Original System Manual
Know-How ecomat*mobile*

CODESYS® V2.3

English

7391020 / 01 04 / 2018

Table of Contents

1 About this manual

Contents

202

1.1 Copyright

6088

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

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

1.2 Overview: documentation modules for CRnnnn

22853

The documentation for this devices consists of the following modules:
(Downloads from ifm's website → www.ifm.com)

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

1.3 Which devices are described in this manual?

14403

Technology and methods can differ from device to device.

These instructions apply to the following devices:
• all **ecomat*mobile*** controllers
• PDM: CR10nn
• PCB controller: CS0015

1.4 What do the symbols and formats mean?

203

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

⚠ WARNING

Death or serious irreversible injuries may result.

⚠ CAUTION

Slight reversible injuries may result.

NOTICE

Property damage is to be expected or may result.

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

1.5 How is this documentation structured?

204
1508

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

How to use this manual:

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

In case of malfunctions or uncertainties please contact the manufacturer at:
Contact → www.ifm.com

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:
→ www.ifm.com

1.6 History of the instructions (SEM)

14337

What has been changed in this manual? An overview:

Date	Theme	Change
2017-01-13	Software manual for CODESYS 2.3	hint to download from the ifm homepage removed
2018--03-05	FB INPUT	Value corrected for analogue input resistance measurement (16...30 000 Ω)
2018-04-16	Chapter "CAN: Hardware"	Now added again

2 Templates and demo programs

Contents

13747

2.1 Introduction

11646

2.1.1 What are ifm templates?

11647

They are templates for CODESYS application programs.
These templates are separately available for all programmable **ecomat*mobile*** devices.

Structure of the file names:
`ifm_template_CRnnnn(CAN)_(V1)_(V2).pro`
While the bracket terms have the following meaning:

(CAN)	CAN protocol: • Layer2 • CANopen master • CANopen slave
(V1)	Version (Vxxyyzz) of the CRnnnn device's runtime system
(V2)	Version (Vnn) of the template

The article number in the template must be exactly identical with the article number of the device to be programmed! → Device manual, chapter "Information concerning the software"

Quick reference guide: ifm templates

18057

This is how you find the **ifm** templates:

► In the CODESYS menu [Datei] > open [Neu aus Vorlage...].
> The dialogue [Öffnen] appears.
► Select the following path in the directory tree:
(*Program drive*) > [Programme] > [ifm electronic] > [CoDeSys (Version)] > [Projects] > (*aktuelle Template-DVD*) > (*requested template*)
► Confirm the selection with [Öffnen].
> A new CODESYS project is created.
This project contains all necessary elements and parameter settings for a project that can run on the selected device.
► Adjust this project manually to the application.
When necessary, integrate individual **ifm** demos (→ chapter **ifm demo programs** (→ p. 18)).

2.1.2 What are ifm demo programs?

11648

ifm demo programs are CODESYS examples for individual functions.
In most cases, the examples do not apply to a specific**ifm** device, as far as nothing else is specified.

Structure of the file names:
(device)demo_(V1)_(V2).pro
While the bracket terms have the following meaning:

(Device)	Article no. of the example device
(V1)	Type of demonstration
(V2)	Version (Vnn) of the demo program

Quick reference guide: ifm demo programs

18058

This is how you find the **ifm** demo programs:

► Open [Projekt] > [öffnen] in the CODESYS menu.
> The dialogue [Öffnen] appears.
► Select the following path in the directory tree:
(*Program drive*) > [Programme] > [ifm electronic] > [CoDeSys (Version)] > [Projects] > (*requested demo directory*) > (*requested demo project*)
► Confirm the selection with [Öffnen].
> The window [Objekte kopieren] appears.
► Highlight the elements containing exclusively the requested function.
► Confirm the selection with [OK].
> The highlighted elements from the demo project are inserted in the current project, .
► Adjust the elements of the application and add e.g. to the module PLC_PRG.

2.2 Set up programming system via templates

Contents

18051

ifm offers ready-to-use templates (program templates) for a fast, simple, and complete setting up of the programming system.

970

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

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

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

2.2.1 About the ifm templates

Contents

3981

As a rule the following templates are offered for each unit:

- `ifm_template_CRnnnnLayer2_Vxxyyzz.pro`
  for the operation of the unit with CAN layer 2
- `ifm_template_CRnnnnMaster_Vxxyyzz.pro`
  for the operation of the unit as CANopen master
- `ifm_template_CRnnnnSlave_Vxxyyzz.pro`
  for the operation of the unit as CANopen slave

The templates described here are for:
• CODESYS from version 2.3.9.6
• on the **ecomat*mobile*** DVD "Software, tools and documentation" from version 020000

The templates all have the same structures.

The selection of this program template for CAN operation already is an important basis for a functioning program.

Folder structure in general

3978

The function elements are sorted in the following folders:

Folder	Description
CAN_OPEN	for Controller and PDM, CAN operation as master or slave: contains the FBs for CANopen.
I_O_CONFIGURATION	for Controller, CAN operation with layer 2 or as master or slave: FBs for parameter setting of the operating modes of the inputs and outputs.
PDM_COM_LAYER2	for Controller, CAN operation as layer 2 or as slave: FBs for basis communication via layer 2 between PLC and PDM.
CONTROL_CR10nn	for PDM, CAN operation with layer 2 or as master or slave: Contains FBs for image and key control during operation.
PDM_DISPLAY_SETTINGS	for PDM, CAN operation with layer 2 or as master or slave: Contains FBs for adjusting the monitor.

Programs and functions in the folders of the templates (C)

18048

The above folders contain the following programs and function blocks (all = function elements):

Function elements in the folder CAN_OPEN	Description
CANopen	CAN operation as master: Contains the following parameterised function elements: • CAN1_MASTER_EMCY_HANDLER, • CAN1_MASTER_STATUS, • SELECT_NODESTATE (→ down).
CANopen	CAN operation as slave: Contains the following parameterised function elements: • CAN1_SLAVE_EMCY_HANDLER, • CAN1_SLAVE_STATUS, • SELECT_NODESTATE (→ down).
Objekt1xxxh	CAN operation as slave: Contains the values [STRING] for the following parameters: • ManufacturerDeviceName, e.g.: 'CR1051' • ManufacturerHardwareVersion, e.g.: 'HW_Ver 1.0' • ManufacturerSoftwareVersion, e.g.: 'SW_Ver 1.0'

Function elements in the folder I_O_CONFIGURATION	Description
CONF_IO_CRnnnn	CAN operation with layer 2 or as master or slave: Parameterises the operating modes of the inputs and outputs.

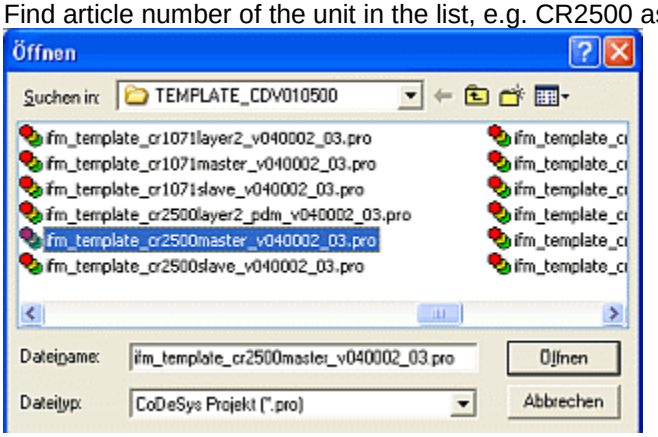
Function elements in the folder PDM_COM_LAYER2	Description
PLC_TO_PDM	CAN operation with layer 2 or as slave: Organises the communication from the Controller to the PDM: • monitors the transmission time, • transmits control data for image change, input values etc.
TO_PDM	CAN operation with layer 2 or as slave: Organises the signals for LEDs and keys between Controller and PDM. Contains the following parameterised function elements: • PACK (→ 3S), • PLC_TO_PDM (→ up), • UNPACK (→ 3S).

Function elements in the root directory	Description
PLC_CYCLE	CAN operation with layer 2 or as master or slave: Determines the cycle time of the PLC in the unit.
PLC_PRG	CAN operation with layer 2 or as master or slave: Main program This is where further program elements are included.

2.2.2 How do you set up the programming system fast and simply? (e.g. CR2500)

18052

- ► In the CODESYS menu select: [File] > [New from template...]
- ► Select directory of the current DVD, e.g. `...\Projects\TEMPLATE_DVD020000.`
- ► Find article number of the unit in the list, e.g. CR2500 as CANopen master:



- ► Mind the correct program version!
- ► How is the CAN network organised?
  Do you want to work on layer 2 basis or is there a master with several slaves (for CANopen)?
- ► Confirm the selection with [Open].

> A new CODESYS project is generated with the following folder structure (left):

Example for CR2500 as CANopen master:	Another example for CR1051 as CANopen slave:
CAN_OPEN CANOPEN (PRG) PLC_CYCLE (PRG) PLC_PRG (PRG)	CAN_OPEN CANOPEN (PRG) Objekt1xxxh (PRG) SELECT_NODESTATE (PRG) CONTROL_CR10xx CONTROL_PDM (PRG) RT_SOFT_KEYS (PRG) SELECT_PAGE (PRG) PDM_DISPLAY_SETTINGS CHANGE_BRIGHTNESS (PRG) DISPLAY_SETTINGS (PRG) READ_SOFTWARE_VERS (PRG) PDM_CYCLE_MS (PRG) PLC_PRG (PRG)

(via the folder structures in templates → section **About the ifm templates** (→ p. 11)).

- ► Save the new project with [file] > [Save as...], and define suitable directory and project name.

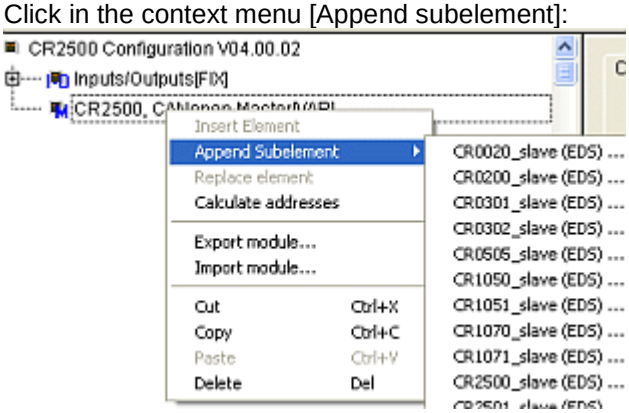
2.2.3 Insert CANopen slave (example: CR2500 <-- CR2011)

18053

► Configure the CAN network in the project:
Double click the element [PLC configuration] above the tabulator [resources] in the CODESYS project.

► **Right** mouse click in the entry [CR2500, CANopen Master]

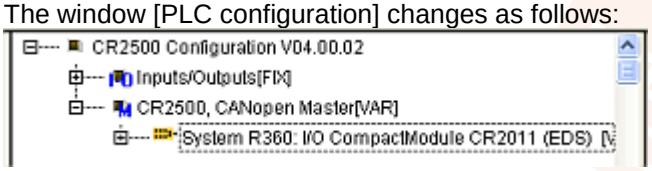
► Click in the context menu [Append subelement]:



> A list of all available EDS files appears in the extended context menu.

► Select requested element, e.g. "System R360": I/O CompactModule CR2011 (EDS)".
The EDS files are in directory `C:\...\CoDeSys V...\Library\PLCConf\`.

> The window [PLC configuration] changes as follows:



► Set CAN parameters, PDO mapping and SDOs for the entered slave according to the requirements.
Better deselect [Create all SDOs].

► With further slaves proceed as described above.

► Save the project!

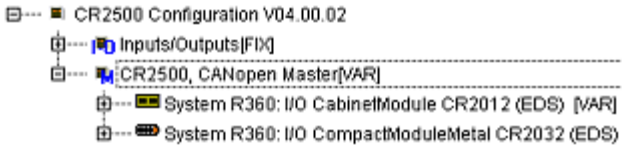
This should be a sufficient description of your project. You want to supplement this project with further elements and functions?
→ chapter **Supplement project with further functions** (→ p. 15)

2.2.4 Supplement project with further functions

3987

You have created a project using an **ifm** template and you have defined the CAN network. Now you want to add further functions to this project.

For the example we take a CabinetController CR2500 as CAN open Master to which an I/O CabinetModule CR2011 and an I/O CompactModule are connected as slaves:

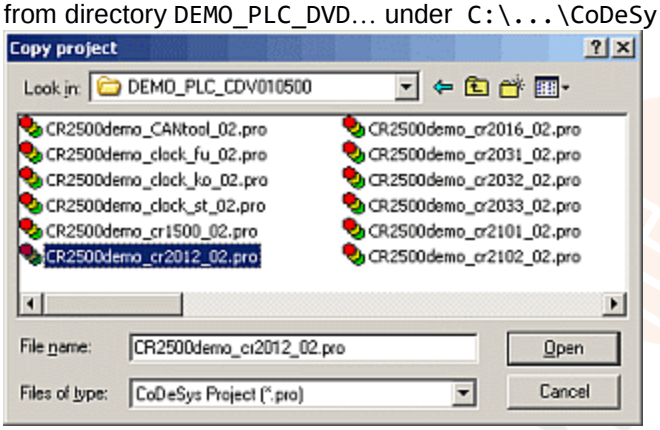


Example: PLC configuration

A joystick is connected to the CR2012 which is to trigger a PWM output on the CR2032. How is that achieved in a fast and simple way?

► Save CODESYS project!

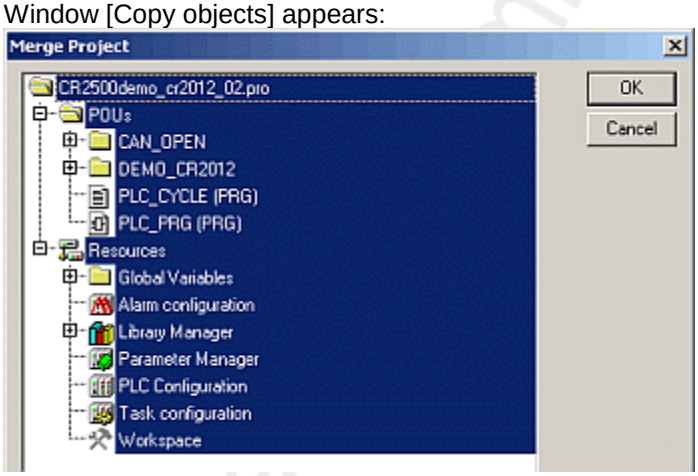
► In CODESYS use [Project] > [Copy...] to open the project containing the requested function:
e.g. `CR2500Demo_CR2012_02.pro`
from directory `DEMO_PLC_DVD...` under `C:\...\CoDeSys V...\Projects\`:



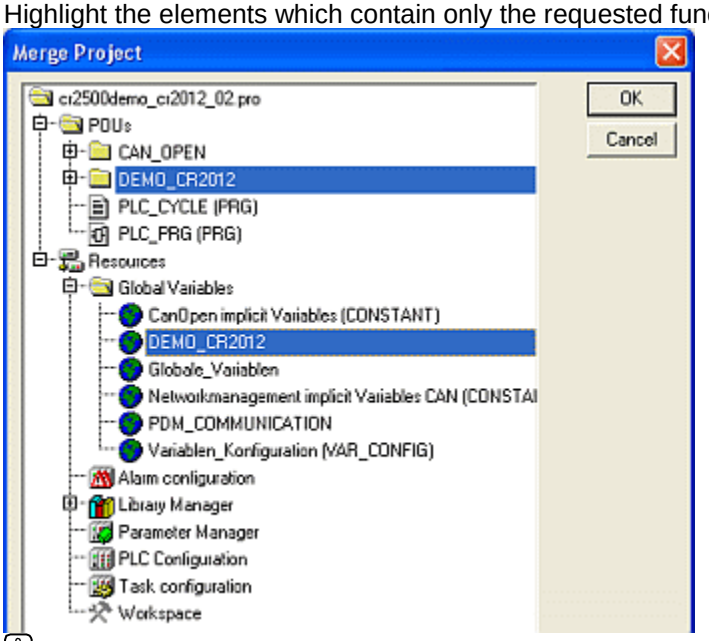
► Confirm the selection with [Open].

► The message "Error when loading the PLC configuration" can be ignored.

> Window [Copy objects] appears:



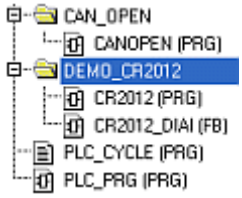
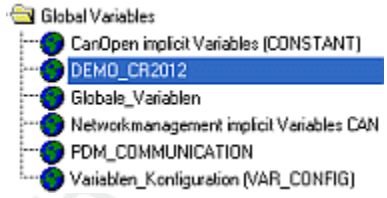
► Highlight the elements which contain only the requested function, in this case e.g.:



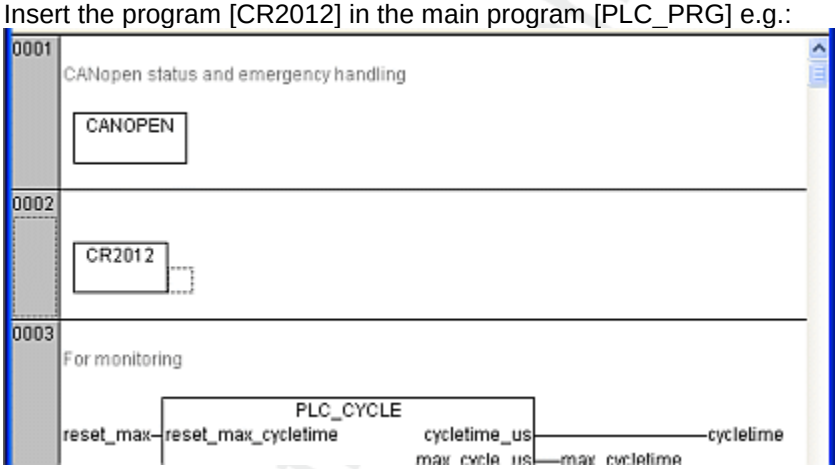
ℹ In other cases libraries and/or visualisations might be required.

► Confirm the selection with [OK].

> In our example project the elements selected in the demo project have been added:

POUs:	Resources:
	

► Insert the program [CR2012] in the main program [PLC_PRG] e.g.:



► The comments of the function elements and global variables usually contain information on how the individual elements have to be configured, included or excluded. This information has to be followed.

► Adapt input and output variables as well as parameters and possible visualisations to your own conditions.

► [Project] > [Save] and
[Project] > [Rebuild all].

► After possibly required corrections and addition of missing libraries (→ Error messages after rebuild) save the project again.

► Follow this principle to step by step (!) add further functions from other projects and check the results.

► [Project] > [Save] and
[Project] > [Rebuild all].

2.3 ifm demo programs

Contents

3982

In the directory
- `DEMO_PLC_DVD...` (for Controller) or
- `DEMO_PDM_DVD...` (für PDMs)

under `C:\...\CoDeSys V...\Projects\`
we explain certain functions in tested demo programs. If required, these functions can be implemented in own projects. Structures and variables of the ifm demo programs match those in the ifm templates.

Each demo program shows just **one** topic. For the Controller as well some visualisations are shown which demonstrate the tested function on the PC screen.

Comments in the function elements and in the variable lists help you adapt the demo programs to your project.

If not stated otherwise the demo programs apply to all controllers or to all PDMs.

The demo programs described here apply for:
- CODESYS from version 2.3.9.6
- on the ecomat*mobile* DVD "Software, tools and documentation" from version 020000

2.3.1 Demo programs for controller

3995

Demo program	Function
`CR2500Demo_CanTool_xx.pro`	separate for PDM360, PDM360compact, PDM360smart and Controller: Contains FBs to set and analyse the CAN interface.

`CR2500Demo_ClockFu_xx.pro`
`CR2500Demo_ClockKo_xx.pro`
`CR2500Demo_ClockSt_xx.pro`
	Clock generator for Controller as a function of a value on an analogue input:
	Fu = in function block diagram
	K0 = in ladder diagram
	St = in structured text

`CR2500Demo_CR1500_xx.pro`	Connection of a keypad module CR1500 as slave of a Controller (CANopen master).
`CR2500Demo_CR2012_xx.pro`	I/O cabinet module CR2012 as slave of a Controller (CANopen master), Connection of a joystick with direction switch and reference medium voltage.
`CR2500Demo_CR2016_xx.pro`	I/O cabinet module CR2016 as slave of a Controller (CANopen master), 4 x frequency input, 4 x digital input minus switching, 4 x digital input plus switching, 4 x analogue input ratiometric, 4 x PWM1000 output and 12 x digital output.
`CR2500Demo_CR2031_xx.pro`	I/O compact module CR2031 as slave of a Controller (CANopen master), Current measurement on the PWM outputs
`CR2500Demo_CR2032_xx.pro`	I/O compact module CR2032 as slave of a Controller (CANopen master), 4 x digital input, 4 x digital input analogue evaluation, 4 x digital output, 4 x PWM output.

Demo program	Function
CR2500Demo_CR2033_xx.pro	I/O compact module CR2033 as slave of a Controller (CANopen master), 4 x digital input, 4 x digital input analogue evaluation, 4 x digital output,
CR2500Demo_CR2101_xx.pro	Inclination sensor CR2101 as slave of a Controller (CANopen master).
CR2500Demo_CR2102_xx.pro	Inclination sensor CR2102 as slave of a Controller (CANopen master).
CR2500Demo_CR2511_xx.pro	I/O smart module CR2511 as slave of a Controller (CANopen master), 8 x PWM output current-controlled.
CR2500Demo_CR2512_xx.pro	I/O smart module CR2512 as slave of a Controller (CANopen master), 8 x PWM output. Display of the current current for each channel pair.
CR2500Demo_CR2513_xx.pro	I/O smart module CR2513 as slave of a Controller (CANopen master), 4 x digital input, 4 x digital output, 4 x analogue input 0...10 V.
CR2500demo_input_from_pdm_CANopen_xx.pro	Use system variables via CANopen: • HANDLE, • INPUT_VALUE, • LENGHT
CR2500demo_input_from_pdm_Layer2_xx.pro	Use system variables via CAN-Layer2: • HANDLE, • INPUT_VALUE, • LENGHT
CR2500Demo_Interrupt_xx.pro	Example with SET_INTERRUPT_XMS.
CR2500Demo_Operating_hours_xx.pro	Example of an operating hours counter with interface to a PDM.
CR2500Demo_PWM_xx.pro	Converts a potentiometer value on an input into a normed value on an output with the following function elements: • INPUT_VOLTAGE, • NORM, • PWM100.
CR2500Demo_RS232_xx.pro	Example for the reception of data on the serial interface by means of the Windows hyper terminal.
StartersetDemo.pro StartersetDemo2.pro StartersetDemo2_fertig.pro	Various e-learning exercises with the starter set EC2074.

_xx = indication of the demo version

3 Using CAN – description

Contents

13743

3.1 General about CAN

Contents

1164

The CAN bus (**C**ontroller **A**rea **N**etwork) belongs to the fieldbuses.

It is an asynchronous serial bus system which was developed for the networking of control devices in automotives by Bosch in 1983 and presented together with Intel in 1985 to reduce cable harnesses (up to 2 km per vehicle) thus saving weight.

3.1.1 CAN: hardware

14103

Topology

1244

The CAN network is set up in a line structure. A limited number of spurs is allowed.
Moreover is possible:
• a star type bus (e.g. central locking).

NOTE

Prevent falsification of the signal quality due to signal echoes at the cable ends:

► Terminate the CAN bus line at both ends using terminating resistors of ≥ 120 Ω!

The devices of **ifm electronic gmbh** equipped with a CAN interface have no terminating resistors.

Together with the terminating resistors the overall resistance (measured between CAN_H and CAN_L) of the voltage-free CAN bus line should be about 60±5 Ω.

The disadvantage of spurs and star-type bus is that the wave resistance is difficult to determine. In the worst case the bus no longer functions.

Network structure

1178

The ISO 11898 standard assumes a line structure of the CAN network.

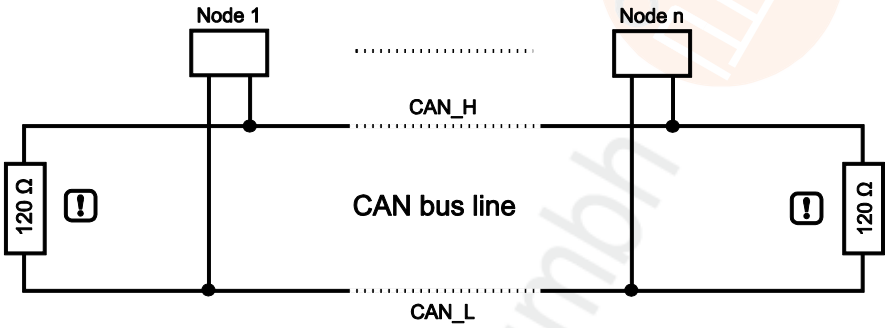


Figure: CAN network structure line topology

The internal resistance of a CAN interface is approx. 40...45 kΩ. With 32 devices on the CAN bus the resulting resistance in the network is only 1.25...1.4 kΩ.

NOTE

Prevent falsification of the signal quality due to signal echoes at the cable ends:

► Terminate the CAN bus line at both ends using terminating resistors of ≥ 120 Ω!

The devices of **ifm electronic gmbh** equipped with a CAN interface have no terminating resistors.

Together with the terminating resistors the overall resistance (measured between CAN_H and CAN_L) of the voltage-free CAN bus line should be about 60±5 Ω.

Spurs

13013

Depending on the total cable length and the time sequences on the bus, signal reflections may result. This is why spurs to the bus participants (node 1...n) should be avoided.

If spurs are not avoidable:

- Individual spurs of a length of up to 2 m (referred to 125 kbits/s) are considered to be uncritical.
- The sum of all spurs in the whole system should not exceed 30 m.

► In special cases the cable lengths of the line and spurs must be calculated exactly.

CAN bus level

1179

The CAN bus is in the inactive (recessive) state if the output transistor pairs are switched off in all bus participants. If at least one transistor pair is switched on, a bit is transferred to the bus. This activates the bus (dominant). A current flows through the terminating resistors and generates a difference voltage between the two bus cables. The recessive and dominant states are converted into voltages in the bus nodes and detected by the receiver circuits.

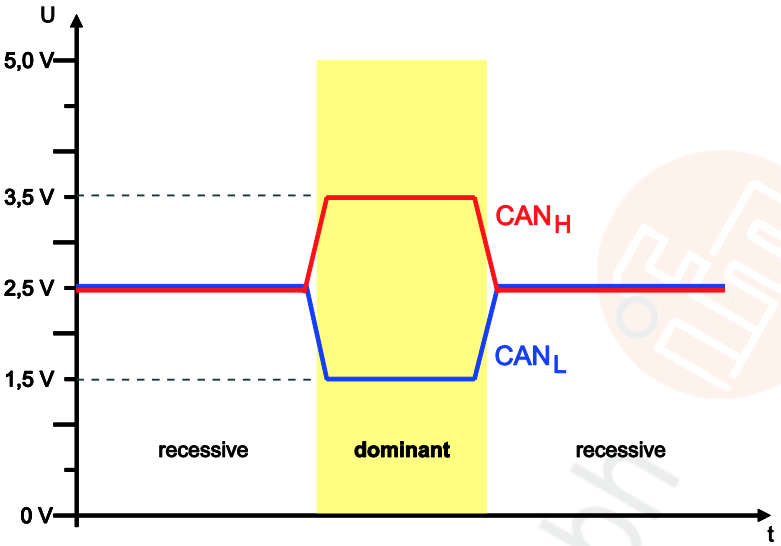


Figure: CAN bus level

This differential transmission with common return considerably improves the transmission security. Noise voltages which interfere with the system externally or shifts of the ground potential influence both signal cables with the same interference. These influences are therefore not considered when the difference is formed in the receiver.

Bus cable length

1180

The length of the bus cable depends on:
- type of the bus cable (cable, connector),
- cable resistance,
- required transmission rate (baud rate),
- length of the spurs.

