

IndraMotion MLC 14VRS

Field Bus Interface CANopen

Application Description
R911344488

Edition 01



Title IndraMotion MLC
14VRS
Field Bus Interface CANopen

Type of Documentation Application Description

Document Typecode DOK-XLCMLC-FB*CAN**V14-AP01-EN-P

Internal File Reference RS-214cced2bf5e22fd0a347ea501100e95-1-en-US-4

Purpose of Documentation This documentation describes the CAN bus connection supported by the IndraMotion MLC.
The documentation focuses on the configuration, parameterization, commissioning and diagnostics of the connection.
The CAN bus support in the PLC user program is also described (CAA_CiA405.library).

Change Record

Edition	Release Date	Notes
Edition 01	2016-09	First edition for CAN bus for IndraMotion MLC 14VRS (IndraControl XM2)

Copyright © Bosch Rexroth AG 2016

This document, as well as the data, specifications and other information set forth in it, are the exclusive property of Bosch Rexroth AG. It may not be reproduced or given to third parties without its consent.

Liability The specified data is intended for product description purposes only and shall not be deemed to be a guaranteed characteristic unless expressly stipulated in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.

Editorial Department DC-IA/EAI4, BAn, CCI (vha/MePe)

Table of Contents

	Page
1 About this documentation.....	5
1.1 Validity of the documentation.....	5
1.2 Documentation structure.....	5
1.3 Use of the safety instructions.....	6
1.3.1 Structure of the safety instructions.....	6
1.3.2 Explaining signal words and safety alert symbol.....	6
1.3.3 Symbols used.....	7
1.3.4 Signal graphic explanation on the device.....	7
1.4 Required and supplementing documentation.....	7
1.5 Terms and abbreviations.....	8
2 Field bus features.....	9
3 Postinstalling CAN device descriptions.....	11
4 CAN bus configuration editor.....	13
4.1 CAN bus configuration editor, general.....	13
4.2 CANopen master node.....	15
4.2.1 Tab "CANopen master - General".....	16
4.2.2 Tab "CANopen master - CAN bus parameters".....	16
4.2.3 Tab "CANopen master - Status".....	16
4.2.4 Tab "CANopen master - Information".....	16
4.3 CANopen manager node.....	17
4.3.1 Tab "CANopen manager - General".....	17
4.3.2 Tab "CANopen manager - Parameters".....	19
4.3.3 Tab "CANopen manager - I/O image".....	19
4.3.4 Tab "CANopen manager - Status".....	20
4.3.5 Tab "CANopen manager - Information".....	20
4.4 CANopen remote device.....	20
4.4.1 Tab "CANopen remote device - General".....	20
4.4.2 Tab "CANopen remote device - PDOs".....	22
4.4.3 Tab "CANopen remote device - SDOs".....	24
4.4.4 Tab "CANopen remote device - Parameters".....	25
4.4.5 Tab "CANopen remote device - I/O image".....	25
4.4.6 Tab "CANopen remote device - Status".....	26
4.4.7 Tab "CANopen remote device - Information"remote device.....	26
4.5 CANopen module.....	26
4.5.1 Tab "CANopen module - Parameters".....	26
4.5.2 Tab "CANopen module - I/O image.....	26
4.5.3 Tab "CANopen module - Status".....	27
4.5.4 Tab "CANopen module - Information".....	27

Table of Contents

	Page
5 CAN bus slave configuration editor.....	29
5.1 CANopen slave node.....	30
5.1.1 Tab "CANopen slave - General".....	30
5.1.2 Tab "CANopen slave - CAN bus parameters".....	31
5.1.3 Tab "CANopen slave - Status".....	31
5.1.4 Tab "CANopen slave - Information".....	31
5.2 CANopen local device.....	32
5.2.1 Tab "CANopen local device - General".....	32
5.2.2 Tab "CANopen local device - Parameters".....	34
5.2.3 Tab "CANopen local device - I/O image".....	34
5.2.4 Tab "CANopen local device - Status".....	34
5.2.5 Tab "CANopen local device - Information".....	34
6 Device database.....	35
6.1 Device database, overview	35
6.2 Managing devices.....	35
6.2.1 Device database, dialog	35
6.2.2 Adding devices.....	36
6.2.3 Removing devices	37
7 CAA_CiA405.library.....	39
7.1 CAA_CiA405.Library, general.....	39
7.2 Data types.....	40
7.2.1 Data types of the CAA_CiA405.library.....	40
7.2.2 DEVICE_STATE.....	40
7.2.3 SDO_MODE.....	41
7.2.4 TRANSITION_STATE.....	42
7.2.5 CANOPEN_KERNEL_ERROR.....	42
7.2.6 EMCY_ERROR.....	42
7.2.7 DEVICE.....	42
7.2.8 SDO_ERROR.....	43
7.3 Global constants.....	43
7.3.1 CANOPEN_KERNEL_ERROR_CODES.....	43
7.4 Function blocks.....	44
7.4.1 Function blocks of the CAA_CiA405.library.....	44
7.4.2 General aspects to structure function blocks.....	44
Data types compared to the data length.....	44
Timeout.....	45
Additional parameters.....	45
Control mechanism.....	46
7.4.3 CIA405_CIA405BASE (behavior model).....	48
CIA405BASE.....	48
7.4.4 Function blocks "Reading and writing SDO".....	49
SDO_WRITE4.....	49
SDO_WRITE_DATA.....	52

Table of Contents

	Page
SDO_READ4.....	53
SDO_READ_DATA.....	55
Obsolete function blocks.....	57
SDO_WRITE.....	57
SDO_READ.....	58
7.4.5 Function block "ID of own node" (GET_Local_NODE_ID).....	59
GET_LOCAL_NODE_ID.....	59
7.4.6 Function blocks "Querying state" (GET_STATE).....	60
GET_STATE.....	60
GET_CANOPEN_KERNEL_STATE.....	62
7.4.7 Function blocks "Network management" (NMT).....	63
NMT.....	63
RECV_EMCY_DEV.....	65
RECV_EMCY.....	67
 8 Service and support.....	 71
 Index.....	 73

1 About this documentation

1.1 Validity of the documentation

Target group This documentation provides information on the CAN bus connection IndraMotion MLC (IndraControl XM2x).

Required qualification: Individual who is able to assess the tasks assigned and to identify possible safety risks owing to qualification in the subject, knowledge and experience. The individual should also be familiar with the standards and regulations.

Application phases *This documentation supports the user during the following phases:*

- Configuration of field bus components in IndraWorks
- Field bus application in the PLC user program.

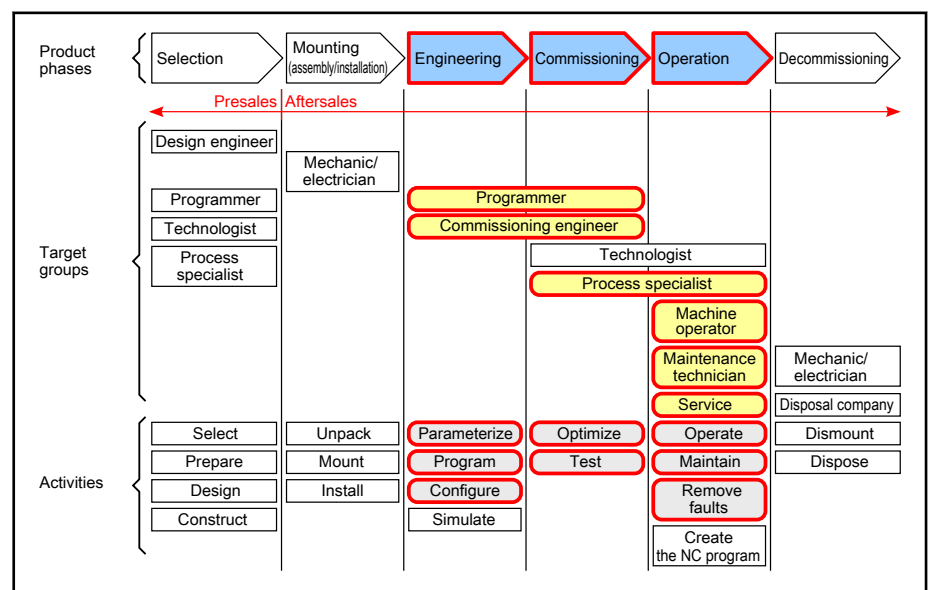


Fig. 1-1: Application phases

1.2 Documentation structure

The following is described

- Field bus characteristics, [Field bus features, page 9](#)
 - Terms and abbreviations
 - Field bus features
 - Transmission modes
- [Postinstalling CAN device descriptions, page 11](#)
- Project planning of the field buses:
 - Configuration of the master connection, [CAN bus configuration editor, page 13](#)
 - Configuration and parameterization of field devices
 - Commissioning and diagnostics
 - Configuration of the slave connection, [CAN bus slave configuration editor, page 29](#)
- Device database, installing additional devices and uninstalling devices that are not required, [Device database, page 35](#)

About this documentation

- CAN field bus support in the PLC user program, [CAA_CiA405.library](#), page 39

1.3 Use of the safety instructions

1.3.1 Structure of the safety instructions

The safety instructions are structured as follows:

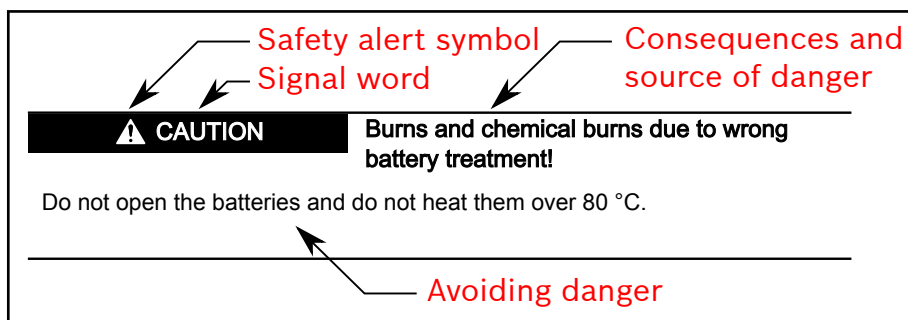


Fig. 1-2: Structure of the safety instructions

1.3.2 Explaining signal words and safety alert symbol

The safety instructions in this documentation contain specific signal words (danger, warning, caution, notice) and, if necessary, a safety alert symbol (according to ANSI Z535.6-2006).

The signal word is used to draw attention to the safety instruction and also provides information on the severity of the hazard.

The safety alert symbol (a triangle with an exclamation point), which precedes the signal words danger, warning and caution is used to alert the reader to personal injury hazards.

⚠ DANGER

In case of non-compliance with this safety instruction, death or serious injury **will** occur.

⚠ WARNING

In case of non-compliance with this safety instruction, death or serious injury **can** occur.

⚠ CAUTION

In case of non-compliance with this safety instruction, minor or moderate injury can occur.

NOTICE

In case of non-compliance with this safety instruction, material damage can occur.

1.3.3 Symbols used

Hints are represented as follows:



This is an information.

Tips are represented as follows:



This is a tip.

1.3.4 Signal graphic explanation on the device



Prior to the installation and commissioning of the device, refer to the device documentation.

1.4 Required and supplementing documentation

The documents listed below contain additional information on this subject.

IndraWorks 14VRS IndraLogic 2G PLC Programming System DOK-IWORKS-IL2GPRO*V14-APRS-EN-P, R911343571 This documentation describes the PLC programming tool IndraLogic 2G and its use. It includes the basic use, first steps, visualization, menu items and editors.
IndraLogic XLC IndraMotion MLC 14VRS Functional Description DOK-XLCMLC-FUNC***V14-APRS-EN-P, R911341700 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.
14VRS Basic Libraries IndraLogic 2G DOK-IL*2G*-BASLIB**V14-LIRS-EN-P, R911343920 This documentation describes the system-comprehensive PLC libraries.
IndraWorks 14VRS Field Buses DOK-IWORKS-FB*****V14-APRS-EN-P, R911341485 This documentation describes the field bus and local periphery connections supported by the IndraLogic XLC, IndraMotion MLC and IndraMotion MTX systems. This documentation focuses on the configuration, parameterization, commissioning and diagnostics of the different periphery connections. It is the basis for the online help.
IndraControl XFE 01.1 Extension Module for XM2x Operating Instructions DOK-CONTRL-XFE**EXTMOD-IT02-EN-P, R911345570 Mounting and operating instructions for the extension modules of the XM2x
IndraControl XFE01.1-FB-20 Extension Module CAN for XM2x Operating Instructions DOK-CONTRL-XFE**xxx, R911xxxxxxx Mounting and operating instructions for the extension modules of the XM2x Being prepared

About this documentation

1.5 Terms and abbreviations

Term	Explanation
PDO	Process Data Object: Inputs/outputs, process data
SDO	Service Data Object: Configuration data
COB-ID	CAN object identifiers, identification of the communication object as synchronization message
CAN ID	CAN identifier, 11-bit CAN message identifier, ID at the beginning of each CAN message on the bus
NMT	Network management
SYNC	Synchronization object (SYNC) protocol

Tab. 1-1: *Terms and abbreviations used. For more terms and abbreviations, refer to the standard.*

Modular and compact field bus connections

Bus coupler

Field bus connection (slave),

Compact

- For a compact (non-modular) slave, the module structure is specified. The modules below the "Slave" object node are already present. No more modules are available in the library

Modular

- The module structure of the slave is variable. The modules (terminals) can be arranged individually, but according to the device mounting specifications.

