
<td>Status messages contain information about the condition of the function block during the normal procedure.</td>
</tr>
<tr>
<td>DIAG</td>
<td>Diagnostic message</td>
<td>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.</td>
</tr>
<tr>
<td>ERR</td>
<td>Error message</td>
<td>Error messages contain information about a failure event. They must be reset in the application after the failure event has disappeared.</td>
</tr>
</tbody>
</table>

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 (→ p. 105)
## 12 Appendix

### Contents

Directory structure and file overview ............................................................... 185
ifm behaviour models for function blocks ......................................................... 186

### 12.1 Directory structure and file overview

The following directories and files are stored in the device:

<table>
<thead>
<tr>
<th>Data name / path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apps</td>
<td>Folder</td>
</tr>
<tr>
<td> standard.app</td>
<td>Application non-safe</td>
</tr>
<tr>
<td> safe.app</td>
<td>Application safe</td>
</tr>
<tr>
<td>os</td>
<td>Folder</td>
</tr>
<tr>
<td> ifmOS.ifm</td>
<td>ifmOS</td>
</tr>
<tr>
<td>boot</td>
<td>Folder</td>
</tr>
<tr>
<td> boot.ifm</td>
<td>Bootloader</td>
</tr>
<tr>
<td>sis</td>
<td>Folder</td>
</tr>
<tr>
<td> sissys.ifm</td>
<td>SIS-SYS</td>
</tr>
<tr>
<td>cfg</td>
<td>Folder</td>
</tr>
<tr>
<td> comconf.cfg</td>
<td>Communication configuration</td>
</tr>
<tr>
<td> memconf.ifm</td>
<td>Memory configuration</td>
</tr>
<tr>
<td>retain</td>
<td>Folder</td>
</tr>
<tr>
<td> standard.ret</td>
<td>Application retain non-safe</td>
</tr>
<tr>
<td> standard.mb</td>
<td>Application memory bytes non-safe</td>
</tr>
<tr>
<td> safe.ret</td>
<td>Application retain safe</td>
</tr>
<tr>
<td> safe.mb</td>
<td>Application memory bytes non-safe</td>
</tr>
<tr>
<td>data</td>
<td>Folder</td>
</tr>
<tr>
<td> <em>.</em></td>
<td>Memory space for user-defined data</td>
</tr>
<tr>
<td>compat</td>
<td>Folder</td>
</tr>
<tr>
<td> compat.ifm</td>
<td>Compatibility File</td>
</tr>
<tr>
<td>cmd</td>
<td>Folder</td>
</tr>
<tr>
<td> cmd.ifm</td>
<td>Command File</td>
</tr>
<tr>
<td>info</td>
<td>Folder</td>
</tr>
<tr>
<td> devinfo.txt</td>
<td>Device information</td>
</tr>
<tr>
<td> swinfo.txt</td>
<td>Software information</td>
</tr>
</tbody>
</table>
This chapter describes the ifm behaviour models for function blocks.

### 12.2.1 General

Ifm function blocks feature the following outputs to return status and error information:

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xError</td>
<td>TRUE: An error has occurred.</td>
</tr>
<tr>
<td></td>
<td>FALSE: No error has occurred.</td>
</tr>
<tr>
<td>eDiagInfo</td>
<td>Diagnostic/error information → Messages / diagnostic codes of the function blocks → p. 184</td>
</tr>
</tbody>
</table>

All inputs and outputs in the function block that belong to the ifm behaviour model are featured at the top.

### 12.2.2 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.

**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:

<table>
<thead>
<tr>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>numerical</td>
<td>0 / 0.0</td>
</tr>
<tr>
<td>String</td>
<td>Empty string</td>
</tr>
<tr>
<td>BOOL/Bit</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
12.2.3 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.

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:

<table>
<thead>
<tr>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>numerical</td>
<td>0 / 0.0</td>
</tr>
<tr>
<td>String</td>
<td>Empty string</td>
</tr>
<tr>
<td>BOOL/Bit</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
13 Glossary of Terms

A

Address
This is the "name" of the bus participant. All participants need a unique address so that the signals can be exchanged without problem.

Application
Software that is programmed by the manufacturer into the machine specifically for the application. The software usually contains logic sequences, limit values and expressions to control the corresponding inputs and outputs, calculations and decisions.

Architecture
Specific configuration of hardware and/or software elements in a system.

B

Baud
Baud, abbrev.: Bd = unit for the data transmission speed. Do not confuse baud with "bits per second" (bps, bits/s). Baud indicates the number of changes of state (steps, cycles) per second over a transmission length. But it is not defined how many bits per step are transmitted. The name baud can be traced back to the French inventor J. M. Baudot whose code was used for telex machines.