To simplify matters, the following dependence between bus length and baud rate can be assumed:

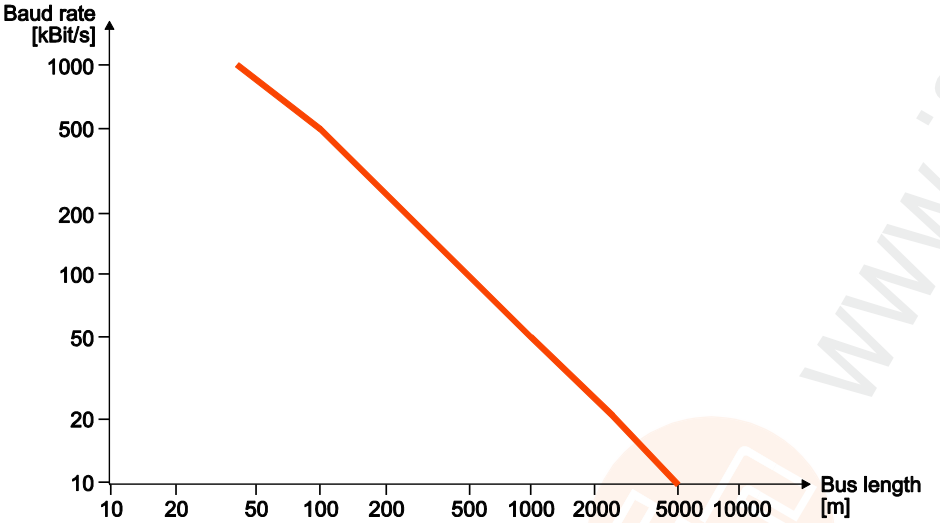


Figure:
bus cable length

Baud rate [kBit/s]	Bus length [m]	Bit time nominal [µs]
1 000	40	1
800	50	1,25
500	100	2
250	250	4
125	500	8
62,5	1 000	20
20	2 500	50
10	5 000	100

Table: Dependencies bus length / baud rate / bit time

Wire cross-sections

1181

► For the layout of the CAN network the wire cross-section of the bus cable used must also be taken into account.

The following table describes the dependence of the wire cross-section referred to the cable length and the number of the connected nodes.

Cable length [m]	**Wire cross-section** [mm²]		
	at 32 nodes	**at 64 nodes**	**at 100 nodes**
≤ 100	0.25	0.25	0.25
≤ 250	0.34	0.50	0.50
≤ 500	0.75	0.75	1.00

Depending on the EMC requirements the bus cables can be laid out as follows:
- in parallel with / without shield
- as twisted pair with / without shield

3.1.2 CAN: software

Contents

14104

IDs (addresses) in CAN

15448

In CANopen there are different types of identifications (here: IDs):

- COB ID
  The **C**ommunication **Ob**ject **Id**entifier addresses the message (= the communication object). A communication object consists of a network-wide CAN message.
  The lower the COB ID, the higher the priority of the message.
- Download ID
  The download ID designates the identification for the program download and the maintenance access in CODESYS.
  For this, ifm uses the SDO mechanism from the CANopen protocol.
  The software adds the download ID to the basis address.
- Node ID
  The **Node id**entifier is a unique identifier for CANopen devices in the CAN network. The Node ID is also part of some predefined connection sets (→ **Function code / Predefined Connectionset** (→ p. 67)). On the basis of the node ID the COB IDs are determined if the predefined connection settings are used.
  The node ID and the download ID must be different in the same CAN network!

13182

Comparison of download ID vs. node ID:

Controller program download		CANopen	
Download ID	**COB ID SDO**	**Node ID**	**COB ID SDO**
1…127	TX: 0x580 + download ID	1…127	TX: 0x580 + node ID
	RX: 0x600 + download ID		RX: 0x600 + node ID

TX = slave sends to master
RX = slave receives from master

NOTE

The CAN download ID of the device must match the CAN download ID set in CODESYS!

In the same CAN network the CAN download IDs must be unique!

The same CAN download ID may be assigned to the different CAN interfaces of a device, provided that these CAN interfaces are connected to separate CAN networks.

COB-ID

18384
18379

Depending of the type the following CAN identifiers are free available for the data transfer:

COB-ID (base)	COB-ID (extended)
11 bits	29 bits
COB identifier: 0...2 047	COB identifier: 0...536 870 911
Standard applications	Engine management (SAE J1939), Truck & Trailer interface (ISO 11992)

18382

Example 11 bits COB-ID (base):

SOF	COB-ID (base) Bit 28 ... bit 18											RTR	IDE
0	0	0	0	0	1	1	1	1	1	1	1	0	0
	0			7				F					

Example 29 bits COB-ID (extended):

SOF	COB-ID (base) Bit 28 ... bit 18										SRR	IDE	COB-ID (extended) Bit 17 ... bit 0																RTR
0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1		F			C				0		0			0			0										

Legend:
SOF = **S**tart **o**f **f**rame
edge of recessive to dominant

RTR = **R**emote **t**ransmission **r**equest
dominant: this message sends data
recessive: this message requests data

IDE = **Id**entifier **e**xtension flag
dominant: after this control bits follows
recessive: after this the second part of the 29 bits identifier follows

SRR = **S**ubstitute **r**emote **r**equest
recessive: extended CAN-ID: replaces the RTR bit at this position

3.2 CAN interfaces

Contents

14101

Connections and data → data sheet

3.2.1 CAN: interfaces and protocols

18060

The devices are equipped with several CAN interfaces depending on the hardware design. Basically, all interfaces can be used with the following functions independently of each other:

- Layer 2: CAN at level 2
- CANopen master
- CANopen slave
- CANopen network variables (via CODESYS)
- SAE J1939 (for drive management)
- Bus load detection
- Error frame counter
- Download interface
- 100 % bus load without package loss

If several CANopen-capable interfaces are available, then the following applies for the assignment of the CANopen protocol to the CAN interface (depending on the device):

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
the sequence in which you add the sub-elements in the controller configuration	the selection of the CAN interface to which you add the sub-element

3.3 CAN: exchange of data

19015

CAN data is exchanged via the CAN protocol of the link layer (level 2) of the seven-layer ISO/OSI reference model specified in the international standard ISO 11898.

Every bus participant can transmit messages (multimaster capability). The exchange of data functions similarly to radio. Data is transferred on the bus without transmitter or address. The data is only marked by the identifier. It is the task of every participant to receive the transmitted data and to check by means of the identifier whether the data is relevant for this participant. This procedure is carried out automatically by the CAN controller together with the runtime system.

For the normal exchange of CAN data the programmer only has to make the data objects with their identifiers known to the system when designing the software. This is done via the FBs for CAN transmit und CAN receive:

- CAN POUs based on layer 2 (RAW CAN):
  simple functions.
- CAN POUs based on SAE J1939:
  high class functions for the engine management.
- CAN POUs based on CANopen:
  complex CAN functions.
- CANopen safety POUs (optional):
  CAN functions for safety applications with SafetyControllers.

Using these FBs the following units are combined into a data object:
- the useful data,
- the frame type (optional),
- selected identifier (ID).

These data objects participate in the exchange of data via the CAN bus. The transmit and receive objects can be defined from all valid IEC data types (e.g. BOOL, WORD, INT, ARRAY).

The CAN message consists of a COB identifier (**COB-ID** (→ p. 27)) and maximum 8 data bytes. The ID does not represent the transmit or receive module but identifies the message. To transmit data it is necessary that a transmit object is declared in the transmit module and a receive object in at least one other module. Both declarations must be assigned to the same identifier and the same message type (base or extended).

3.3.1 Data reception

19016

In principle the received data objects are automatically stored in a buffer (i.e. without influence of the user).

Each identifier has such a buffer (queue). Depending on the application software this buffer is emptied according to the FiFo principle (**F**irst **I**n, **F**irst **O**ut) via the device-specific CAN receive FB.

3.3.2 Transmit data

19017

By calling the device-specific CAN transmit FB the application program transmits exactly one CAN message to the buffer (queue). As feedback you are informed whether there was still enough space in the buffer and whether the message was successfully transferred. The buffer autonomously transfers the message to the CAN controller, which transmits it on the bus.

The transmit order is denied if the buffer is already full. The transmit order must then be repeated by the application program. This information is indicated to the programmer by a bit.

In the buffer, no transmit priority is assigned to the messages based on their **COB-ID** (→ p. 27). The programmer must therefore carefully assign the order in which he transfers the messages.

3.4 Technical details on CANopen

Contents

13822

3.4.1 CANopen network configuration, status and error handling

13824

The network configuration and parameter setting of the connected devices are carried out via the programming software CODESYS.

For some devices, the error messages can only be reached via nested variable structures in the CANopen stack.

The documentation below shows you the structure and use of the network configuration.

The following chapters describe the internal function elements of the CODESYS CANopen stack and their use. They also give information of how to use the network configurator.

NOTE

It is absolutely necessary to use only the corresponding device-specific library. The context can be seen from the integrated article number of the device.
Example CR0032 with CANopen master for CAN interface 1:
→ `ifm_CR0032_CANopen1Master_Vxxyyzz.lib`

→ device manual, chapter 'Set up target'

When other libraries are used the device can no longer function correctly.

3.4.2 CANopen support by CoDeSys

1857

General information about CANopen with CODESYS

13826

CODESYS is one of the leading systems for programming control systems to the international standard IEC 61131. To make CODESYS more interesting for users many important functions were integrated in the programming system, among them a configurator for CANopen. This CANopen configurator allows you to configure CANopen networks (with some restrictions) under CODESYS.

An **ecomat*mobile*** controller can be used as a CANopen master and as a CANopen slave.

> **NOTE**
>
> For all **ecomat *mobile*** controllers and PDM360smart you must use the 3S CANopen libraries with the following addition: "**OptTableEx**"
>
> If a new project is created, these libraries are in general automatically loaded. If you add the libraries via the library manager, you must ensure a correct selection.
>
> The CANopen libraries without this addition are used for other programmable **ifm** devices.

CANopen terms and implementation

1858

In respect of the transmission of process data in CANopen there are no masters and slaves in a CAN network. In CANopen there is, however, a master/slave architecture for the network management (NMT) and for the configuration.

The CAN protocol (below the CANopen protocol) does not know any master/slave relationship.

Implementation assumes that a CAN network serves as periphery of a CODESYS programmable controller.

The master is an NMT master and configuration master. Normally the master ensures that the network is put into operation. The master takes the initiative to start the individual nodes (= network nodes) known via the configuration. These nodes are called slaves.

To bring the master closer to the status of a CANopen slave an object directory was introduced for the master. The master can also act as an SDO server (SDO = Service Data Object) and not only as SDO client in the configuration phase of the slaves.

3.4.3 CANopen master

Contents

1859

CANopen libraries

Contents

18033

Libraries: required by the system for CANopen

14356

The following libraries are automatically integrated in the CODESYS project when the CANopen functionality is used:

- the CODESYS library `3S_CanDrvOptTableEx.LIB`
- the CODESYS library `3S_CANopenMasterOptTableEx.LIB`
- the CODESYS library `3S_CANopenManagerOptTableEx.LIB`
- the CODESYS library `3S_CanOpenDeviceOptTableEx.LIB`
- the CODESYS library `3S_CanOpenNetVarOptTableEx.LIB`
- the CODESYS library `SysLibCallback.LIB`

The contained function blocks and functions must NOT be called directly in the code of the application program!

Functions of the CANopen libraries

1990

The following functions defined in CANopen are at present supported by the CODESYS CANopen library:

- **Transmitting PDOs:** master transmits to slaves (slave = node, device)
  - transmitting event-controlled (i.e. in case of a change),
  - transmitting time-controlled (RepeatTimer) or
  - transmitting as synchronous PDOs, i.e. always when a SYNC was transmitted by the master.

  An external SYNC source can also be used to initiate transmission of synchronous PDOs.
- **Receiving PDOs:** master receives from slave
  Depending on the slave: event-controlled, request-controlled, acyclic and cyclic.
- **PDO mapping**
  Assignment between a local object directory and PDOs from/to the CANopen slave (if supported by the slave).
- **Transmitting and receiving SDOs** (unsegmented, i.e. 4 bytes per entry in the object directory)
  - automatic configuration of all slaves via SDOs at the system start.
  - application-controlled transmission and reception of SDOs to/from configured slaves.
- **Synchronisation**
  Automatic transmission of SYNC messages by the CANopen master.
- **Nodeguarding**
  Automatic transmission of guarding messages and lifetime monitoring for every slave configured accordingly.
  ⓘ We recommend: It is better to work with the heartbeat function for current devices since then the bus load is lower.
- **Heartbeat**
  Automatic transmission and monitoring of heartbeat messages.

The following functions defined in CANopen are at present **not** supported by the CODESYS CANopen library:

- Dynamic identifier assignment
- Dynamic SDO connections
- SDO transfer block by block (can be implemented for some **ifm** devices by means of function blocks in the respective ifm device library)
- segmented SDO transfer (can be implemented by means of function blocks in the respective ifm device library)
- All options of the CANopen protocol which are not mentioned above.

The following functions are supported by the **ifm** CANopen library:

- **Emergency**
  Reception of emergency messages from the configured slaves and message storage.
- Set **Node-ID** and **baud rate** in the slaves
  By calling a simple function, node ID and baud rate of a slave can be set at runtime of the application.

Create a CANopen project

19021

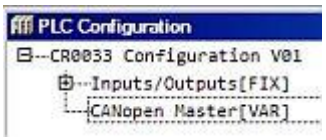
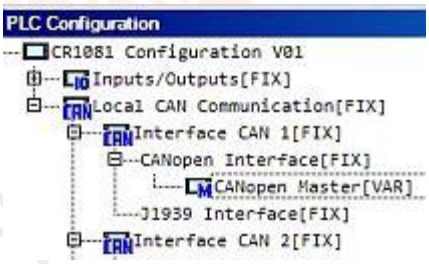
Below the creation of a new project with a CANopen master is described step by step. It is assumed that you have already installed CODESYS on your processor and the target and EDS files have also been correctly installed or copied.

You will find a detailed description about the adjustment and application of the controller and CANopen configuration dialogue here:
- in the CODESYS manual see [Ressourcen] > [Steuerungskonfiguration]
- in the CODESYS online help.

► Add the CANopen master in the controller configuration after creating a new project (→ device manual, chapter 'Set up the target'):

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
► Right mouse click on the first line ("CRnnnn Configuration Vnn") ► [Unterelement anhängen] > [CANopen Master]	► At the requested CAN interface: Right mouse click on [CANopen Interface] ► [Unterelement anhängen] > [CANopen Master]

If several CANopen-capable interfaces are available, then the following applies for the assignment of the CANopen protocol to the CAN interface (depending on the device):

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
the sequence in which you add the sub-elements in the controller configuration	the selection of the CAN interface to which you add the sub-element
Example with CR0033: 	Example with CR1081: 

> The following libraries and software modules are automatically integrated:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
The `STANDARD.LIB` which provides the standard functions for the controller defined in IEC 61131,	The `STANDARD.LIB` which provides the standard functions for the controller defined in IEC 61131,
The `3S_CanOpenManager.LIB` which provides the CANopen basic functionalities (if needed `3S_CanOpenManagerOptTable.LIB` for C167 controller),	–
one or several of the libraries `3S_CANopenNetVar.LIB`, `3S_CANopenDevice.LIB` and `3S_CANopenMaster.LIB` (if needed, `3S_...OptTable.LIB` für C167-Controller), depending on the requested functionality,	–
the system libraries, e.g.: `SysLibSem.LIB` and `SysLibCallback.LIB`.	the system libraries, e.g.: `SysLibSem.LIB` and `SysLibCallback.LIB`.

► To use the prepared network diagnostics, status and EMCY function, add the ifm CANopen master library (or the ifm CANopen_NT library) manually in the library manager. Without this library the network information must be directly read from the nested structures of the 3S CANopen libraries.

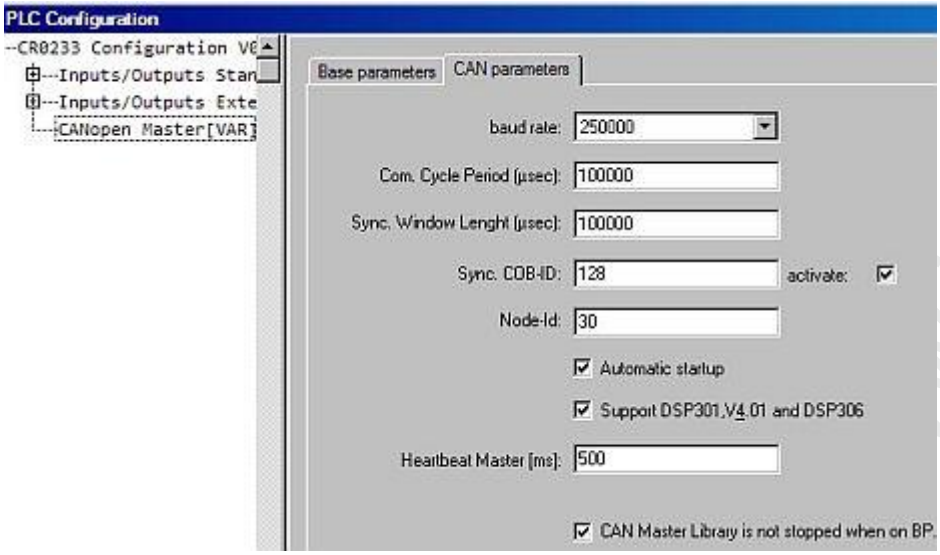
► Additionally, integrate the following libraries and software modules:

- The device library for the corresponding hardware, e.g. `ifm_CR0032_Vxxyyzz.LIB`. This library provides all device-specific functions.
- EDS files for all slaves to be operated on the network. The EDS files for all **ifm** CANopen slaves are provided by **ifm electronic gmbh**.
  For the EDS files of other manufacturers' nodes, contact the corresponding manufacturer.

CANopen master: Tab [CAN parameters]

1967

► Set the most important parameters for the master in this dialogue window.



Example: Control configuration for CR0233 CANopen master at CAN interface 1

Legend:

CAN parameters: Baud rate

10028

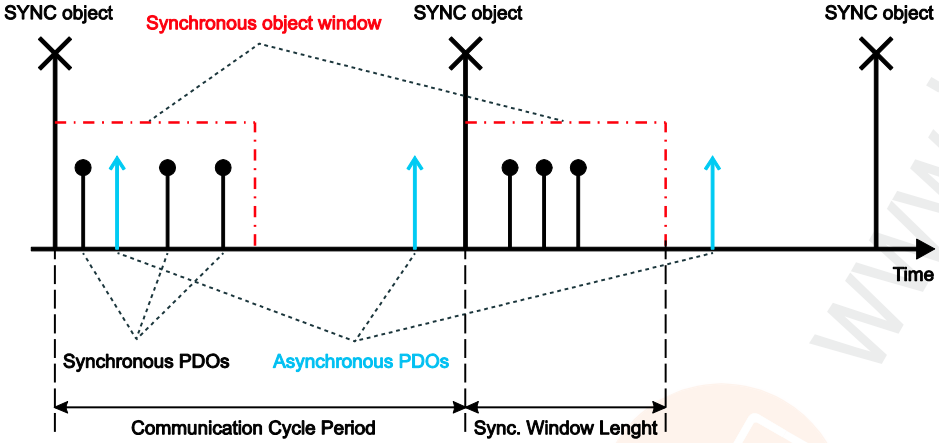
► Select the baud rate for the master.

- The baud rate must correspond to the transmission speed of the other network participants.

CAN parameters: Communication Cycle Period/Sync. Window Length

10029

After expiry of the [Communication Cycle Period] a SYNC message is transmitted by the master.



The [Sync. Window Length] indicates the time during which synchronous PDOs are transmitted by the other network participants and must be received by the master.

As in most applications no special requirements are made for the SYNC object, the same time can be set for [Communication Cycle Period] and [Sync. Window Length].
Please ensure the time is entered in [µs] (the value 50 000 corresponds to 50 ms).

CAN parameters: Sync. COB ID

10030

In this field the identifier for the SYNC message can be set. It is always transmitted after the communication cycle period has elapsed. The default value is 128 and should normally not be changed. To activate transmission of the SYNC message, the checkbox [activate] must be set.

> **NOTE**
>
> The SYNC message is always generated at the start of a program cycle. The inputs are then read, the program is processed, the outputs are written to and then all synchronous PDOs are transmitted.
>
> Please note that the SYNC time becomes longer if the set SNYC time is shorter than the program cycle time.
>
> **Example:** communication cycle period = 10 ms and program cycle time = 30 ms.
> The SYNC message is only transmitted after 30 ms.

CAN parameters: Node ID

10031

► Enter the node number (not the download ID!) of the master in this field.

The node number may only occur once in the network, otherwise the communication is disturbed.

CAN parameters: Automatic startup

10032

After successful configuration the network and the connected nodes are set to the state [operational] and then started.

If the checkbox is not activated, the network must be started manually.

CAN parameters: Heartbeat

10033

If the other participants in the network support heartbeat, the option [support DSP301, V4.01...] can be selected.

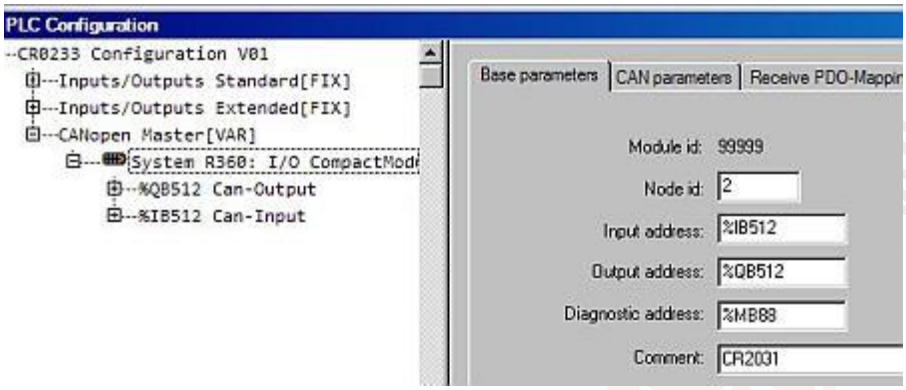
If necessary, the master can generate its own heartbeat signal after the set time has elapsed.

Add and configure CANopen slaves

Contents

1861

Next you can add the CANopen slaves. To do so, you must call again the dialogue in the controller configuration [Insert] > [Append subelement]. A list of the CANopen device descriptions (EDS files) stored in the directory PLC_CONF is available. By selecting the corresponding device it is directly added to the tree of the controller configuration.



Example: PLC configuration for CR0020 CANopen master with connected I/O CompactModule

NOTE

If a slave is added via the configuration dialogue in CoDeSys, source code is dynamically integrated in the application program for every node. At the same time every additionally inserted slave extends the cycle time of the application program. This means: In a network with many slaves the master can process no further time-critical tasks (e.g. FB OCC_TASK).

A network with 27 slaves has a basic cycle time of 30 ms.

Please note that the maximum time for a PLC cycle of approx. 50 ms should not be exceeded (watchdog time: 100 ms).

CANopen slave: Tab [CAN parameters]

1968

CAN parameters: Node ID

10036

The node ID is used to clearly identify the CAN module and corresponds to the number on the module set between 1 and 127.

The ID is entered decimally and is automatically increased by 1 if a new module is added.

CAN parameters: Write DCF

10037

If [Write DCF] is activated, a DCF file is created after adding an EDS file to the set directory for compilation files. The name of the DCF file consists of the name of the EDS file and appended node ID.

CAN parameters: Create all SDOs

10038

If this option is activated, SDOs are generated for all communication objects.

Default values are not written again!

CAN parameters: Node reset

10039

The slave is reset ("load" and NMT command "Reset Node") during initialisation of the CANopen network after the master has been rebooted.

Then the slave is configured.

CAN parameters: Optional device

10040

If the option [optional device] is activated, the master tries only once to read from this node. In case of a missing response, the node is ignored and the master goes to the normal operating state.

If the slave is connected to the network and detected at a later point in time, it is automatically started. To do so, you must have selected the option [Automatic startup] in the CAN parameters of the master.

CAN parameters: No initialization

10041

If this option is activated, the master immediately takes the node into operation without transmitting configuration SDOs. (Nevertheless, the SDO data is generated and stored in the controller.)

CAN parameters: Nodeguarding / heartbeat settings

10042

Depending on the device you can choose:
- [nodeguarding] and [life time factor] must be set OR
- [heartbeat] must be set.

If both are set, only heartbeat is executed.

We recommend: It is better to work with the heartbeat function for current devices since then the bus load is lower.

CAN parameters: Emergency telegram

10043

This option is normally selected. The EMCY messages are transferred with the specified identifier.

CAN parameters: Communication cycle

10044

In special applications a monitoring time for the SYNC messages generated by the master can be set here.

Please note that this time must be longer than the SYNC time of the master. The optimum value must be determined experimentally.

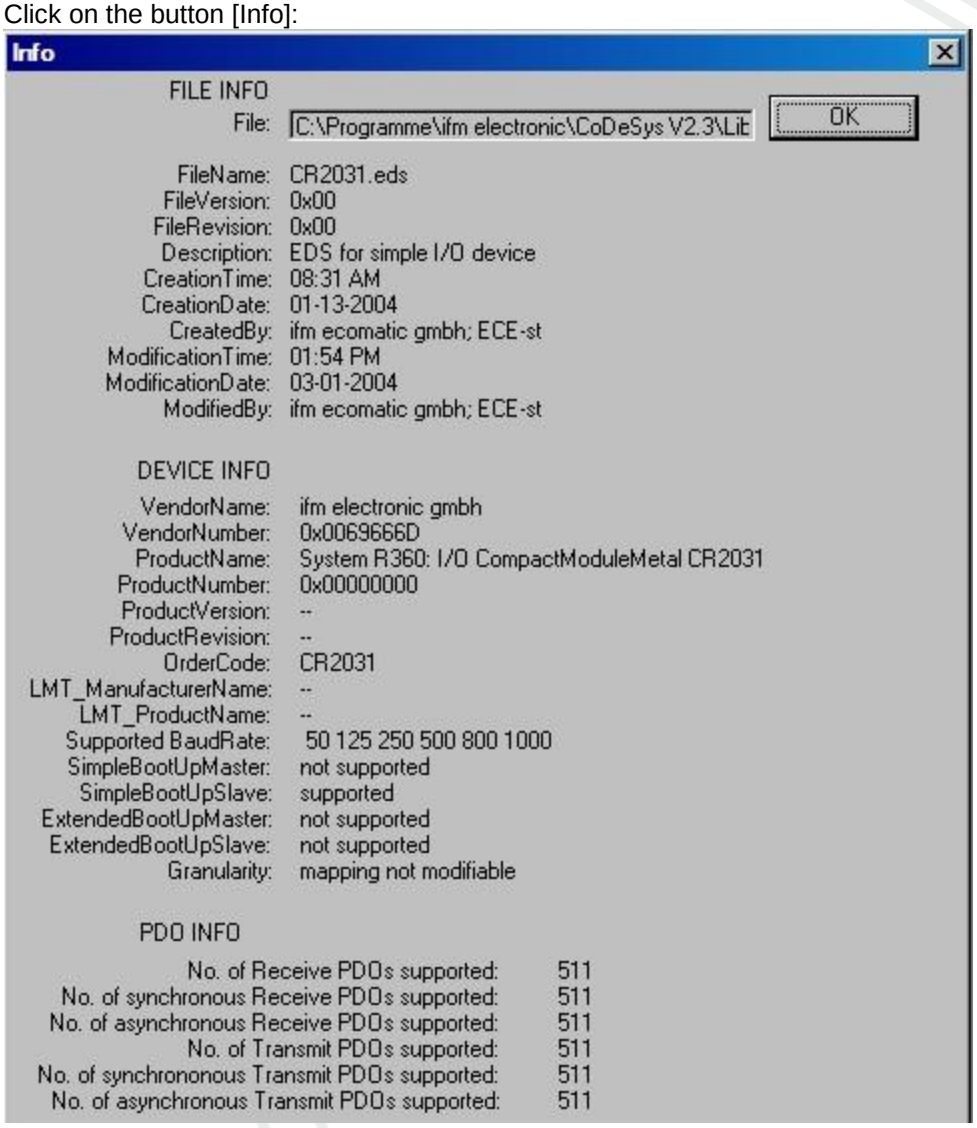
In most cases nodeguarding or heartbeat are sufficient for node monitoring.

CAN parameters: Info

18062

Display the description in the EDS file for this slave:

► Click on the button [Info]:



Example: Info about slave CR2031

► Close display again with [OK].

Tab [Receive PDO-Mapping] and [Send PDO-Mapping]

1969

With the tabs [Receive PDO-Mapping] and [Send PDO-Mapping] in the configuration dialogue of a CAN module the module mapping (assignment between local object directory and PDOs from/to the CANopen slave) described in the EDS file can be changed (if supported by the CAN module).