Module, terminal

Smallest I/O unit to assemble a field bus connection.

Device description files

The device description file describes the properties of the field bus device. This file contains all data significant for the engineering as well as for the data exchange with the I/O device.

The following file format is specified for the CAN bus:

EDS file

These are device descriptions used by IndraWorks to make the CAN field bus devices known (EDS: Electronic Data Sheet).

2 Field bus features

The implementation of a CANopen interface (master and slave) is described for an IndraMotion MLC (IndraControl XM) via extension modules.

The software and firmware components are included in the IndraWorks Engineering Suite or the MLC firmware.

The CAN components are not visible in the standard installation of the IndraWorks Engineering Suite. The CAN functionality (CAN extension module and configuration dialogs) is provided by postinstalling the device descriptions for the CAN extension module as well the CAN master and slave connection for the user.

Up to three CAN extension modules can be operated at the MLC controls.

Two CAN interfaces are provided on each CAN extension module. The interfaces can be configured either as CANopen master or as CANopen slave.

Thus, up to six CAN interfaces can be provided at the XM2x.



In IndraWorks 14VRS, CAN is supported at the control type IndraControl XM based on an extension module, in contrast to IndraWorks 13VRS, in which CAN is supported at the control IndraControl Lx5 based on a function module.

Coupling possible between an XM (14VRS) and an Lx5 (13VRS).

General characteristics of the CANopen implementation

- CANopen stack acc. to CANopen communication profile (CiA 301)
- State machine acc. to CiA 302
- **Error control:** Heartbeat/node guarding
- Display of diagnostic data and emergency messages
- Sync producing/consuming

Characteristics of the CANopen master implementation

- Auto PDO mapping for modular slaves acc. to CiA 401 (modular-dependent)
- Configurable stack response
- Function blocks of the CiA405 (CAA_CIA405.lib)
- Support of multiple SDO channels
- Import of CANopen EDS and DCF files
- Configurable stack response

Properties of the CANopen slave implementation

- Static PDO mapping
- Definition of I/O areas with automatic generation of the PDO mapping
- Definition of SDO parameter areas
- API to affect the NMT state machine
- Generation of a CiA 306-compliant EDS file

3 Postinstalling CAN device descriptions

To provide a CAN extension module as well as the CAN configuration dialogs (editors) to the user, postinstall the following device descriptions in IndraWorks:

1. EM_CANXM.devdesc.xml
Device description for two CANopen interfaces on one XM extension module
2. XM2_CANopen_Master.devdesc.xml
Device description for the CANopen master on an XM extension module
3. XM2_CANopen_Slave.devdesc.xml
Device description for the CANopen slave on an XM extension module
4. EmptyCANModule.devdesc.xml
Device description for an empty module at the CAN node of the CAN function module
5. CanOpenManager.devdesc.xml
Device description for the CANopen master
6. CanLocalDevice.devdesc.xml
Device description for the CANopen slave

To postinstall the "device descriptions", go **Tools ► Device database** in the IndraWorks menu.

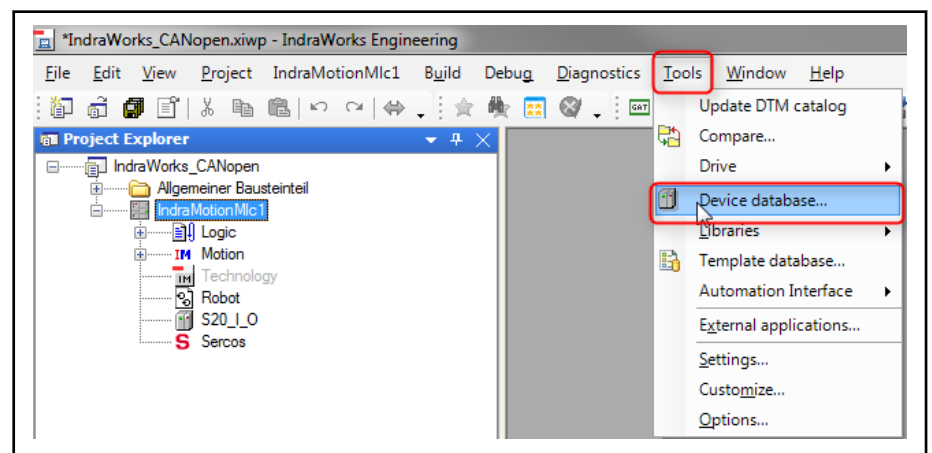


Fig. 3-1: "Device database" menu item

Press "Add device" to open a browser window in the following "Device database" dialog. Use this browser window to navigate to the storage location of the "device descriptions".

Select "Open" to add the listed *.devdesc.XML files to the IndraWorks device database. A multiple selection is possible.



Select the type "Device description files (*.devdesc.xml)" in the field "File type".

Postinstalling CAN device descriptions

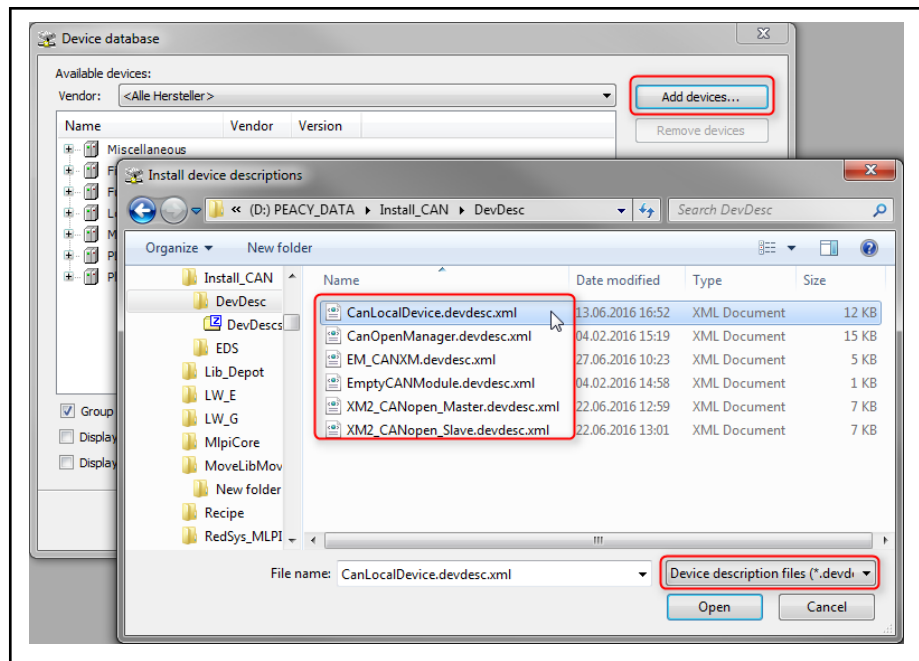


Fig. 3-2: "Install device descriptions" dialog

The devices are then displayed in the IndraWorks device database:

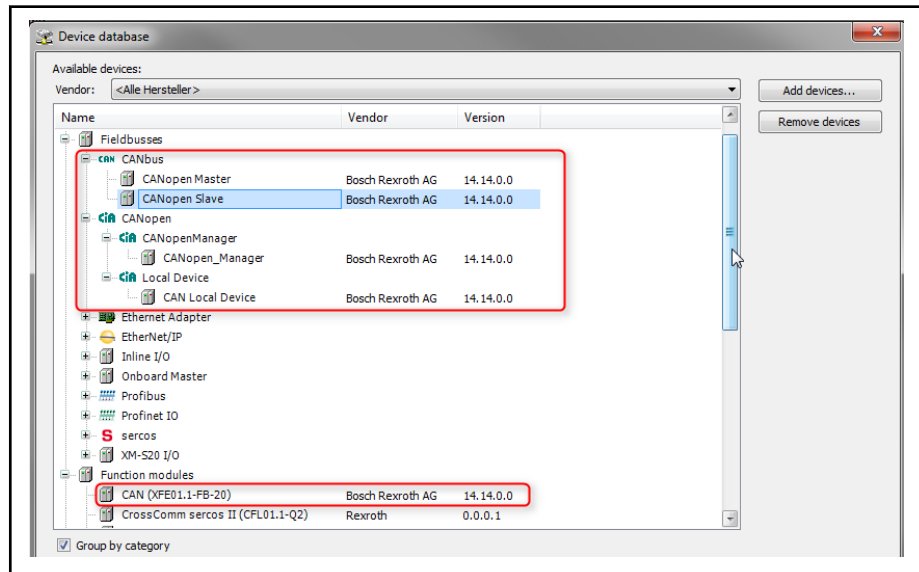


Fig. 3-3: "Device database" dialog

Importing CAN device descriptions via the IndraWorks project archive

The device descriptions to project a CAN application are saved - like the device descriptions for the field bus devices - to an IndraWorks project archive.

When restoring an IndraWorks project archive containing a CAN application, the required CAN device descriptions are automatically taken from the archive and imported into IndraWorks. The CAN components are then available in the IndraWorks installation. They do not have to be installed as described above.



Storing the CAN project archive on the control (e.g. MLC XM2). Thus, after restoring, a service technician is provided with the required device descriptions without installing them!

4 CAN bus configuration editor

4.1 CAN bus configuration editor, general

The CAN bus configuration editor provides dialogs to configure CANopen devices and modules acc. to the standard CANopen Draft Standard 301.

To map a CAN bus hardware in the project, the corresponding device objects have to be inserted into the Project Explorer and configured. Thus, special CAN bus configuration dialogs are provided in addition to the standard dialogs of the device editor.

The CAN bus devices and modules obtain a configuration defining their time- and error-related response during the data exchange.

Furthermore, the inputs and outputs of each module are mapped on **PDOs** (Process Data Objects) used to send and receive process data.

Subsequently, a list of **SDOs** (Service Data Objects) is created transferring all configuration data to the CAN bus target system using request and response services. Communication data and process data are managed in the "Object directory".

The CANopen module configuration is based on the device description files of the master devices and slave devices used and can be modified in the configuration dialogs by the user. Observe the recommendation to use the auto configuration mode for standard applications and modular devices ([Tab "CANopen remote device - General", page 20](#))!

Perform the following steps to create a CAN bus configuration in the Project Explorer of the project:

1. [Installing CANopen device files, page 13](#)
2. Mapping the CAN bus structure by inserting the respective device objects in the device window ([Basic hierarchy of a CAN bus configuration in IndraLogic/inserting the CAN bus device objects, page 13](#)).
3. Checking or modifying the configuration parameters in the dialogs of the device editors ([Configuration, page 14](#)).

Installing CANopen device files

To map the CAN bus hardware in the device tree of the project, install the respective device description files on the local system via the dialogs of the device database.

This refers to the description files (*.devdesc.xml) for the basic modules "CANbus" and "CANopen_Manager" and to the vendor-specific **EDS files** (Electronic Data Sheet) for the CANopen devices and modules used in the bus system.

These files contain parameter definitions and configuration options of a module ([Device database, page 35](#)).

Basic hierarchy of a CAN bus configuration in IndraLogic/inserting the CAN bus device objects

- To create a CAN configuration, first insert the extension module "CAN_XFE01_1_FB_20" below the control node.

Both communication channels of the CAN extension module can be set independently of one another as "CANopen_Master" or "CANopen_Slave" via the "Set device" function in the context menu of the respective node. The project planning of communication channel 1 as "CANopen master" is described in the following.

CAN bus configuration editor

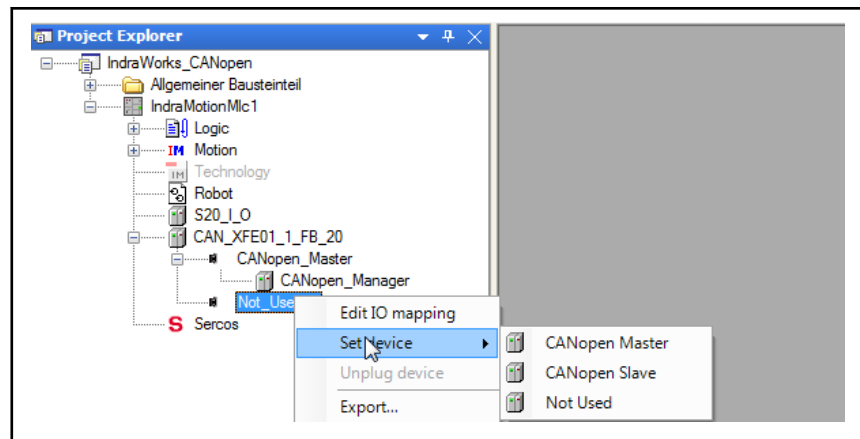


Fig. 4-1: "CANopen_Master_" in the Project Explorer, communication channel 1 configured as CANopen_Master.

- In the CANopen_Master node, the "baud rate" should be checked or adjusted for the data transfer to the bus and the number of the CAN network connected via the CAN bus interface (Tab "CANopen master - General", page 16).
- The mandatory object "CANopen_Manager" is available below a CANopen_Master entry.

This manager can be considered as CAN bus "master". It currently supports the CAN bus configuration exclusively by implicit functions. Thus, the user does not have to make any configurations.