1 MBd = 1024 x 1024 Bd = 1 048 576 Bd

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

Bus
Serial data transmission of several participants on the same cable.

C

CAN
CAN = Controller Area Network
CAN is a priority-controlled fieldbus system for large data volumes. There are several higher-level protocols that are based on CAN, e.g. 'CANopen' or 'J1939'.

CAN stack
CAN stack = software component that deals with processing CAN messages.

CiA
CiA = CAN in Automation e.V.
User and manufacturer organisation in Germany / Erlangen. Definition and control body for CAN and CAN-based network protocols.
Homepage → www.can-cia.org
**CiA DS 304**

DS = Draft Standard
CANopen device profile for safety communication

**CiA DS 401**

DS = Draft Standard
CANopen device profile for binary and analogue I/O modules

**CiA DS 402**

DS = Draft Standard
CANopen device profile for drives

**CiA DS 403**

DS = Draft Standard
CANopen device profile for drives

**CiA DS 404**

DS = Draft Standard
CANopen device profile for measurement and control technology

**CiA DS 405**

DS = Draft Standard
CANopen specification of the interface to programmable controllers (IEC 61131-3)

**CiA DS 406**

DS = Draft Standard
CANopen device profile for encoders

**CiA DS 407**

DS = Draft Standard
CANopen application profile for local public transport

**Clamp 15**

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

**COB ID**

COB = Communication Object
ID = Identifier
ID of a CANopen communication object
Corresponds to the identifier of the CAN message with which the communication project is sent via the CAN bus.

**CODESYS**

CODESYS® is a registered trademark of 3S – Smart Software Solutions GmbH, Germany.
'CODESYS for Automation Alliance' associates companies of the automation industry whose hardware devices are all programmed with the widely used IEC 61131-3 development tool CODESYS®.
Homepage → www.codesys.com
CSI

CSI = Current Sinking

Current Sinking Input
In = connection of binary input n
(S) = sensor

Binary input block diagram, plus-switching for positive sensor signal
Input = open ⇒ Signal = Low (GND)
→ Low-Side Input (B_L)

Current Sinking Output
Qn = connection of output n
(L) = load

Output block diagram, minus-switching (B_L) for negative output signal
Low-Side Output (B_L)

CSO

CSO = Current Sourcing

Current Sourcing Input
In = connection of binary input n
(S) = Sensor

Binary input block diagram, minus-switching for negative sensor signal:
Input = open ⇒ Signal = High (Supply)
→ High-Side Input (B_H)

Current Sourcing Output
Qn = connection of output n
(L) = load

Output block diagram, plus-switching for positive output signal
High-Side Output (B_H)

CSV file

CSV = Comma Separated Values (also: Character Separated Values)
A CSV file is a text file for storing or exchanging simply structured data. The file extension is .csv.

Example: Source table with numerical values:

<table>
<thead>
<tr>
<th>value 1.0</th>
<th>value 1.1</th>
<th>value 1.2</th>
<th>value 1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>value 2.0</td>
<td>value 2.1</td>
<td>value 2.2</td>
<td>value 2.3</td>
</tr>
<tr>
<td>value 3.0</td>
<td>value 3.1</td>
<td>value 3.2</td>
<td>value 3.3</td>
</tr>
</tbody>
</table>

This results in the following CSV file:

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

Cycle time

This is the time for a cycle. The PLC program performs one complete run.
Depending on event-controlled branchings in the program this can take longer or shorter.
## Glossary of Terms

### D

#### Data type

Depending on the data type, values of different sizes can be stored.

<table>
<thead>
<tr>
<th>Data type</th>
<th>min. value</th>
<th>max. value</th>
<th>size in the memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>FALSE</td>
<td>TRUE</td>
<td>8 bits = 1 byte</td>
</tr>
<tr>
<td>BYTE</td>
<td>0</td>
<td>255</td>
<td>8 bits = 1 byte</td>
</tr>
<tr>
<td>WORD</td>
<td>0</td>
<td>65 535</td>
<td>16 bits = 2 bytes</td>
</tr>
<tr>
<td>DWORD</td>
<td>0</td>
<td>4 294 967 295</td>
<td>32 bits = 4 bytes</td>
</tr>
<tr>
<td>SINT</td>
<td>-128</td>
<td>127</td>
<td>8 bits = 1 byte</td>
</tr>
<tr>
<td>USINT</td>
<td>0</td>
<td>255</td>
<td>8 bits = 1 byte</td>
</tr>
<tr>
<td>INT</td>
<td>-32 768</td>
<td>32 767</td>
<td>16 bits = 2 bytes</td>
</tr>
<tr>
<td>UINT</td>
<td>0</td>
<td>65 535</td>
<td>16 bits = 2 bytes</td>
</tr>
<tr>
<td>DINT</td>
<td>-2 147 483 648</td>
<td>2 147 483 647</td>
<td>32 bits = 4 bytes</td>
</tr>
<tr>
<td>UDINT</td>
<td>0</td>
<td>4 294 967 295</td>
<td>32 bits = 4 bytes</td>
</tr>
<tr>
<td>REAL</td>
<td>-3.402823466 • 10^{38}</td>
<td>3.402823466 • 10^{38}</td>
<td>32 bits = 4 bytes</td>
</tr>
<tr>
<td>ULINT</td>
<td>0</td>
<td>18 446 744 073 709 551 615</td>
<td>64 Bit = 8 Bytes</td>
</tr>
<tr>
<td>STRING</td>
<td></td>
<td></td>
<td>number of char. + 1</td>
</tr>
</tbody>
</table>