All [mappable] objects of the EDS file are available on the left and can be added to or removed from the PDOs (Process Data Objects) on the right.

The [StandardDataTypes] can be added to generate spaces in the PDO.

PDO-Mapping: Insert

10046

With the button [Insert] you can generate more PDOs and insert the corresponding objects. The inputs and outputs are assigned to the IEC addresses via the inserted PDOs.

In the controller configuration the settings made can be seen after closing the dialogue. The individual objects can be given symbolic names.

PDO-Mapping: Properties

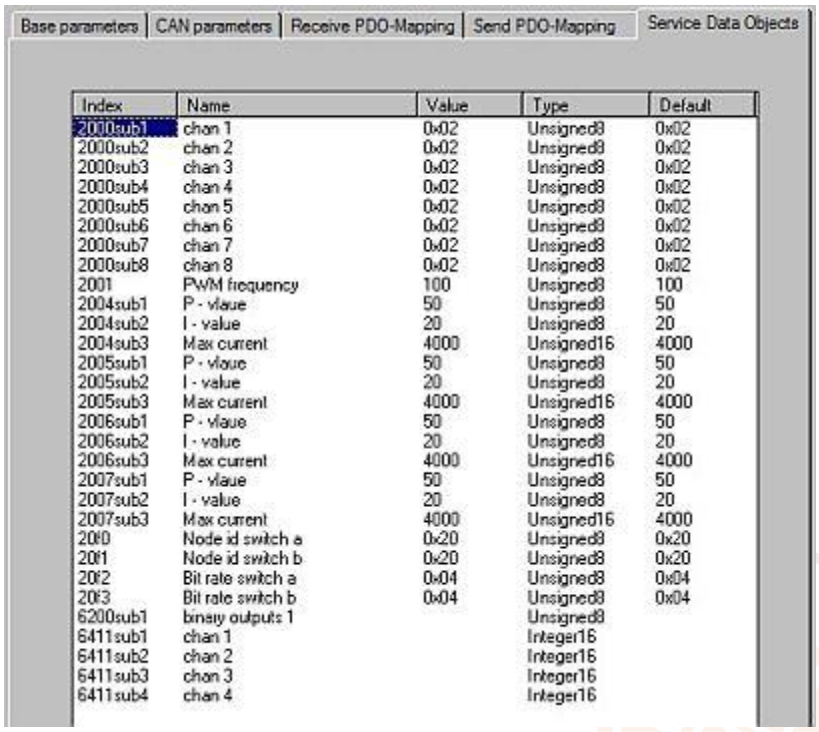
10047

The PDO properties defined in the standard can be edited in a dialogue via properties.

COB-ID	Every PDO message requires a clear COB ID (communication object identifier). If an option is not supported by the module or the value must not be changed, the field is grey and cannot be edited.
Inhibit Time	The inhibit time (100 µs) is the minimum time between two messages of this PDO so that the messages which are transferred when the value is changed are not transmitted too often. The unit is 100 µs.
Transmission Type	For transmission type you receive a selection of possible transmission modes for this module: **acyclic – synchronous** After a change the PDO is transferred with the next SYNC. **cyclic – synchronous** The PDO is transferred synchronously. [Number of SYNCs] indicates the number of the synchronisation messages between two transmissions of this PDO. **asynchronous – device profile specific** The PDO is transmitted on event, i.e. when the value is changed. The device profile defines which data can be transferred in this way. **asynchronous – manufacturer specific** The PDO is transmitted on event, i.e. when the value is changed. The device manufacturer defines which data is transferred in this way. **(a)synchronous – RTR only** These services are not implemented. **Number of SYNCs** Depending on the transmission type this field can be edited to enter the number of synchronisation messages (definition in the CAN parameter dialogue of [Com. Cycle Period], [Sync Window Length], [Sync. COB ID]) after which the PDO is to be transmitted again. **Event-Time** Depending on the transmission type the maximum period in milliseconds [ms] required between two transmissions of the PDO is indicated in this field.

CANopen slave: Register [Service Data Objects]

18036



Index	Name	Value	Type	Default
2000sub1	chan 1	0x02	Unsigned8	0x02
2000sub2	chan 2	0x02	Unsigned8	0x02
2000sub3	chan 3	0x02	Unsigned8	0x02
2000sub4	chan 4	0x02	Unsigned8	0x02
2000sub5	chan 5	0x02	Unsigned8	0x02
2000sub6	chan 6	0x02	Unsigned8	0x02
2000sub7	chan 7	0x02	Unsigned8	0x02
2000sub8	chan 8	0x02	Unsigned8	0x02
2001	PWM frequency	100	Unsigned8	100
2004sub1	P - value	50	Unsigned8	50
2004sub2	I - value	20	Unsigned8	20
2004sub3	Max current	4000	Unsigned16	4000
2005sub1	P - value	50	Unsigned8	50
2005sub2	I - value	20	Unsigned8	20
2005sub3	Max current	4000	Unsigned16	4000
2006sub1	P - value	50	Unsigned8	50
2006sub2	I - value	20	Unsigned8	20
2006sub3	Max current	4000	Unsigned16	4000
2007sub1	P - value	50	Unsigned8	50
2007sub2	I - value	20	Unsigned8	20
2007sub3	Max current	4000	Unsigned16	4000
20f0	Node id switch a	0x20	Unsigned8	0x20
20f1	Node id switch b	0x20	Unsigned8	0x20
20f2	Bit rate switch a	0x04	Unsigned8	0x04
20f3	Bit rate switch b	0x04	Unsigned8	0x04
6200sub1	binary outputs 1		Unsigned8	
6411sub1	chan 1		Integer16	
6411sub2	chan 2		Integer16	
6411sub3	chan 3		Integer16	
6411sub4	chan 4		Integer16	

Here all objects of the EDS or DCF file are listed which are within the range from index 0x2000...0x9FFF and defined as writable.

Index, name, current value, type and default are indicated for every object.
Only the value in [Wert] can be changed.

SDOs: Change value

18039

The value in [Wert] can be changed:

► double-click on the requested entry.
► Enter the new value.
► Confirm the change with [Eingabe] or reject with [ESC].

During the initialisation of the CAN:

> The values that differ from the default values are transferred to the CAN modules in the form of SDOs (Service Data Objects).
> Consequently, these values have a direct influence on the object directory of the CANopen slave.
> These values are usually rewritten with each start of the application program – irrespective of whether they are permanently stored in the CANopen slave.
> If the value was deleted without entering a new value, the default value will be transmitted during initialisation.

Master at runtime

Contents

8569

Here you find information about the functionality of the CANopen master libraries at runtime.

The CANopen master library provides the CODESYS application with implicit services which are sufficient for most applications. These services are integrated for users in a transparent manner and are available in the application without additional calls. The following description assumes that the CANopenMaster library (or the CANopen_NT library) was manually added to the library manager to use the network diagnostic, status and EMCY functions.

Services of the CANopen master library:

Reset of all configured slaves on the bus at the system start

19027

The individual NMT commands are described in the CAN document DSP301.
According to CANopen, NMT stands for **N**etwork **M**anagmen**t**.

Reset slaves one by one

19029

To reset the slaves, the NMT command "Reset Node" is used as standard, explicitly for each slave separately.

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
with FB CANx_MASTER_STATUS: • GLOBAL_START = FALSE • RESET_ALL_NODES = FALSE	with FB CANOPEN_NMTSERVICES: • NODE = node ID of the slave • NMTSERVICE = 3

Reset all slaves at once

19031

In order to avoid overload of slaves having less powerful CAN controllers, it is useful to reset all connected slaves at once using the command "Reset All Nodes".

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
with FB CANx_MASTER_STATUS: • GLOBAL_START = TRUE • RESET_ALL_NODES = TRUE	with FB CANOPEN_NMTSERVICES: • NODE = 0 • NMTSERVICE = 3

Polling of the slave device type

8021

Polling of the slave device type using SDO (polling for object 0x1000) and comparison with the configured slave ID:

> The request is repeated after 0.5 s if ...
- no device type was received
- AND the slave was **not** identified as optional in the configuration
- AND the timeout has **not** elapsed.

Indication of an error status for the slaves from which a wrong device type was received.

Configuration of all correctly detected devices

8022

Every SDO is monitored for a response and repeated if the slave does not respond within the monitoring time.

Automatic configuration of slaves

8023

Automatic configuration of slaves using SDOs while the bus is in operation:
Prerequisite: The slave logged in the master via a bootup message.

Start of all slaves configured without errors

19032

Start of all correctly configured slaves after the end of the configuration of the corresponding slave:

Start slaves one by one

19034

The NMT command "Start Node" is usually used to start the slaves.

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
with FB CANx_MASTER_STATUS: • GLOBAL_START = FALSE • START_ALL_NODES = FALSE	with FB CANOPEN_NMTSERVICES: • NODE = node ID of the slave • NMTSERVICE = 2

Start all slaves at once

19036

As is the case for the "reset", this command can be replaced by "Start All Nodes".

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
with FB CANx_MASTER_STATUS: • GLOBAL_START = TRUE • START_ALL_NODES = TRUE	with FB CANOPEN_NMTSERVICES: • NODE = 0 • NMTSERVICE = 2

Cyclical transmission of the SYNC message

8025

This value can only be set during the configuration.

Node guarding with lifetime monitoring

19038

Our recommendation: It is better to work with the heartbeat function for current devices since then the bus load is lower.

Node guarding with lifetime monitoring for every slave can be set:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
The FB CANx_MASTER_STATUS shows the error status of max. 8 slaves: > ERROR_CONTROL = list of missing network nodes (guard or heartbeat error)	List all nodes in an array for which the master has detected an error with FB CANOPEN_GETGUARDHBERRLIST: guarding error, heartbeat error > N_NODES = number of nodes with heartbeat or guarding errors > NODEID = list of node IDs with heartbeat of guarding error

Heartbeat from master to the slaves

19039

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
The FB CANx_MASTER_STATUS shows the error status of max. 8 slaves: > ERROR_CONTROL = list of missing network nodes (guard or heartbeat error)	List all nodes in an array for which the master has detected an error with FB CANOPEN_GETGUARDHBERRLIST: guarding error, heartbeat error > N_NODES = number of nodes with heartbeat or guarding errors > NODEID = list of node IDs with heartbeat of guarding error

Receiving emergency messages

19042

Receiving emergency messages for each slave with storage of the last received emergency messages:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
Read error messages with FB CANx_MASTER_STATUS: > EMERGENCY_OBJECT_SLAVES = list of current EMCY messages > GET_EMERGENCY = last generated EMCY message	Read error messages with FB CANOPEN_GETEMCYMESSAGES: > N_MSGS = number of accumulated messages > EMCY = list of emergency messages The most recent entry is in index 0

Start CANopen network

19048

Here you find information about how to start the CANopen network.

After downloading the project to the controller or a reset of the application, the master starts up the CAN network again. This always happens in the same sequence of actions:

- All slaves are reset unless they are marked as [nicht initialisieren] in the configurator. They are reset one by one using the NMT command "Reset Node" (0x81) with the node ID of the slave.
  → chapter **Start of all slaves configured without errors** (→ p. 45)
- All slaves are configured. To do so, the object 0x1000 of the slave is polled.
  - If the slave responds within the monitoring time of 0.5 seconds:
    > the next corresponding configuration SDO is sent.
  - If a slave is configured as [optional] and does not respond within the monitoring time to the polling for object 0x1000:
    > it is marked as 'not available' and
    > no further SDOs are sent to it.
  - If a slave responds to the polling for object 0x1000 with a different type than the configured one (in the lower 16 bits):
    > it will not be configured and
    > it will be marked as 'wrong type'.
- All SDOs are repeated as long as a response of the slave was seen within the monitoring time. The application program can monitor start-up of the individual slaves.

► If necessary, cancel the initialisation of a slave, if...
  • slave is not present **and**
  • slave is not configured as [optional]:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
Set the flag SET_TIMEOUT_STATE = TRUE of the FB CANx_MASTER_STATUS in the array NODE_STATE_SLAVE	When the timeout has elapsed, the FB stops waiting.

► If necessary, skip the initialisation of a slave that has responded to the polling for object 0x1000 with a different type than the configured one:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
Set the flag SET_NODE_STATE = TRUE of the FB CANx_MASTER_STATUS in the array NODE_STATE_SLAVE	with FB CANOPEN_NMTSERVICES: • NODE = node ID of the slave • NMTSERVICE = 1

> Thereby set the slave to the PRE-OPERATIONAL state.

- If the slave is later set to the OPERATIONAL state:
  > the master sends no PDOs to the slave
  > the PDOs sent by the slave are ignored.
- If the master has configured a heartbeat time that is not 0:
  > the heartbeat is generated immediately after the master controller is started.
- After all slaves have obtained their configuration SDOs:
  > the guarding begins for slaves with configured node guarding.
- If the master was configured to [automatisch starten]:
  > the NMT command "Start Remote Node" (0x01) is used
  > each slave is started individually by the master.
- If the flag GLOBAL_START was set:
  > the NMT command is used with node ID 0
  > all slaves are started with a "Start all Nodes".
- At least once all configured TX-PDOs are sent
  (for the slaves, these are RX-PDOs).

► It [automatisch starten] is deactivated, start each slave individually:
  • via the flag START_NODE in the NODE_STATE_SLAVE array or
  • → chapter **Start of all slaves configured without errors** (→ p. 45).

Network states

Contents

19050

Here you read how to interpret the states of the CANopen network and how to react.

For the start-up of the CANopen network (→ Chapter **Start CANopen network** (→ p. 47)) and during operation the individual function blocks of the library pass different states.

> **NOTE**
>
> In the monitor mode (online mode) of CODESYS the states of the CAN network can be seen in the global variable list "CANOpen implicit variables". This requires exact knowledge of CANopen and the structure of the CODESYS CANopen libraries.

To facilitate the access, the following FB is available from the CANopen master library (specific for device and CAN channel):

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
CANx_MASTER_STATUS	CANOPEN_GETSTATE

Boot up of the CANopen master

19056

During boot-up of the CAN network, the master passes different states which you can read here:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
FB CANx_MASTER_STATUS: > NODE_STATE = current status of the CANopen master	FB CANOPEN_GETSTATE: > MASTERSTATE = internal state of the master > CANSTATE = status of the CANopen network

Details→ chapter **NMT state for CANopen master** (→ p. 72)

Whenever a slave does not respond to an SDO request (upload or download), the request is repeated. The master leaves state 3, as described above, but not before all SDOs have been transmitted successfully. So it can be detected whether a slave is missing or whether the master has not correctly received all SDOs. It is of no importance for the master whether a slave responds with an acknowledgement or an abort. It is only important for the master whether he received a response at all.

An exception is a slave marked as [optional]. Optional slaves are asked only once about their object 0x1000. If they do not respond within 0.5 s, the slave is first ignored by the master and the master goes to state 5 without further reaction of this slave.

Boot up of the CANopen slaves

19057

You can see the status of a slave here (from the master's view):

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
FB CANx_SLAVE_STATUS: > NODE_STATE = current status of the CANopen slave	FB CANOPEN_GETNMTSTATESLAVE: > NMTSTATE = network operating status of the node

Details→ chapter **NMT state for CANopen slave** (→ p. 73)

Start-up of the network without [Automatic startup]

8583

Sometimes it is necessary that the application determines the instant to start the CANopen slaves. To do so, the option [Automatic startup] of the CANopen master must be deactivated in the configuration. It is then up to the application to start the slaves.

Starting the network with GLOBAL_START

19073

In a CAN network with many participants (in most cases more than 8) it often happens that NMT messages in quick succession are not detected by all (mostly slow) IO nodes (e.g. CompactModules CR2013). The reason for this is that these nodes must listen to all messages with the ID 0. NMT messages transmitted at too short intervals overload the receive buffer of such nodes.

A help for this is to reduce the number of NMT messages in quick succession:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
with FB CANx_MASTER_STATUS: ► GLOBAL_START = TRUE	with FB CANOPEN_SETSTATE: ► GlobalStart = TRUE

> The CANopen master library uses the command "Start All Nodes" instead of starting all nodes individually using the command "Start Node".

> GLOBAL_START is executed only once when the network is initialised.

> If this input is set, the controller also starts nodes with status 98 (see above). However, the PDOs for these nodes remain deactivated.

Starting the network with START_ALL_NODES

19074

If the network is not started automatically with GLOBAL_START, each node can be started one by one.

If this is not requested, the option is as follows:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
with FB CANx_MASTER_STATUS: • GLOBAL_START = FALSE • START_ALL_NODES = FALSE	with FB CANOPEN_NMTSERVICES: • NODE = node ID of the slave • NMTSERVICE = 2

- START_ALL_NODES is typically set by the application program at runtime.

> If this input is set, nodes with status 98 (see above) are started. However, the PDOs for these nodes remain deactivated.

Initialisation of the network with RESET_ALL_NODES

19075

The same reasons which apply to the command "Start All Nodes" also apply to the NMT command "Reset All Nodes" (instead of "Reset Nodes" for every individual node).

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
with FB CANx_MASTER_STATUS: • GLOBAL_START = TRUE • RESET_ALL_NODES = TRUE	with FB CANOPEN_NMTSERVICES: • NODE = 0 • NMTSERVICE = 3

> This resets all nodes once and simultaneously.

Access to the status of the CANopen master

19076

You should poll the status of the master so that the application code is not processed before the IO network is ready. The following code fragment example shows one option:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:

Variable declaration
`VAR`
`  FB_MasterStatus := CR0020_MASTER_STATUS;`
`  :`
`END_VAR`
	Variable declaration
	`VAR`
	`  FB_MasterStatus : CANOPEN_GETSTATE;`
	`  ...`
	`END_VAR`

Program code
`IF  FB_MasterStatus.NODE_STATE = 5 THEN`
`  <application code>`
`END_IF`
	Program code
	`IF  FB_MasterStatus.MASTERSTATE = 5 THEN`
	`  <application code>`
	`END_IF`

By setting the flag TIME_OUT_STATE in the array NODE_STATE_SLAVE of the FB CANx_MASTER_STATUS the application can react and, for example, jump the non configurable node.	By setting the value for the input NODE of the FB CANOPEN_GETSTATE, the application can react and skip, for example, the node that cannot be configured.

The object directory of the CANopen master

Contents

19156

In some cases it is helpful if the CANopen master has its own object directory. This enables, for example, the exchange of data of the application with other CAN nodes.

The object directory of the master is generated using an EDS file named `CRnnnnMasterODEntry.EDS` during compilation and is given default values. This EDS file is stored in the directory `CoDeSys Vn\Library\PLCconf`. The content of the EDS file can be viewed via the button [EDS...] in the configuration window [CAN parameters].

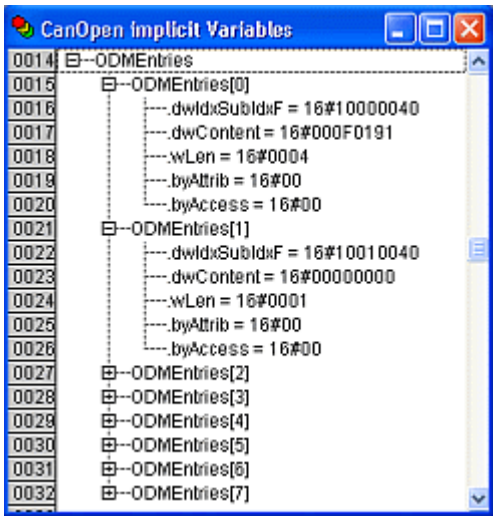
Even if the object directory is not available, the master can be used without restrictions.

Access to the object directory (controllers)

19157

For all controllers (except for CR04nn, CR253n) the following applies:

The object directory is accessed by the application via an array with the following structure:



Structure element	Description
.dwIdxSubIdxF	Structure of the component 0xiiiissff: iiii – index (2 bytes, bits 16...31), Idx ss – sub-index (1 byte, bits 8...15), SubIdx ff – flags (1 byte, bits 0...7), F Meaning of the flag bits: bit 0: write bit 1: content is a pointer to an address bit 2: mappable bit 3: swap bit 4: signed value bit 5: floating point bit 6: contains more sub-indices
.dwContent	contains the contents of the entry
.wLen	length of the data
.byAttrib	initially intended as access authorisation can be freely used by the application of the master
.byAccess	in the past access authorisation can be freely used by the application of the master

On the platform CODESYS has no editor for this object directory.

The EDS file only determines the objects used to create the object directory. The entries are always generated with length 4 and the flags (least significant byte of the component of an object directory entry `.dwIdxSubIdxF`) are always given the value 1. This means both bytes have the value 0x41.

If an object directory is available in the master, the master can act as SDO server in the network. Whenever a client accesses an entry of the object directory by writing, this is indicated to the application via the flag OD_CHANGED in CANx_MASTER_STATUS. After evaluation this flag must be reset.

The application can use the object directory by directly writing to or reading the entries or by pointing the entries to IEC variables. This means: when reading/writing to another node these IEC variables are directly accessed.

If index and sub-index of the object directory are known, an entry can be addressed as follows:

```
I := GetODMEntryValue(16#iiiiss00, pCanOpenMaster[0].wODMFirstIdx,
pCanOpenMaster[0].wODMFirstIdx + pCanOpenMaster[0].wODMCount;
```

For "iii" the index must be used and for "ss" the sub-index (as hex values).

The number of the array entry is available in I. You can now directly access the components of the entry.
It is sufficient to enter address, length and flags so that this entry can be directly transferred to an IEC variable:

```
ODMEntries[I].dwContent := ADR(<variable name>);
ODMEntries[I].wLen := sizeof(<variable name>);
ODMEntries[I].dwIdxSubIdxF := ODMEntries[I].dwIdxSubIdxF OR OD_ENTRYFLG_WRITE OR
OD_ENTRYFLG_ISPOINTER;
```

It is sufficient to change the content of "`.dwContent`" to change only the content of the entry.

Access to the object directory (others)

19158

For all CR04nn, CR1nnn, CR253n the following applies:

The object directory is accessed by the application via function blocks:
- CANOPEN_GETODCHANGEDFLAG,
- CANOPEN_READOBJECTDICT,
- CANOPEN_WRITEOBJECTDICT.

On the platform CODESYS has no editor for this object directory.

The EDS file only determines the objects used to create the object directory.

If an object directory is available in the master, the master can act as SDO server in the network. Whenever a client accesses an entry of the object directory by writing, this is indicated to the application via the flag ODCHANGED in CANOPEN_GETODCHANGEDFLAG. After evaluation this flag must be reset by the input RESETFLAG=TRUE.

The application can use the object directory by directly writing to or reading the entries.

3.4.4 CANopen slave

Contents

1865

A CODESYS programmable controller can also be a CANopen slave in a CAN network.

Functionality of the CANopen slave library

19161

The CANopen slave library in combination with the CANopen configurator provides the user with the following options:

- In CODESYS: configuration of the properties for nodeguarding/heartbeat, emergency, node ID and baud rate at which the device is to operate.
- Together with the parameter manager in CODESYS, a default PDO mapping can be created which can be changed by the master at runtime. The configuration of the PDO mapping is changed by the master during the configuration phase. By means of mapping IEC variables of the application can be mapped to PDOs. This means IEC variables are assigned to the PDOs to be able to easily evaluate them in the application program.
- The CANopen slave library provides an object directory. The size of this object directory is defined while compiling CODESYS. This directory contains all objects which describe the CANopen slave and in addition the objects defined by the parameter manager. In the parameter manager only the list types parameters and variables can be used for the CANopen slave.
- The CANopen slave manages the access to the object directory, i.e. it acts as SDO server on the bus.
- The CANopen slave monitors nodeguarding or the heartbeat consumer time (always only of one producer) and sets corresponding error flags for the application.
- An EDS file can be generated which describes the configured properties of the CANopen slave so that the device can be integrated and configured as a slave under a CANopen master.

The CANopen slave library explicitly does not provide the following functionalities described in CANopen (all options of the CANopen protocol which are not indicated here or in the above section are not implemented either):

- Dynamic SDO and PDO identifiers
- SDO block transfer
- Automatic generation of emergency messages. Emergency messages must always be generated by the application program. To do so, the CANopen slave library provides these FBs:
  - **For all controllers (except for CR04nn, CR253n) the following applies:**

CANx_SLAVE_EMCY_HANDLER	Handles the device-specific error status of the CANopen slave on CAN interface x: • error register (index 0x1001) and • error field (index 0x1003) of the CANopen object directory x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_SLAVE_SEND_EMERGENCY	Sends application-specific error status of the CANopen slave on CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)

  - **For all CR04nn, CR1nnn, CR253n the following applies:**

CANOPEN_GETERRORREGISTER	= Get CANopen error register Reads the error registers 0x1001 and 0x1003 from the controller The registers can be reset by setting the respective inputs.
CANOPEN_GETEMCYMESSAGES	= Get CANopen emergency messages Lists all emergency messages that have been received by the controller from other nodes in the network since the last deletion of messages The list can be reset by setting the according input.
CANOPEN_SENDEMCYMESSAGE	= CANopen send emergency message Sends an EMCY message. The message is assembled from the according parameters and entered in register 0x1003

- Dynamic changes of the PDO properties are currently only accepted on arrival of a StartNode NMT message, not with the mechanisms defined in CANopen.

Configure CANopen slave

Contents

19163
1980

Request: using the controller as CANopen slave:

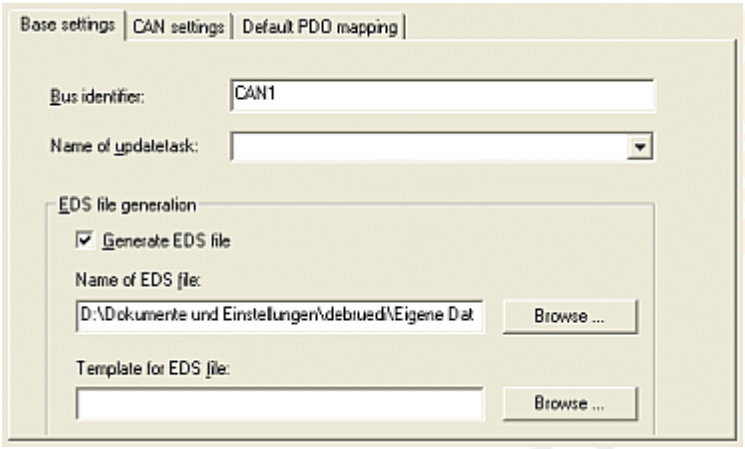
► In the PLC configuration the CANopen slave must be added:
right click on the first line ("CRnnnn Configuration Vnn")
[Append Subelement] > [CANopen Slave...]

> If several CANopen-capable interfaces are available, then the following applies for the assignment of the CANopen protocol to the CAN interface (depending on the device):
→ chapter 'CAN interfaces and CAN protocols'

> All required libraries are automatically added to the library manager.

Tab [Base settings]

1981



Base settings: Bus identifier

10049

Parameter is currently not used.

Base settings: Name of updatetask

10050

Name of the task where the CANopen slave is called.

Base settings: Generate EDS file

10051

If an EDS file is to be generated from the settings to be able to add the CANopen slave to any master configuration, the option [Generate EDS file] must be activated and the name of a file must be indicated. As an option a template file can be indicated whose entries are added to the EDS file of the CANopen slave. In case of overlapping the template definitions are not overwritten.

Example of an object directory

1991

The following entries could for example be in the object directory:

```
[FileInfo]
FileName=D:\CoDeSys\lib2\plcconf\MyTest.eds
FileVersion=1
FileRevision=1
Description=EDS for CoDeSys-Project: D:\CoDeSys\CANopenTestprojekte\TestHeartbeatODsettings_Device.pro
CreationTime=13:59
CreationDate=09-07-2005
CreatedBy=CoDeSys
ModificationTime=13:59
ModificationDate=09-07-2005
ModifiedBy=CoDeSys

[DeviceInfo]
VendorName=3S Smart Software Solutions GmbH
ProductName=TestHeartbeatODsettings_Device
ProductNumber=0x33535F44
ProductVersion=1
ProductRevision=1
OrderCode=xxxx.yyyy.zzzz
LMT_ManufacturerName=3S GmbH
LMT_ProductName=3S_Dev
BaudRate_10=1
BaudRate_20=1
BaudRate_50=1
BaudRate_100=1
BaudRate_125=1
BaudRate_250=1
BaudRate_500=1
BaudRate_800=1
BaudRate_1000=1
SimpleBootUpMaster=1
SimpleBootUpSlave=0
ExtendedBootUpMaster=1
ExtendedBootUpSlave=0

...

[1018sub0]
ParameterName=Number of entries
ObjectType=0x7
DataType=0x5
AccessType=ro
DefaultValue=2
PDOMapping=0

[1018sub1]
ParameterName=VendorID
ObjectType=0x7
DataType=0x7
AccessType=ro
DefaultValue=0x0
PDOMapping=0

[1018sub2]
ParameterName=Product Code
ObjectType=0x7
DataType=0x7
AccessType=ro
DefaultValue=0x0
PDOMapping=0
```

For the meaning of the individual objects please see the CANopen specification DS301.