The CANopen manager facilitates establishing the connection to the network especially by simplifying the PDO links between the individual modules.

- Below the CANopen manager, the individual CANopen device objects ("slaves") provided with the hardware, are inserted.

Two types are possible: Non-modular devices and modular devices.

Further CAN modules can be inserted below the modular devices and the automatic I/O image can be used.

Configuration After the CAN bus hardware has been mapped in the Project Explorer, use the configuration dialogs for the individual objects to check the default settings and to modify them if required:

For the object currently selected in the tree, double-click on the entry or select the command "Edit object" to display the respective configuration dialog in the window of the device editor.

The configuration options of a device or module always depend on the definitions in the device description file. Observe the general settings for the device editor in the "Options" dialog: Generic dialogs can for example be hidden (Tools ► Options ► IndraLogic 2G ► Device editor).

A CANopen configuration dialog is always called with the name of the respective module/device (as used in the Project Explorer) and can include the following subdialogs:

For the CANopen master object

- CANopen master node, page 15, (network number, baud rate)
- Tab "CANopen master - General", page 16, (network number, baud rate)

CAN bus configuration editor

- [Tab “CANopen master - CAN bus parameters”, page 16](#)
- [Tab “CANopen master - Status”, page 16](#)
- [Tab “CANopen master - Information”, page 16](#)

For the CANopen manager object:

- [CANopen manager node, page 17](#)
- [Tab “CANopen manager - General”, page 17](#)
- [Tab “CANopen manager - Parameters”, page 19](#)
- [Tab “CANopen manager - I/O image”, page 19](#)
- [Tab “CANopen manager - Status”, page 20](#)
- [Tab “CANopen manager - Information”, page 20](#)

For an XM_CAN device:

- [CANopen remote device, page 20](#) (main settings of the device)
- [Tab “CANopen remote device - General”, page 20](#) (main settings of the device)
- [Tab “CANopen remote device - PDOs”, page 22](#)
- [Tab “CANopen remote device - SDOs”, page 24](#)
- [Tab “CANopen remote device - Parameters”, page 25](#)
- [Tab “CANopen remote device - I/O image”, page 25](#)
- [Tab “CANopen remote device - Status”, page 26](#)
- [Tab “CANopen remote device - Information” remote device, page 26](#)

For a CANopen module:

- [Tab “CANopen module - Parameters”, page 26](#), (parameter as defined in the EDS file)
- [Tab “CANopen module - I/O image, page 26](#)
- [Tab “CANopen module - Status”, page 27](#)
- [Tab “CANopen module - Information”, page 27](#)

4.2 CANopen master node

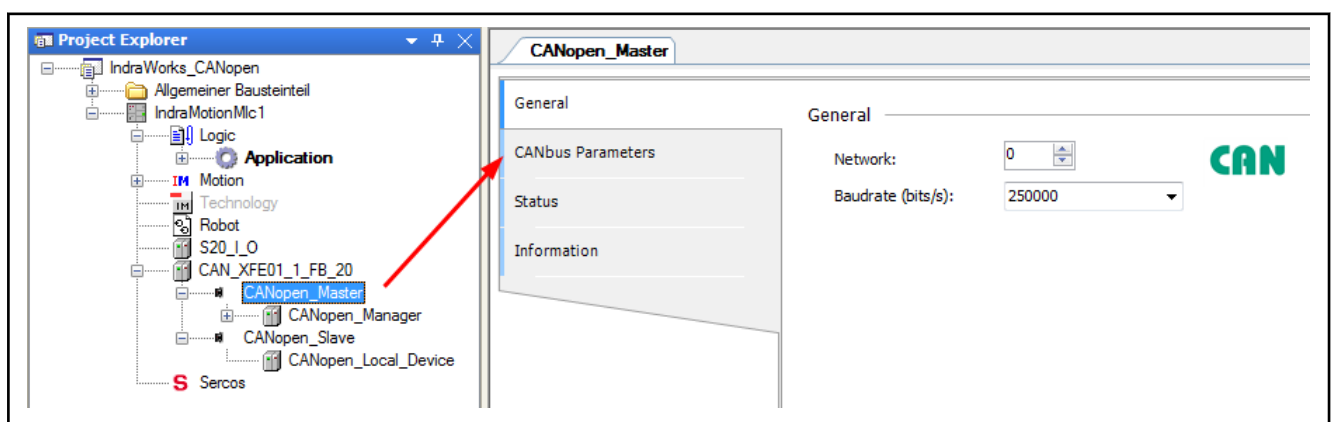


Fig. 4-2: "CANopen master" dialog

4.2.1 Tab “CANopen master - General”

The CAN bus object has to be entered into the Project Explorer as uppermost node of a CAN bus configuration ([CAN bus configuration editor, page 13](#)).

The following is defined in the respective configuration dialog:

- **Network:** Number of the CAN network to be connected via the CAN bus interface.

Permitted values: 0 to 63.



If multiple CAN extension modules and multiple CAN master and slave interfaces are projected at the XM2x, an individual, unique network number has to be set for each CAN interface.

- **Baud rate (bits/s):** Baud rate in bits per second for the data transfer to the bus.

The default value is taken from the device description file (*.devdesc) of the CAN bus device.

The following baud rates can be set:

10000, 20000, 50000, 100000, 125000, 250000, 500000, 800000 und 1000000.

The unit is bits/s (bits per second). The baud rate is saved in KBit/s in the configuration data.

4.2.2 Tab “CANopen master - CAN bus parameters”

Currently, this configuration dialog for a CAN bus node has no function, as the device description does not include any configuration parameters except for the baud rate and the network number (NetID).

4.2.3 Tab “CANopen master - Status”

This dialog of a CANopen device or module configuration displays status information (e.g. "Running", "Stopped"), specific diagnostic messages of the device, information on the card used and on the internal bus system ([CAN bus configuration editor, page 13](#)).

4.2.4 Tab “CANopen master - Information”

This dialog of a CANopen device or module configuration displays information on the device or on the module that can be read from the device description.

Possible specifications: Name, Vendor, Type, Version number, Categories, Ordering number, Image ([CAN bus configuration editor, page 13](#)).

4.3 CANopen manager node

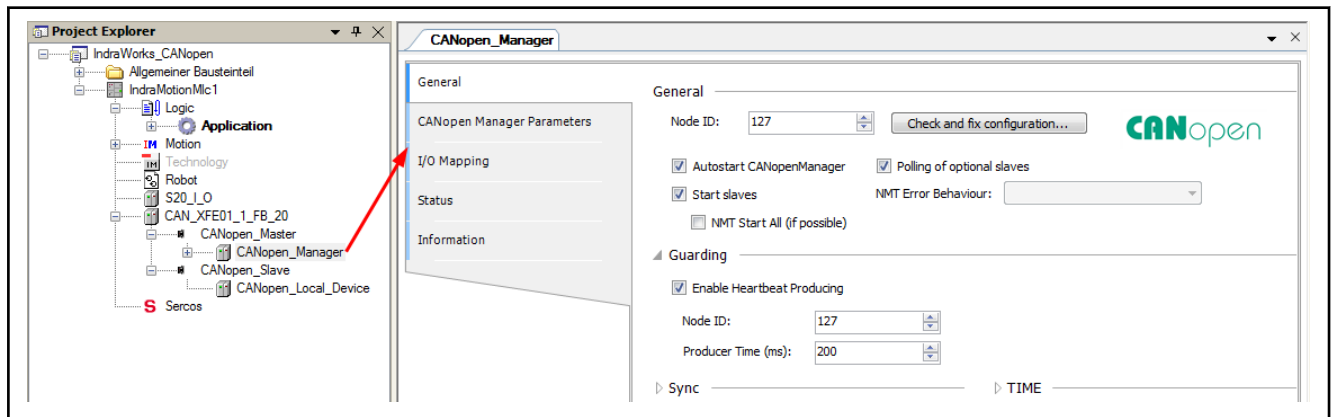


Fig. 4-3: "CANopen manager" dialog

4.3.1 Tab "CANopen manager - General"

General information

Node ID	The node number uniquely identifies the CANopen manager and corresponds to the number set on the module (value between 1 and 127).
Check and fix configuration	Opens the dialog with the same name. For details, refer to the following.
Autostart CANopenManager	☒ : The CANopen manager starts automatically (changes to the mode OPERATION-AL) after all mandatory slaves are ready. ☐ : Start the CANopen manager from the application. Use the NMT function block CiA405. <i>Tip:</i> If the CANopen manager is not in the mode OPERATIONAL, no PDOs are sent (outputs updated).
Polling of optional slaves	☒ : If a slave does not send any response during the boot sequence, the CANopen manager queries it each second until a positive response is sent. Permanent polling of the slaves causes an increase in the bus cycle time which can affect the application (especially for Motion applications). To avoid this behavior, disable polling. If polling is disabled, a slave is only detected again if it sends a bootup telegram.
Start slaves	☒ : The CANopen manager is responsible for the start of the slaves. ☐ : Start the slaves from the application. Use the NMT function block CiA405.
NMT start all (if possible)	If the "Start slaves" option is enabled, the CANopen manager starts all slaves with an "NMT start all" command. "NMT start all" is not executed until optional slaves are ready for the start. In this case, the CANopen manager starts every slave separately. An "NMT start all" can only be ensured in a project without optional slaves.
NMT error behavior	• Restart slaves. If an error occurs during slave monitoring (NMT error event), the slave is automatically restarted by the stack (NMT reset + SDO configuration + NMT start). • Stop slaves. If an error occurs during the slave monitoring (NMT error event), the slave is stopped. Then, reset the slave via the application using the CiA405 NMT function blocks.

CAN bus configuration editor

Guarding

Working with heartbeats is an alternative mechanism for monitoring: In contrast to the node guarding functionality, it can execute the master node as well as the slave node. Normally, the master sends heartbeats to the slaves.	
Enable "Heartbeat producing"	☒ : Master sends heartbeats. The time interval is defined in the Producer time . If attaching slaves with heartbeat functionality, a heartbeat consuming entry is created for the master at the slave. Node ID as well as 1.5 times the heartbeat interval of the master are applied. ☐ : Node guarding is enabled for the slaves. The settings from the EDS file of the slaves are used. If no useful values are defined there, default values are entered. Observe that a CANopen slave device can also be configured as heartbeat generator.
Node ID	Unique ID (1-27) of the heartbeat generator on the bus.
Producer time (ms)	Interval length of subsequent heartbeats in milliseconds.

Sync

Enable "Sync producing":	☒ : (Disabled by default). CANopen manager sends sync telegrams. The synchronous PDOs are directly sent after the synchronization message.
COB ID (hex)	CAN ID of the SYNC message. Possible value range: [1, 2047].
Cycle period (µs)	Interval length (in microseconds) after which the synchronization message is sent.
Window length (µs)	Length of the time slot for synchronous PDOs in microseconds.
Enable sync consuming:	☒ : (Disabled by default). Another device has to generate the synchronization messages received by the CANopen manager.



When enabling the sync producing for the CANopen manager, the option "Enable sync producing" may not be selected for all other bus devices.

TIME

Enable TIME producing:	☒ : (Disabled by default). CANopen manager sends TIME messages.
COB ID (hex)	Communication Object Identifier labeling the time stamp of the message. Default values : [0, 2047], presetting 16#100.
Producer time (ms):	Interval in milliseconds after which the time stamp is sent. This value has to be a multiple of the task cycle time. Possible values [0, 65535].



The runtime system has to support "high resolution timestamp". Otherwise, an error message is output.

"Check and fix configuration" dialog

If there are multiple devices below the CANopen manager, error messages might be displayed reporting node IDs or invalid COB IDs assigned multiple times.

Press the button "Check and fix configuration" to open a dialog by resolving these conflicts.

For conflicts with node IDs or PDO-COB IDs, open a dialog with detailed information using the command "Edit conflicts...".

Node ID and COB ID conflicts

Duplicate node number	List of all devices with identical IDs. New node numbers can be entered for the respective devices into the fields of the "Node ID" column.
Incorrectly or double assigned PDO-COB IDs	The COB IDs automatically created from the device description files can be invalid. All incorrect entries are listed with the respective node names, node IDs and index. There are three options to correct the invalid COB IDs: • Correction of the formula to calculate the COB IDs to ensure a valid COB ID. Change the formula in the corresponding table field. • Click on the respective button to apply the automatic suggestion for the COB ID. • Click on the button "Use suggested COB ID" to accept all automatically created suggestions.

Corrected entries are automatically removed from the displayed lists.

Solve timing problems automatically using the command "Automatic repair". The command adjusts all timing values to respective compatible values (time should be a multiple of the task time).

4.3.2 Tab "CANopen manager - Parameters"

This tab shows the parameters of the CANopen manager in a table.

A part of the parameters can be edited (for editing, click on the "Value" field). The font color of these parameters is black.

The font color of the fields "Value" and "Type" are grey, only the current values are displayed.

4.3.3 Tab "CANopen manager - I/O image"

The tab shows the name of the automatically created instance of the master under "IEC objects" in the "Variable" column.

Use the application to access this instance. Thus, the target device can be restarted, stopped or its status can be checked using an application.

Please also refer to the description of the general "I/O image" dialog.