#### DC

Direct Current

#### Diagnosis

During the diagnosis, the "state of health" of the device is checked. It is to be found out if and what faults are given in the device.

Depending on the device, the inputs and outputs can also be monitored for their correct function.
- wire break,
- short circuit,
- value outside range.

For diagnosis, configuration and log data can be used, created during the "normal" operation of the device.

The correct start of the system components is monitored during the initialisation and start phase.

Errors are recorded in the log file.

For further diagnosis, self-tests can also be carried out.

#### Dither

Dither is a component of the PWM signals to control hydraulic valves. It has shown for electromagnetic drives of hydraulic valves that it is much easier for controlling the valves if the control signal (PWM pulse) is superimposed by a certain frequency of the PWM frequency. This dither frequency must be an integer part of the PWM frequency.

#### DLC

Data Length Code = in CANopen the number of the data bytes in a message.

For SDO: DLC = 8
Glossary of Terms

DRAM
DRAM = Dynamic Random Access Memory.
Technology for an electronic memory module with random access (Random Access Memory, RAM). The memory element is a capacitor which is either charged or discharged. It becomes accessible via a switching transistor and is either read or overwritten with new contents. The memory contents are volatile: the stored information is lost in case of lacking operating voltage or too late restart.

DTC
DTC = Diagnostic Trouble Code = error code
In the protocol J1939 faults and errors well be managed and reported via assigned numbers – the DTCs.

E

ECU
(1) Electronic Control Unit = control unit or microcontroller
(2) Engine Control Unit = control device of an engine

EDS-file
EDS = Electronic Data Sheet, e.g. for:
• File for the object directory in the CANopen master,
• CANopen device descriptions.
Via EDS devices and programs can exchange their specifications and consider them in a simplified way.

Embedded software
System software, basic program in the device, virtually the →runtime system.
The firmware establishes the connection between the hardware of the device and the application program. The firmware is provided by the manufacturer of the controller as a part of the system and cannot be changed by the user.

EMC
EMC = Electro Magnetic Compatibility.
According to the EC directive (2004/108/EEC) concerning electromagnetic compatibility (in short EMC directive) requirements are made for electrical and electronic apparatus, equipment, systems or components to operate satisfactorily in the existing electromagnetic environment. The devices must not interfere with their environment and must not be adversely influenced by external electromagnetic interference.

EMCY
Abbreviation for emergency
Message in the CANopen protocol with which errors are signalled.

Ethernet
Ethernet is a widely used, manufacturer-independent technology which enables data transmission in the network at a speed of 10...10 000 million bits per second (Mbps). Ethernet belongs to the family of so-called “optimum data transmission” on a non exclusive transmission medium. The concept was developed in 1972 and specified as IEEE 802.3 in 1985.
Glossary of Terms

**EUC**

EUC = Equipment Under Control.

EUC is equipment, machinery, apparatus or plant used for manufacturing, process, transportation, medical or other activities (→ IEC 61508-4, section 3.2.3). Therefore, the EUC is the set of all equipment, machinery, apparatus or plant that gives rise to hazards for which the safety-related system is required. If any reasonably foreseeable action or inaction leads to hazards with an intolerable risk arising from the EUC, then safety functions are necessary to achieve or maintain a safe state for the EUC. These safety functions are performed by one or more safety-related systems.

**F**

**FiFo**

FIFO (First In, First Out) = Operating principle of the stack memory: The data packet that was written into the stack memory first, will also be read first. Each identifier has such a buffer (queue).

**Flash memory**

Flash ROM (or flash EPROM or flash memory) combines the advantages of semiconductor memory and hard disks. Similar to a hard disk, the data are however written and deleted blockwise in data blocks up to 64, 128, 256, 1024, ... bytes at the same time.

**Advantages of flash memories**

- The stored data are maintained even if there is no supply voltage.
- Due to the absence of moving parts, flash is noiseless and insensitive to shocks and magnetic fields.