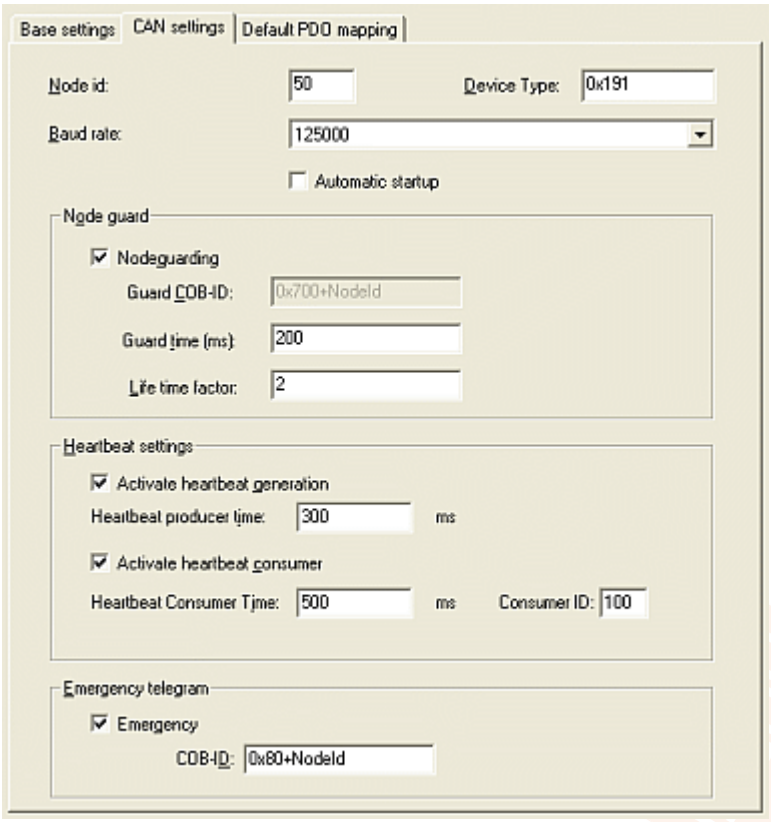
In addition to the prescribed entries, the EDS file contains the definitions for SYNC, guarding, emergency and heartbeat. If these objects are not used, the values are set to 0 (preset). But as the objects are present in the object directory of the slave at runtime, they are written to in the EDS file.

The same goes for the entries for the communication and mapping parameters. All 8 possible sub-indices of the mapping objects 0x16nn or 0x1Ann are present, but possibly not considered in the sub-index 0.

Bit mapping is not supported by the library!

Tab [CAN settings]

1982



Here you can set the **node ID** and the **baud rate**.

Device type
(this is the default value of the object 0x1000 entered in the EDS) has 0x191 as default value (standard IO device) and can be freely changed.
The index of the CAN controller results from the position of the CANopen slave in the controller configuration.

The **nodeguarding** parameters, the **heartbeat** parameters and the emergency COB ID can also be defined in this tab. The CANopen slave can only be configured for the monitoring of a heartbeat.
We recommend: It is better to work with the heartbeat function for current devices since then the bus load is lower.

NOTE

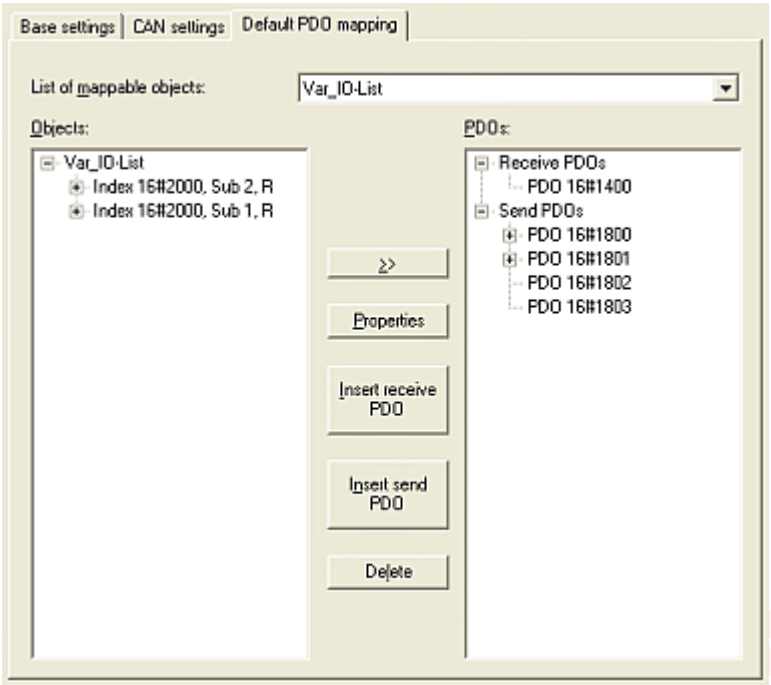
When applying guarding or heartbeat AND
when creating an EDS file for integration with a CANopen master:

► enter guard time = 0
enter life time factor = 0
enter heartbeat time = 0

> The values set for the CANopen master are transmitted to the CANopen slave during configuration. Thus, the CANopen master has safely activated the guarding or heartbeat for this node.

Tab [Default PDO mapping]

1983



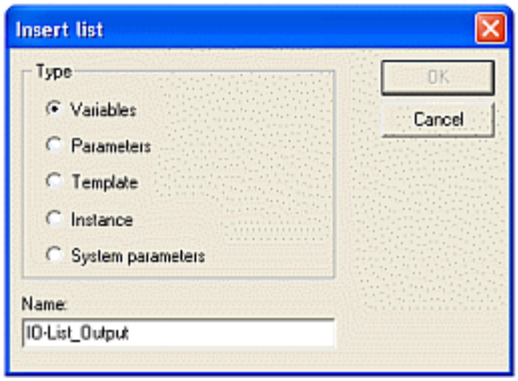
In this tab the assignment between local object directory (OD editor) and PDOs transmitted/received by the CANopen slave can be defined. Such an assignment is called "mapping".

In the object directory entries used (variable OD) the connection to variables of the application is made between object index/sub-index. You only have to ensure that the sub-index 0 of an index containing more than one sub-index contains the information concerning the number of the sub-indices.

Example: list of variables

10052

On the first receive PDO (COB ID = 512 + node ID) of the CANopen slave the data für variable PLC_PRG.a shall be received.



Info

[Variables] and [parameters] can be selected as list type.

For the exchange of data (e.g. via PDOs or other entries in the object directory) a variable list is created.

The parameter list should be used if you do not want to link object directory entries to application variables. For the parameter list only the index 1006_{16} / SubIdx 0 is currently predefined. In this entry the value for the "Com. Cycle Period" can be entered by the master. This signals the absence of the SYNC message.

So you have to create a variable list in the object directory (parameter manager) and link an index/sub-index to the variable `PLC_PRG.a`.

- ► To do so, add a line to the variable list (a click on the right mouse button opens the context menu) and enter a variable name (any name) as well as the index and sub-index.
- ► The only allowed access right for a receive PDO is [write only].
- ► Enter "`PLC_PRG.a`" in the column [variable] or press [F2] and select the variable.

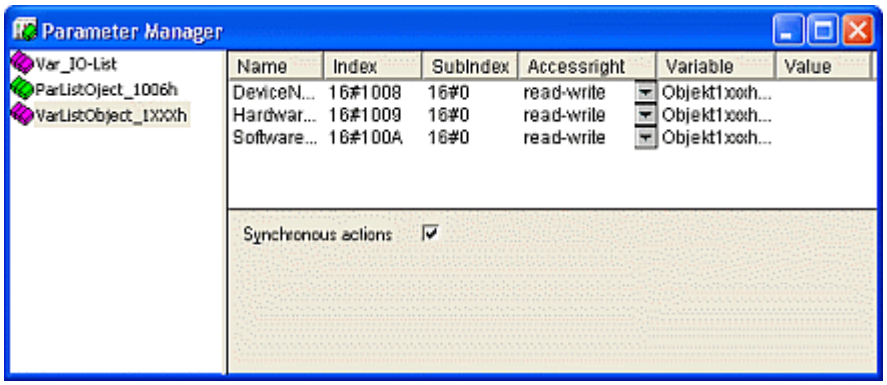
NOTE

Data to be read by the CANopen master (e.g. inputs, system variables) must have the access right [read only].

Data to be written by the CANopen master (e.g. outputs in the slave) must have the access right [write only].

SDO parameters to be written and at the same time to be read from and written to the slave application by the CANopen master must have the access right [read-write].

To be able to open the parameter manager the parameter manager must be activated in the target settings under [Network functionality]. The areas for index/sub-index already contain sensible values and should not be changed.



In the default PDO mapping of the CANopen slave the index/sub-index entry is then assigned to a receive PDO as mapping entry. The PDO properties can be defined via the dialogue known from chapter **Add and configure CANopen slaves** (→ p. 39).

Only objects from the parameter manager with the attributes [read only] or [write only] are marked in the possibly generated EDS file as mappable (= can be assigned) and occur in the list of the mappable objects. All other objects are not marked as mappable in the EDS file.

Changing the standard mapping by the master configuration

1984

You can change the default PDO mapping (in the CANopen slave configuration) within certain limits by the master.

The following applies:
The CANopen slave can only recreate entries in the object directory which are already available in the standard mapping (default PDO mapping in the CANopen slave configuration).
For a PDO, for example, which contains a mapped object in the default PDO mapping no second object can be mapped in the master configuration.

So the mapping changed by the master configuration can at most contain the PDOs available in the standard mapping. Within these PDOs there are 8 mapping entries (sub-indices).

Possible errors which may occur are not displayed, i.e. the supernumerary PDO definitions / supernumerary mapping entries are processed as if not present.

In the master the PDOs must always be created as follows:
- starting from 0x1400 (receive PDO communication parameter) or
- starting from 0x1800 (transmit PDO communication parameter)
- and follow each other without interruption.

Access to the CANopen slave at runtime

1985

Setting the node number of a CANopen slave

19165

The node number can be set at the CANopen slave during the runtime of the application program:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
use the FB CANx_SLAVE_NODEID from the CANopen slave library	the FB CANOPEN_SETSTATE from the library `ifm_CANopen_NT_Vxxyyzz.LIB` panels

Setting the baud rate of a CANopen slave

19166

The baud rate can be set at the CANopen slave during the runtime of the application program:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
use one of the following FBs from the device `ifm_CRnnnn_Vxxyyzz.LIB` library: • CAN1_BAUDRATE or • CAN1_EXT or • CANx.	use the FB CANOPEN_ENABLE from the `ifm_CANopen_NT_Vxxyyzz.LIB` library

Access to the OD entries by the application program

19167

As standard, there are entries in the object directory which are mapped to variables (parameter manager).

However, there are also automatically generated entries of the CANopen slave which cannot be mapped to the contents of a variable via the parameter manager. These entries are available here:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
use the FB CANx_SLAVE_STATUS from the CANopen slave library	• CANOPEN_READOBJECTDICT • CANOPEN_WRITEOBJECTDICT from the library `ifm_CANopen_NT_Vxxyyzz.LIB`

Change the PDO properties at runtime

1988

If the properties of a PDO are to be changed at runtime, this is done by another node via SDO write access as described by CANopen.

As an alternative, it is possible to directly write a new property, e.g. the "event time" of a send PDO and then transmit a command "StartNode-NMT" to the node although it has already been started. As a result of this the device reinterprets the values in the object directory.

Send emergency messages via the application program

19168

Send an emergency message from the application program:

- **For all controllers (except for CR04nn, CR253n) the following applies:**

CANx_SLAVE_EMCY_HANDLER	Handles the device-specific error status of the CANopen slave on CAN interface x: • error register (index 0x1001) and • error field (index 0x1003) of the CANopen object directory x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_SLAVE_SEND_EMERGENCY	Sends application-specific error status of the CANopen slave on CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)

- **For all CR04nn, CR1nnn, CR253n the following applies:**

CANOPEN_GETERRORREGISTER	= Get CANopen error register Reads the error registers 0x1001 and 0x1003 from the controller The registers can be reset by setting the respective inputs.
CANOPEN_GETEMCYMESSAGES	= Get CANopen emergency messages Lists all emergency messages that have been received by the controller from other nodes in the network since the last deletion of messages The list can be reset by setting the according input.
CANOPEN_SENDEMCYMESSAGE	= CANopen send emergency message Sends an EMCY message. The message is assembled from the according parameters and entered in register 0x1003

3.4.5 CANopen tables

Contents

9941

The following tables will inform you about important values and settings of the CANopen interfaces.

Structure of CANopen messages

Contents

9971

A CANopen message consists of the COB ID and up to 8-byte data:

COB ID			DLC	Byte 1		Byte 2		Byte 3		Byte 4		Byte 5		Byte 6		Byte 7		Byte 8	
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Details are given in the following chapters.

Please note the reversed byte order! (⇨ Little Endian or Intel format)

Examples:

Value [hex]	Data type	Byte 1		Byte 2		Byte 3		Byte 4		Byte 5		Byte 6		Byte 7		Byte 8	
12	BYTE	1	2	–	–	–	–	–	–	–	–	–	–	–	–	–	–
1234	WORD	3	4	1	2	–	–	–	–	–	–	–	–	–	–	–	–
12345678	DWORD	7	8	5	6	3	4	1	2	–	–	–	–	–	–	–	–

Structure of the COB ID

9972

The first part of a message is the COB ID. Structure of the 11-bit COB ID:

Nibble 0				Nibble 1				Nibble 2			
11	10	9	8	7	6	5	4	3	2	1	0
--	**3**	**2**	**1**	**0**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
--	**function code**				**node ID**						

The COB ID consists of the **Function code / Predefined Connectionset** (→ p. 67) and the node ID.

Example:
Communication object = TPDO1 (TX)
Node number of the device = 0x020 = 32

Calculation:
Function code for the communication object TPDO1 = 0x03
Significance of the function code in the 11-bit COB ID = 0x03 • 0x80 = 0x180
Add the node number (0x020) ⇒ the COB ID is: 0x1A0

1				A				0			
3	2	1	0	3	2	1	0	3	2	1	0
0	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
--	0x03 = 3				0x020 = 32						

Function code / Predefined Connectionset

9966

In the "CANopen Predefined Connectionset" some function codes are predefined.

When using the predefined connectionset you can operate a CANopen network of up to 127 participants without the risk of a double assignment of COB IDs.

Broadcast or multicast messages:

Communication object	Function code [hex]	COB ID [hex]	Related parameter objects [hex]
NMT	0	000	
SYNC	1	080	1005, 1006, 1007, 1028
TIME	2	100	1012, 1013

Point-to-point messages:

Communication object	Function code [hex]	COB ID [hex]	Related parameter objects [hex]
EMERGENCY	1	080 + node ID	1014, 1015
TPDO1 (TX)	3	180 + node ID	1800
RPDO1 (RX)	4	20016 + node ID	1400
TPDO2 (TX)	5	280 + node ID	1801
RPDO2 (RX)	6	30016 + node ID	1401
TPDO3 (TX)	7	380 + node ID	1802
RPDO3 (RX)	8	400 + node ID	1402
TPDO4 (TX)	9	480 + node ID	1803
RPDO4 (RX)	A	500 + node ID	1403
Default SSDO (TX)	B	58016 + node ID	1200
Default CSDO (RX)	C	60016 + node ID	1280
NMT Error Control	E	70016 + node ID	1016, 1017

TX = slave sends to master
RX = slave receives from master

SSDO = server SDO
CSDO = client SDO

SDO command bytes

9968

Structure of an SDO message:

COB-ID	DLC	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
XXX	8	Command	Index		Sub index	Data depending on the data to be transmitted			

Please note the reversed byte order! (⇨ Little Endian or Intel format)

An SDO COB ID consists of:

CANopen	
Node ID	**COB ID SDO**
1…127	TX: 0x580 + node ID
	RX: 0x600 + node ID

TX = slave sends to master
RX = slave receives from master

DLC = **D**ata **L**ength **C**ode = in CANopen the number of the data bytes in a message.
For →SDO: DLC = 8

SDO command bytes:

Command hex	dec	Message	Data length	Description
21	33	request	more than 4 bytes	send data to slave
22	34	request	1…4 bytes	send data to slave
23	35	request	4 bytes	send data to slave
27	39	request	3 bytes	send data to slave
2B	43	request	2 bytes	send data to slave
2F	47	request	1 byte	send data to slave
40	64	request	---	request data from slave
42	66	response	1…4 bytes	send data from slave to master
43	67	response	4 bytes	send data from slave to master
47	71	response	3 bytes	send data from slave to master
4B	75	response	2 bytes	send data from slave to master
4F	79	response	1 byte	send data from slave to master
60	96	response	---	data transfer ok: send confirmation of receipt from slave to master
80	128	response	4 bytes	data transfer failed send abort message from slave to master → chapter **SDO abort code** (→ p. 69)

SDO abort code

9970

The SDO abort code is NOT part of the emergency message!

Abord code [hex]	Description
0503 0000	toggle bit not alternated
0504 0000	SDO protocol timed out
0504 0001	client/server command specifier not valid or unknown
0504 0002	invalid block size (block mode only)
0504 0003	invalid sequence number (block mode only)
0504 0004	CRC error (block mode only)
0504 0005	out of memory
0601 0000	unsupported access to an object
0601 0001	attempt to read a write only object
0601 0002	attempt to write a read only object
0602 0000	object does not exist in the object dictionary
0604 0041	object cannot be mapped to the PDO
0604 0042	the number and length of the objects to be mapped would exceed PDO length
0604 0043	general parameter incompatibility reason
0604 0047	general internal incompatibility in the device
0606 0000	access failed due to an hardware error
0607 0010	data type does not match, length of service parameter does not match
0607 0012	data type does not match, length of service parameter too high
0607 0013	data type does not match, length of service parameter too low
0609 0011	sub-index does not exist
0609 0030	value range of parameter exceeded (only for write access)
0609 0031	value of parameter written too high
0609 0032	value of parameter written too low
0609 0036	maximum value is less than minimum value
0800 0000	general error
0800 0020	data cannot be transferred or stored to the application
0800 0021	data cannot be transferred or stored to the application because of local control
0800 0022	data cannot be transferred or stored to the application because of the present device state
0800 0023	object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error)

Boot-up message

9961

After booting, the CAN participate sends the boot-up message once:

	COB ID	DLC	Byte 1
hex	0x700 + node ID	0x1	0x00
dec	1 792 + node ID	1	0

The participant is now capable of communicating in the CAN network.

DLC = **D**ata **L**ength **C**ode = in CANopen the number of the data bytes in a message.
For →SDO: DLC = 8

Example:
The node ID of the participant is 0x7D = 125.
The COP ID of the boot-up message is: 0x77D = 1 917

Deviation:
There are devices that cannot send a [0x700 + Node ID] (these are devices that were built before version 4 of the CANopen specification).
These devices send the following boot-up message and without status instead:

	COB ID	DLC
hex	0x080 + node ID	0x0
dec	128 + node ID	0

Network management (NMT)

Contents

9974

Network management commands

9962

With the following network management commands the user can influence the operating mode of individual or all CAN participants. Structure:

COB ID	DLC	Byte 1	Byte 2
0x000	X	Command	Node ID

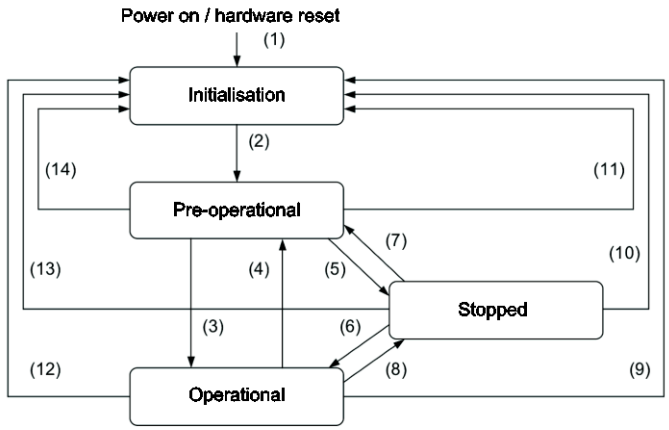
Node ID = 00 ⇨ command valid for all nodes in the network at the same time

COB ID	NMT command		Description	
0x000	0x01 = 01	Node ID	start_remode_node	Set the node to the "operational" state
0x000	0x02 = 02	Node ID	stop_remode_node	Set the node to the "STOPPED" state
0x000	0x80 = 128	Node ID	enter_pre-operational	Set the node to the "PRE-OPERATIONAL" state
0x000	0x81 = 129	Node ID	reset_node	Node reset
0x000	0x82 = 130	Node ID	reset_communication	Reset the CAN communication of the node

NMT state

9963

The status byte informs about the state of the CAN participant.



Graphics: state transitions under CANopen

Permitted transitions:

(1) State is automatically reached at power on

(2) Internal initialisation completed – node automatically goes to PRE-OPERATIONAL

(3) NMT service "Start Remote Node"

(4) + (7) NMT service "Enter PRE-OPERATIONAL"

(5) + (8) NMT service "Stop Remote Node"

(6) NMT service "Start Remote Node"

(9)...(11) NMT service "Reset Node"

(12)...(14) NMT service "Reset Communication"

NMT state for CANopen master

9964

These statuses show the internal status of the CANopen master stack.
They are not defined by the CANopen specification.

State hex	dec	Description
00	0	not defined
01	1	Master waits for a boot-up message of the node. OR: Master waits for the expiry of the given guard time.
02	2	• Master waits for 300 ms. • Master requests the object 0x1000. • Then the state is set to 3.
03	3	The master configures its slaves. To do so, all SDOs generated by the configurator are transmitted to the slaves one after the other. Then the master changes to status 5.
05	5	After transmission of all SDOs to the slaves the master goes to state 5 and remains in this state. State 5 is the normal operating state for the master.

To read the node state out of the FB:

Used function block	Node state is found here
CANx_MASTER_STATUS	output NODE_STATE
CANOPEN_GETSTATE	output MASTERSTATE

NMT state for CANopen slave

9965

State hex	dec	Description
FF	-1	The slave is reset by the NMT message "Reset Node" and automatically goes to state 1.
00	0	not defined
01	1	state = waiting for BOOTUP After max. 2 s or immediately on reception of its boot up message the slave goes to state 2.
02	2	state = BOOTUP After a delay of 0.5 s the slave automatically goes to state 3.
03	3	state = PREPARED The slave is configured in state 3. The slave remains in state 3 as long as it has received all SDOs generated by the configurator. It is not important whether during the slave configuration the response to SDO transfers is abort (error) or whether the response to all SDO transfers is no error. Only the response as such received by the slave is important – not its contents. If in the configurator the option "Reset node" has been activated, a new reset of the node is carried out after transmitting the object 0x1011 sub-index 1 which then contains the value "load". The slave is then polled again with the upload of the object 0x1000. Slaves with a problem during the configuration phase remain in state 3 or directly go to an error state (state > 5) after the configuration phase.
04	4	state = PRE-OPERATIONAL A node always goes to state 4 except for the following cases: • it is an "optional" slave and it was detected as non available on the bus (polling for object 0x1000) OR: • the slave is present but reacted to the polling for object 0x1000 with a type in the lower 16 bits other than expected by the configurator.
05	5	state = OPERATIONAL State 5 is the normal operating state of the slave: [Normal Operation]. If the master was configured to [Automatic startup], the slave starts in state 4 (i.e. a "start node" NMT message is generated) and the slave goes automatically to state 5. If the flag GLOBAL_START was set, the master waits until all slaves are in state 4. All slaves are then started with the NMT command [Start All Nodes].
61	97	A node goes to state 97 if it is optional (optional device in the CAN configuration) and has not reacted to the SDO polling for object 0x1000. If the slave is connected to the network and detected at a later point in time, it is automatically started. To do so, you must have selected the option [Automatic startup] in the CAN parameters of the master.
62	98	A node goes to state 98 if the device type (object 0x1000) does not correspond to the configured type.
63	99	In case of a nodeguarding timeout the slave is set to state 99. As soon as the slave reacts again to nodeguard requests and the option [Automatic startup] is activated, it is automatically started by the master. Depending on the status contained in the response to the nodeguard requests, the node is newly configured or only started. To start the slave manually it is sufficient to use the method [NodeStart].

Nodeguard messages are transmitted to the slave ...
- if the slave is in state 4 or higher AND
- if nodeguarding was configured.

To read the node state out of the FB:

Used function block	Node state is found here
CANx_MASTER_STATUS CANx_SLAVE_STATUS	output NODE_STATE
CANOPEN_GETSTATE	output NODESTATE

CANopen status of the node

1973

Node status according to CANopen (with these values the status is also coded by the node in the corresponding messages).

Status hex | dec		CANopen status:	Description
00	0	BOOTUP	BOOTUP message of the node
04	4	STOPPED	The node is in the status STOPPED. There is no exchange of data and the node cannot be configured, either.
05	5	OPERATIONAL	The node is in the status OPERATIONAL and participates in the normal exchange of data.
7F	127	PRE-OPERATIONAL	The node is in the status PRE-OPERATIONAL and can be configured by the master.

If nodeguarding active: the most significant status bit toggles between the messages.

Read the node status from the function block:

Function block used	Node status is found here
CANx_MASTER_STATUS	Structure element LAST_STATE from the array NODE_STATE_SLAVE
CANx_SLAVE_STATUS	Output NODE_STATE
CANOPEN_GETSTATE	Output LASTNODESTATE

3.5 CANopen network variables

Contents

1868

3.5.1 General information

2076

CAN network variables are one option to exchange data between two or several controllers. For users the mechanism should be easy to use. At present network variables are implemented on the basis of CAN and Ethernet (UDP/IP).

The variable values are automatically exchanged on the basis of broadcast messages.
- in UDP as broadcast messages,
- in CAN as messages comparable to the PDOs of CANopen.

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).

The CANopen object directory is another option to exchange variables. This is a 1 to 1 connection using a confirmed protocol. The user can check whether the message arrived at the receiver. The exchange is not carried out automatically but via the call of FBs from the application program.
→ chapter **The object directory of the CANopen master** (→ p. 52)

3.5.2 Configuration of CANopen network variables

Contents

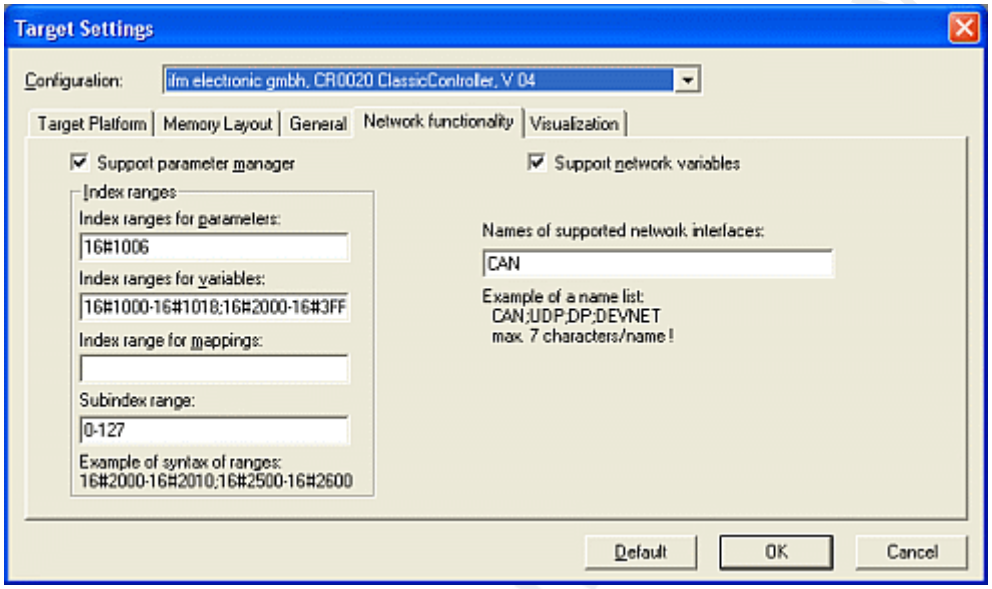
1869

To use the network variables with CODESYS you need the following libraries:
- 3s_CanDrv.lib
- 3S_CANopenManager.lib
- 3S_CANopenNetVar.lib
- SysLibCallback.lib.