Bus cycle options

Bus cycle task	The selection list provides all tasks defined in the task configuration of the active application, such as "MainTask". For Use parent bus cycle setting, IndraLogic uses the settings of the father node. If no task is defined here, the task set in the PLC settings applies. If no task has been set here as well, the task with the highest priority is used as bus cycle task.
----------------	--

Tab "<Device> I/O image"

In this tab of the generic device editor, map the input, output and memory addressed of the control device to the variables of the application. This is used to create the so-called "I/O image" (mapping).

IEC objects

	This section of the dialog is only displayed if an instance of the device object is implicitly applied that can be accessed in the application. This information can for example be used to restart a bus or to query information. Whether such an instance is available and how it can be used depends on the device type.
--	--

CAN bus configuration editor

4.3.4 Tab “CANopen manager - Status”

This dialog of a CANopen device or module configuration displays status information (e.g "Running", "Stopped"), specific diagnostic messages from the device, information on the card used and on the internal bus system ([CAN bus configuration editor, page 13](#)).

4.3.5 Tab “CANopen manager - Information”

This dialog of a CANopen device or module configuration displays information on the device or the module read from the device description ([CAN bus configuration editor, page 13](#)).
Possible specifications: Name, Vendor, Type, Version number, Categories, Ordering number, Image.

4.4 CANopen remote device

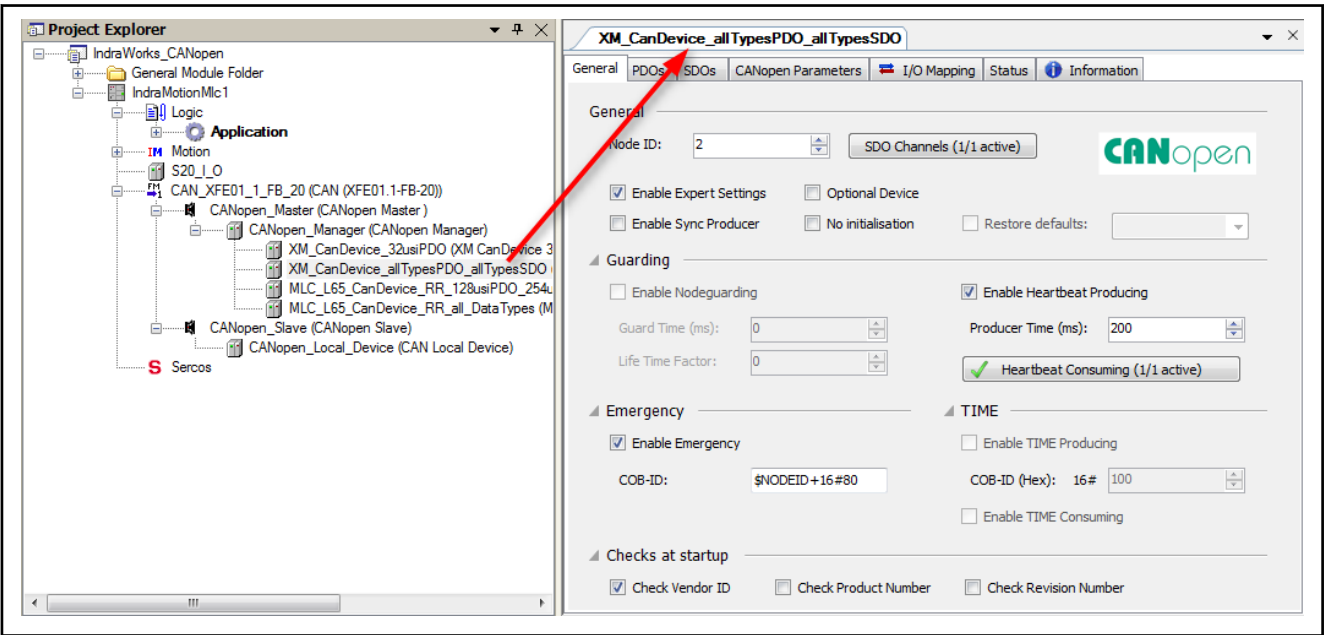


Fig. 4-4: CANopen remote device

4.4.1 Tab “CANopen remote device - General”

... In this dialog of a CANopen remote device (slave), the general setting of the CANopen slave are defined.

General information

Node ID	The node number uniquely identifies the CAN module. It corresponds to the number (value between 1 and 127) set on the module (hardware). Enter the node ID as decimal.
Expert settings	☒ : All settings specified for the device by the device description (EDS file) are displayed.
SDO channels(...)	Press to button to open a dialog to enable the SDO channels predefined in the EDS file. Service Data Objects (SDOs) allow to access all entries of the CANopen object directory. An SDO creates a peer-to-peer communication channel between two devices (SDO server or client channel).

CAN bus configuration editor

Optional device	☒ : The slave is optional and not required to start the CAN network.
Sync generation	Only available if the option "Enable Sync generation" of the CANopen manager is displayed. ☒: The I/O transfer is synchronized on the bus. The slave operates as sync generator. Define the parameters of the sync interval in the settings of the CANopen manager.
No initialization	This option is not intended for non-configurable slaves starting with an already valid configuration ☒: The master does not send any configuration SDOs and no NMT starting command to the slave. PDO communication and monitoring (heartbeat, node guarding) take place if enabled in the configurator. If the slave does not start automatically, an NMT starting command can be sent by the user using the CiA405 NMT function block.
Factory settings	If this option is available depends on the content of the device description file ☒: Enabled by default. The slave nodes are reset to the default parameters before loading the configuration to the device or when the slave is configured. Which parameters can be set depends on the device. The concrete specification is given when selecting a subindex from the drop-down list. • Sub:001: All parameters are reset. • Sub:002: Communication parameters (index 1000h - 1FFFh vendor-specific communication parameters) are reset. • Sub:003: Application parameters (index 6000h - 9FFFh vendor-specific application parameters) are reset. • Sub:004 - Sub:127: Vendor-specific individual selection of parameters is reset • Sub:128 - Sub:254: Reserved for future use
Auto config. PDO mapping	This option is only available for modular devices. ☒: The PDO mapping is automatically created from the definitions in the device description and cannot be changed in both the mapping dialogs. If the automatically created mapping does not match your application, disable the option and configure the mapping manually. Enabling this option is recommended for the standard applications.

Node guarding

Node guarding is an outdated monitoring mechanism and should not be used anymore, as it uses RTR frames. If possible, always use heartbeat. In individual cases, e.g. in case of older slaves, only node guarding can sometimes be used.	
Enable node guarding	☒: The CANopen manager sends a message to the slave in the Interval guard time (ms). If the slave does not report with the specified Guard COB ID (Communication Object Identifier), the CANopen manager sends this message as defined under Life Time Factor or until the slave responds. If the slave does not respond, it is labeled as "not available".
Guard time (ms)	Interval of the message sending (200 ms by default)
Life time factor	If the slave does not respond, Life Time Factor is multiplied with Guard Time after the period and a node guarding error is determined.
Enable heartbeat producing	☒ : The module sends heartbeats in the time intervals specified at the producer time (ms).

CAN bus configuration editor

Producer time (ms)	Set to 200 by default if not specified or if the specification is 0 in the device description file.
Heartbeat consuming (...)	Heartbeat consuming (...): Opens a "Heartbeat consuming" properties dialog. Enable the slaves to be monitored. The number of possible slaves to be monitored is described in the EDS file. Enable the "Activate" field and enter the node ID of the slaves and the value in the field "Heartbeat time in milliseconds". The slave then monitors the heartbeats sent by the respective slaves (defined by the node ID). If no heartbeats are received anymore, the slave disables the I/Os. If a slave is monitored, there is a green checkmark on the button "Heartbeat consuming". <i>Note:</i> When inserting a device with heartbeat functionality, its heartbeat settings are adjusted automatically to those of the master (CANopen manager).

Emergency

Enable emergency	☒ : In case of internal errors, the slave sends emergency messages with a unique COB ID. These messages can be read using the function blocks of the CiA405 library. (RECV_EMCY_DEF, RECV_EMCY).
COB-ID	CAN ID of the EMCY message. Possible value range: [1, 2047].

TIME

The availability of this function depends on the device description.	
Enable TIME producing	☒ : Device sends TIME messages.
COB ID (hex)	Communication Object Identifier labeling the time stamp of the message.
Enable TIME consuming	☒ : Device uses TIME messages.

Checks upon start

The respective information is retrieved from the firmware of the CANopen slave (object 0x1018 identity object) and compared to the information from the EDS file. If different, the configuration is stopped and the slaves are not started.	
Vendor ID	☒ : Check the vendor ID upon start
Product number	☒ : Check the product number upon start
Revision number	☒ : Check revision number upon start

4.4.2 Tab "CANopen remote device - PDOs"

This dialog is only available in the device editor of a CANopen slaves of the version V3.5.6.0 or higher. It shows all PDOs and their default settings. New objects can be added or existing objects can be edited or deleted in this dialog.

The PDOs, which the slave receives from the master, are shown in the left section.

The PDOs, which the slave sends to the master, are shown in the right section.

CAN bus configuration editor

Add PDO	Opens the dialog "Select PDO" that displays all available PDOs. In the dialog, select the PDOs to be added to the "receiving PDOs" or "sending PDOs".
Add mapping	Opens the "Select element from object directory" dialog. It displays objects that can be added to the PDO mapping.
Edit	After selecting a PDO, the dialog "PDO properties" open. After selecting a PDO mapping, the dialog "Select element from object directory" opens.
Delete	Deletes the selected objects from the list
Up	Moves up the selected object by one line
Down	Moves down the selected object by one line

"PDO properties" dialog

COB-ID	Each PDO message has to be provided with a COB ID (Communication Object Identifier). Explicit values (example: 16#201) and formulas (example: \$NODEID+16#200) can be entered.
RTR	Remote Transmission Request. This option is only available for sending PDOs. ☒ : The PDO can be queried externally via an RTR frame.
Inhibit time (x 100µs)	This field can only be edited if the device supports the functionality. The inhibit time is the minimum period between two messages of this PDO. Use this setting to avoid that PDOs transferred upon value change, are sent too often. Default: "0". Possible values: 0-65535.
Transmission type	• Acyclic - Synchronous: The PDO is synchronously transferred upon change, but not periodically (default). • Cyclic - Synchronous: The PDO transfers each nth sync. • Synchronous – Only RTR: Only available for sending PDOs. The PDO is updated after a synchronization message, but not sent. Only transferred if requested explicitly (Remote Transmission Request) • Asynchronous – Only RTR: Only available for sending PDOs. The PDO is updated and transferred only if requested explicitly (Remote Transmission Request) • Asynchronous – Vendor-specific: The PDO is only transferred after certain events. • Asynchronous– Device profile: The PDO is transferred as per the CiA device profile.
Number of syncs	For the transmission type Cyclic - Synchronous . Enter the interval after which the PDOs are sent. The value is a multiple of the cycle period (µs) of the CANopen manager. Default: 1. Possible values: 1-240. Example: Number of syncs = 4, cycle period = 1000 µs per transmission interval = 4000 µs.
Event time (x 1ms)	Only for the transmission types Asynchronous - Vendor-specific and Asynchronous - Device profile . This field can only be edited if the device supports the functionality. Enter the time between two transfers of the PDO (in ms). Default: "0". Possible values: 0-65535.
Processed by CANopen manager	☒ : Default setting. ☐ : CANopen manager does not process the PDO anymore. It is not sent or received anymore.

CAN bus configuration editor

"Select element from object directory" dialog

For modular slaves, disable the option "Autoconfig. PDO mapping" to manually configure the mapping.

The table shows all object directory entries from the EDS file of the device. For receiving PDOs, IndraLogic only provides objects with write access (flag = w) and for sending PDOs, objects with read access are provided.

Name	COB ID of the PDO or the name of the "mapped" object as used in the device description and in the object directory.
Index	Object index
Subindex	Subindex of the object
Access type	• RW: Read and write • RO: Read-only • WO: Write-only • RWW: It can be written on and read via SDO; written on via PDO (==> RPDO, output seen from the master, input seen from the slave). • RWR: It can be written on and read via SDO; read via PDO (==> TPDO, input seen from the master, output seen from the slave). • CONST = Constant
Data type	Data type of the object
Default value	Default value of the object
Bit length	Object length

4.4.3 Tab "CANopen remote device - SDOs"

While initializing the CAN bus, IndraLogic transfers the current configuration settings to the CAN modules using Service Data Objects (SDO).

Configure the SDOs in this tab. They specify the sequence of the objects to be transferred and the procedure in case of a transfer error.

The sequence of the objects in this list corresponds to the transfer sequence of the SDOs to the module.



If the option "Expert settings" has not been enabled for the current device, only the user-defined SDOs are displayed.

Add SDO	Opens the "Select element from object directory" dialog. The chosen object is inserted after the selected object.
Edit	Opens the "Select element from object directory" dialog and highlights the respective object. The parameters of the object can be changed or the object can be replaced by another one.
Delete	Deletes the selected objects from the list.
Top	Moves up the selected object by one line
Down	Moves down the selected objects by one line
Abort in case of error	☒ : If an error is detected for this SDO, the stack stops the configuration phase of the current slave. The slave remains in the status PREOPERATIONAL.
Jump to line in case of error	☒ : The transfer is continued with the SDO specified in the column "Next line".