**Disadvantages of flash memories**

- A storage cell can tolerate a limited number of write and delete processes:
  - Multi-level cells: typ. 10 000 cycles
  - Single level cells: typ. 100 000 cycles
- Given that a write process writes memory blocks of between 16 and 128 Kbytes at the same time, memory cells which require no change are used as well.

**FRAM**

FRAM, or also FeRAM, means Ferroelectric Random Access Memory. The storage operation and erasing operation is carried out by a polarisation change in a ferroelectric layer.

**Advantages of FRAM as compared to conventional read-only memories:**

- non-volatile,
- compatible with common EEPROMs, but:
  - access time approx. 100 ns,
  - nearly unlimited access cycles possible.

**H**

**Heartbeat**

The participants regularly send short signals. In this way the other participants can verify if a participant has failed.

**HMI**

HMI = Human Machine Interface
Glossary of Terms

I

ID
ID = Identifier
Name to differentiate the devices / participants connected to a system or the message packets transmitted between the participants.

IEC 61131
Standard: Basics of programmable logic controllers
• Part 1: General information
• Part 2: Production equipment requirements and tests
• Part 3: Programming languages
• Part 5: Communication
• Part 7: Fuzzy Control Programming

IEC user cycle
IEC user cycle = PLC cycle in the CODESYS application program.

Instructions
Superordinate word for one of the following terms:
installation instructions, data sheet, user information, operating instructions, device manual, installation information, online help, system manual, programming manual, etc.

Intended use
Use of a product in accordance with the information provided in the instructions for use.

IP address
IP = Internet Protocol.
The IP address is a number which is necessary to clearly identify an internet participant. For the sake of clarity the number is written in 4 decimal values, e.g. 127.215.205.156.

ISO 11898
Standard: Road vehicles – Controller area network
• Part 1: Data link layer and physical signalling
• Part 2: High-speed medium access unit
• Part 3: Low-speed, fault-tolerant, medium dependent interface
• Part 4: Time-triggered communication
• Part 5: High-speed medium access unit with low-power mode

ISO 11992
Standard: Interchange of digital information on electrical connections between towing and towed vehicles
• Part 1: Physical and data-link layers
• Part 2: Application layer for brakes and running gear
• Part 3: Application layer for equipment other than brakes and running gear
• Part 4: Diagnostics
ISO 16845
Standard: Road vehicles – Controller area network (CAN) – Conformance test plan

J

J1939
→ SAE J1939

L

LED
LED = Light Emitting Diode.
Light emitting diode, also called luminescent diode, an electronic element of high coloured luminosity at small volume with negligible power loss.

Link
A link is a cross-reference to another part in the document or to an external document.

LSB
Least Significant Bit/Byte

M

MAC-ID
MAC = Manufacturer's Address Code
= manufacturer's serial number.
→ ID = Identifier
Every network card has a MAC address, a clearly defined worldwide unique numerical code, more or less a kind of serial number. Such a MAC address is a sequence of 6 hexadecimal numbers, e.g. "00-0C-6E-D0-02-3F".

Master
Handles the complete organisation on the bus. The master decides on the bus access time and polls the → slaves cyclically.

Misuse
The use of a product in a way not intended by the designer.
The manufacturer of the product has to warn against readily predictable misuse in his user information.

MMI
→ HMI (→ p. 193)
MRAM

MRAM = Magnetoresistive Random Access Memory

The information is stored by means of magnetic storage elements. The property of certain materials is used to change their electrical resistance when exposed to magnetic fields.

Advantages of MRAM as compared to conventional RAM memories:
- non volatile (like FRAM), but:
- access time only approx. 35 ns,
- unlimited number of access cycles possible.

MSB

Most Significant Bit/Byte

N

NMT

NMT = Network Management = (here: in the CANopen protocol).

The NMT master controls the operating states of the NMT slaves.

Node

This means a participant in the network.

Node Guarding

Node = here: network participant

Configurable cyclic monitoring of each →slave configured accordingly. The →master verifies if the slaves reply in time. The slaves verify if the master regularly sends requests. In this way failed network participants can be quickly identified and reported.

Obj / object

Term for data / messages which can be exchanged in the CANopen network.

Object directory

Contains all CANopen communication parameters of a device as well as device-specific parameters and data.

OBV

Contains all CANopen communication parameters of a device as well as device-specific parameters and data.

OPC

OPC = OLE for Process Control

Standardised software interface for manufacturer-independent communication in automation technology

OPC client (e.g. device for parameter setting or programming) automatically logs on to OPC server (e.g. automation device) when connected and communicates with it.
Operating system
Basic program in the device that establishes the connection between the hardware of the device and
the application program.
→ Chapter Software module for the device

Operational
Operating state of a CANopen participant. In this mode →SDOs, →NMT commands and →PDOs can
be transferred.