CODESYS automatically generates the required initialisation code and the call of the network blocks at the start and end of the cycle.

Settings in the target settings

1994



Example: target settings for ClassicController CR0020

► Select the dialogue box [Target settings].
► Select the tab [Network functionality].
► Activate the check box [Support network variables].
► Enter the name of the requested network, here CAN, in [Names of supported network interfaces].
► To use network variables you must also add a CANopen master or CANopen slave (device) to the controller configuration.
► Please note the particularities when using network variables for the corresponding device types.
→ Chapter **Particularities for network variables** (→ p. 80)

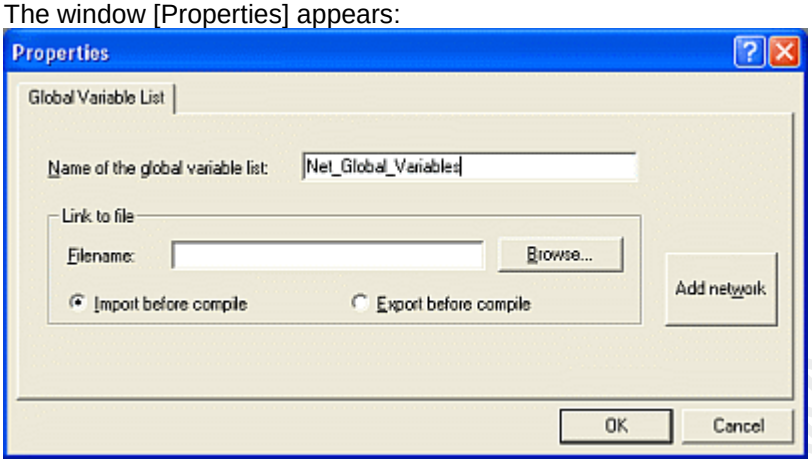
Settings in the global variable lists

1995

► Create a new global variable list. In this list the variables to be exchanged with other controllers are defined.

► Open the dialogue with the menu point [Object Properties].

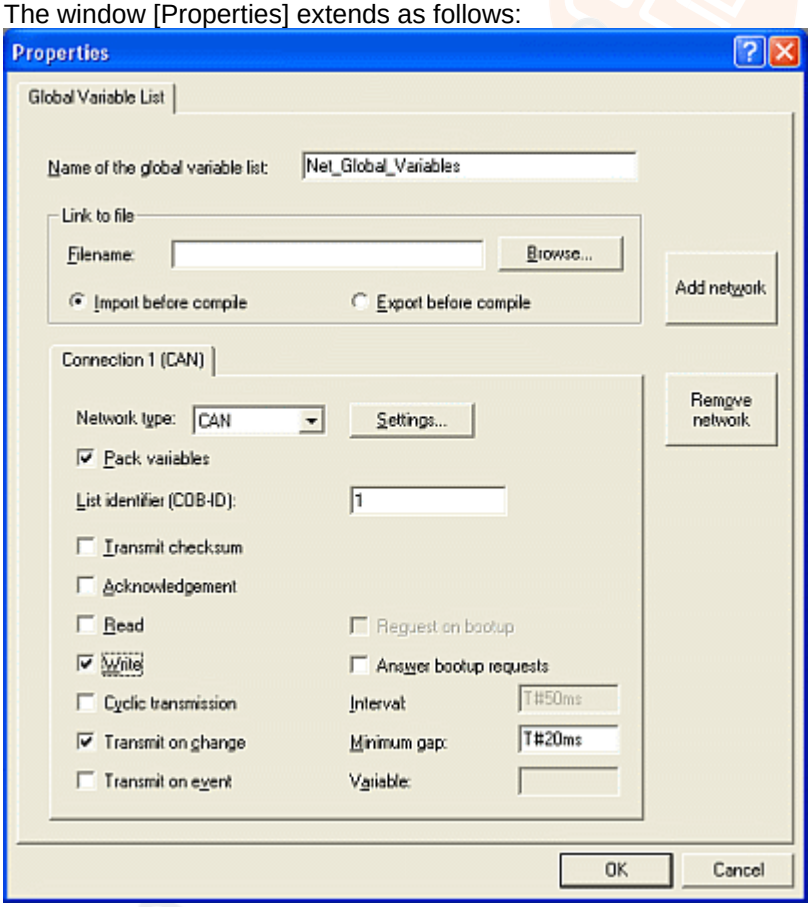
> The window [Properties] appears:



If you want to define the network properties:

► Click the button [Add network].
If you have configured several network connections, you can also configure here several connections per variable list.

> The window [Properties] extends as follows:



Meaning of the options:

Global variable list: Network type

10055

As network type you can enter one of the network names indicated in the target settings.
If you click on the button [Settings] next to it, you can select the CAN interface:

1. CAN interface: value = 0
2. CAN interface: value = 1

etc.

Global variable list: Pack variables

10056

If this option is activated with [**v**], the variables are combined, if possible, in one transmisson unit. For CAN the size of a transmission unit is 8 bytes.

If it is not possible to include all variables of the list in one transmission unit, several transmission units are formed for this list.

If the option is not activated, every variable has its own transmission unit.

If [Transmit on change] is configured, it is checked separately for every transmission unit whether it has been changed and must be transmitted.

Global variable list: List identifier (COB-ID)

10057

The basic identifier is used as a unique identification to exchange variable lists of different projects. Variable lists with identical basic identifier are exchanged.

► Note that the definitions of the variable lists with the same basic identifier match in the different projects.

NOTE

In CAN networks the COB ID is directly used as identifier of the CAN messages. It is not checked whether the identifier is also used in the remaining CAN configuration.

To ensure a correct exchange of data between two controllers the global variable lists in the two projects must match. To ensure this you can use the feature [Link to file]. A project can export the variable list file before compilation, the other projects should import this file before compilation.

In addition to simple data types a variable list can also contain structures and arrays. The elements of these combined data types are transmitted separately.

► Strings must not be transmitted via network variables!
Otherwise a runtime error will occur and the watchdog will be activated.

If a variable list is larger than a PDO of the corresponding network, the data is split up to several PDOs. Therefore it cannot be ensured that all data of the variable list is received in **one** cycle. Parts of the variable list can be received in different cycles. This is also possible for variables with structure and array types.

Global variable list: Transmit checksum

10058

This option is not supported.

Global variable list: Acknowledgement

10059

This option is not supported.

Global variable list: Read

10060

The variable values of one (or several) controllers are read.

Global variable list: Write

10061

The variables of this list are transmitted to other controllers.

> **NOTE**
>
> You should only select one of these options for every variable list, i.e. either only read or only write.
>
> If you want to read or write several variables of a project, please use several variable lists (one for reading, one for writing).
>
> To get the same data structure for the communication between two participants you should copy the variable list from one controller to the other.
>
> In a network the same variable list should only be exchanged between two participants.

Global variable list: Cyclic transmission

10062

Only valid if [write] is activated. The values are transmitted in the specified [interval] irrespective of whether they have changed.

Global variable list: Transmit on change

10063

The variable values are only transmitted if one of the values has been changed. With [Minimum gap] (value > 0) a minimum time between the message packages can be defined.

Global variable list: Transmit on event

13327

If this option is selected, the CAN message is only transmitted if the indicated binary [variable] is set to TRUE. This variable cannot be selected from the list of the defined variables via the input help.

3.5.3 Particularities for network variables

11811

Network variables are supported on the following interface(s):
- CAN 1 (value = 0)
- CAN 2 (value = 1)
- CAN 3 (value = 2)
- CAN 4 (value = 3)

11818

NOTE

► Enter the identifier of the network variables and of the receive PDOs as **decimal** values!

3.6 Summary CAN / CANopen / network variables

7946

- The COB ID of the network variables must differ from the CANopen slave ID in the controller configuration and from the IDs of the transmit and receive blocks!
- If more than 8 bytes of network variables are put into one COB ID, CODESYS automatically expands the data packet to several successive COB IDs. This can lead to conflicts with manually defined COB IDs!
- Network variables cannot transport any string variables.
- Network variables can be transported...
  - if a variable becomes TRUE (Event),
  - in case of data changes in the network variable or
  - cyclically when the timer has elapsed
- The interval time is the period between transmissions if cyclical transmission has been selected. The minimum distance is the waiting time between two transmissions, if the variable changes too often.
- To reduce the bus load, split the messages via network variables or CANx_TRANSMIT to several plc cycles using several events.
- In the controller configuration the values for [Com Cycle Period] and [Sync. Window Length] should be identical.
- If [Com Cycle Period] is selected for a slave, the slave searches for a Sync object of the master during exactly this period. This is why the value for [Com Cycle Period] must be higher than the [Master Synch Time].
- We recommend to select "optional startup" for slaves and "automatic startup" for the network. This reduces unnecessary bus load and allows a briefly lost slave to integrate into the network again.
- We recommend to set analogue inputs to "synchronous transmission" to avoid bus overload.
- Binary inputs, especially the irregularly switching ones, should best be set to "asynchronous transmission" using an event timer.
- To be considered during the monitoring of the slave status:
  - after the start of the slaves it takes a while until the slaves are operational.
  - when the system is switched off, slaves can indicate an incorrect status change due to early voltage loss.

3.7 CAN for the drive engineering

Contents

7678

With the standard J1939 the SAE offers to the user a CAN bus protocol for the drive engineering. The CAN messages are transferred with a 29-bit identifier. Due to the longer identifier numerous messages can be directly assigned to the identifier.

For writing the protocol this advantage was used and certain messages were combined in ID groups. The ID assignment is specified in the standards SAE J1939 and ISO 11992. The protocol of ISO 11992 is based on the protocol of SAE J1939.

Standard	Application area
SAE J1939	Engine management
ISO 11992	Truck & Trailer Interface

As for the software protocol the two standards do not differ because ISO 11992 is based on SAE J1939. Concerning the hardware interface, however, there is one difference: higher voltage level for ISO 11992.

To use the functions to SAE J1939 / ISO 11992 the protocol description of the aggregate manufacturer (e.g. for engines, gears) is definitely needed. For the messages implemented in the aggregate control device this description must be used because not every manufacturer implements all messages or implementation is not useful for all aggregates.

The following information and tools should be available to develop programs for functions to SAE J1939:

- list of the data to be used by the aggregates
- overview list of the aggregate manufacturer with all relevant data
- CAN monitor with 29-bit support
- if required, the standard SAE J1939

3.7.1 Identifier acc. to SAE J1939

7675

For the data exchange with SAE J1939 the 29 bit identifiers are determinant. This identifier is pictured schematically as follows:

A	SOF	Identifier 11 bits											SRR	IDE	Identifier 18 bits																	RTR
B	SOF	Priority			R	DP	PDU format (PF) 6+2 bits						SRR	IDE	still PF		PDU specific (PS) destination address group extern or proprietary								Source address								RTR
	1	3	2	1	1	1	8	7	6	5	4	3	1	1	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	1
C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
D	-	28	27	26	25	24	23	22	21	20	19	18	-	-	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-

Legend:

A = CAN extended message format
B = J1939 message format
C = J1939 message bit position
D = CAN 29 bit ID position

SOF = **S**tart **o**f **f**rame
SRR = **S**ubstitute **r**emote **r**equest
IDE = **Id**entifier **e**xtension flag
RTR = **R**emote **t**ransmission **r**equest
PDU = **P**rotocol **D**ata **U**nit
PGN = **P**arameter **G**roup **N**umber = PDU format (PF) + PDU source (PS)

(→ **COB-ID** (→ p. 27))

To do so, the 3 essentially communication methods with SAE J1939 are to be respected:

- destination specific communication with PDU1 (PDU format 0...239)
- broadcast communication with PDU2 (PDU format 240...255)
- proprietary communication with PDU1 or PDU2

3.7.2 Example: Detailed message documentation

7679

ETC1: Electronic Transmission Controller #1 (3.3.5)
Value = 0x0CF00203
The following details result:

Designation	Parameter	Value in the example above
transmission repetition rate	RPT	10 ms
data length	LEN	8 bytes
PDU format	PF	240
PDU specific	PS	2
default priority	PRIO	3
data page	PG	0
source address destination address	SA DA	3
parameter group number	PGN	0x00F002
identifier	ID	0x0CF00203
data field	SRC DST	Array address (Meaning of the data bytes 1...8 à manufacturer's documentation)

As in the example of the manufacturer all relevant data has already been prepared, it can be directly transferred to the function blocks.

Depending on the required function the corresponding values are set. For the fields SA / DA or SRC / DST the meaning (but not the value) changes according to the receive or transmit function.

The individual data bytes must be read from the array and processed according to their meaning.

3.7.3 Example: Short message documentation

7680

But even if the aggregate manufacturer only provides a short documentation, the FB parameters can be derived from the identifier. In addition to the ID, the "transmission repetition rate" and the meaning of the data fields are also always needed.

If the protocol messages are not manufacturer-specific, the standard SAE J1939 or ISO 11992 can also serve as information source.

Structure of the identifier 0x0CF00203:

DATA	PRIO, reserv., PG	PF	PS	SA / DA
hex	0C	F0	02	03
dec	→ following table	240	2	3

As these values are hexadecimal numbers of which individual bits are sometimes needed, the numbers must be further broken down:

DATA	PRIO, reserv., PG
hex	0C
bin	0000 1100

That is separated into:

DATA	not relevant	PRIO	reserved	PG
bin	000	011	0	0
dec	---	3	0	0

Other typical combinations for "PRIO, reserv., PG"

0x18:

DATA	PRIO, reserv., PG
hex	18
bin	0001 1000

That is separated into:

DATA	not relevant	PRIO	reserved	PG
bin	000	110	0	0
dec	---	6	0	0

0x1C:

DATA	PRIO, reserv., PG
hex	1c
bin	0001 1100

That is separated into:

DATA	not relevant	PRIO	reserved	PG
bin	000	111	0	0
dec	---	7	0	0

3.8 CAN / CANopen: errors and error handling

Contents

1171

The error mechanisms described are automatically processed by the CAN controller integrated in the controller. This cannot be influenced by the user. (Depending on the application) the programmer should react to signalled errors in the application software.

Goal of the CAN error mechanisms:

- Ensuring uniform data objects in the complete CAN network
- Permanent functionality of the network even in case of a faulty CAN participant
- Differentiation between temporary and permanent disturbance of a CAN participant
- Localisation and self-deactivation of a faulty participant in 2 steps:
  - error passive
  - disconnection from the bus (bus off)

  This gives a temporarily disturbed participant a "rest".

To give the interested user an overview of the behaviour of the CAN controller in case of an error, error handling is easily described below. After error detection the information is automatically prepared and made available to the programmer as CAN error bits in the application software.

3.8.1 CAN errors

8589

Error message

1172

If a bus participant detects an error condition, it immediately transmits an error message. The transmission is then aborted or the correct messages already received by other participants are rejected. This ensures that correct and uniform data is available to all participants. Since the error message is directly transmitted the sender can immediately start to repeat the disturbed message as opposed to other fieldbus systems (they wait until a defined acknowledgement time has elapsed). This is one of the most important features of CAN.

One of the basic problems of serial data transmission is that a permanently disturbed or faulty bus participant can block the complete system. Error handling for CAN would increase such a risk. To exclude this, a mechanism is required which detects the fault of a participant and disconnects this participant from the bus, if necessary.

Error counter

1173

A transmit and receive error counter are integrated in the CAN controller. They are counted up (incremented) for every faulty transmit or receive operation. If a transmission was correct, these counters are counted down (decremented).

However, the error counters are more incremented in case of an error than decremented in case of success. Over a defined period this can lead to a considerable increase of the counts even if the number of the undisturbed messages is greater than the number of the disturbed messages. Longer undisturbed periods slowly reduce the counts. So the counts indicate the relative frequency of disturbed messages.

If the participant itself is the first to detect errors (= self-inflicted errors), the error is more severely "punished" for this participant than for other bus participants. To do so, the counter is incremented by a higher amount.

If the count of a participant exceeds a defined value, it can be assumed that this participant is faulty. To prevent this participant from disturbing bus communication by active error messages (error active), it is switched to "error passive".

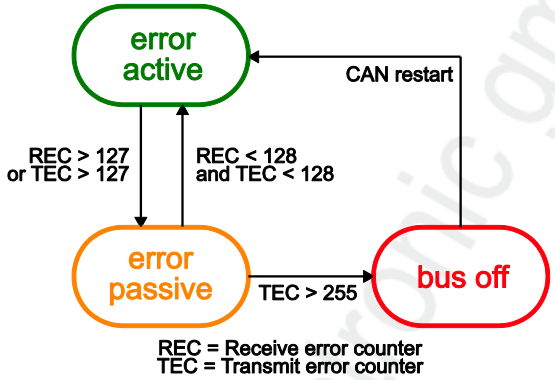


Figure: mechanism of the error counter

error active
→ **Participant, error active** (→ p. 88)
error passive
→ **participant error passive** (→ p. 88)
bus off
→ **participant bus off** (→ p. 88)
CAN restart
→ participant, bus off

Participant, error active

1174

An error active participant participates in the bus communication without restriction and is allowed to signal detected errors by transmitting the active error message. As already described the transmitted message is destroyed.

participant error passive

19173

An error passive participant can also still communicate without restriction. However, it is only allowed to identify a detected error by a passive error flag, which does not interfere with the bus communication. An error passive participant becomes error active again if it is below a defined count value.

About the reaction in the application program:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
In case of an error counter value ≥ 96: System variable CANx_WARNING = TRUE.	In case the warning threshold for the TX error counter is exceeded: in the FB CAN_STATUS, the output WARNING_TX becomes = TRUE In case the warning threshold for the RX error counter is exceeded: in the FB CAN_STATUS, the output WARNING_RX becomes = TRUE

In this state, the participant is error passive.

participant bus off

19174

If the error count value continues to be incremented, the participant is disconnected from the bus (bus off) after exceeding a maximum count value.

About the reaction in the application program:

For all controllers (except for CR04nn, CR253n) the following applies:	For all CR04nn, CR1nnn, CR253n the following applies:
The system variable CANx_BUSOFF = TRUE	In case the upper threshold for the TX error counter is exceeded: in the FB CAN_STATUS, the output BUSOFF becomes = TRUE In case the upper threshold for the RX error counter is exceeded: in the FB CAN_STATUS, the output BUSOFF becomes = TRUE
> The error CANx_BUSOFF is automatically handled and reset by the runtime system. If a precise error treatment and evaluation is to take place via the application program: ► Use the FB CANx_ERRORHANDLER!	> The error BUSOFF is treated automatically by the runtime system (recovery). Attempted reboot of the corresponding CAN interface up to 4 times every second. > If the bus-off is not repaired after the 4th attempt, the device cuts itself off from the interface and no longer participates in the bus traffic.

► Always reset the bus error indication in the application program:

► Explicitly reset the error CANx_BUSOFF in the application program!	► Set the input CLEAR = TRUE in the FB CAN_STATUS. > The error indication is reset. If these errors are no longer initially set in the next cycle: > the bus-off is repaired > the device is ERROR ACTIVE again, i.e. it participates as usual in the bus communication.

3.8.2 CANopen errors

Contents

11670

Structure of an EMCY message

Contents

19177

Under CANopen error states are indicated via a simple standardised mechanism. For a CANopen device every occurrence of an error is indicated via a special message which details the error.

If an error or its cause disappears after a certain time, this event is also indicated via the EMCY message. The errors occurred last are stored in the object directory (object 0x1003) and can be read via an SDO access (→ **CANx_SDO_READ**). In addition, the current error situation is reflected in the error register (object 0x1001).

Read the errors via SDO access:

- **For all controllers (except for CR04nn, CR253n) the following applies:**

CANx_SDO_READ	CAN interface x: Reads the SDO with the indicated indices from the node x = 1...n = number of the CAN interface (depending on the device, → data sheet)

- **For all CR04nn, CR1nnn, CR253n the following applies:**

CANOPEN_SDOREAD	= CANopen read SDO Reads an "Expedited SDO" = Expedited Service Data Object
CANOPEN_SDOREADBLOCK	= CANopen read SDO block Reads the indicated entry in the object directory of a node in the network via SDO block transfer
CANOPEN_SDOREADMULTI	= CANopen read SDO multi Reads the indicated entry in the object directory of a node in the network

A distinction is made between the following errors:

8046

Communication error (examples:)

- The CAN controller signals CAN errors.
  (The frequent occurrence is an indication of physical problems. These errors can considerably affect the transmission behaviour and thus the data rate of a network.)
- Life guarding or heartbeat error
- SYNC error (slave only)

Application error (examples:)

- Short circuit or wire break
- Temperature too high

Emergency messages

9973

Device errors in the slave or problems in the CAN bus trigger emergency messages:

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x80 + node ID		error code		object 0x1001	device-specific				

Please note the reversed byte order! (⇨ Little Endian or Intel format)

Identifier

8048

The identifier for the error message consists of the sum of the following elements:
EMCY default identifier 128 (0x80)
+
node ID

EMCY error code

8049

It gives detailed information which error occurred. A list of possible error codes has already been defined in the communication profile. Error codes which only apply to a certain device class are defined in the corresponding device profile of this device class.

Overview CANopen error codes

545

Error Code (hex)	Meaning
00xx	Reset or no error
10xx	Generic error
20xx	Current
21xx	Current, device input side
22xx	Current inside the device
23xx	Current, device output side
30xx	Voltage
31xx	Mains voltage
32xx	Voltage inside the device
33xx	Output voltage
40xx	Temperature
41xx	Ambient temperature
42xx	Device temperature
50xx	Device hardware
60xx	Device software
61xx	Internal software
62xx	User software
63xx	Data set
70xx	Additional modules
80xx	Monitoring
81xx	Communication
8110	CAN overrun-objects lost
8120	CAN in error passiv mode
8130	Life guard error or heartbeat error
8140	Recovered from bus off
8150	Transmit COB-ID collision
82xx	Protocol error
8210	PDO not processed due to length error
8220	PDO length exceeded
90xx	External error
F0xx	Additional functions
FFxx	Device specific

Object 0x1001 (error register)

8547

This object reflects the general error state of a CANopen device. The device is to be considered as error free if the object 0x1001 signals no error any more.

Bit	Meaning (Bedeutung)
0	generic error
1	current
2	voltage
3	temperature
4	communication error
5	device profile specific
6	reserved – always 0
7	manufacturer specific

For an error message more than one bit in the error register can be set at the same time.

Example: CR2033, message "wire break" at channel 2 (→ installation manual of the device):

COB-ID	DLC	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x80 + node ID		00	FF	81	10	00	00	00	00

Error code = 0xFF00

Error register = 0x81 = 0b1000 0001, thus it consists of the following errors:
- generic error
- manufacturer specific

Concerned channel = 0x0010 = 0b0000 0000 0001 0000 = wire break channel 2

Object 0x1003 (error field)

8050

The object 0x1003 represents the error memory of a device. The sub-indices contain the errors occurred last which triggered an error message.

If a new error occurs, its EMCY error code is always stored in the sub-index 0x01. All other older errors are moved back one position in the error memory, i.e. the sub-index is incremented by 1. If all supported sub-indices are used, the oldest error is deleted. The sub-index 0x00 is increased to the number of the stored errors. After all errors have been rectified the value "0" is written to the error field of the sub-index 0x01.

To delete the error memory the value "0" can be written to the sub-index 0x00. Other values must not be entered.

Signalling of device errors

19178

As described, EMCY messages are transmitted if errors occur in a device. In contrast to programmable devices error messages are automatically transmitted by decentralised input/output modules (e.g. CompactModules CR2033).
Corresponding error codes → corresponding device manual.

Programmable devices only generate an EMCY message automatically (e.g. for "short circuit on output Q07" if one of the following FBs is integrated in the application program:

- **For all controllers (except for CR04nn, CR253n) the following applies:**

CANx_MASTER_EMCY_HANDLER	Handles the device-specific error status of the CANopen master on CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_SLAVE_EMCY_HANDLER	Handles the device-specific error status of the CANopen slave on CAN interface x: • error register (index 0x1001) and • error field (index 0x1003) of the CANopen object directory x = 1...n = number of the CAN interface (depending on the device, → data sheet)

- **For all CR04nn, CR1nnn, CR253n the following applies:**

CANOPEN_GETERRORREGISTER	= Get CANopen error register Reads the error registers 0x1001 and 0x1003 from the controller The registers can be reset by setting the respective inputs.
CANOPEN_GETEMCYMESSAGES	= Get CANopen emergency messages Lists all emergency messages that have been received by the controller from other nodes in the network since the last deletion of messages The list can be reset by setting the according input.

Overview of the automatically transmitted EMCY error codes for all **ecomat*mobile*** devices programmable with CODESYS → chapter **Overview CANopen error codes** (→ p. 91).

If in addition application-specific errors are to be transmitted by the application program, one of the following FBs is used:

- **For all controllers (except for CR04nn, CR253n) the following applies:**

CANx_MASTER_SEND_EMERGENCY	Sends application-specific error status of the CANopen master on CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)
CANx_SLAVE_SEND_EMERGENCY	Sends application-specific error status of the CANopen slave on CAN interface x x = 1...n = number of the CAN interface (depending on the device, → data sheet)

- **For all CR04nn, CR1nnn, CR253n the following applies:**

CANOPEN_SENDEMCYMESSAGE	= CANopen send emergency message Sends an EMCY message. The message is assembled from the according parameters and entered in register 0x1003

Manufacturer specific information

8548

A device manufacturer can indicate additional error information. The format can be freely selected.

Example:

In a device two errors occur and are signalled via the bus:

- Short circuit of the outputs:
  Error code 0x2308,
  the value 0x03 (0b0000 0011) is entered in the object 0x1001
  (generic error and current error)
- CAN overrun:
  Error code 0x8110,
  the value 0x13 (0b0001 0011) is entered in the object 0x1001
  (generic error, current error and communication error)

>> CAN overrun processed:
Error code 0x0000,
the value 0x03 (0b0000 0011) is entered in the object 0x1001
(generic error, current error, communication error reset)

It can be seen only from this information that the communication error is no longer present.

Overview of CANopen EMCY codes (standard page)

18071

- **For all controllers (except for CR04nn, CR253n) the following applies:**
  The following EMCY messages are sent automatically when the FB CANx_MASTER_EMCY_HANDLER is called cyclically.
- **For all CR04nn, CR1nnn, CR253n the following applies:**
  In the CANopen stack, none of these EMCY codes has a fixed implementation yet. Suggestion:
  ▶ Generate these EMCY codes with the FB CANOPEN_SENDEMCYMESSAGE.

EMCY code object 0x1003		Object 0x1001	Manufactor specific information					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Description
00	21	03						wire break, inputs
08	21	03						short circuit, inputs
10	21	03						overcurrent 0…20 mA
00	23	03						wire break, outputs
08	23	03						short circuit, outputs
10	23	03						overload, outputs
00	31	05						supply voltage VBBS
00	33	05						output voltage VBBO
08	33	05						output voltage VBBR
00	42	09						Excess temperature

The entries for bytes 3...7 depend on the concrete distribution of inputs and outputs of the device (→ Programming manual).

Overview of CANopen EMCY codes (extended page)

18072

- **For all controllers (except for CR04nn, CR253n) the following applies:**
  The following EMCY messages are sent automatically when the FB CANx_MASTER_EMCY_HANDLER is called cyclically.
- **For all CR04nn, CR1nnn, CR253n the following applies:**
  In the CANopen stack, none of these EMCY codes has a fixed implementation yet. Suggestion:
  ▶ Generate these EMCY codes with the FB CANOPEN_SENDEMCYMESSAGE.

EMCY code object 0x1003		Object 0x1001	Manufactor specific information					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Description
01	21	03						wire break, inputs
09	21	03						short circuit, inputs
11	21	03						overcurrent 0…20 mA
01	23	03						wire break, outputs
09	23	03						short circuit, outputs
10	23	03						overload, outputs
10	33	05						output voltage VBB1
11	33	05						output voltage VBB2
12	33	05						output voltage VBB3
13	33	05						output voltage VBB4
18	33	05						relay supply

The entries for bytes 3...7 depend on the concrete distribution of inputs and outputs of the device (→ Programming manual).