CAN bus configuration editor

Next line	Line number, at which the processing continues in case of error.
SDO timeout (ms)	Timeout for SDO transfer. If the slave does not respond to the SDO request during this period, the transfer is aborted with timeout.
Generate all SDOs	☒ : Creates an SDO for all writable objects from the index 16#2000 for which a default value is specified in the EDS. The option should only be used by experts and has to be disabled by default.
Write complete PDO configuration	☒ : This option forces to write all PDO configuration objects. This ensures that the settings in the project correspond to those of the slaves. ☐ : PDOs are not disabled explicitly. The prerequisite is that the "Factory settings" option is enabled in the general settings of the slave and that the PDOs are also disabled in the EDS. If the default values in the EDS do not correspond to the "factory settings" of the slave firmware, this procedure can cause problems. In this case, select this option.

"Select element from object directory" dialog

This table shows all object directory entries from the EDC file of the device not bigger than four bytes that can be written via SDO. Before adding an SDO to the SDO dialog for selection, its parameters can be changed in the fields below the table. Thus, and SDO that is not written on the EDS file, can also be created by entering a new index/subindex value.

Name	COB ID of the PDO or the name of the "mapped" object as used in the device description and in the object directory.
Index	Object index
Subindex	Subindex of the object
Access type	• RW: Read and write • RO: Read-only • WO: Write-only • RWW: It can be written on and read via SDO; written on via PDO (==> RPDO, output seen from the master, input seen from the slave). • RWR: It can be written on and read via SDO; can be read via PDO (==> TPDO, input seen from the master, output seen from the slave).
Data type	Data type of the object
Default value	Default value of the object
Bit length	Object length
Value	

4.4.4 Tab "CANopen remote device - Parameters"

This tab shows the parameters of the CANopen remote device in a table.

A part of the parameters can be edited (for editing, click on the "Value" field). The font color of these parameters is black.

The font color of the fields "Value" and "Type" are grey, only the current values are displayed.

4.4.5 Tab "CANopen remote device - I/O image"

The tab shows the name of the automatically created instance of the slave under IEC objects in the "Variable" column. Thus, access this instance via the

CAN bus configuration editor

application. Thus, the slave can be restarted, stopped or its status can be checked using an application.

In the tab, project variables are mapped to the inputs and outputs of the slave.



For a modular slave, no project variables can be mapped on I/Os in this dialog. This is executed in the editor of the subordinate modules.

Please also refer to the description of the general “I/O image” dialog.

Reset mapping	IndraLogic resets the mapping settings to the default values defined in the device description file.
Update variables	Specification for the device object regarding I/O variable update. The default value is defined in the device description file: Use parent device setting: Update acc. to the settings of the parent device. Enabled 1 (use bus cycle task if not used in any task): IndraLogic updates the I/O variables in the bus cycle task if not used in any other task. Enabled 2 (always in bus cycle task): IndraLogic updates all variables in each cycle of the bus cycle task irrespective of whether they are used or mapped on an input or output channel.

4.4.6 Tab “CANopen remote device - Status”

This dialog of a CANopen device or module configuration displays status information (e.g. "Running", "Stopped") and specific diagnostic messages from the device and information on the card used and the internal bus system.

4.4.7 Tab “CANopen remote device - Information”remote device

This dialog of a CANopen device or module configuration displays information on the device or the module read from the device description.

Possible specifications: Name, Vendor, Type, Version number, Categories, Ordering number, Image.

4.5 CANopen module

CANopen modules are modular components that can be inserted below a CANopen remote device.

4.5.1 Tab “CANopen module - Parameters”

The respective dialog is only available in the device editor of the CANopen manager if the option "Show generic configuration editors" was enabled (Tools ► Options ► IndraLogic 2G ► Device editor).

The dialog corresponds to the "Parameters" dialog also provided for other device modules by the device editor.

In this dialog of a CANopen configuration of a CANopen module, the parameters of the module from the module description can be displayed and edited. It is a standard configuration dialog of a device editor to parameterize a device or a module.

4.5.2 Tab “CANopen module - I/O image

The tab shows the name of the automatically created instance of the module under "IEC objects" in the "Variable" column. Thus, access the instance via

CAN bus configuration editor

the application. Thus, the module can be restarted, stopped or its status can be checked using an application.

In the tab, project variables are mapped to the inputs and outputs of the module.

Please also refer to the description of the general "I/O image" dialog.

Reset mapping	IndraLogic resets the mapping settings to the default values defined in the device description file.
Update variables	Specification for the device object regarding I/O variable update. The default value is defined in the device description file: Use parent device setting: Update acc. to the settings of the parent device. Enabled 1 (use bus cycle task if not used in any task): IndraLogic updates the I/O variables in the bus cycle task if not used in any other task. Enabled 2 (always in bus cycle task): IndraLogic updates all variables in each cycle of the bus cycle task irrespective of whether they are used or mapped on an input or an output channel.

4.5.3 Tab "CANopen module - Status"

This dialog of a CANopen device or module configuration displays status information (e.g. "Running", "Stopped") and specific diagnostic messages from the device and information on the card used and the internal bus system.

4.5.4 Tab "CANopen module - Information"

This dialog of a CANopen device or module configuration displays information on the device or the module read from the device description. Possible specifications: Name, Vendor, Type, Version number, Categories, Ordering number, Image.

5 CAN bus slave configuration editor

The CAN bus slave configuration editor provides dialogs to configure a local CAN bus slave device that can be created in the device tree below a CAN extension module. It allows to use the IndraLogic runtime system as CANopen slave.

A slave description file *.eds can be directly created from the parameters set in the configuration editor. This file can then be used in any standard CANopen configurator, such as IndraLogic.

For the standard CAN bus device configuration, refer to: [CAN bus configuration editor, page 13](#)

As for the other CAN modules, specific dialogs are also added to the standard device editor to configure the "local" CAN bus slave described here.

Supported functionalities

The CANopen slave device, which can be configured here, supports the following functionalities:

- SDO server functionality (only "Expedited mode")
- NMT client
- Guarding: Heartbeat producing, heartbeat consuming (only one "consumer" entry), node-guarding
- Sync producing/consuming
- Emergency (EMCY)
- Synchronous and asynchronous PDOs (all transmission types specified in CiA301)
- Inhibit time, event timer for PDOs
- Static PDO mapping
- Max. 16 TxPDOs and 16 RxPDOs
- Reading and writing parameters via SDO

The following is not supported:

- TIME
- Dynamic PDO mapping

Hierarchy of the CAN bus configuration in IndraLogic/inserting the CAN bus slave device

The basic node, i.e. the uppermost entry in the CAN bus configuration tree has to be a CAN bus object. See [CAN bus configuration editor, page 13](#)

To create a CAN configuration, first insert the extension module "CAN_XFE01_1_FB_20" below the control node.

The local CANopen slave device is created (as the CANopen master) via "Set device" in the context menu of the node of the CAN extension module (in this example, communication channel 2).

CAN bus slave configuration editor

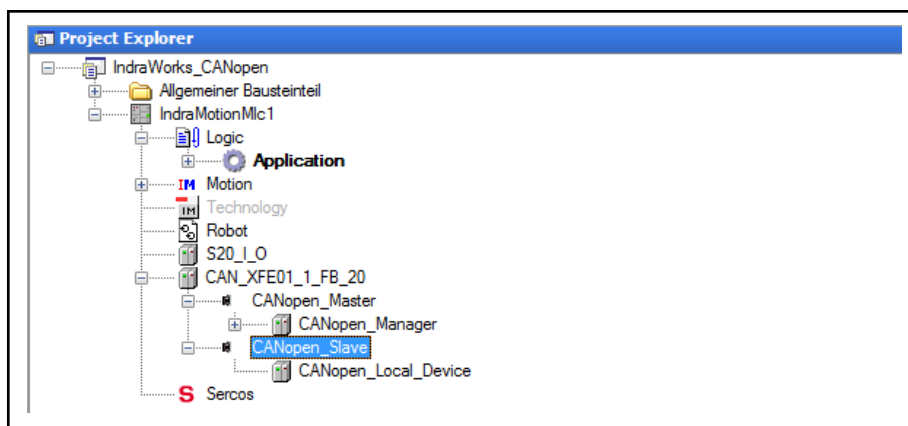


Fig. 5-1: Example: CAN bus configuration tree

Configuration dialogs

For the CANopen slave object

- CANopen slave node, page 30, (network number, baud rate)
- Tab “CANopen slave - General”, page 30, (network number, baud rate)
- Tab “CANopen slave - CAN bus parameters”, page 31
- Tab “CANopen slave - Status”, page 31
- Tab “CANopen slave - Information”, page 31

Double-click on the "CAN_Local_Device" object in the Project Explorer to open the configuration editor providing the following subdialogs to set the slave parameters:

- CANopen local device, page 32
- Tab “CANopen local device - General”, page 32
- Tab “CANopen local device - Parameters”, page 34
- Tab “CANopen local device - I/O image”, page 34
- Tab “CANopen local device - Status”, page 34
- Tab “CANopen local device - Information”, page 34

5.1 CANopen slave node

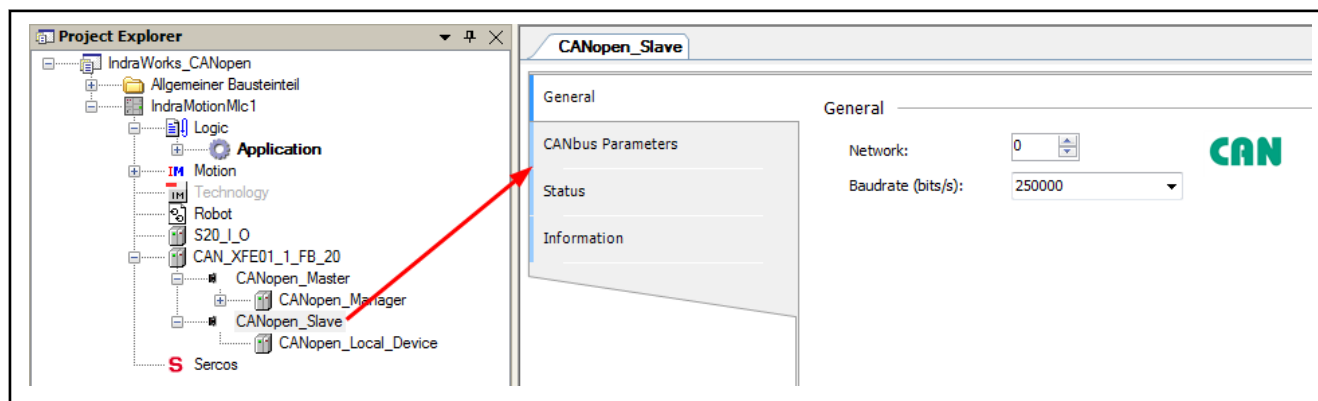


Fig. 5-2: "CANopen slave" dialog

5.1.1 Tab “CANopen slave - General”

The CAN bus object has to be entered into the Project Explorer as uppermost node of a CAN bus configuration (CAN bus configuration editor, page 13).

CAN bus slave configuration editor

The following is defined in the respective configuration dialog:

- **Network:** Number of the CAN network to be connected via the CAN bus interface.

Permitted values: 0 to 63.



If multiple CAN extension modules and multiple CAN master and slave interfaces are projected at the XM2x, an individual, unique network number has to be set for each CAN interface.

- **Baud rate (bits/s):** Baud rate in bits per second for the data transfer to the bus.

The default value is taken from the device description file (*.devdesc) of the CAN bus device.

The following baud rates can be set:

10000, 20000, 50000, 100000, 125000, 250000, 500000, 800000 und 1000000.

The unit is bits/s (bits per second). The baud rate is saved in KBit/s in the configuration data.

5.1.2 Tab “CANopen slave - CAN bus parameters”

Currently, this configuration dialog for a CAN bus node has no function, as the device description does not include any configuration parameters except for baud rate and network number (NetID).

5.1.3 Tab “CANopen slave - Status”

This dialog of a CANopen device or module configuration displays status information (e.g. "Running", "Stopped") and specific diagnostic messages from the device and information on the card used and the internal bus system ([CAN bus configuration editor, page 13](#)).

5.1.4 Tab “CANopen slave - Information”

This dialog of a CANopen device or module configuration displays information on the device or the module that can be read from the device description.

Possible specifications: Name, Vendor, Type, Version number, Categories, Ordering number, Image ([CAN bus configuration editor, page 13](#)).