P

PC card
→PCMCIA card

PCMCIA card
PCMCIA = Personal Computer Memory Card International Association, a standard for expansion
cards of mobile computers.
Since the introduction of the cardbus standard in 1995 PCMCIA cards have also been called PC card.

PDM
PDM = Process and Dialogue Module.
Device for communication of the operator with the machine / plant.

PDO
PDO = Process Data Object.
The time-critical process data is transferred by means of the "process data objects" (PDOs). The
PDOs can be freely exchanged between the individual nodes (PDO linking). In addition it is defined
whether data exchange is to be event-controlled (asynchronous) or synchronised. Depending on the
type of data to be transferred the correct selection of the type of transmission can lead to considerable
relief for the →CAN bus.
According to the protocol, these services are unconfirmed data transmission: it is not checked whether
the receiver receives the message. Exchange of network variables corresponds to a "1 to
n connection" (1 transmitter to n receivers).

PDU
PDU = Protocol Data Unit = protocol data unit.
The PDU is a term from the →CAN protocol →SAE J1939. It refers to a component of the target
address (PDU format 1, connection-oriented) or the group extension (PDU format 2, message-
oriented).

PES
Programmable Electronic System ...
• for control, protection or monitoring,
• dependent for its operation on one or more programmable electronic devices,
• including all elements of the system such as input and output devices.
Glossary of Terms

PGN
PGN = Parameter Group Number
PGN = 6 zero bits + 1 bit reserved + 1 bit data page + 8 bit PDU Format (PF) + 8 PDU Specific (PS).
The parameter group number is a term from the →CAN protocol →SAE J1939.

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

PID controller
The PID controller (proportional–integral–derivative controller) consists of the following parts:
• P = proportional part
• I = integral part
• D = differential part (but not for the controller CR04nn, CR253n).

PLC configuration
Part of the CODESYS user interface.
► The programmer tells the programming system which hardware is to be programmed.
> CODESYS loads the corresponding libraries.
> Reading and writing the periphery states (inputs/outputs) is possible.

Pre-Op
Pre-Op = PRE-OPERATIONAL mode.
Operating status of a CANopen participant. After application of the supply voltage each participant automatically passes into this state. In the CANopen network only →SDOs and →NMT commands can be transferred in this mode but no process data.

Process image
Process image is the status of the inputs and outputs the PLC operates with within one →cycle.
• At the beginning of the cycle the PLC reads the conditions of all inputs into the process image.
  During the cycle the PLC cannot detect changes to the inputs.
• During the cycle the outputs are only changed virtually (in the process image).
• At the end of the cycle the PLC writes the virtual output states to the real outputs.

PWM
PWM = pulse width modulation
The PWM output signal is a pulsed signal between GND and supply voltage.
Within a defined period (PWM frequency) the mark-to-space ratio is varied. Depending on the mark-to-space ratio, the connected load determines the corresponding RMS current.

R

ratiometric
Measurements can also be performed ratiometrically. If the output signal of a sensor is proportional to its supply voltage then via ratiometric measurement (= measurement proportional to the supply) the influence of the supply’s fluctuation can be reduced, in ideal case it can be eliminated.
→ analogue input
RAW-CAN
RAW-CAN means the pure CAN protocol which works without an additional communication protocol on the CAN bus (on ISO/OSI layer 2). The CAN protocol is international defined according to ISO 11898-1 and guarantees in ISO 16845 the interchangeability of CAN chips in addition.

remanent
Remanent data is protected against data loss in case of power failure. The runtime system for example automatically copies the remanent data to a flash memory as soon as the voltage supply falls below a critical value. If the voltage supply is available again, the runtime system loads the remanent data back to the RAM memory. The data in the RAM memory of a controller, however, is volatile and normally lost in case of power failure.

ro
RO = read only for reading only
Unidirectional data transmission: Data can only be read and not changed.

RTC
RTC = Real Time Clock
Provides (batter-backed) the current date and time. Frequent use for the storage of error message protocols.

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

S

SAE J1939
The network protocol SAE J1939 describes the communication on a CAN bus in commercial vehicles for transmission of diagnosis data (e.g. engine speed, temperature) and control information. Standard: Recommended Practice for a Serial Control and Communications Vehicle Network
• Part 2: Agricultural and Forestry Off-Road Machinery Control and Communication Network
• Part 3: On Board Diagnostics Implementation Guide
• Part 5: Marine Stern Drive and Inboard Spark-Ignition Engine On-Board Diagnostics Implementation Guide
• Part 11: Physical Layer – 250 kBits/s, Shielded Twisted Pair
• Part 13: Off-Board Diagnostic Connector
• Part 15: Reduced Physical Layer, 250 kBits/s, Un-Shielded Twisted Pair (UTP)
• Part 21: Data Link Layer
• Part 31: Network Layer
• Part 71: Vehicle Application Layer
• Part 73: Application Layer – Diagnostics
• Part 81: Network Management Protocol

SD card
An SD memory card (short for Secure Digital Memory Card) is a digital storage medium that operates to the principle of flash storage.
Glossary of Terms

**SDO**
SDO = Service Data Object.
The SDO is used for access to objects in the CANopen object directory. 'Clients' ask for the requested data from 'servers'. The SDOs always consist of 8 bytes.