Overview of CANopen EMCY codes (CANx)

18073

- **For all controllers (except for CR04nn, CR253n) the following applies:**
  The following EMCY messages are sent automatically when the FB CANx_MASTER_EMCY_HANDLER is called cyclically.
- **For all CR04nn, CR1nnn, CR253n the following applies:**
  In the CANopen stack, none of these EMCY codes has a fixed implementation yet. Suggestion:
  ▶ Generate these EMCY codes with the FB CANOPEN_SENDEMCYMESSAGE.

13094

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

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

4 Control outputs – description

Contents

13741

4.1 PWM functions – description

Contents

13831

In this chapter you will find out more about the pulse width modulation in the **ecomat*mobile*** device.

Availability of PWM or PWMi:
→ Data sheet of the device
→ Device manual of the device

4.1.1 PWM signal processing – description

Contents

13832
6889

The abbreviation PWM stands for **p**ulse **w**idth **m**odulation. It is mainly used to trigger proportional valves (PWM valves) for mobile and robust controller applications. Also, with an additional component (accessory) for a PWM output the pulse-width modulated output signal can be converted into an analogue output voltage.

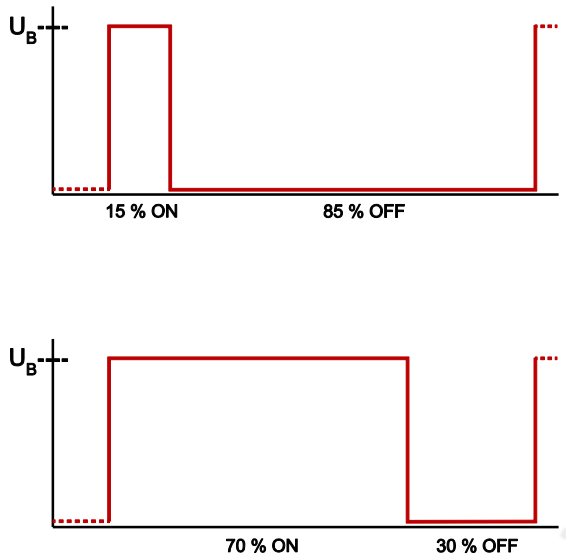


Figure: PWM principle

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 then varied. Depending on the mark-to-space ratio, the connected load determines the corresponding RMS current.

The PWM function of the **ecomat*mobile*** controller is a hardware function provided by the processor. To use the integrated PWM outputs of the controller, they must be initialised in the application program and parameterised corresponding to the requested output signal.

PWM: What does a PWM output do?

1563

PWM stands for "**p**ulse **w**idth **m**odulation" which means the following principle:

In general, digital outputs provide a fixed output voltage as soon as they are switched on. The value of the output voltage can*not* be changed here. The PWM outputs, however, split the voltage into a quick sequence of many square-wave pulse trains. The pulse duration [switched on] / pulse duration [switched off] ratio determines the effective value of the requested output voltage. This is referred to as the switch-on time in [%].

In the following sketches the current profiles are shown as a stylised straight line. In reality the current flows to an e-function.

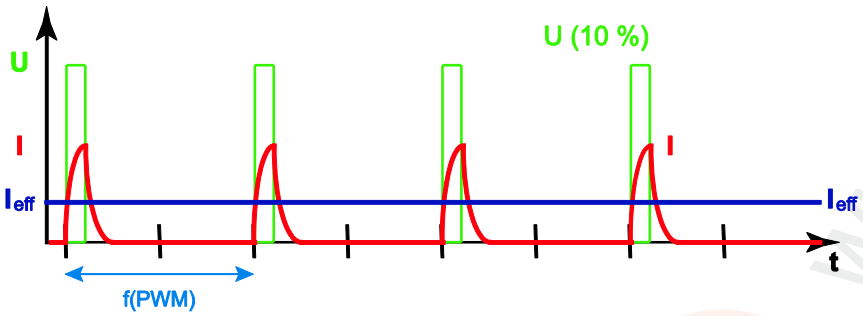


Figure: The profile of the PWM voltage U and the coil current I at 10 % switch-on time: The effective coil current I_{eff} is also 10 %

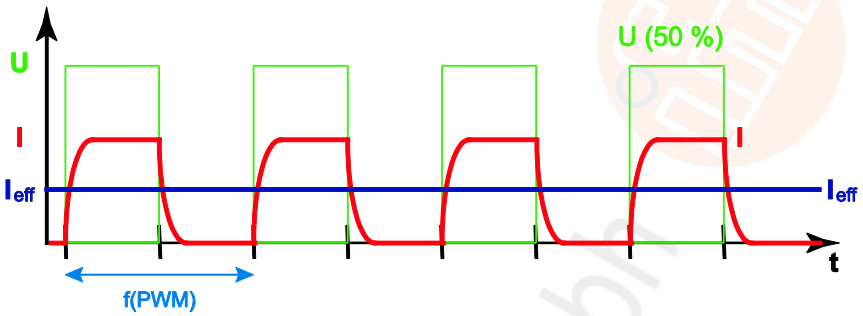


Figure: The profile of the PWM voltage U and the coil current I at 50 % switch-on time: The effective coil current I_{eff} is also 50 %

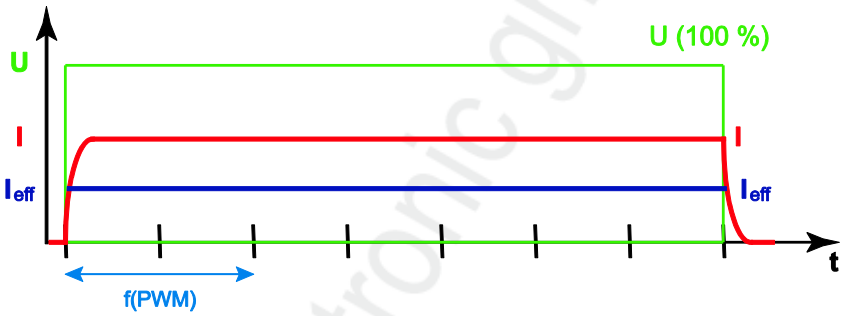


Figure: The profile of the PWM voltage U and the coil current I at 100 % switch-on time: The effective coil current I_{eff} is also 100 %

PWM: What is the dither?

1564

If a proportional hydraulic valve is controlled, its piston does not move right away and at first not proportional to the coil current. Due to this "slip stick effect" – a kind of "break-away torque" – the valve needs a slightly higher current at first to generate the power it needs to move the piston from its off position. The same also happens for each other change in the position of the valve piston. This effect is reflected in a jerking movement of the valve piston, especially at very low manipulating speeds.

Technology solves this problem by having the valve piston move slightly back and forth (dither). The piston is continuously vibrating and cannot "stick". Also a small change in position is now performed without any delay, a "flying splice" so to speak.

Advantage: The hydraulic cylinder controlled in that way can be moved more sensitively.

Disadvantage: The valve becomes measurably hotter with dither than without because the valve coil is now working continuously.

That means that the "golden means" has to be found.

When is a dither useful?

1565

When the PWM output provides a pulse frequency that is small enough (standard value: up to 250 Hz) so that the valve piston continuously moves at a minimum stroke, an additional dither is not required (→ next figure):

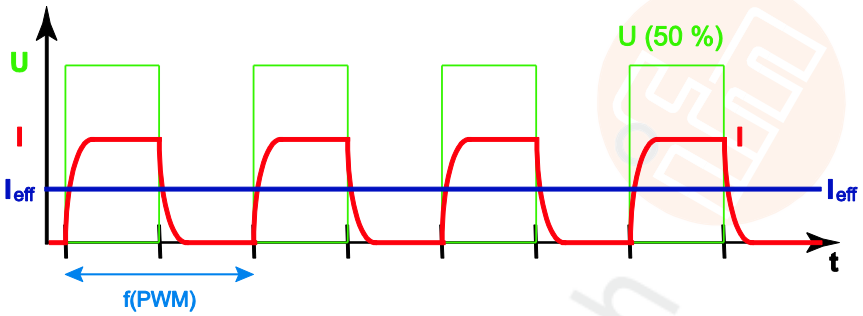


Figure: Balanced PWM signal; no dither required.

At a higher PWM frequency (standard value 250 Hz up to 1 kHz) the remaining movement of the valve piston is so short or so slow that this effectively results in a standstill so that the valve piston can again get stuck in its current position (and will do so!) (→ next figures):

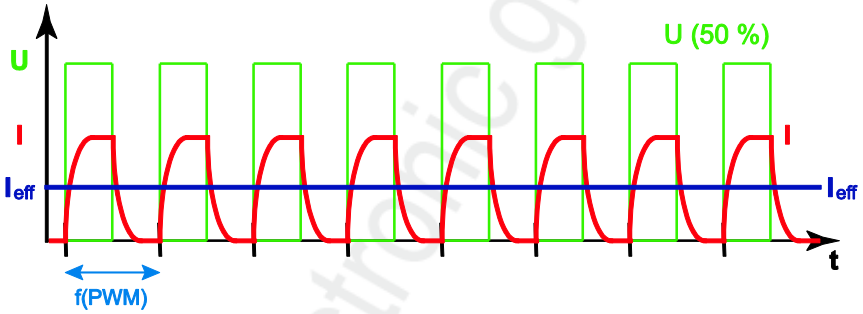


Figure: A high frequency of the PWM signal results in an almost direct current in the coil. The valve piston does not move enough any longer. With each signal change the valve piston has to overcome the break-away torque again.

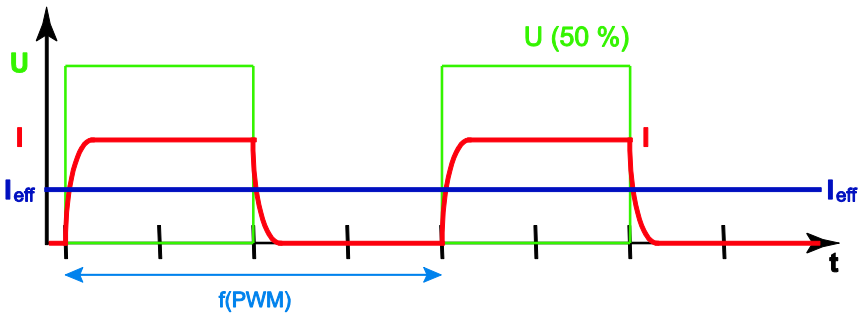


Figure: Too low frequencies of the PWM signal only allow rare, jerking movements of the valve piston. Each pulse moves the valve piston again from its off position; every time the valve piston has to overcome the break-away torque again.

> In case of a PWM switch-on time under 10 % and over 90 %, it is adequate and necessary to superimpose the PWM signal with a dither signal.

Dither frequency and amplitude

1566

The mark/space ratio (the switch-on time) of the PWM output signal is switched with the dither frequency. The dither amplitude determines the difference of the switch-on times in the two dither half-waves.

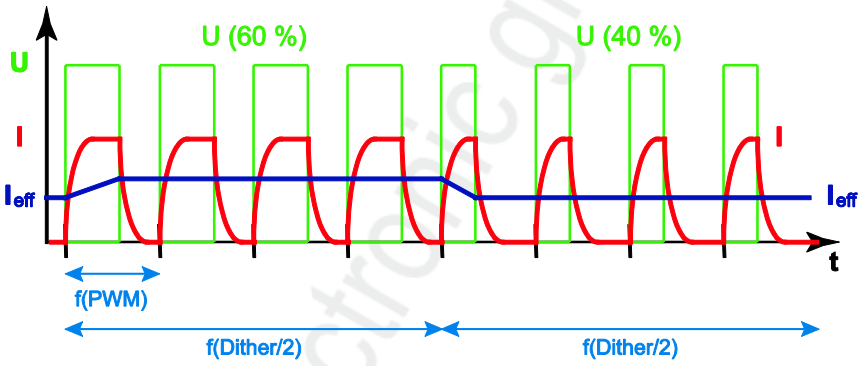
> The dither frequency must be an integer part of the PWM frequency. Otherwise the hydraulic system would not work evenly but it would oscillate.

Example Dither

1567

The dither frequency is 1/8 of the PWM frequency.
The dither amplitude is 10 %.

With the switch-on time of 50 % in the figure, the actual switch-on time for 4 pulses is 60 % and for the next 4 pulses it is 40 % which means an average of 50 % switch-on time. The resulting effective coil current will be 50 % of the maximum coil current.



The result is that the valve piston always oscillates around its off position to be ready to take a new position with the next signal change without having to overcome the break-away torque before.

PWM: Function blocks

19183

You can access PWM functions for the PWM-compatible outputs with the following function blocks:

Some of the function blocks listed here are only available for individual devices.

- **For all controllers (except for CR04nn, CR253n) the following applies:**

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

- **For all CR04nn, CR253n the following applies:**

CURRENT_CONTROL	Current controller for a PWMi output channel
H_BRIDGE	H-bridge on a PWM channel pair
PWM1000	Initialises and configures a PWM-capable output channel the mark-to-space ratio can be indicated in steps of 1 ‰
PWM1000_LOW	Initialises and configures a PWM-capable output channel minus switched the mark-to-space ratio can be indicated in steps of 1 ‰

PWM: Description of the parameters

13833

The FB PWM... contains a set of parameters. Here, some of them are explained in detail.

PWM frequency

2304

Depending on the valve type, a corresponding PWM frequency is required. The PWM frequency is directly transferred as numerical value in [Hz] for PWM1000.

All PWM channels behave in the same way. Every PWM channel can be set to its own frequency separately. The PWM frequency is in the range 20...250 Hz.

PWM dither

2306

For certain hydraulic valve types a so-called dither frequency must additionally be superimposed on the PWM frequency. If these valves were triggered over a longer period by a constant PWM value, they could block due to the high system temperatures.

To prevent this, the PWM value is increased or reduced on the basis of the dither frequency by a defined value (DITHER_VALUE). As a consequence a vibration with the dither frequency and the amplitude DITHER_VALUE is superimposed on the constant PWM value. The dither frequency must be an integer part of the PWM frequency.

4.1.2 Hydraulic control with PWMi

Contents

19185
1559

ifm electronic offers the user special functions to control hydraulic systems as a special field of current regulation with PWM (= PWMi).

A hydraulics library is only available for controllers, but not for CR04nn, CR253n.

The purpose of this library? – An introduction

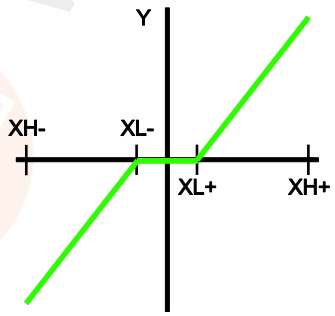
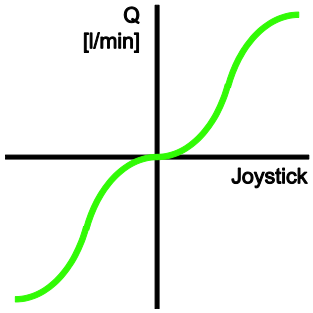
1560

Thanks to the FBs of this library you can fulfil the following tasks:

Standardise the output signals of a joystick

1561

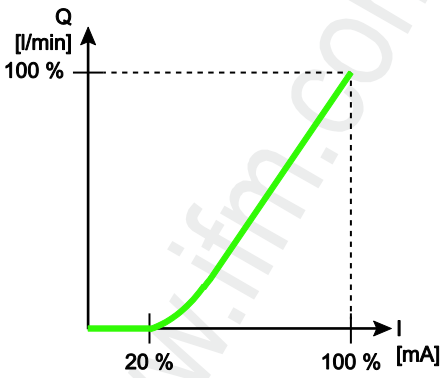
It is not always intended that the whole movement area of the joy stick influences the movement of the machine.

Often the area around the neutral position of the joy stick is to be spared because the joy stick does not reliably supply 0 V in this neutral position. Here in this figure the area between XL- and XL+ is to be spared.	
The FBs of this library enable you to adapt the characteristic curve of your joy stick according to your requirements – on request even freely configurable:	

Control hydraulic valves with current-controlled outputs

1562

As a rule hydraulic valves do not have a completely linear characteristic:

Typical characteristic curve of a hydraulic valve: The oil flow starts at approx. 20 % of the coil current. The initial oil flow is not linear. This has to be taken into account for the calculation of the preset values for the coil current. The FBs of this library support you here.	

4.2 Controller – description

13830
1623

Controlling is a process during which the unit to be controlled (control variable x) is continuously detected and compared with the reference variable w. Depending on the result of this comparison, the control variable is influenced for adaptation to the reference variable.

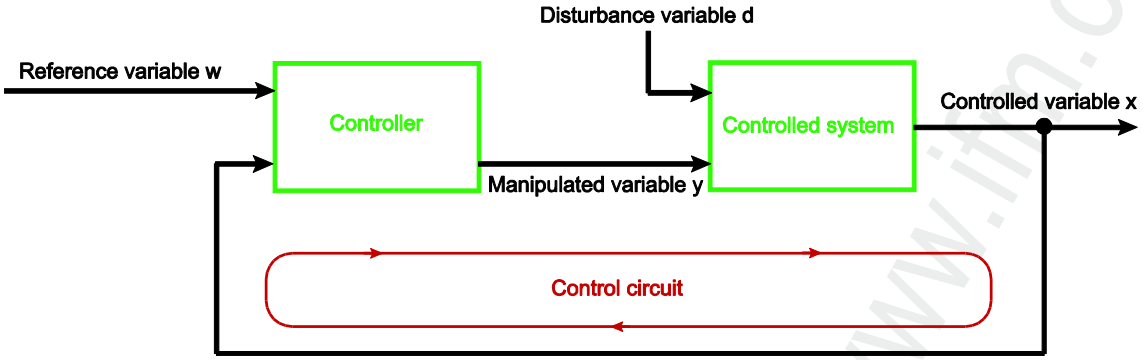


Figure: Principle of controlling

The selection of a suitable control device and its optimum setting require exact indication of the steady-state behaviour and the dynamic behaviour of the controlled system. In most cases these characteristic values can only be determined by experiments and can hardly be influenced.

Three types of controlled systems can be distinguished:

4.2.1 Self-regulating process

1624

For a self-regulating process the control variable x goes towards a new final value after a certain manipulated variable (steady state). The decisive factor for these controlled systems is the amplification (steady-state transfer factor KS). The smaller the amplification, the better the system can be controlled. These controlled systems are referred to as P systems (P = proportional).

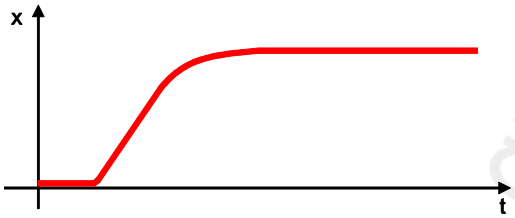


Figure: P controller = self-regulating process

4.2.2 Controlled system without inherent regulation

1625

Controlled systems with an amplifying factor towards infinity are referred to as controlled systems without inherent regulation. This is usually due to an integrating performance. The consequence is that the control variable increases constantly after the manipulated variable has been changed or by the influence of an interfering factor. Due to this behaviour it never reaches a final value. These controlled systems are referred to as I systems (I = integral).

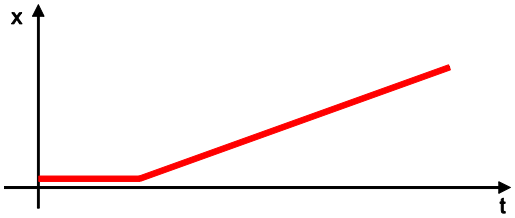


Figure: I controller = controlled system without inherent regulation

4.2.3 Controlled system with delay

1626

Most controlled systems correspond to series systems of P systems (systems with compensation) and one or several T1 systems (systems with inertia). A controlled system of the 1st order is for example made up of the series connection of a throttle point and a subsequent memory.

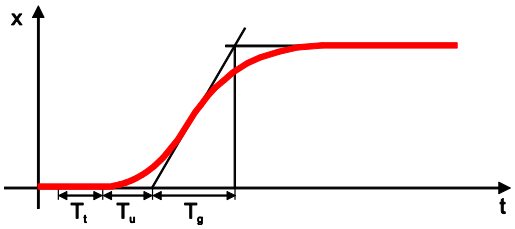


Figure: PT system = controlled system with delay

For controlled systems with dead time the control variable does not react to a change of the control variable before the dead time T_t has elapsed. The dead time T_t or the sum of $T_t + T_u$ relates to the controllability of the system. The controllability of a system is the better, the greater the ratio T_g/T_u.

The controllers which are integrated in the library are a summary of the preceding basic functions. It depends on the respective controlled system which functions are used and how they are combined.

5 Working with the user flash memory

Contents

11607

Some ifm devices feature a user flash memory. This is a flash memory area which is intended for the customer's application data.

Application examples:
- Message texts (several languages can be selected) for display in the PDM and on the display
- Load limit value tables e.g. for lifts, cranes and turntable ladders

The programmer creates lists or tables.
The program used for this must be able to convert the source file into a CSV file.
Suitable are e.g. spreadsheet programs such as Microsoft Excel or OpenOffice Calc.

7317

NOTE

CSV files must not contain any safety-related data.
No suitable backup measures are provided.

5.1 Flash memory – what is that?

11608

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.

5.2 What is a CSV file?

11627

CSV = **C**omma **S**eparated **V**alues (also: **C**haracter **S**eparated **V**alues)
A CSV file is a text file for storing or exchanging simply structured data.
The file extension is `.csv`.

Example: Source table with numerical values:

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

This results in the following CSV file:

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

Please note:

- The downloader and the maintenance tool expect a separator between the columns of the source table, e.g. a semicolon (;).
- CODESYS expects a null byte (NUL) as a terminator of a string.
- Each data set (each table row to be transmitted) should have the same number of table columns.

5.3 CSV file and the ifm maintenance tool

Contents

11619

The following devices can communicate with the ifm maintenance tool:
via AddIn BasicSystem
- BasicController: CR040n, CR041n, CR043n
- BasicDisplay: CR045n
- SmartController: CR253n

via Addin R360System
- Controller: CR0n3n, CR7n3n
- Controller: CR0020, CR0200, CR0505
- CabinetController: CR0303
- SmartController: CR2500

5.3.1 Requirements for the CSV file

11630

► The CSV file must have a specific header structure.
All header data begin with '#'.
  - 1st line: CSV file type
    e.g.: `#File Type=0`
    allowed: 0/1
  - 2nd line (option): Project name of the CSV file
    e.g.: `#Name=Demo Textmessages`
    allowed: 0...20 characters
  - 3rd line (option): Version of the CSV file
    e.g.: `#Version=V01.00.00`
    allowed: 0...12 characters

- The ifm maintenance tool knows the start address of the user flash memory.
The address does not have to be indicated in the CSV file.

► Data must follow directly after the header data lines without any gaps!
Structure:
relative address;date or text;data type;{comment}
Example:
31;excess temperature;string(20);text 02
A semicolon (;) MUST follow after the data type!

- The ifm maintenance tool itself generates the correct data length from the data type.
Therefore, the string data does not have to be indicated in the CSV file in full length.

► The semicolon (;) is used as field delimiter.

5.3.2 Creation of a CSV file using a spreadsheet program

11639

Example:

- bilingual message texts for BasicDisplay CR0451
- 20 texts with up to 30 characters each in German
- 20 texts with up to 30 characters each in English

DemoFlashTestMessageD_E.xls

	A	B	C	D
1	#FileType=0			
2	#Name=Messages D/E CR0451			
3	#Version=V01.00.00			
4	0	Dies ist Text 1	String(30)	Text 01
5	31	Dies ist Text 2	String(30)	Text 02
6	62	Dies ist Text 3	String(30)	Text 03
7	93	Dies ist Text 4	String(30)	Text 04
8	124	Dies ist Text 5	String(30)	Text 05
9	155	Dies ist Text 6	String(30)	Text 06
10	186	Dies ist Text 7	String(30)	Text 07
11	217	Dies ist Text 8	String(30)	Text 08
12	248	Dies ist Text 9	String(30)	Text 09
13	279	Dies ist Text 10	String(30)	Text 10
14	310	Dies ist Text 11	String(30)	Text 11
15	341	Dies ist Text 12	String(30)	Text 12
16	372	Dies ist Text 13	String(30)	Text 13
17	403	Dies ist Text 14	String(30)	Text 14
18	434	Dies ist Text 15	String(30)	Text 15
19	465	Reserve	String(30)	Text 16
20	496	Reserve	String(30)	Text 17
21	527	Reserve	String(30)	Text 18
22	558	Reserve	String(30)	Text 19
23	589	Reserve	String(30)	Text 20
24	620	This is text 1	String(30)	Text 21
25	651	This is text 2	String(30)	Text 22
26	682	This is text 3	String(30)	Text 23
27	713	This is text 4	String(30)	Text 24
28	744	This is text 5	String(30)	Text 25
29	775	This is text 6	String(30)	Text 26
30	806	This is text 7	String(30)	Text 27
31	837	This is text 8	String(30)	Text 28
32	868	This is text 9	String(30)	Text 29
33	899	This is text 10	String(30)	Text 30
34	930	This is text 11	String(30)	Text 31
35	961	This is text 12	String(30)	Text 32
36	992	This is text 13	String(30)	Text 33
37	1023	This is text 14	String(30)	Text 34
38	1054	This is text 15	String(30)	Text 35
39	1085	reserve	String(30)	Text 36
40	1116	reserve	String(30)	Text 37
41	1147	reserve	String(30)	Text 38
42	1178	reserve	String(30)	Text 39
43	1209	reserve	String(30)	Text 40

Legend:

Field no.	Description
A1...A3	Header; entries begin with '#'
A1	#FileType= 0: When compiling a CSV file, the set parameters are directly stored in the user flash in the given order. 1: When compiling a CSV file, the set parameters are stored in the user flash in such a way that the data can be directly read using the CoDeSys structure.
A2	#Name= Name for the definition of the table and for finding the table in the application program Length = 0...20 characters
A3	#Version= Version of the table (e.g. for different vehicles) Length = 0...12 characters
A4...A43	Byte number for the beginning of a message text
A4	First text of the first language (byte number = 0)
A24	Here: first text of the second language (byte number = 620 = offset)

Field no.	Description
B4...B23 B24...B43	Message texts (first language) The same message texts (second language) The ifm maintenance tool only transfers these texts into the device
C4...C43	Data type, here: string(30)
D4...D43	Comments (option) • only for information when creating the table • comments are not transferred into the device

► This structure is necessary for the generated CSV file to be understood by CoDeSys.

► Save the spreadsheet:
Select a memory location and enter a file name.
Choose a sensible file name so that you can identify the right file later on.

► Convert the spreadsheet into a CSV file.
Select the semicolon ';' as column delimiter.

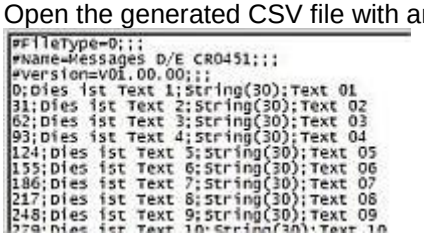
For **Excel**: [Save As...] > [Save as type:] = CSV file

For **OpenOffice**: [Save As...] > [Save as type] = Text CSV > [Keep Current Format]
In the window [Export of text files] set:
• semicolon as field delimiter
• text delimiter = (empty)

► Acknowledge the warning (regarding loss of formatting).

► Close the spreadsheet program.

► Open the generated CSV file with an editor:



```
#FileType=D;;;
#Name=Messages D/E CR0451;;;
#version=V01.00.00;;;
0;Dies ist Text 1;String(30);Text 01
31;Dies ist Text 2;String(30);Text 02
62;Dies ist Text 3;String(30);Text 03
93;Dies ist Text 4;String(30);Text 04
124;Dies ist Text 5;String(30);Text 05
155;Dies ist Text 6;String(30);Text 06
186;Dies ist Text 7;String(30);Text 07
217;Dies ist Text 8;String(30);Text 08
248;Dies ist Text 9;String(30);Text 09
```

► Remove all semicolons behind the header lines (starting with'#').

► Close the editor and save the file.

5.3.3 Creation of a CSV file using an editor

11640

► Enter the requested contents of the CSV file manually.