CAN bus slave configuration editor

5.2 CANopen local device

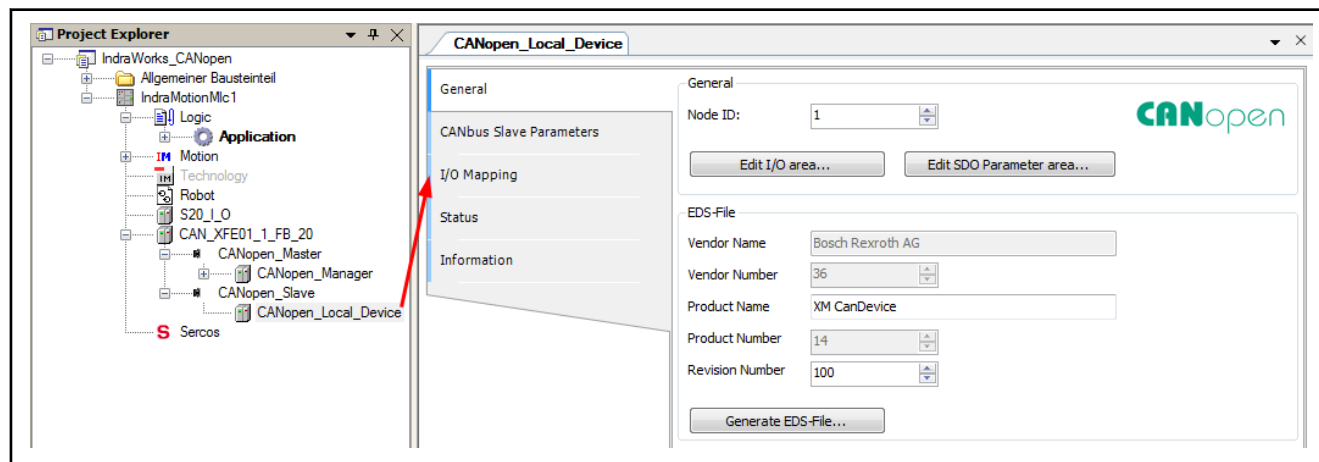


Fig. 5-3: "CANopen local device" dialog

5.2.1 Tab "CANopen local device - General"

General information

Node ID	The node number uniquely identifies the CAN module (value between 1 and 127).
Device profile	The device profile to be implemented by the CANopen slave. If a database (*.CODB) is available for object directories, the selection is adjusted according to the device profile ID in the "Object directory" dialog. The device profile IDs are defined in the CiA specifications.
Edit I/O area	The inputs and outputs of the CANopen slave are represented by the I/O areas. An I/O area is a collection of inputs and outputs of the same data type. One I/O channel and one mapping variable are automatically created for each input/output. The mapping variables allow to read and write the I/Os in the application program. The "Edit I/O area..." button opens a dialog (same name) to define the I/O areas and the respective object directory index of the slave. The possible number and size of the I/O areas depends on the number of the unused PDOs. The current number TxPDOs used and RxPDOs used is always displayed below the table. From the version 14.14.0.0 of the CANopen Local_Device, up to 512 PDOs (Tx like Rx) are possible (in older versions, only up to 16 PDOs).
Edit SDO parameter area	Opens the "Edit SDO parameter area" dialog. For details, refer to the description of the dialog.

EDS file

Vendor name	General specification to be saved in the EDS file.
Vendor number	
Product name	
Product number	
Revision number	
Export EDS file	The EDS file is created with the specified settings acc. to the standard CiA DS 306 V1.3 and saved on the local computer.
Import EDS file	Data import from an existing EDS file.

"Edit I/O area" dialog The dialog displays the currently valid I/O areas of the slave.

CAN bus slave configuration editor

Add area	Opens the dialog "Add I/O area". If all PDOs have already been used, the dialog cannot be edited.
TxPDOs used	Number of currently used TxPDOs/maximum possible TxPDOs
RxPDOs used	Number of currently used RxPDOs/maximum possible RxPDOs

I/O type	Input or output
Area name	A unique default name is entered automatically. However, it can be changed.
Object index	Index in the object directory of the slave to be assigned to the new object.
Number	Number of I/Os. The possible values depend on the data type and the current number of the unused PDOs.
Data type	Selection of possible data types.
Force new PDO	☒ : (Default). Each new I/O area is mapped on a new PDO even if the present PDO mapping is not yet full. ☐ : The current PDO mapping is filled completely (if possible) before a new PDO is created.
OK	Enables the new I/O area. Input/output channels and mapping variables are created. In the object directory of the device, each I/O area has its own index (output area: 0x3000 – 0x30FF, input area: 0x3800 – 0x38FF) and each individual input/output has a subindex within the I/O area. When creating an I/O area, PDOs are created automatically (RxPDOs start with 0x1400, TxPDOs with 0x1800). If the option "Force new PDO" is selected (see above), the I/Os are automatically and subsequently mapped on these new PDOs.

"Edit SDO parameter area" dialog

They can define read/write parameters (SDO parameters) that can be read or written by SDOs. Therefore, define parameter areas being a collection of parameters of the same data type. One I/O channel and one mapping variable are automatically created for each SDO parameter. The mapping variables allow to read or write I/Os using the application program.

In the object directory of the device, each SDO parameter area has its own index and each individual SDO parameter has a subindex within the parameter area:

- SDO write parameter area: 0x5000 - 0x50FF
- SDO read parameter area: 0x5800 – 0x58FF

The dialog shows the currently valid parameter areas and its object directory indices.

Add area	Opens the "Add parameter area" dialog.
----------	--

Dialog "Add parameter area"

The dialog is used to easily add I/Os.	
SDO access type	Read-only or Read/write .
Range name	A unique name is entered automatically. However, it can be changed.
Object index	Index in the object directory of the slave to be assigned to the new object.

CAN bus slave configuration editor

Number	Number of parameters
Data type	Selection of possible data types

5.2.2 Tab “CANopen local device - Parameters”

This tab shows the parameters of the CANopen Local_Device in a table.
A part of the parameters can be edited (for editing, click on the “Value” field).
The font color of these parameters is black.
The font color of the fields “Value” and “Type” are grey, only the current values are displayed.

5.2.3 Tab “CANopen local device - I/O image”

The tab shows the name of the automatically created instance of the Local_Device under "IEC objects" in the "Variable" column.
Use this application to access the instance. Thus, the target device can be restarted, stopped or its status can be checked using an application.
Please also refer to the description of the general “I/O image” dialog.

Bus cycle options

Bus cycle task	The selection list provides all tasks defined in the task configuration of the active application, such as “MainTask”. For Use parent bus cycle setting, IndraLogic uses the settings of the father node. If no task is defined here, the task set in the PLC settings applies. If no task has been set here as well, the task with the highest priority is used as bus cycle task.
----------------	--

Tab “<Device> I/O image”

In this tab of the generic device editor, map the input, output and memory addressed of the control device to the variables of the application. This is used to create the so-called "I/O image" ("mapping").

IEC objects

	This section of the dialog is only displayed if an instance of the device object is implicitly applied that can be accessed in the application. This information can for example be used to restart a bus or to query information. Whether such an instance is available and how it can be used depends on the device type.
--	--

5.2.4 Tab “CANopen local device - Status”

This subdialog of the configuration editor of a local CAN bus slave device displays status information (e.g. "Running", "Stopped") and device-specific diagnostic messages as well as information on the card used and the internal bus system.

5.2.5 Tab “CANopen local device - Information”

This dialog in the local CAN bus slave device configuration displays information on the device currently selected in the Project Explorer.
The following general information is output for the device if available: Name, Vendor, Type, Version number, Categories, Ordering number, Description, Image.

6 Device database

6.1 Device database, overview

The device database supplements the device library (IndraWorks, right side).

It contains device descriptions installed on the local system to provide them to projects in IndraWorks. The properties of the respective devices (e.g. parameters, vendor, device name) are stored in the device description files.

There are different formats for different devices

- Profibus DP: *.gs?,
- Profinet I/O: GSDML*.xml,
- EtherNet/IP: EDS*.xml,
- CAN: *.eds
- Sercos III I/O: SDDML file, *.xml.
- Device description files: *devdesc.xml

The device database is installed when installing IndraWorks

Additional devices can be installed and devices which are not required anymore can be uninstalled in the device database.

Postinstallation of the *.devdesc.xml as well as of the *.eds files is required for the CAN bus.

6.2 Managing devices

6.2.1 Device database, dialog

Icon:

Use the "Device database" command to install and uninstall device description files. In the main menu, click on **Tools ► Device database...** to open the "Device database" dialog:

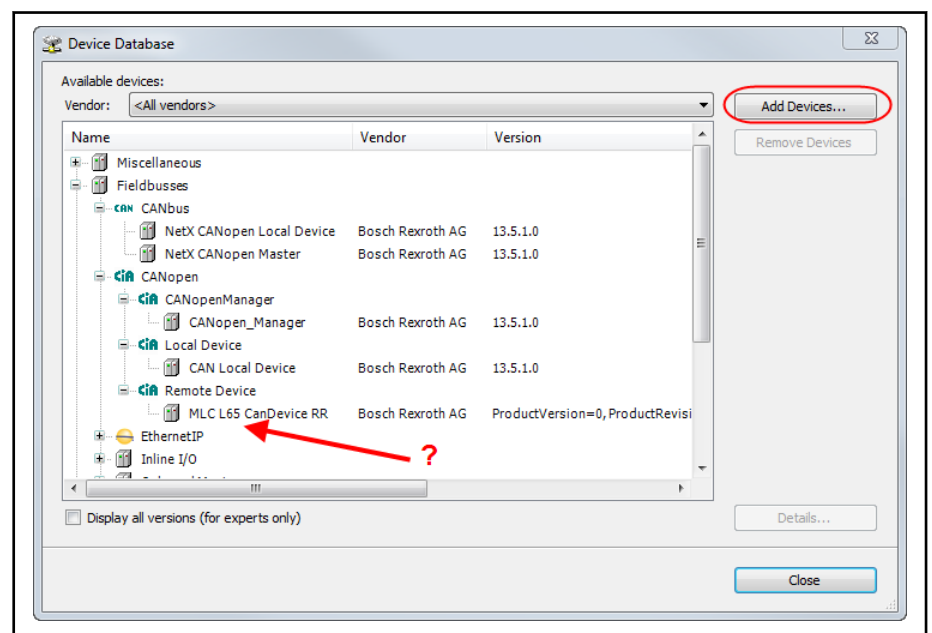


Fig. 6-1: Device database

Device database

- Available devices** The currently installed devices are listed under "Available devices".
- The devices are hierarchically structured, that is, controls in the upper part, then function modules below, then integrated interfaces, etc.
- Set the "Vendor" filter to restrict the list.
- Device can be added or removed under "Available devices".
- Normally, the devices matching the IndraWorks version are displayed. However, for a complete overview, select "Display all versions".
- After selecting a device, more information is available under "Details".

6.2.2 Adding devices

To add devices to IndraWorks, proceed with the following steps:

1. In the main menu, click on **Tools ► Device database...** to open the "Device database" dialog.
2. Click on "Add devices..." in the "Device database" dialog.

The file selection dialog "Install device descriptions" opens:

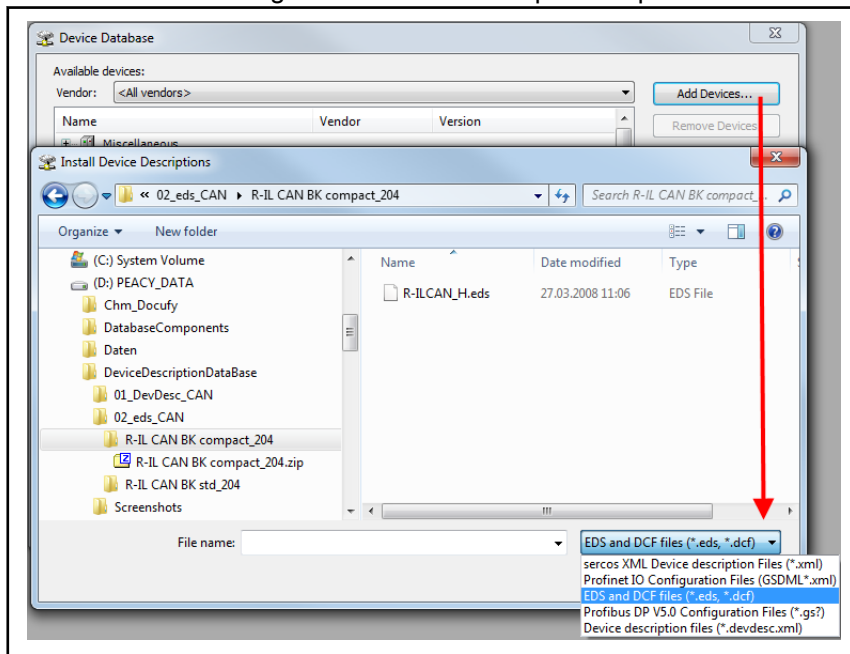


Fig. 6-2: Installing device descriptions

3. Select the device description files to be installed.



Select the correct file type in the drop-down list "File type".