**Examples:**
- Automatic configuration of all slaves via SDOs at the system start,
- Reading error messages from the object directory.

Every SDO is monitored for a response and repeated if the slave does not respond within the monitoring time.

**Self-test**
Test program that actively tests components or devices. The program is started by the user and takes a certain time. The result is a test protocol (log file) which shows what was tested and if the result is positive or negative.

**Slave**
Passive participant on the bus, only replies on request of the master. Slaves have a clearly defined and unique address in the bus.

**stopped**
Operating status of a CANopen participant. In this mode only NMT commands are transferred.

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

**System variable**
Variable to which access can be made via IEC address or symbol name from the PLC.

**T**

**Target**
The target contains the hardware description of the target device for CODESYS, e.g.: inputs and outputs, memory, file locations. Corresponds to an electronic data sheet.

**TCP**
The Transmission Control Protocol is part of the TCP/IP protocol family. Each TCP/IP data connection has a transmitter and a receiver. This principle is a connection-oriented data transmission. In the TCP/IP protocol family the TCP as the connection-oriented protocol assumes the task of data protection, data flow control and takes measures in the event of data loss. (compare: →UDP)

**Template**
A template can be filled with content.
Here: A structure of pre-configured software elements as basis for an application program.
UDP
UDP (User Datagram Protocol) is a minimal connectionless network protocol which belongs to the transport layer of the internet protocol family. The task of UDP is to ensure that data which is transmitted via the internet is passed to the right application. At present network variables based on →CAN and UDP are implemented. The values of the variables are automatically exchanged on the basis of broadcast messages. In UDP they are implemented as broadcast messages, in CAN as →PDOs. According to the protocol, these services are unconfirmed data transmission: it is not checked whether the receiver receives the message. Exchange of network variables corresponds to a "1 to n connection" (1 transmitter to n receivers).

Use, intended
Use of a product in accordance with the information provided in the instructions for use.

Watchdog
In general the term watchdog is used for a component of a system which watches the function of other components. If a possible malfunction is detected, this is either signalled or suitable program branchings are activated. The signal or branchings serve as a trigger for other co-operating system components to solve the problem.
14 Index

A
About this manual ........................................... 6
Access inputs ........................................... 86
Access outputs ........................................... 87
Access to inputs ........................................... 83
Access to outputs ........................................... 84
Activate the access protection for a project ............ 65
Add ifm function libraries to the application ........ 64
Address ........................................... 188
Address assignment in Ethernet networks ............ 71
Address assignment of the outputs .................... 172
Allocate inputs/outputs ................................ 67
Allocate memory partition ................................ 66
Appendix .................................................. 185
Application .............................................. 58, 188
Architecture .............................................. 188
Available memory ........................................ 31

B
Baud ......................................................... 188
Behaviour model ENABLE ................................ 186
Behaviour model EXECUTE ................................ 187
Binary input block diagram plus/minus-switching .... 38
Binary output block diagram plus/minus-switching ... 44
Block diagram ............................................. 27
Block diagram of the supply and of the output deactivation ... 29
Bootsloader .............................................. 57, 188
Bus ......................................................... 188