► Save the file as CSV file.
File type = ANSI

Example:

- bilingual message texts for BasicDisplay CR0451
- 20 texts with up to 30 characters each in German
- 20 texts with up to 30 characters each in English

```
#FileType=0
#Name=Messages D/E CR0451
#version=V01.00.00
0;Dies ist Text 1;String(30);
31;Dies ist Text 2;String(30);
62;Dies ist Text 3;String(30);
93;Dies ist Text 4;String(30);
124;Dies ist Text 5;String(30);
155;Dies ist Text 6;String(30);
186;Dies ist Text 7;String(30);
217;Dies ist Text 8;String(30);
248;Dies ist Text 9;String(30);
279;Dies ist Text 10;String(30);
310;Dies ist Text 11;String(30);
341;Dies ist Text 12;String(30);
372;Dies ist Text 13;String(30);
403;Dies ist Text 14;String(30);
434;Dies ist Text 15;String(30);
465;Reserve;String(30);
496;Reserve;String(30);
527;Reserve;String(30);
558;Reserve;String(30);
589;Reserve;String(30);
620;This is text 1;String(30);
651;This is text 2;String(30);
682;This is text 3;String(30);
713;This is text 4;String(30);
744;This is text 5;String(30);
775;This is text 6;String(30);
806;This is text 7;String(30);
837;This is text 8;String(30);
868;This is text 9;String(30);
899;This is text 10;String(30);
930;This is text 11;String(30);
961;This is text 12;String(30);
992;This is text 13;String(30);
1023;This is text 14;String(30);
1054;This is text 15;String(30);
1085;reserve;String(30);
1116;reserve;String(30);
1147;reserve;String(30);
1178;reserve;String(30);
1209;reserve;String(30);
```

Legend:

Field no.	Description
Lines 1...3	Header; entries begin with '#'
Line 1	#FileType= 0: When compiling a CSV file, the set parameters are directly stored in the user flash in the given order. 1: When compiling a CSV file, the set parameters are stored in the user flash in such a way that the data can be directly read using the CoDeSys structure.
Line 2 (optional)	#Name= Name for the definition of the table and for finding the table in the application program Length = 0...20 characters
Line 3 (optional)	#Version= Version of the table (e.g. for different vehicles) Length = 0...12 characters
0; 31; 62; ...	Byte number for the beginning of a message text • start with the relative address 0 • the following addresses in steps of (line length plus 1) bytes
;text;	Message texts (or reserve) • each line is precisely 20 characters long (fill with any characters, here: dots) • the data type results automatically from the line length • only these data are later transferred into the device

Field no.	Description
String(30)	Data type A semicolon (;) MUST follow after the data type! The ifm maintenance tool automatically generates the correct length of the message texts based on this information.

5.3.4 Transfer of a CSV file with the maintenance tool

19189

► Connect the programming interface of the ifm device to the PC.
► If not yet done, transfer the application program (as boot project) into the ifm device using CODESYS.
► Start the ifm maintenance tool.

E.g. for the BasicSystem:

► If not yet done, open the following menu:
[ecomat mobile] > [CAN] > [Basic System]
► Select the following menu in the maintenance tool in the left column of the user interface:
[ecomat mobile] > [CAN] > [Basic System]
► Select the following menu from the column in the middle of the user interface:
[Basic System] > [System information] > [Identity]
> After clicking on [Read] the device information appears in the right-hand section of the user interface.

If the data of the correct device appears:

► Select the following menu from the column in the middle of the user interface:
[Basic System] > [Software] > [Load]
► Click on [to Basic System] in the right-hand section of the user interface
► Click on [Import *.csv file...] in the field [Load software] .
> The window [Load software] appears.
► Select the memory location and file and confirm with [Open].
> An information window with the following information appears:
- memory location, path
- memory: n bytes of m bytes used
- file type
- name (from the CSV header data, can be edited)
- version (from the CSV header data, can be edited)

► Import the file into the list of files to be transferred with [Import].
► Mark the files to be transferred (or: all).
► Transfer the CSV file into the ifm device with [Load].
> A progress bar indicates how the process is progressing.
> Then a finished message appears.
► Reboot the device.

5.3.5 Access to the flash data: Function blocks

19190

The application program can only access the data with the following function blocks:

- **For all controllers (except for CR04nn, CR253n) the following applies:**

FLASHREAD	transfers different data types directly from the flash memory to the RAM
FLASHWRITE	writes different data types directly into the flash memory

- **For all CR04nn, CR1nnn, CR253n the following applies:**

FLASH_INFO	Reads the information from the user flash memory: • name of the memory area (user defined), • software version, • start address (for simple reading with IEC structure)
FLASH_READ	transfers different data types directly from the flash memory to the RAM

6 Visualisations in the device

Contents

3111

In this chapter you find important information about bitmap graphics in CODESYS visualisations.

6.1 General

10465
10464

In addition to the graphical elements created with the CODESYS visualisation editor, you can also integrate graphics created with other programs. Such graphics files can, for example, be pictograms, logos or smaller images. But before you integrate such an "external graphics" some basics have to be taken into account which will be explained in the following chapters.

More information is given here:

- Creation and parameter setting of visualisations:
  → CODESYS programming manual (→ **ecomat*mobile*** DVD "Software, tools and documentation")
  → **ifm** manual "PDM – Handbuch zur Einführung"
- See the **Limitations and programming notes**!

NOTE

► Immediately save the new project or the project created from a template under a project name on the PC under CODESYS!

If a project without a filename is loaded to the device, no visualisation is shown. The device has no filename and therefore the visualisation cannot start.

6.2 Recommendations for user interfaces

Contents

7435

User-friendliness is a decisive criterion for the acceptance and use of technical products!

In this chapter we will give some recommendations how the user interface (also called **H**uman-**M**achine-**I**nterface HMI) of a machine can be designed as user-friendly as possible.

6.2.1 Recommendations for a user-friendly product design

7436

All important interfaces between humans and machines are determined by the user platform and design. Important criteria for the design of interfaces between humans and machines are...

- Clear condition:
  - For each function a clear description.
  - Design according to expectations, learned contents remain the same
- Readability:
  - Take the environment (illumination, read distance) into account.
- Intuitive handling:
  - Operating element / function must be obvious.
  - User interface must be self-explanatory.
- Sensuality
  - Operating elements must be user-friendly.
  - Clear differentiation from other displays and operating elements.
- Feedback
  - Quick reaction to user activities.
  - Cause for a message must be clearly obvious.
- Environment of the product because of distraction or irritation by...
  - noise
  - darkness
  - light reflection
  - vibrations
  - extreme temperatures

From the manufacturer's view the following features are also important:

- Display as a brand-specific feature.
- Display must meet standards.

6.2.2 Do you know the future users?

7444

The future users of the product should be known:

- Age
- Gender
- Senses:
  - Eyesight
  - Hearing ability
  - Preferred hand (right or left hander)
  - Tactile ability
- Training and education:
  - General education level
  - Specific training seminars and experience
- Motivation and cognitive abilities:
  - Perception (sense organs): Not all available information is used but massively filtered, integrated and changed in many ways before it comes into awareness.
  - Thinking: The working memory where intellectual manipulation of information takes place has a very small capacity.
  - Learning: The information stored in the long-term memory is often changed in advance (e.g. due to expectations) and subsequently (e.g. by subsequent information).
  - Remembering: The information which is "actually" present in the long-term memory is often not retrievable.
  - Motivation and concentration: fatigue, weariness, distractibility etc. can affect the cognitive capability.
- Familiarity with the problem or application area:
  - Be able to recognise dangers
  - Know what is to happen after an action
- Intensity of the application (how often and how intensely is the product used)
- Culture, e.g.:
  - Language
  - Meaning of colours and symbols
  - Reading direction

6.2.3 Check suitability for use

7422

In many cases a test set-up with potential users can provide important results where and how the product is/has to be improved to be successful in the market.

For this "usability test" the following steps must be carried out:

- Determine the user group (target group):
  - Who is to handle the product?
- Prepare an interview guideline:
  - What method do I use to interview what user (operator, fitter, maintenance personnel)?
  - What do I want to achieve with the interviews? (Improvement potentials)
- Conduct and evaluate interviews.
- Create context scenarios:
  - Create an evaluable test environment.
  - Identify critical user scenarios.
- Carry out usability test:
  - How do the test persons cope with the product in the test set-up?
  - Where is what corrective action needed for the product?
- After the product has been optimised repeat the tests, if necessary.

6.2.4 Language as an obstacle

7454

In order to produce equipment which satisfies end users worldwide, language must be taken into account. The operator is not able to effectively carry out his tasks if he cannot understand the instructions on the screen. Manufacturers are still trying to solve this problem considering the many different languages in the world. A few languages are listed below:

Chinese characters

A Chinese character, also known as a Han character, is a logogram, i.e. it can be represented as a word. The number of characters in the Kangxi dictionary is over 47000 but in China knowledge of three to four thousand characters is sufficient. In modern times the Chinese characters have been greatly simplified and are used in mainland China while traditional Chinese characters are still used in Honk Kong and Taiwan. The Chinese characters have been romanised. They are called Pinyin and are also widely used in China.

Japanese characters

The modern Japanese writing system uses three main scripts:

- Kanji are ideographs from Chinese characters
- Hiragana is used for native Japanese words and
- Katakana is used for loanwords
- Romanised Japanese characters, called Romanji, are also used in Japanese texts.

Korean characters

The modern Korean writing system is called Hangul and officially used in North and South Korea. In addition, Hanja is used which refers to the characters borrowed from Chinese.

Arabic alphabet

This script is used for writing several languages in Asia (e.g. Middle East, Pakistan,) and Africa (e.g. Arabic and Urdu). It is written from right to left in a cursive style and includes 28 letters.

Unicode

Unicode is a standard for the consistent representation and use of characters found in the writing systems of the world. It war not easy to adapt languages to computers, partly due to the large number of characters of some languages. It is possible to encode one English character with just one byte because written English only needs a small number of characters. This does not apply to languages like Japanese, Chinese or Korean which have more than 256 characters and therefore require double byte or multi-byte encoding. Several encoding methods are used and Unicode seems to be the most universal method. It obviously encodes into all languages in the world.

For example the Han unification, contracted to Unihan, is an approach by Unicode and the Universal Character Set (according to ISO 10646) to map several character sets of the Chinese, Japanese and Korean languages in a single set of unified characters.

Arabic characters can be encoded by Unicode from Version 5.0 or higher (several character sets and ISO 8859-6).

ISO 10646 specifies the Universal Multiple Octet Coded Character Set. It is used for the representation, interchange, processing, storage and input of the written form of the languages in the world as well as for additional symbols.

The Unicode standard versions 4...6 all comply with ISO 10646.

Pictogram

This is a graphical symbol, also called a pictograph, representing a concept, object, event or an activity by illustration. Pictograms have been used for many thousand years. They are still important in the event of language barriers and illiteracy in the modern world and are used as pictorial signs, representational signs, instructions or statistical diagrams. Due to their graphical nature they are used in different areas of life. To indicate, for example, to toilets and airports a standard set of pictograms is defined in the standard ISO 7001 "Graphical Symbols - Public Information Symbols".

A pictogram has been developed into a functional visual language for people with cognitive problems. Each image represents a word or concept. It comprises two elements, drawn images and text. The symbols are mostly white on a black square.

6.2.5 Cultural details are often not transferable

7461

Country, culture or language-specific details should be avoided in the source text because their use is often not necessary and adaptation to the target culture is time-consuming. In most cases the author does not know that his texts or graphics are characterised in terms of culture or language or lead to localisation problems due to other design-related decisions. Problems can, for example, occur in the following areas:

- Colours
- Symbols
- Illustrations
- Reading direction

Colours

7464

The selection of the "right" colour is an important element for the text and product design. Many colours are culture-specific and can lead to misunderstandings if used incorrectly and even to an image loss of the product as a result of handling faults.

Examples:

	Colour	Meaning in Europe + USA	Meaning in other cultures
	Red	Drama, turmoil, blood (fight, revenge and death), love, danger, nobility	China: fortune, cheerful Russia: beautiful Egypt: death India: life, creative Japan: anger, danger
	Yellow	Caution, warning, sunlight, eternity, envy, hate	China: birth, health, force Egypt: cheerful, property India: success Japan: nobility
	Green	Nature, ecology, hope, immortal, fortune	China: eternity, family, harmony, health, peace, posterity Egypt: fertile, strength India: property, fertile Japan: future, youth, energy
	Blue	Water, sky, loyalty, freedom, reliable, joy, friendship, male	Asia: richness, strength Egypt: virtue, faith, truth
	White	Light, pure, wise, life, perfect, ideal, good, matter of fact, clear, innocent, honest	Asia: death, grief, purity Egypt: joy
	Black	Death, grief, darkness, evil. Also: fraternity, power and unity	(Grief not in Buddhism) Egypt: resurrection
	Grey	Wisdom and age	Asia: helpful

Symbols

7465

As symbols are often produced in analogy to culture-specific concepts or use allusions to familiar areas of the source culture, they pose a problem for localisation.

Example:

The symbol for a house that is to stand for start or beginning is not clearly understandable because the English term "home" cannot be transferred without problem.

Illustrations

7466

An image is not always a sensible substitute for a text.

The representation of more complex processes can become impossible. How is, for example, the request "press the button until you feel a slight resistance" to be illustrated?

Even if an illustration is a good representation of a fact, its use has to be well considered at international level. Replacing text by images is only sensible and reduces cost if the illustrations are independent of culture, i.e. can be used in ALL intended target countries without adaptations. Many things which are self-evident for us are not self-evident in other cultures.

The illustration of people can lead to problems: What sex must or may a person have? What skin colour? What age? Eventually, the addressees in all target countries are to feel equally addressed. Clothing which does not stand out in Western Europe can lead to irritations in Arabic or African countries. Gestures and individual body parts, especially hands and eyes, should not be represented because they often trigger an offensive or insulting association.

Reading direction

7468

In most cultures reading is done from left to right and from top to bottom.

Some Asian cultures, however, read from bottom to top and from back to front.

Many Arabic cultures read from right to left.

These particularities have to be taken into account for graphical instructions!

6.2.6 Directives and standards

Contents

7445

The following list is only a selection and is not complete.

ISO 7001 _ Graphical symbols – Public information symbols

7456

A graphical symbol, also called a pictograph, represents a concept, object, event or an activity by illustration. Pictograms have been used for many thousand years. They are still important in the event of language barriers and illiteracy in the modern world and are used as pictorial signs, representational signs, instructions or statistical diagrams. Due to their graphical nature they are used in different areas of life.

Examples:

ISO 9126 _ Software engineering – Product quality

7446

The standard describes the following criteria:

Functionality: To what extent does the software have the required functions?

- Suitability: suitability of functions for specified tasks, e.g. task-oriented composition of functions from sub-functions.
- Correctness: providing the correct or agreed results or effects, e.g. necessary accuracy of calculated values.
- Interoperability: ability to interact with specified systems.
- Security: ability to prevent unauthorised access (inadvertent or intentional) to programs and data.
- Compliance: software features that cause the software to comply with application-specific standards or agreements or legal provisions and similar regulations.

Reliability: Can the software maintain a defined performance level under defined conditions for a defined period?

- Maturity: low failure frequency by error states.
- Error tolerance: ability to maintain a specified performance level in case of software errors or non-compliance with its specified interface.
- Robustness: ability to ensure a stable system in case of inputs which have not been intended. The software withstands "lusers".
- Restorability: ability to restore the performance level in case of a failure and to retrieve the directly involved data. The time and the needed level of input have to be taken into account.
- Conformity: degree to which the software complies with standards or agreements on reliability.

Usability: What level of input does the use of the software require from users and how is it assessed by them?

- Understandability: level of input required from the user to understand the concept and its application.
- Learnability: level of input required from the user to learn the application (e.g. handling, input, output).
- Usability: level of input required from the user to handle the application.
- Attractiveness: attractiveness of the application for the user.
- Conformity: degree to which the software complies with standards or agreements on usability.

Efficiency: How is the relationship between performance level of the software and equipment used?

- Time behaviour: response and processing times as well as data processing speed when executing the function.
- Consumption behaviour: Number and actuation time of the required operating elements to carry out the functions. Resource consumption, such as CPU time, hard disc access, etc.
- Conformity: degree to which the software complies with standards or agreements on efficiency.

Changeability: What level of input is required make the defined changes in the software? Changes can include corrections, improvements or adaptations to changes of the environment, requirements or functional specifications.

- Analysability: level of input required to diagnose defects or causes of failure or to determine parts that need to be changed.
- Modifiability: level of input required to carry out improvements, eliminate faults or adapt to a changed environment.
- Stability: probability of the occurrence of unexpected effects of changes.
- Testability: level of input required to test the changed software.

Transferability: How easily can the software be transferred to another environment? An environment can be an organisational, hardware or software environment.

- Adaptability: ability of the software to adapt to different environments.
- Installability: level of input required to install the software in a defined environment.
- Coexistence: ability of the software to function in parallel with another software having similar or identical functions.
- Exchangeability: possibility to use this software instead of a another specified software in the environment of that software as well as the level of input required to do so.
- Conformity: degree to which the software complies with standards or agreements on transferability.

ISO 9241 _ Ergonomics of human-system interaction

7447

The standard ISO 9241 is an international standard describing the guidelines of interaction between humans and computers. The series of standards describes requirements for the work environment, hardware and software. The goal of the guideline is to avoid health damage at computer workplaces and to make it easier for the user to carry out his tasks.

The following parts (incomplete list) are part of the standard:

Part 1: General introduction
Part 2: Guidance on task requirements
Part 3: Visual display requirements
Part 4: Keyboard requirements
Part 5: Workstation layout and postural requirements
Part 6: Guidance on the work environment
Part 7: Requirements for display with reflections
Part 8: Requirements for displayed colours
Part 9: Requirements for non-keyboard input devices
(Part 10: Dialogue principles (obsolete, was replaced by part 110 in 2006))
Part 11: Guidance on usability
Part 12: Presentation of information
Part 13: User guidance
Part 14: Menu dialogues
Part 15: Command dialogues
Part 16: Direct manipulation dialogues
Part 17: Form filling dialogues
Part 110: Dialogue principles (replaces part 10)
Part 151: Guidance on World Wide Web user interfaces
Part 171: Guidance on software accessibility (published in October 2008)
Part 300: Introduction to electronic visual display requirements
Part 302: Terminology for electronic visual displays (at present in the draft stage)
Part 303: Requirements for electronic visual displays (at present in the draft stage)
Part 304: User performance test methods
Part 305: Optical laboratory test methods for electronic visual displays (at present in the draft stage)
Part 306: Field assessment methods for electronic visual displays (at present in the draft stage)
Part 307: Analysis and compliance test methods for electronic visual displays (at present in the draft stage)
Part 400: Principles and requirements for physical input devices
Part 410: Design criteria for physical input devices (at present in the draft stage)

Parts 5 and 6 deal with the work environment. Parts 3, 4, 7, 8 and 9 deal with hardware requirements, parts 11...17 and 110 deal with aspects of software ergonomics. Mainly the parts **ISO 9241-110 _ Dialogue principles** (→ p. 127) and **ISO 9241-11 _ Guidance on usability** (→ p. 126) contain some criteria for the ergonomic design of interactive systems.

ISO 9241-11 _ Guidance on usability

7448

The usability of a software depends on its context of use. In part 11 of ISO 9241 three main criteria are defined for the usability of a software:

- Effectivity to solve a task
- Efficiency to use the system
- Satisfaction of the software user

ISO 9241-110 _ Dialogue principles

7450

User interfaces of interactive systems such as websites or software should be easy to use. Part 110 of ISO 9241 describes the following principles for the design and evaluation of an interface between the user and system (dialogue design):

- Suitability for the task
  Suitable functionality, minimisation of unnecessary interactions
- Self-descriptiveness
  Understandability by means of support / feedback
- Suitability for learning
  User guidance, suitable metaphors, goal: minimum learning time
- Controllability
  Dialogue control by the user
- Conformity with user expectations
  Consistency, adaptation to the user model
- Suitability for individualisation
  Adaptability to the user and his context of work
- Error tolerance
  Intelligent dialogue principles so that the user avoids error is given priority. Other aspects:
  Detected user errors do not prevent the user's goal.
  Undetected errors: slight correction by the user.

ISO 10646 _ Information technology – Universal multiple-octet coded character set (UCS)

7455

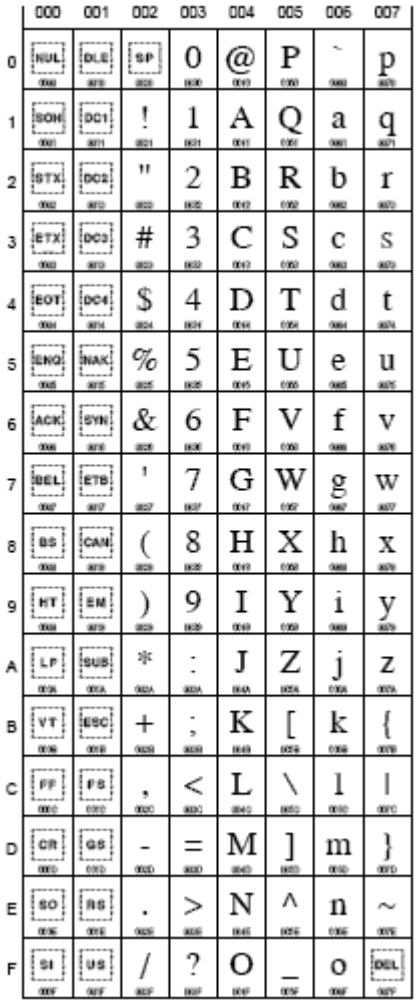
The universal character set (UCS) is a standard set of characters which is defined in the international standard ISO 10646. For all practical purposes this is the same as Unicode.

Per character a memory space of 2 bytes is used. Unicode is a 16-bit code which represents 2^{16} = 65536 characters. The first goal is a clear and standardised encoding of the characters of all national languages.

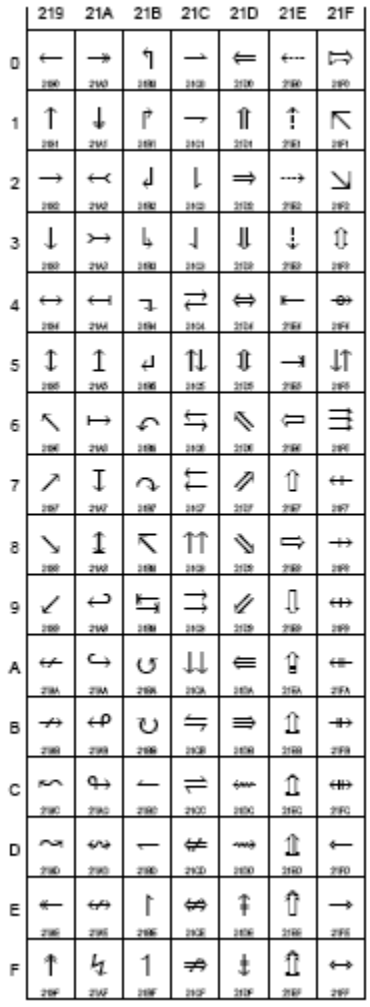
Not all of these 65536 character addresses are used. A user-defined area enables approx. 2000 addresses with user-specific characters.

Another 1408576 characters can be encoded via the combination of two 16-bit codes. The hope is to be able to cover all characters that exist or have ever existed. Furthermore, technical symbols, musical signs, phonetics, etc. are mapped. However, one is still far from using all character addresses.

Examples:



Unicode: control characters and basic characters



Unicode: arrows

ISO 13406 _ Ergonomic requirements for work with visual displays based on flat panels

7453

Part 2: Ergonomic requirements for flat panel displays

According to the international standard ISO 13406-2 LCD screens are classified on the basis of the following criteria:

- Luminance, contrast and colour measured by the viewer's direction
- Reflections and contrast in case of incident illumination
- Image set-up time
- Faults (pixel faults)

ISO 13407 _ Human-centred design processes for interactive systems

7452

ISO 13407 is a standard which describes a prototypical human-centred software development process. A special development process can be considered to conform to the standard if its recommendations are met.

The standard represents human-centred design as an interdisciplinary activity covering knowledge of human factors and ergonomic information and techniques. The ISO process consists of four essential sub-activities:

- Understand the context of use:
  The result of this activity is a documented description of the relevant users, their tasks and their environment.
- Specify requirements:
  During this phase the targets are deducted from the existing documentation at a compromise level. The division of the system tasks is defined in...
  - tasks to be carried out by people
  - tasks to be carried out by technology
- Produce solutions:
  This can be done following a prototype development or another iterative process. These prototypes can be paper drafts (mocks) or executable program versions. If there are company-internal design rules for user interfaces, they should be used.
- Evaluate solutions:
  The solutions are checked for compliance with the defined requirements. To do so, expert assessments, usability tests, interviews or a combination of these can be used. The determined deviations are evaluated for their relevance and are a starting point of the next iteration of the development process.

This method is complementary with existing process models of the software development. According to the standard the human-centred design process should start in the earliest stage of the project and should be repeated until the system meets the requirements. The significance and required level of input for the human-centred design depend on the size and type of the product to be developed. For smaller projects this is controlled by individuals.

ISO 20282 _ Ease of operation of everyday products

7443

This draft consists of

- Part 1: Design requirements for context of use and user characteristics
  The following criteria are described:
  - Scope
  - User interface
  - User
  - Psychological and social characteristics
  - Physical and social environmental factors
  - Physical and sensory characteristics
- Part 2: Test method for walk-up-and-use products
  This part is a technical specification for the test methods.

6.3 Basic information about colours and bitmap graphics

Contents

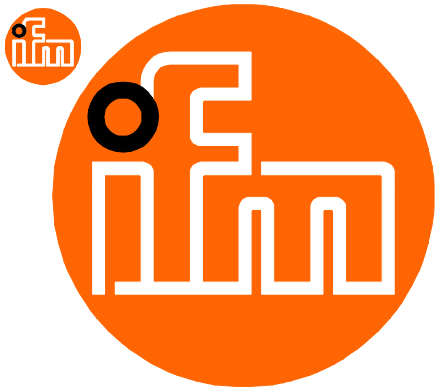
3112

For graphics and image files two basic types are distinguished:

	Vector graphics	Pixel graphics
Examples:	Drawings of CAD programs Character sets type TrueType, PostScript or OpenType	Digital photos Files from the scanner or capture programs
Principle:	Vector graphics are based on an image description which exactly defines the objects from which the image is made. A circle is, for example, defined via the position of the centre (coordinates), radius, line thickness and colour.	A raster graphics, also pixel graphics or bitmap, is a way of describing an image which consists of a raster-type arrangement of pixels to which one colour each is assigned. The main characteristics of a raster graphics are therefore width and height in pixels (image resolution) as well as the colour depth.
Required memory space:	Required memory space relatively small	Depending on the resolution the required memory space is high or very high: the files become larger with every additionally pixel to be stored.
Loss when scaling:	Loss-free resampling (scaling) to any image sizes possible	Resampling (scaling) to other image sizes leads to quality loss in most cases.
Hardware performance:	Since monitors are in principle based on a raster matrix, all graphics must be resampled to individual pixels (= rastered) to display them on the monitor. Depending on the complexity of the graphics very powerful computers are needed to enable quick processing and display.	Requirements relatively low
Typical file extensions:	*.cdr (Corel Draw) *.dwg (AutoCAD) *.ai (Adobe Illustrator) *.svg (Scalable Vector Graphics)	*.bmp (Bitmap) *.gif (Compuserv GIF) *.jpg (Joint Photographic Experts Group) *.png (Portable Network Graphics)

6.3.1 Image size vector graphics / pixel graphics

7380

Vector graphics	Pixel graphics
Graphical elements are described as vectors: information about start and end point, thickness and colour of a line, possibly fill pattern and colour gradient.	Pixel graphics of modern digital cameras have 5 million and more pixels (resolution = 5 megapixels). A special data compression tries to reduce the required high memory space. Unfortunately, compression leads to a poorer quality.
Reduction or enlargement is easy and leads to no quality loss (→ example below).	Enlargement leads to block graphics or blurry images (→ example below). Reducing such a megapixel image results in high loss of image information.
Example:  Original Ø 10 mm / enlargement 5 times EPS file 35 kB	Example:  Original 30 x 30 px / enlargement 5 times BMP file 3 kB / 62 kB

Example: reducing a pixel image for CR108n

19193
7402

Task: An existing digital photo with a resolution of 5 megapixels has, for example, an image size of 2560 x 1920 pixels (= 4,915,200 pixels).
This photo is to be displayed in an image size of only 800 x 480 pixels (= monitor size for PDM360NG).