4. Click on "Open" to start the installation.
- After the installation, the imported devices are displayed and highlighted under "Available devices":

Device database

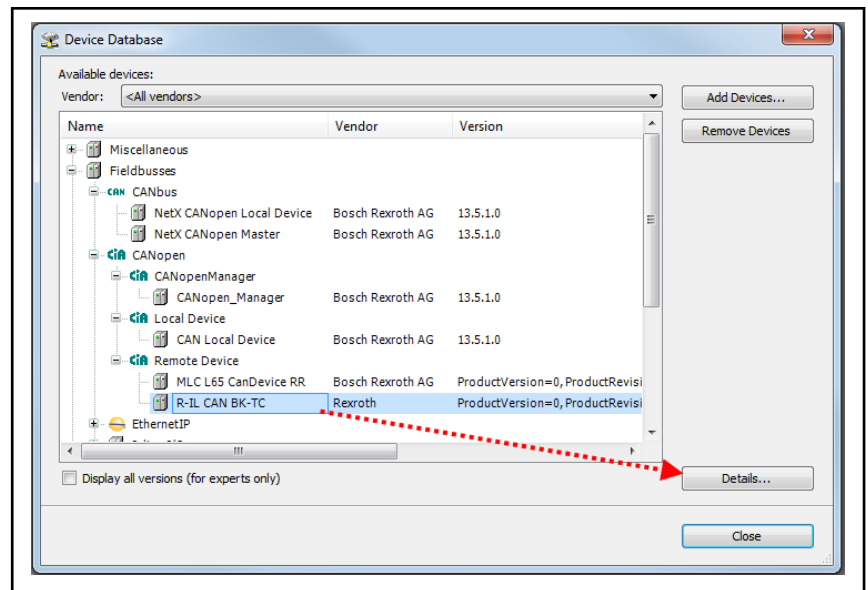


Fig. 6-3: Device descriptions installed

5. Select the device and further information is provided under "Details".

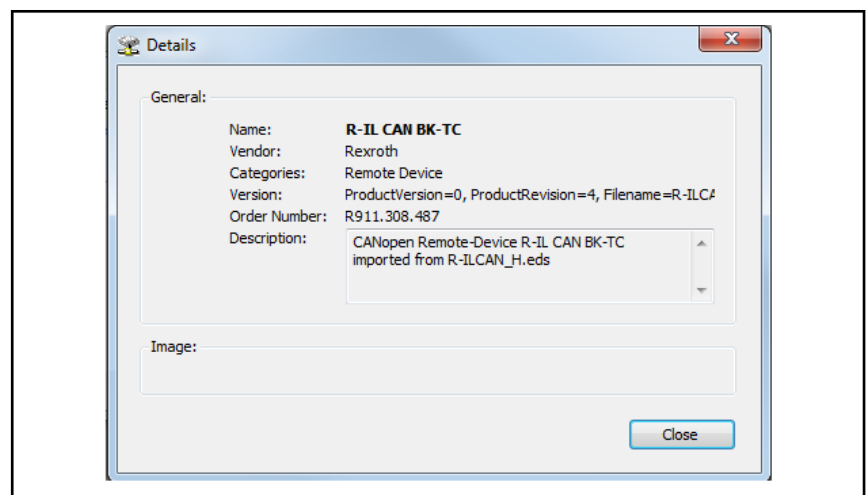


Fig. 6-4: Details on the installed device description file

6.2.3 Removing devices

To remove the devices from IndraWorks, proceed with the following steps:

1. In the main menu, click on **Tools ► Device database...** to open the "Device database" dialog.
2. Highlight the devices to be removed in the device tree.
3. Click on "Remove devices".

The "Confirm device removal" dialog opens:

4. Click on "Yes" to remove the selected device.

The device is removed from the device database and from the library.

5. To close the "Device database" dialog, press "Close".

7 CAA_CiA405.library

7.1 CAA_CiA405.Library, general

This document uses common CANopen interface for IEC 61131-3 programmable devices such as PLC.

The devices use communication methods corresponding to those described in the CiA standard draft DS-301 (Application Layer and Communication Profile), the standard draft DSP-302 (Framework for Programmable CANopen Devices) and the standard draft DS-405 (Interface and Device Profile for IEC 61131-3 programmable devices).

Use these documents as well when using this description.

- References**
- /DS-301/ CiA Draft Standard DS 301 V4.02 CANopen Application Layer and Communication Profile
 - /DSP-302/ CiA Draft Standard DSP 302 V3.1 CANopen Framework for Programmable CANopen Devices
 - /DS-405/ CiA Draft Standard Proposal DS 405 V2.0 CANopen Interface and Device Profile for IEC 61131-3 Programmable Devices

In general, the creation of an application implements the handling of up to five interfaces:

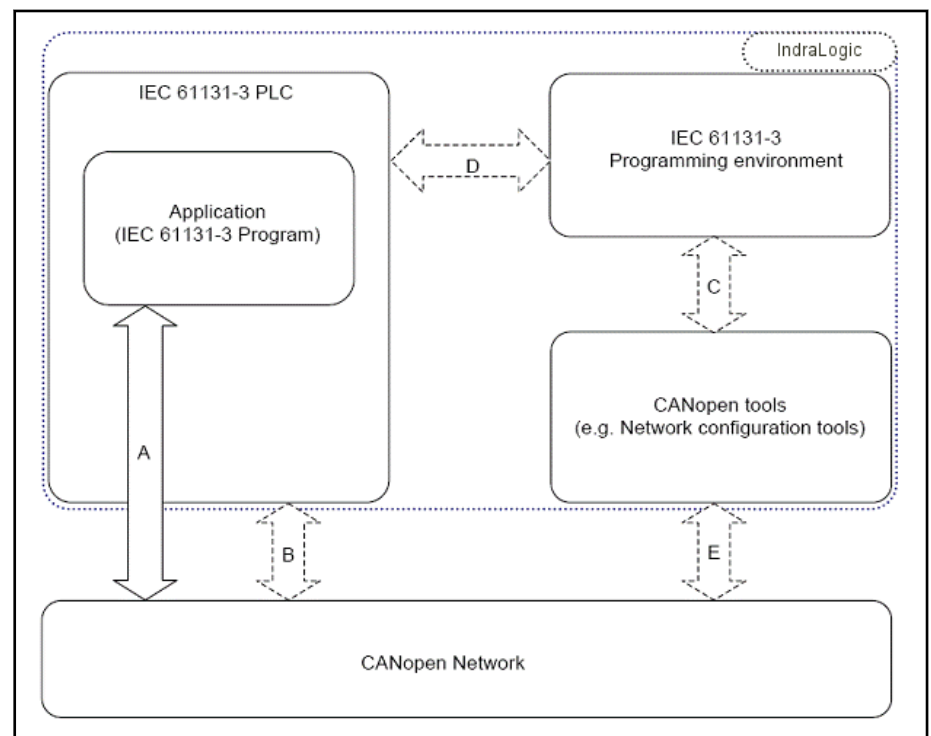


Fig. 7-1: Creating an application/handling interface

This description is only about the interface:

A. means accessing a CANopen communication system from an IEC 61131-3 program based on function block calls

This description does not (yet) include the interfaces:

B. Service function for troubleshooting, monitoring and network management

CAA_CiA405.library

- C. Interface between the CANopen tools and the IEC 61131-3 programming environment
- D. Vendor-specific interface between the programming environment and the PLC runtime
- E. CANopen tool-specific interface between the configuration tool and the network



The figure does not necessarily describe the use of different tools for programming and configuration.

A tool might be able to handle both functionalities and to cover the interfaces. An example is the programming tool IndraLogic (see the dashed line) that includes the CANopen configuration and the runtime software in the PLC.

User interface

Each application in an IEC 61131-3 layer has to be able to read in a device object directory of a node connected to CANopen or to write on it (SDO). Other elements of a CANopen communication are Process Data Objects (PDO) supported as inputs (PDO received; i.e.: %IX..) or outputs (PDO transferred; i.e.: %QW..) in the process mapping within an IEC application.

Received asynchronous error messages (EMCY) are registered for each node in a buffer. The network status is checked by the network management (NMT). Due to CANopen, the "Node guarding" functionality as well as the synchronized communication is possible for process data.

The present description explains the requirements on a function block library for the CANopen communication.

7.2 Data types

7.2.1 Data types of the CAA_CiA405.library

The following data types are mandatory due to the library requirements.

ENUM

- [DEVICE_STATE](#), page 40
- [SDO_MODE](#), page 41
- [TRANSITION_STATE](#), page 42
- [CANOPEN_KERNEL_ERROR](#), page 42

STRUCT

- [EMCY_ERROR](#), page 42

Type derivation

- [DEVICE](#), page 42
- [SDO_ERROR](#), page 43

7.2.2 DEVICE_STATE

Brief description The enumeration type `DEVICE_STATE` (INT) describes the state of the CANopen network layer as defined in /DS-301/.

The states `INIT`, `RESET_COMM`, `RESET_APP`, `PRE_OPERATIONAL`, `STOPPED`, `OPERATIONAL` correspond to the same states as in /DS-301/.

The state `UNKNOWN` has to be used if the current state of the device is unknown (e.g. if no guarding of the device is executed).

The state NOT_AVAIL has to be used when it is known that the device is not available (e.g. when guarding is executed and the device does not respond).

INIT	0
RESET_COMM	1
RESET_APP	2
PRE_OPERATIONAL	3
STOPPED	4
OPERATIONAL	5
UNKNOWN	6
NOT_AVAIL	7

Tab. 7-1: Elements of the DEVICE_STATE enumeration type

The state provided by [GET_STATE](#), page 60, is:

- If node guarding/heartbeat is not enabled for the selected DEVICE: UNKNOWN (6) (even if a restart message has been received)
- If heartbeat is enabled, but the NMT master is not the heartbeat consumer: UNKNOWN (6).
- If node guarding/heartbeat is enabled and the device reports its NMT status:

Correct NMT state (value defined in DS301 V4.02 chap. "Error backup protocol")	STATE (value, defined in DS405 V2.0 chapter "Data types")
Initialization (0)	INIT (0)
Pre-operational (127)	PRE-OPERATIONAL (3)
Operational (5)	OPERATIONAL (5)
Stopped (4)	STOPPED (4)

- If node guarding/heartbeat is active and the device did not correctly report its NMT state before the timeout, e.g. "Node life time" (node guarding protocol) or "Heartbeat consumer time" (heartbeat protocol): NOT_AVAIL (7)

7.2.3 SDO_MODE

Brief description

The parameter mode for [SDO_WRITE](#), page 57, is a value from the following:

AUTO	0	Client selects the mode automatically
EXPEDITED	1	Client uses the accelerated protocol
SEGMENTED	2	Client uses the segmented transmission protocol
BLOCK	3	Client uses the block transfer protocol

Tab. 7-2: Elements of the SDO_MODE enumeration type

The client uses the most efficient transmission protocol in the mode "AUTO" to transfer data from or to the server. It queries the options and selects the best for both parties.

The SDO protocol is defined in /DS 301/.

CAA_CiA405.library

7.2.4 TRANSITION_STATE

Brief description The enumeration type (UINT) describes the state transitions of the CANopen network layer as defined in /DS-301/.

STOP_REMOTE_NODE	16#04	
START_REMOTE_NODE	16#05	
RESET_NODE	16#06	
RESET_COMMUNICATION	16#07	
ENTER_PRE_OPERATIONAL	16#7F	
ALL_EXCEPT_NMT_AND_SENDER	16#800	A flag to be used together with other state transitions (logical OR). It is to be used if NMT commands are addressed to all devices (DEVICE=0) except for the NMT sender and the NMT master (= CANopenManager).

Tab. 7-3: Elements of the TRANSITION_STATE enumeration type

7.2.5 CANOPEN_KERNEL_ERROR

Brief description The type CANOPEN_KERNEL_ERROR includes error information on the CANopen kernel.

For possible assignments, go to "Global constants" [CANOPEN_KERNEL_ERROR_CODES](#), page 43,.

Name	Type	Description
CANOPEN_KERNEL_ERROR	WORD	

Tab. 7-4: Type assignment SDO_ERROR

7.2.6 EMCY_ERROR

Brief description The structure EMCY_ERROR includes emergency error information as specified in /DS-301/:

Name	Type	Description
EMCY_ERROR_CODE	WORD	
ERROR_REGISTER	BYTE	
ERROR_FIELD	ARRAY [1..5] OF BYTE	

Tab. 7-5: Elements of the EMCY_ERROR structure

7.2.7 DEVICE

Brief description The type DEVICE represents the Node_ID of a device.

Name	Type	Description
DEVICE	USINT (0..127)	

Tab. 7-6: Type assignment DEVICE

7.2.8 SDO_ERROR

Brief description The type SDO_ERROR represents error information as defined in / DS-301/:

Name	Type	Description
SDO_ERROR	UDINT	

Tab. 7-7: Type assignment SDO_ERROR

7.3 Global constants

7.3.1 CANOPEN_KERNEL_ERROR_CODES

The following table contains possible assignments of the data type [CANOPEN_KERNEL_ERROR](#), page 42.

	Value	Description
CANOPEN_KERNEL_NO_ERROR	16#0	No error No error was reported by the CANopen kernel. The DEVICE output is invalid and can contain any value. However, there is the option that an SDO abort message or an EMCY message was received by any device, refer to SDO_READ , page 58, SDO_WRITE , page 57, and RECV_EMCI , page 67 / RECV_EMCI_DEV , page 65
CANOPEN_KERNEL_OTHER_ERROR	16#1	Other error If there is a "further error register" output at the calling function block, this output contains more detailed information, e.g. via the function blocks SDO_READ , page 58, and SDO_WRITE , page 57 to which the contents of the SDO abort message were provided. For the function blocks RECV_EMCI_DEV , page 65, and RECV_EMCI , page 67,, the content of the received EMCY message is provided. For the function block GET_CANOPEN_KERNEL_STATE , page 62,, this value is never provided, as no "further error register" output is available.
CANOPEN_KERNEL_DATA_OVERFLOW	16#2	Data overflow For implemented sending or receiving buffers for CAN objects, it can be possible that these buffers overflow.
CANOPEN_KERNEL_TIMEOUT	16#3	Timeout There is a timeout if a node does not respond to an SDO query for example. Time measurement starts when the application sends the query, i.e. at a rising edge at the ENABLE signal. Also refer to Additional parameters , page 45
	16#4.. 16#F	For further SDO errors Reserved for CIA
CANOPEN_KERNEL_CANBUS_OFF	16#10	CAN bus is in BUSOFF or in the bus alarm state The node is separated from the bus.
CANOPEN_KERNEL_CAN_ERROR_PASSIVE	16#11	CAN error passive The node is error-passive. The node can communicate, but it cannot send an active error flag if an error is detected, i.e. it cannot disturb the bus communication. Additionally, if it waits for 8 bits longer than another node before another transfer is started.