C
CAN .......................................................... 188
Interfaces and protocols ................................ 54
CAN stack ................................................. 188
CAN_BAUDRATE (ENUM) ................................ 116
CAN_BUS_STATE (STRUCT) ......................... 182
CAN_CHANNEL (ENUM) ................................ 116
CAN_Enable .............................................. 166
CAN_Info (GVL) ........................................... 182
CAN_Recover ............................................. 168
CAN_RemoteRequest .................................... 170
CAN_RemoteResponse ................................... 172
CAN_Rx ..................................................... 174
CAN_RxMask ............................................. 176
CAN_RxRange ............................................ 178
CAN_Tx ...................................................... 180
CANconstants (GVL) ..................................... 116
CANopen
Network Management (NMT) ................................ 91
Send and receive SDO ..................................... 91
Carry out installation ..................................... 19
Check the hardware version of the device ............ 23
Check the operating system version of the device .... 22
CIA ......................................................... 188
CIA DS 304 .............................................. 189
CIA DS 401 .............................................. 189
CIA DS 402 .............................................. 189
CIA DS 403 .............................................. 189
CIA DS 405 .............................................. 189
CIA DS 406 .............................................. 189
CIA DS 407 .............................................. 189
Clamp 15 ................................................. 189
COB ID ...................................................... 189
CODESYS ............................................... 189
CODESYS programming software .................... 19
Complete package for ecocatController CR721S .... 20
Components of the complete package ................. 20
Configure CAN interfaces ................................ 72
Configure Ethernet interface .......................... 70
Configure IEC watchdog ................................ 37
Configure inputs and outputs .......................... 69
Configure interfaces ..................................... 70
Configure PLC ........................................... 66
Configure programming interface ..................... 63
Configure serial interface .............................. 70
Configure task processing ................................ 82
Control device ........................................... 84
Controlling LEDs in the applications .................. 98
COP_GetNodeState ....................................... 106
COP_SDOn read .......................................... 108
COP_SDOn write .......................................... 110
COP_SendNMT ........................................... 112
COUNT_DIRECTION (ENUM) ......................... 125
Create CODESYS project ................................ 59
Create new project with CR721S ...................... 60
Create PLC application .................................. 80
CSI .......................................................... 190
CSO .......................................................... 190
CSV file ..................................................... 190
CurrentControl .......................................... 157
Cycle time .................................................. 190

D
Data transmission for series production ............... 101
Data transmission with TFTP ............................ 102
Data type .................................................. 191
DC .......................................................... 191
Delete application from CR721S ...................... 94
Device supply (technology) ............................ 33
Diagnosis .................................................. 191
Directory structure and file overview ................. 185
Display system information ........................... 104
DIO .......................................................... 191
DLC .......................................................... 191
DRAM ...................................................... 192
DTC .......................................................... 192

E
ECU .......................................................... 192
EDS-file .................................................... 192
Embedded software ....................................... 192
EMC .......................................................... 192
EMCY ....................................................... 192
ENCODER_RESOLUTION (ENUM) ................ 125
Error classes .............................................. 183
## T

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>200</td>
</tr>
<tr>
<td>TCP</td>
<td>200</td>
</tr>
<tr>
<td>Temperature</td>
<td>140</td>
</tr>
<tr>
<td>Template</td>
<td>200</td>
</tr>
<tr>
<td>The the IP parameter of the Ethernet interface</td>
<td>70</td>
</tr>
<tr>
<td>Transfer CODESYS project to device</td>
<td>93</td>
</tr>
<tr>
<td>Transmission of the files with CODESYS</td>
<td>101</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>183</td>
</tr>
<tr>
<td>Types of inputs</td>
<td>38</td>
</tr>
</tbody>
</table>

## U

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>201</td>
</tr>
<tr>
<td>Uninstall package (PC/laptop)</td>
<td>22</td>
</tr>
<tr>
<td>Update package (PC/laptop)</td>
<td>21</td>
</tr>
<tr>
<td>Update the operating system of the device</td>
<td>24</td>
</tr>
<tr>
<td>Update the operating system of the device with the batch file</td>
<td>24</td>
</tr>
<tr>
<td>Use CANopen</td>
<td>91</td>
</tr>
<tr>
<td>Use CODESYS user manual</td>
<td>62</td>
</tr>
<tr>
<td>Use ifm function libraries</td>
<td>83</td>
</tr>
<tr>
<td>Use IO mapping</td>
<td>85</td>
</tr>
<tr>
<td>Use RawCAN (CAN Layer 2)</td>
<td>89</td>
</tr>
<tr>
<td>Use SAE J1939</td>
<td>92</td>
</tr>
<tr>
<td>Use, intended</td>
<td>201</td>
</tr>
</tbody>
</table>

## V

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>V model</td>
<td>14</td>
</tr>
<tr>
<td>via function block</td>
<td>69</td>
</tr>
<tr>
<td>RAW-CAN</td>
<td>77</td>
</tr>
<tr>
<td>via system configuration</td>
<td>69</td>
</tr>
<tr>
<td>CANopen device</td>
<td>75</td>
</tr>
<tr>
<td>CANopen Manager</td>
<td>73</td>
</tr>
<tr>
<td>J1939 manager</td>
<td>76</td>
</tr>
<tr>
<td>Voltage ranges of the on-board system</td>
<td>33</td>
</tr>
</tbody>
</table>