Problem 1: The side to height ratio is 4:3 (1.33:1) for the source but 15:9 (1.66:1) for the target.

Solution (anisotropic): Scale the height and side of the image in different scales to represent an undistorted image on the display.
For a uniform (isotropic) scaling the image is distorted compared to the original.

Problem 2: After scaling there are only 384,000 pixels (= 7.8 % of the original image), the other 4,531,200 pixels are no longer available.
In other words: Horizontally only every 3rd pixel is used, vertically only every 4th pixel.

Therefore such a transformed photo can no longer have the quality of the original. Important information is lost and the image is distorted.

► **Solution:** Create images in the required size and resolution right from the start.

The problem applies correspondingly to other devices with different monitor sizes.

Adapt bitmap graphics

9996

You can adapt existing bitmap graphics by means of common graphics software.
Please ask your ***ecomatmobile*** specialist!

6.3.2 Colour for bitmap graphics

3121

A second important factor is the colour information (the RGB value) which is stored for every pixel.

RGB stands for red, green and blue. For each of these three primary colours 255 intensity levels are available. By mixing these three primary colours in different intensities approx. 16.6 million colours can be created via the **Additive colour mixing** (→ p. 134). To represent this quantity suitable monitors and powerful processors are needed.

6.3.3 Which colours are shown?

19194

Colours in the CR108n

19195
7381

The display can represent a colour depth of 6 bits per primary colour, i.e. 64 colour grades. Consequently, from the total spectrum of 256 addressable colour grades only every fourth can be used:

Colour		Allowed colour values
Red		R = 0, 4, 8, 12, 16, ..., 236, 240, 244, 248, 252
Green		G = 0, 4, 8, 12, 16, ..., 236, 240, 244, 248, 252
Blue		B = 0, 4, 8, 12, 16, ..., 236, 240, 244, 248, 252

Values which do not fit into this pattern are not shown.

Colours in the CR045n

19196
8367

The device can represent a colour depth of 8 bits, i.e. a total of 256 colour grades. From the total spectrum of 256 addressable colour grades per colour channel (= 16,777,216 colours) only every 65,536th can be used:

Colour		Allowed colour values
Red		R (3 bits = 8 grades) = 0, 32, 64, 96, 128, 160, 192, 224
Green		G (3 bits = 8 grades) = 0, 32, 64, 96, 128, 160, 192, 224
Blue		B (2 bits = 4 grades) = 0, 64, 128, 192

The colour palette was specified in the factory and is permanently stored in the device.

Values which do not fit into this pattern are not shown.

6.4 Special information about bitmap graphics

Contents

3113

Here the interested reader finds more details about bitmap graphics.

6.4.1 Additive colour mixing

3123

RGB	Mixed colours
Monitors and many printers make mixed colours from the 3 primary colours red, green and blue.	Mixed colours are made by adding the colours in the required mix ratio. This method is therefore called additive colour mixing.

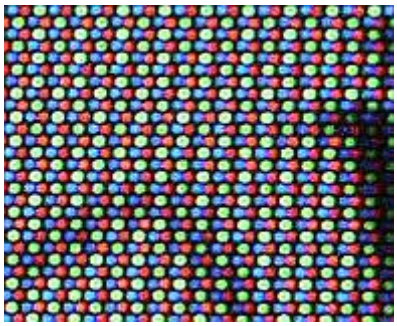
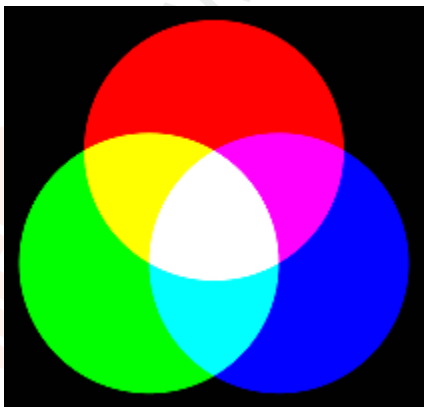

	

Photo: RGB raster of a monitor, considerably enlarged	Figure: additive colour mixing

Table: examples of colour mixtures

100 % red		+ 100 % green		= 100 % yellow	
100 % green		+ 100 % blue		= 100 % cyan	
100 % blue		+ 100 % red		= 100 % magenta	

Nuances in the colour saturation result from smaller shares of the respective primary colour:

Screenshot: RGB colour mixture at Photoshop; 100 % ⇒ 255_{dez} = FF_{hex}

6.4.2 What graphics are suitable for which PDM and what steps must be carried out?

7387

Not all bitmaps are suitable for display on the PDM.

- In principle, photos should be transformed so that they are optimised when displayed in the given resolution and colour depth.
- Images with a low contrast are not suitable because the colour differences cannot be displayed on the PDM.
- If needed, logos and symbols should be optimised for the display on the PDM or drawn again.

7 Overview of the files and libraries used

Contents

2711

(as on 2014-06-25)

Depending on the unit and the desired function, different libraries and files are used. Some are automatically loaded, others must be inserted or loaded by the programmer.

7.1 General overview

2712

File name	Description and memory location [1])
ifm_CRnnnn_Vxx.CFG	PLC configuration per device only 1 device-specific file inlcudes: IEC and symbolic addresses of the inputs and outputs, the flag bytes as well as the memory allocation …\CoDeSys V*\Targets\ifm\ifm_CRnnnncfg\Vxxyyzz
CAA-*.CHM	Online help per device only 1 device-specific file inlcudes: online help for this device …\CoDeSys V*\Targets\ifm\Help\… (language)
ifm_CRnnnn_Vxxyyzz.H86 ifm_CRnnnn_Vxxyyzz.RESX	Runtime system (must be loaded into the controller / monitor when used for the first time) per device only 1 device-specific file …\CoDeSys V*\Targets\ifm\Library\ifm_CRnnnn
ifm_Browser_CRnnnn.INI	CODESYS browser commands (CODESYS needs the file for starting the project) per device only 1 device-specific file inlcudes: commands for the browser in CODESYS …\CoDeSys V*\Targets\ifm
ifm_Errors_CRnnnn.INI	CODESYS error file (CODESYS needs the file for starting the project) per device only 1 device-specific file inlcudes: device-specific error messages from CODESYS …\CoDeSys V*\Targets\ifm
ifm_CRnnnn_Vxx.TRG	Target file per device only 1 device-specific file inlcudes: hardware description for CODESYS, e.g.: memory, file locations …\CoDeSys V*\Targets\ifm
ifm_*_Vxxyyzz.LIB	General libraries per device several files are possible …\CoDeSys V*\Targets\ifm\Library
ifm_CRnnnn_Vxxyyzz.LIB	Device-specific library per device only 1 device-specific file inlcudes: function elements of this device …\CoDeSys V*\Targets\ifm\Library\ifm_CRnnnn
ifm_CRnnnn_*_Vxxyyzz.LIB	Device-specific libraries per device several files are possible → following tables …\CoDeSys V*\Targets\ifm\Library\ifm_CRnnnn

Legend:

*	any signs
CRnnnn	article number of the controller / monitor
V*	CODESYS version
Vxx	version number of the ifm software
yy	release number of the ifm software
zz	patch number of the ifm software

[1]) memory location of the files:
System drive (C: / D:) \ program folder\ ifm electronic

7.2 What are the individual files and libraries used for?

Contents

2713

The following overview shows which files/libraries can and may be used with which unit. It may be possible that files/libraries which are not indicated in this list can only be used under certain conditions or the functionality has not yet been tested.

7.2.1 Files for the runtime system

2714

File name	Function	Available for:
`ifm_CRnnnn_Vxxyyzz.H86` `ifm_CRnnnn_Vxxyyzz.RESX`	runtime system	• all ecomat*mobile* controllers • BasicDisplay: CR045n • PDM: CR10nn
`ifm_Browser_CRnnnn.INI`	CODESYS browser commands	• all ecomat*mobile* controllers • PDM: CR10nn
`ifm_Errors_CRnnnn.INI`	CODESYS error file	• all ecomat*mobile* controllers • PDM: CR10nn

7.2.2 Target file

2715

File name	Function	Available for:
`ifm_CRnnnn_Vxx.TRG`	Target file in CODESYS	• all ecomat*mobile* controllers • BasicDisplay: CR045n • PDM: CR10nn

7.2.3 PLC configuration file

2716

File name	Function	Available for:
`ifm_CRnnnn_Vxxyyzz.CFG`	PLC configuration in CODESYS	• all ecomat*mobile* controllers • BasicDisplay: CR045n • PDM: CR10nn

7.2.4 ifm device libraries

2717

File name	Function	Available for:
`ifm_CRnnnn_Vxxyyzz.LIB`	device-specific library	• all ecomat*mobile* controllers • BasicDisplay: CR045n • PDM: CR10nn
`ifm_CR0200_MSTR_Vxxyyzz.LIB`	library without extended functions	• ExtendedController: CR0200
`ifm_CR0200_SMALL_Vxxyyzz.LIB`	library without extended functions, reduced functions	• ExtendedController: CR0200

7.2.5 ifm CANopen libraries master / slave

2718

These libraries are based on the CODESYS libraries (3S CANopen function elements) and make the functions available to the user in a simple way.

File name	Function	Available for:

`ifm_CRnnnn_CANopenMaster_Vxxyyzz.LIB`
`ifm_CRnnnn_CANopen`**`x`**`Master_Vxxyyzz.LIB`
`ifm_CRnnnn_CAN`**`x`**`openMaster_Vxxyyzz.LIB`
	CANopen master emergency and status handler	• all ecomat*mobile* controllers *)
		• PDM: CR10nn *)

`ifm_CRnnnn_CANopenSlave_Vxxyyzz.LIB`
`ifm_CRnnnn_CANopen`**`x`**`Slave_Vxxyyzz.LIB`
`ifm_CRnnnn_CAN`**`x`**`openSlave_Vxxyyzz.LIB`
	CANopen slave emergency and status handler	• all ecomat*mobile* controllers *)
		• PDM: CR10nn *)

`ifm_CANx_SDO_Vxxyyzz.LIB`	CANopen SDO read and SDO write	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
`ifm_CANopen_NT_Vxxyyzz.LIB`	CANopen function elements in the CAN stack	• BasicController: CR040n, CR041n, CR043n • BasicDisplay: CR045n • PDM360 NG: CR108n, CR120n • SmartController: CR253n

*) but NOT for...
- BasicController: CR040n, CR041n, CR043n
- BasicDisplay: CR045n
- PDM360 NG: CR108n, CR120n
- SmartController: CR253n

7.2.6 CODESYS CANopen libraries

2719

For the following devices these libraries are NOT useable:
- BasicController: CR040n, CR041n, CR043n
- BasicDisplay: CR045n
- PDM360 NG: CR108n, CR120n
- SmartController: CR253n

File name	Function	Available for:
3S_CanDrvOptTableEx.LIB	CANopen driver	• all ecomatmobile controllers • PDM360smart: CR1070, CR1071
3S_CanDrv.LIB [1])		• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
3S_CANopenDeviceOptTableEx.LIB	CANopen slave driver	• all ecomatmobile controllers • PDM360smart: CR1070, CR1071
3S_CANopenDevice.LIB [1])		• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
3S_CANopenManagerOptTableEx.LIB	CANopen network manager	• all ecomatmobile controllers • PDM360smart: CR1070, CR1071
3S_CANopenManager.LIB [1])		• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
3S_CANopenMasterOptTableEx.LIB	CANopen master	• all ecomatmobile controllers • PDM360smart: CR1070, CR1071
3S_CANopenMaster.LIB [1])		• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
3S_CANopenNetVarOptTableEx.LIB	Driver for network variables	• all ecomatmobile controllers • PDM360smart: CR1070, CR1071
3S_CANopenNetVar.LIB [1])		• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056

[1]) For the following devices: This library is without function used as placeholder:
- BasicController: CR040n, CR041n, CR043n
- BasicDisplay: CR045n
- PDM360 NG: CR108n, CR120n
- SmartController: CR253n

7.2.7 Specific ifm libraries

2720

File name	Function	Available for:
ifm_RawCAN_NT_Vxxyyzz.LIB	CANopen function elements in the CAN stack based on Layer 2	• BasicController: CR040n, CR041n, CR043n • BasicDisplay: CR045n • PDM360 NG: CR108n, CR120n • SmartController: CR253n
ifm_J1939_NT_Vxxyyzz.LIB	J1939 communication function elements in the CAN stack	• BasicController: CR040n, CR041n, CR043n • BasicDisplay: CR045n • PDM360 NG: CR108n, CR120n • SmartController: CR253n
ifm_NetVarLib_NT_Vxxyyzz.lib	additional driver for network variables	• BasicController: CR040n, CR041n, CR043n • BasicDisplay: CR045n • PDM360 NG: CR108n, CR120n • SmartController: CR253n
ifm_J1939_Vxxyyzz.LIB	J1939 communication function elements	up to runtime system V04: • CabinetController: CR0303 • ClassicController: CR0020, CR0505 • ExtendedController: CR0200 • SafetyController: CR7020, CR7200, CR7505 • SmartController: CR2500 • PDM360smart: CR1070, CR1071
ifm_J1939_x_Vxxyyzz.LIB	J1939 communication function elements	from runtime system V05: • CabinetController: CR0303 • ClassicController: CR0020, CR0505 • ExtendedController: CR0200 • SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506 • SmartController: CR2500 • sPDM360smart: CR1070, CR1071
ifm_CRnnnn_J1939_Vxxyyzz.LIB	J1939 communication function elements	• Controller: CR0n3n, CR7n3n
ifm_PDM_J1939_Vxxyyzz.LIB	J1939 communication function elements	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
ifm_CANx_LAYER2_Vxxyyzz.LIB	CAN function elements on the basis of layer 2: CAN transmit, CAN receive	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
ifm_CAN1E_Vxxyyzz.LIB	changes the CAN bus from 11 bits to 29 bits	up to runtime system V04: • PDM360smart: CR1070, CR1071
ifm_CAN1_EXT_Vxxyyzz.LIB	changes the CAN bus from 11 bits to 29 bits	from runtime system V05: • CabinetController: CR030n • ClassicController: CR0020, CR0505 • ExtendedController: CR0200 • PCB controller: CS0015 • SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506 • SmartController: CR250n • PDM360smart: CR1070, CR1071
ifm_CAMERA_O2M_Vxxyyzz.LIB	camera function elements	• PDM360: CR1051
CR2013AnalogConverter.LIB	analogue value conversion for I/O module CR2013	• all ecomat*mobile* controllers • PDM: CR10nn
ifm_Hydraulic_16bitOS04_Vxxyyzz.LIB	hydraulic function elements for R360 controllers	up to runtime system V04: • ClassicController: CR0020, CR0505 • ExtendedController: CR0200 • SafetyController: CR7020, CR7200, CR7505 • SmartController: CR250n

File name	Function	Available for:
ifm_Hydraulic_16bitOS05_Vxxyyzz.LIB	hydraulic function elements for R360 controllers	from runtime system V05: • ClassicController: CR0020, CR0505 • ExtendedController: CR0200 • SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506 •SmartController: CR250n
ifm_Hydraulic_32bit_Vxxyyzz.LIB	hydraulic function elements for R360 controllers	• Controller: CR0n3n, CR7n3n
ifm_Hydraulic_CR0303_Vxxyyzz.LIB	hydraulic function elements for R360 controllers	• CabinetController: CR0303
ifm_SafetyIO_Vxxyyzz.LIB	safety function elements	• SafetyController: CR7020, CR7021, CR7200, CR7201, CR7505, CR7506
ifm_SafetyPLCopen_Vxxyyzz.LIB	safety function elements	• SafetyController: CR7032, CR7132
ifm_PDM_UTIL_Vxxyyzz.LIB	auxiliary functions PDM	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
ifm_PDMng_UTIL_Vxxyyzz.LIB	auxiliary functions PDM	• PDM360 NG: CR108n, CR120n
ifm_PDMsmart_UTIL_Vxxyyzz.LIB	auxiliary functions PDM	• PDM360smart: CR1070, CR1071
ifm_PDM_Input_Vxxyyzz.LIB	alternative input function elements PDM	• PDM: CR10nn
ifm_CR107n_Init_Vxxyyzz.LIB	initialisation function elements PDM360smart	• PDM360smart: CR1070, CR1071
ifm_PDM_File_Vxxyyzz.LIB	file function elements PDM360	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056 • PDM360 NG: CR108n, CR120n
ifm_PDM360NG_linux_syscall_asynch_LIB	send Linux commands to the system	• PDM360 NG: CR108n, CR120n
ifm_PDM360NG_USB_Vxxyyzz.LIB	manage devices at the USB interface	• PDM360 NG: CR108n, CR120n
ifm_PDM360NG_USB_LL_Vxxyyzz.LIB	auxiliary library for ifm_PDM360NG_USB_Vxxyyzz.LIB	• PDM360 NG: CR108n, CR120n
Instrumente_x.LIB	predefined display instruments	• PDM: CR10nn
Symbols_x.LIB	predefined symbols	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056
Segment_x.LIB	predefined 7-segment displays	• PDM360: CR1050, CR1051 • PDM360compact: CR1052, CR1053, CR1055, CR1056

Further libraries on request.

8 Diagnosis and error handling

Contents

19598

The runtime-system (RTS) checks the device by internal error checks:
• during the boot phase (reset phase)
• during executing the application program
→ chapter Operating states

In so doing a high operating reliability is provided, as much as possible.

8.1 Overview

12217

When errors are detected the system flag ERROR can also be set in the application program. Thus, in case of a fault, the controller reacts as follows:

> the operation LED lights red,
> the output relays switch off,
> the outputs protected by the relays are disconnected from power,
> the logic signal states of the outputs remain unchanged.

NOTE

If the outputs are switched off by the relays, the logic signal states remain unchanged.

► The programmer must evaluate the ERROR bit and thus also reset the output logic in case of a fault.

Complete list of the device-specific error codes and diagnostic messages
→ chapter system flags.

9 Terms and abbreviations

A

Address

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

Application software

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

Architecture

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

B

Baud

Baud, abbrev.: Bd = unit for the data transmission speed. Do not confuse baud with "bits per second" (bps, bits/s). Baud indicates the number of changes of state (steps, cycles) per second over a transmission length. But it is not defined how many bits per step are transmitted. The name baud can be traced back to the French inventor J. M. Baudot whose code was used for telex machines.
1 MBd = 1024 x 1024 Bd = 1 048 576 Bd

Boot loader

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

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

Bus

Serial data transmission of several participants on the same cable.

C

CAN

CAN = **C**ontroller **A**rea **N**etwork
CAN is a priority-controlled fieldbus system for large data volumes. There are several higher-level protocols that are based on CAN, e.g. 'CANopen' or 'J1939'.

CAN stack

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

CiA

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

CiA DS 304

DS = **D**raft **S**tandard
CANopen device profile for safety communication

CiA DS 401

DS = **D**raft **S**tandard
CANopen device profile for binary and analogue I/O modules

CiA DS 402

DS = **D**raft **S**tandard
CANopen device profile for drives

CiA DS 403

DS = **D**raft **S**tandard
CANopen device profile for HMI

CiA DS 404

DS = **D**raft **S**tandard
CANopen device profile for measurement and control technology

CiA DS 405

DS = **D**raft **S**tandard
CANopen specification of the interface to programmable controllers (IEC 61131-3)

CiA DS 406

DS = **D**raft **S**tandard
CANopen device profile for encoders

CiA DS 407

DS = **D**raft **S**tandard
CANopen application profile for local public transport

Clamp 15

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

COB ID

COB = **C**ommunication **Ob**ject
ID = **Id**entifier
ID of a CANopen communication object
Corresponds to the identifier of the CAN message with which the communication project is sent via the CAN bus.

CODESYS

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

CSV file

CSV = **C**omma **S**eparated **V**alues (also: **C**haracter **S**eparated **V**alues)
A CSV file is a text file for storing or exchanging simply structured data.
The file extension is `.csv`.

Example: Source table with numerical values:

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

This results in the following CSV file:

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

Cycle time

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

D

Data type

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

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

DC

Direct **C**urrent

Diagnosis

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

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

For diagnosis, configuration and log data can be used, created during the "normal" operation of the device.
The correct start of the system components is monitored during the initialisation and start phase. Errors are recorded in the log file.
For further diagnosis, self-tests can also be carried out.

Dither

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

DLC

Data **L**ength **C**ode = in CANopen the number of the data bytes in a message.
For →SDO: DLC = 8

DRAM

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

DTC

DTC = **D**iagnostic **T**rouble **C**ode = error code
In the protocol J1939 faults and errors well be managed and reported via assigned numbers – the DTCs.

E

ECU

(1) **E**lectronic **C**ontrol **U**nit = control unit or microcontroller
(2) **E**ngine **C**ontrol **U**nit = control device of a engine

EDS-file

EDS = **E**lectronic **D**ata **S**heet, e.g. for:
- File for the object directory in the CANopen master,
- CANopen device descriptions.

Via EDS devices and programs can exchange their specifications and consider them in a simplified way.

Embedded software

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

EMC

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

EMCY

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

Ethernet

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

EUC

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

F

FiFo

FIFO (**F**irst **I**n, **F**irst **O**ut) = Operating principle of the stack memory: The data packet that was written into the stack memory first, will also be read first. Each identifier has such a buffer (queue).

Flash memory

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

Advantages of flash memories

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

Disadvantages of flash memories

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

FRAM

FRAM, or also FeRAM, means **Fe**rroelectric **R**andom **A**ccess **M**emory. The storage operation and erasing operation is carried out by a polarisation change in a ferroelectric layer.
Advantages of FRAM as compared to conventional read-only memories:

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

H

Heartbeat

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

HMI

HMI = **H**uman **M**achine **I**nterface

I

ID

ID = **Id**entifier

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

IEC 61131

Standard: Basics of programmable logic controllers

- Part 1: General information
- Part 2: Production equipment requirements and tests
- Part 3: Programming languages
- Part 5: Communication
- Part 7: Fuzzy Control Programming

IEC user cycle

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

Instructions

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

Intended use

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

IP address

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

ISO 11898

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

ISO 11992

Standard: Interchange of digital information on electrical connections between towing and towed vehicles
- Part 1: Physical and data-link layers
- Part 2: Application layer for brakes and running gear
- Part 3: Application layer for equipment other than brakes and running gear
- Part 4: Diagnostics

ISO 16845

Standard: Road vehicles – Controller area network (CAN) – Conformance test plan

J

J1939

→ SAE J1939

L

LED

LED = **L**ight **E**mitting **D**iode.
Light emitting diode, also called luminescent diode, an electronic element of high coloured luminosity at small volume with negligible power loss.

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

LSB
Least **S**ignificant **B**it/Byte

M

MAC-ID
MAC = **M**anufacturer‘s **A**ddress **C**ode
= manufacturer's serial number.
→ID = **Id**entifier
Every network card has a MAC address, a clearly defined worldwide unique numerical code, more or less a kind of serial number. Such a MAC address is a sequence of 6 hexadecimal numbers, e.g. "00-0C-6E-D0-02-3F".

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

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

MMI
→ **HMI** (→ p. 149)

MRAM
MRAM = **M**agnetoresistive **R**andom **A**ccess **M**emory
The information is stored by means of magnetic storage elements. The property of certain materials is used to change their electrical resistance when exposed to magnetic fields.
Advantages of MRAM as compared to conventional RAM memories:
- non volatile (like FRAM), but:
- access time only approx. 35 ns,
- unlimited number of access cycles possible.

MSB
Most **S**ignificant **B**it/Byte

N

NMT
NMT = **N**etwork **M**anagemen**t** = (here: in the CANopen protocol).
The NMT master controls the operating states of the NMT slaves.

Node
This means a participant in the network.

Node Guarding

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

O

Obj / object

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

Object directory

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

OBV

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

OPC

OPC = **O**LE for **P**rocess **C**ontrol
Standardised software interface for manufacturer-independent communication in automation technology
OPC client (e.g. device for parameter setting or programming) automatically logs on to OPC server (e.g. automation device) when connected and communicates with it.

Operational

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

P

PC card

→PCMCIA card

PCMCIA card

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

PDM

PDM = **P**rocess and **D**ialogue **M**odule.
Device for communication of the operator with the machine / plant.

PDO

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

PDU

PDU = **P**rotocol **D**ata **U**nit = protocol data unit.
The PDU is a term from the →CAN protocol →SAE J1939. It refers to a component of the target address (PDU format 1, connection-oriented) or the group extension (PDU format 2, message-oriented).

PES

Programmable **E**lectronic **S**ystem ...
- for control, protection or monitoring,
- dependent for its operation on one or more programmable electronic devices,
- including all elements of the system such as input and output devices.

PGN

PGN = **P**arameter **G**roup **N**umber
PGN = 6 zero bits + 1 bit reserved + 1 bit data page + 8 bit PDU Format (PF) + 8 PDU Specific (PS)
The parameter group number is a term from the →CAN protocol →SAE J1939.

Pictogram

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

PID controller

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

PLC configuration

Part of the CODESYS user interface.

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

Pre-Op

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

Process image

Process image is the status of the inputs and outputs the PLC operates with within one →cycle.

- At the beginning of the cycle the PLC reads the conditions of all inputs into the process image. During the cycle the PLC cannot detect changes to the inputs.
- During the cycle the outputs are only changed virtually (in the process image).
- At the end of the cycle the PLC writes the virtual output states to the real outputs.

PWM

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

R

ratiometric

Measurements can also be performed ratiometrically. If the output signal of a sensor is proportional to its suppy voltage then via ratiometric measurement (= measurement proportional to the supply) the influence of the supply's fluctuation can be reduced, in ideal case it can be eliminated.
→ analogue input

RAW-CAN

RAW-CAN means the pure CAN protocol which works without an additional communication protocol on the CAN bus (on ISO/OSI layer 2). The CAN protocol is international defined according to ISO 11898-1 and garantees in ISO 16845 the interchangeability of CAN chips in addition.

remanent

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

ro

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

RTC

RTC = **R**eal **T**ime **C**lock
Provides (batter-backed) the current date and time. Frequent use for the storage of error message protocols.

Runtime system

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

rw

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

S

SAE J1939

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

SD card

An SD memory card (short for **S**ecure **D**igital Memory Card) is a digital storage medium that operates to the principle of →flash storage.

SDO

SDO = **S**ervice **D**ata **O**bject.
The SDO is used for access to objects in the CANopen object directory. 'Clients' ask for the requested data from 'servers'. The SDOs always consist of 8 bytes.

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

Every SDO is monitored for a response and repeated if the slave does not respond within the monitoring time.

Self-test

Test program that actively tests components or devices. The program is started by the user and takes a certain time. The result is a test protocol (log file) which shows what was tested and if the result is positive or negative.

Slave

Passive participant on the bus, only replies on request of the →master. Slaves have a clearly defined and unique →address in the bus.

stopped

Operating status of a CANopen participant. In this mode only →NMT commands are transferred.

Symbols

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

System variable

Variable to which access can be made via IEC address or symbol name from the PLC.

T

Target

The target contains the hardware description of the target device for CODESYS, e.g.: inputs and outputs, memory, file locations.
Corresponds to an electronic data sheet.

TCP

The **T**ransmission **C**ontrol **P**rotocol is part of the TCP/IP protocol family. Each TCP/IP data connection has a transmitter and a receiver. This principle is a connection-oriented data transmission. In the TCP/IP protocol family the TCP as the connection-oriented protocol assumes the task of data protection, data flow control and takes measures in the event of data loss. (compare: →UDP)

Template

A template can be filled with content.
Here: A structure of pre-configured software elements as basis for an application program.

U

UDP

UDP (**U**ser **D**atagram **P**rotocol) is a minimal connectionless network protocol which belongs to the transport layer of the internet protocol family. The task of UDP is to ensure that data which is transmitted via the internet is passed to the right application.
At present network variables based on →CAN and UDP are implemented. The values of the variables are automatically exchanged on the basis of broadcast messages. In UDP they are implemented as broadcast messages, in CAN as →PDOs.
According to the protocol, these services are unconfirmed data transmission: it is not checked whether the receiver receives the message. Exchange of network variables corresponds to a "1 to n connection" (1 transmitter to n receivers).

Use, intended

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

W

Watchdog

In general the term watchdog is used for a component of a system which watches the function of other components. If a possible malfunction is detected, this is either signalled or suitable program branchings are activated. The signal or branchings serve as a trigger for other co-operating system components to solve the problem.

10 Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

11 Notizen • Notes • Notes