## W

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warnings used</td>
<td>16</td>
</tr>
<tr>
<td>watchdog</td>
<td>201</td>
</tr>
<tr>
<td>Watchdog</td>
<td>37, 201</td>
</tr>
<tr>
<td>What previous knowledge is required?</td>
<td>12</td>
</tr>
</tbody>
</table>
Version: 2016-11-29

ifm electronic gmbh • Friedrichstraße 1 • 45128 Essen
www.ifm.com • Email: info@ifm.com
Service hotline: 0800 / 16 16 16 (only Germany, Mo-Fr 07.00...18.00 h)

ifm Niederlassungen • Sales offices • Agences

D
Niederlassung Nord • 31136 Hildesheim • Tel. 0 51 21 / 76 67-0
Niederlassung West • 45128 Essen • Tel. 02 01 / 3 64 75-0
Niederlassung Mitte-West • 58511 Lüdenscheid • Tel. 0 23 51 / 43 01-0
Niederlassung Süd-West • 64466 Heppenheim • Tel. 0 62 52 / 79 05-0
Niederlassung Baden-Württemberg • 73230 Kirchheim • Tel. 0 70 21 / 80 86-0
Niederlassung Bayern • 82178 Puchheim • Tel. 0 89 / 8 00 91-0
Niederlassung Ost • 07639 Tautenhain • Tel. 0 36 601 / 771-0

A, SL
ifm electronic gmbh • 1120 Wien • Tel. +43 16 17 45 00

AUS
ifm efector Pty ltd. • Mulgrave Vic 3170 • Tel. +61 3 00 365 088

B, L
ifm electronic N.V. • 1731 Zellik • Tel. +32 2 / 4 81 02 20
ifm electronic eood • 1202 Sofia • Tel. +359 2 807 59 69

BR
ifm electronic Ltda. • 03337-000, Sao Paulo SP • Tel. +55 11 / 2672 4800

DK
ifm electronic SpA • Oficina 5032 Comuna de Conchalí • Tel. +55 11 / 2672-1730

E
ifm electronic s. a. • 08620 El Prat de Llobregat • Tel. +34 93 479 30 80

F
ifm electronic s. a. • 93192 Noisy-le-Grand Cedex • Tel. +33 0820 22 30 01

FIN
ifm electronic oy • 00440 Helsinki • Tel. +358 75 329 5000

GB, IRL
ifm electronic Ltd. • Hampton, Middlesex TW12 2HD • Tel. +44 208 / 213-0000

GR
ifm electronic Monoprosopi E.P.E. • 15125 Amaroussio • Tel. +30 210 / 6180090

H
ifm electronic kft. • 9028 Győr • Tel. +36 96 / 518-397

I
ifm electronic s. a. • 20041 Agrate-Brianza (MI) • Tel. +39 039 / 68.99.982

IL
ifm electronic India Branch Office • Kolhapur, 416234 • Tel. +91 231-267 27 70

J
efector co., ltd. • Chiba-shi, Chiba 261-7118 • Tel. +81 043-299-2070

MAL
ifm electronic Sp. z o.o. • 835 54 Bratislava • Tel. +421 2 / 44 87 23 29

MEX
ifm electronic S. de R. L. de C. V. • Monterrey, N. L. 64630 • Tel. +52 81 8040-3535

N
Sivilingeniør J. F. Knudtzen A/S • 1396 Billingstad • Tel. +47 66 / 98 33 50

NL
ifm electronic b.v. • 3843 GA Harderwijk • Tel. +31 341 / 438 438

NZ
ifm efector ltd. • 930 Great South Road Penrose, Auckland • Tel. +64 95 79 69 91

P
ifm electronic s. a. • 4410-136 São Félix da Marinha • Tel. +55 21 / 391 71 71 08

PL
ifm electronic Sp. z o.o. • 40-106 Katowice • Tel. +48 32-608 74 54

RA, ROU
ifm electronic s.r.l. • 1107 Buenos Aires • Tel. +54 11 / 5353 3436

RO
ifm electronic s.r.l. • Sibiu 557260 • Tel. +40 269 224550

RUS
ifm electronic • 105318 Moscow • Tel. +7 495 921-44-14

S
ifm electronic a b • 41250 Göteborg • Tel. +46 31 / 750 23 00

SGP
ifm electronic Pte. Ltd. • Singapore 609 916 • Tel. +65 6562 8661/2/3

SK
ifm electronic s.r.o. • 835 54 Bratislava • Tel. +421 2 / 44 87 23 29

THA
SCM Allianze Co., Ltd. • Bangkok 10 400 • Tel. +66 02 615 4888

TR
ifm electronic Ltd. • Istanbul 34381 Silivri/Istanbul • Tel. +90 212 / 210 50 80

UA
TOV ifm electronic • 02600 Kiev • Tel. +380 44 501 8543

USA
ifm efector inc. • Exton, PA 19341 • Tel. +1 610 / 5 24-2000

VN
ifm electronic • Ho Chi Minh city 700000 • Tel. +84-8-35125177

ZA
ifm electronic (Pty) Ltd. • 0157 Pretoria • Tel. +27 12 345 44 49

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