CAA_CiA405.library

	Value	Description
	16#12.. 16#1F	For further internal errors Reserved for CIA
CANOPEN_INTERNAL_FB_ERROR	16#21	Vendor-specific Up to 222 vendor-specific errors (16#21 – 16#FF). This is the free value range that should be used for error codes originating from the software of the subordinate level, e.g. kernel, driver, services, operating system.
	16#100.. 16#FFFF	Reserved for CIA

7.4 Function blocks

7.4.1 Function blocks of the CAA_CiA405.library

The common function blocks to access the CANopen of IEC 61131-3 are described in the following.

The function blocks are mandatory due to the requirements on the library.

Optional function blocks acc. to the CANopen standardization are additionally described to put all function blocks defined in /DS-405/ on one level.

IEC 61131-3-compliant systems can use the following table to identify the hidden properties.

Basic function block (behavior model, basic class)

- [CIA405BASE](#), page 48

Mandatory function blocks

- [RECV_EM CY_DEV](#), page 65
- [RECV_EM CY](#), page 67
- [SDO_WRITE4](#), page 49
- [SDO_WRITE_DATA](#), page 52
- [SDO_READ4](#), page 53
- [SDO_READ_DATA](#), page 55
- [GET_LOCAL_NODE_ID](#), page 59
- [GET_STATE](#), page 60
- [GET_CANOPEN_KERNEL_STATE](#), page 62
- [NMT](#), page 63

Optional function blocks

- [SDO_WRITE](#), page 57
- [SDO_READ](#), page 58

7.4.2 General aspects to structure function blocks

Data types compared to the data length

It is possible to represent the transfer of data of any length within the IEC 61131-3.

However, the interfaces to function blocks permitting that might be far more difficult to understand and to use.

Thus, different function blocks are defined to transfer specified (maximum) amount of data and function blocks for any lengths.

Timeout

The interface design allows to pack calls with a time encoder block to implement timeouts.

It is additionally permitted that the CANopen software of the subordinate level uses its own timeouts and messages, e.g. errors to the caller.

Additional parameters

Some systems might support or request additional information in the function blocks. Thus, one PLC can for example be provided with multiple CAN channels.

In this case, the function blocks require the channel number of the service.

Another requirement is to restrict the time that a function block requires until it reports to the caller whether the function block was executed successfully or not executed successfully. Thus, a timeout input (milliseconds) is required.

Thus, these parameters are added to the function blocks.

Adding the function blocks to the following parameters is permitted in / DS405/ and mandatory within CAA_CiA405.Library:

NETWORK: USINT; (*CAN channel number*)	Determines the CAN channel used to execute the required service (from number 1). A device with only one CAN channel has to accept either only the number 1 or all numbers, i.e. it has to ignore the value of this input.
CHANNEL: USINT; (*SDO channel number*)	Determines the SDO channel used to execute the required service (from number 1). A device with only one SDO channel has to accept either only the number 1 or all numbers, i.e. it has to ignore the value of this input. When using the number 0, the "Auto channel" function is enabled to execute the required service automatically using the next free SDO channel. The CAN channel number is not equal to the Net_ID specified in the CAN bus configurator. Use the following formula when calculating the CAN channel number: $\text{Net_ID} + 1$
TIMEOUT: UDINT; (*milliseconds; maximum waiting time for response*)	The timeout specifies the maximum waiting time for a response of the function block. The time measurement starts at the rising edge at the ENABLE input of the function block called. If the specified maximum time elapsed without a response being available for the querying service, the function block then completes the execution or reports a timeout. In case of an unlimited response time, 0 is used in this input.
DEVICE: DEVICE; (*source of STATE *)	In the CAA_CiA405.Library, another output of the function block GET_CANOPEN_KERNEL_STATE, page 62, is defined: DEVICE contains either the device number of the device connected to the STATE (e.g. if an unknown data format is received) or the number of the device, at which the kernel is positioned.

CAA_CiA405.library

In the CAA_CiA405.Library, another input of the function block **SDO_WRITE**, page 57, is defined:

MODE: SDO_MODE;
(*Protocol type*)

MODE determines the method for sending/receiving data from the server with regard to the SDO protocol specification in /CIA301/

Control mechanism

In general, the task of each defined function block of the CAA_CiA405.Library is to read or write one or multiple parameters.

Such parameters can be variables or states of devices connected to CANopen (SDO, EMCY, NMT state) or local nodes (individual node number, CANopen kernel state). That is why each function block generally has the same behavior.

One typical response of the signals ENABLE, CONFIRM and ERROR are displayed upon SDO access:

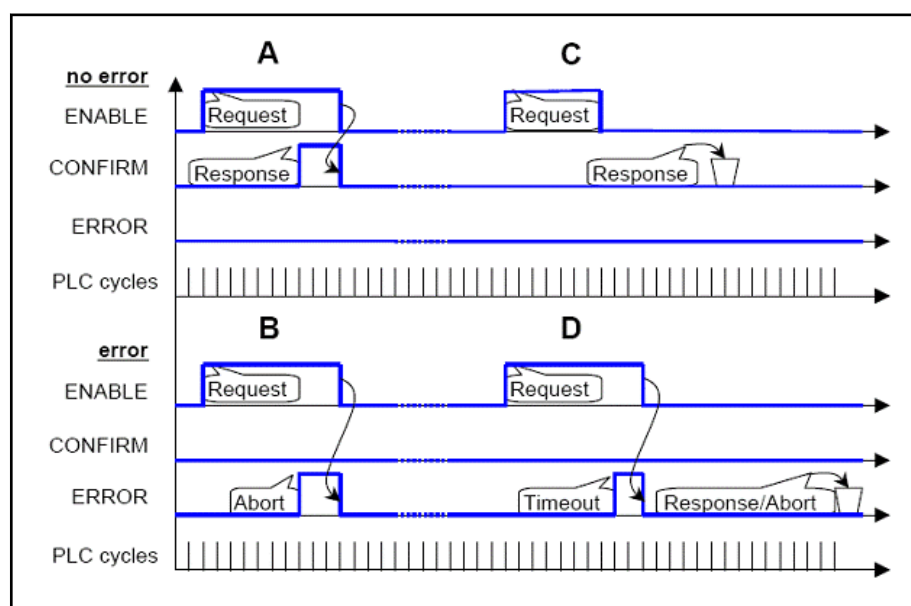


Fig. 7-2: Request/response diagram

- The tip at the symbol shows the event time.
The symbol shows that the message is ignored.
High levels at ENABLE and CONFIRM signal TRUE. Low levels signal FALSE. High level at ERROR signals that the value is greater than 0. Lower level signals a value equal to 0.
- PLC cycles:** The diagram shows that there is no relation between the signal sequence and the PLC cycles, e.g. the response can either be immediately available or after some PLC cycles.
- Timeout:** The time control starts at a rising edge at "ENABLE". A timeout event occurs if not response is available before the time control reaches the timeout value. If there is a timeout, ERROR = 3 and ERRORINFO can contain error information in the SDO abort data format. If no additional error information is available, ERRORINFO is 0.

Request: The state "no_error" also shows like "error" that the request event occurs immediately with the rising edge being detected at the ENABLE input, i.e. it is transferred to a subordinate software level. The outputs remain on

their initialization values. The program control is returned to the caller if the response or the abort are not immediately available.

A: Response: When an SDO response message is available, the output CONFIRM is set to TRUE and remains TRUE as long as the function block is called with the input ENABLE = TRUE. If the function block is called with ENABLE = FALSE, CONFIRM is set to its initialization value FALSE.

B: Abort: When the SDO abort message is available, the output ERROR is set to 1 and further error information is available at the output ERRORINFO ("further errors register" /DS405/). The outputs remain as long as the function block is called with the input ENABLE = TRUE. If the function block is called with ENABLE = FALSE, ERROR and ERRORINFO are set to 0, i.e. to their initialization values.

C: ENABLE = FALSE available before response or abort: When the function block is called with ENABLE = FALSE, the outputs are set to their initialization values, i.e. CONFIRM = FALSE, ERROR = 0, ERRORINFO = 0. If a response or an abort are ignored if available like the later incoming response or abort.

D: Timeout: If a function block is called and a timeout is detected and response or abort are not available, ERROR = 3 and ERRORINFO can contain error information in the SDO abort data format.

If no additional error information is available, ERRORINFO is 0.

Flow control

The function block remains in idle mode, i.e. all outputs are set to their initialization values and no operation is executed as long as the ENABLE input is FALSE. The number of calls is of no importance.

As soon as ENABLE realizes a rising edge, the function starts the execution, i.e. the input values are read and a query is sent to the subordinate software to e.g. read or write a certain parameter. If a response to the query is immediately available, as the return value after a function call, the function block sets its outputs immediately and completes its task. A function receiving the ID of the local node could operate as follows for example. If the response is not available immediately, the function block returns the program control to the caller when setting all output signals to the active state, e.g. CONFIRM = FALSE und ERROR = 0.

Upon each function block call, the availability of the response and the timeout are checked if required. Input changes are ignored, except ENABLE.

If the response reports an error state or if there is a timeout, the function block sets the outputs to the error state, e.g. CONFIRM = FALSE and ERROR <> 0 and completes its task.

If the response reports success, the function blocks sets the outputs to state confirmation, i.e. CONFIRM = TRUE and ERROR = 0 and, if it was a reading requests, it sets the data outputs to the currently queried values and completes its task.

CAA_CiA405.library



It is not clearly defined in /DS405/ if the function block is executed

- a) immediately,
- b) after the next call or
- c) after the next rising edge of the ENABLE input

.

To facilitate access to the same SDO channel on the same device for other function blocks, it is recommended to wait for at least one PLC cycle (b) before the execution is started again. Thus, it is prevented that a function block that is continuously called with the ENABLE = TRUE input blocks communication with a device.

In cases a) and b), there is the potential disadvantage that the function block cannot refer to the availability of a new response, e.g. new data. The user does not know how old the provided data is. In case of write requests, the user does not know whether the input data is transferred to the device before it is changed. Furthermore, it is not secure to transfer the same data multiple times. Thus, it is recommended not to restart the execution if ENABLE remains TRUE (c). If ENABLE is FALSE, all outputs are immediately set to their initialization values. A possible available response is ignored like a running query. A rising edge at ENABLE restarts the flow control.



In /DS301/ (V 4.02 chap. 9.2.2.1.7), the following is defined with regard to the SDO abort:

"The service can be executed by either the client or the server of an SDO at any time. If an SDO customer confirmed a pending service, the cancelation is considered as confirmation for this service."

7.4.3 CIA405_CIA405BASE (behavior model)

CIA405BASE

Brief description

To be able to implement the general behavior model for all CIA405 function blocks, this function block is used as a basic class. All other CIA405 function blocks in the present document extend this function block.

Interface description

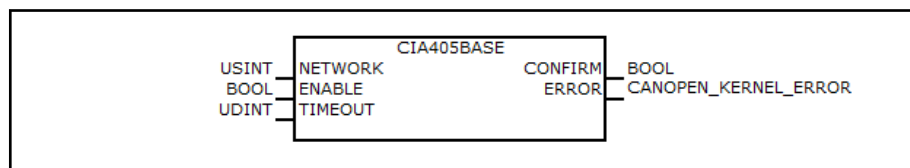


Fig. 7-3: Interface of the CIA405BASE function block

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	CAN channel number: Determines the CAN bus used by a CiA405 function block to execute its action. MAX_Networks = 64 The CAN_ChannelNumber is not equal to the Net_ID specified in the CAN bus configurator. Use the following formula when calculating the CAN_ChannelNumber: CAN_ChannelNumber: = Net_ID + 1
	ENABLE	BOOL	Enable execution
	TIMEOUT	UDINT	Maximum execution time in ms 0 means that the timeout is disabled
VAR OUTPUT	CONFIRM	BOOL	Successfully executed Range: True/false Initialization value: False
	ERROR	CANOPEN_KERNEL_ERROR	Error code Range: 0000 ... FFFF Initialization value: 0 CANOPEN_KERNEL_ERROR, page 42

Tab. 7-8: Interface of the CIA405BASE function block

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NET- WORKS (max. 64)	1	0/1 at ENABLE
TIMEOUT	UDINT	0		0	0/1 at ENABLE

Tab. 7-9: Input behavior

7.4.4 Function blocks "Reading and writing SDO"

SDO_WRITE4

Brief description The function block is used to send SDOs (Service Data Objects).
 The function block extends [CIA405BASE, page 48](#).

Interface description

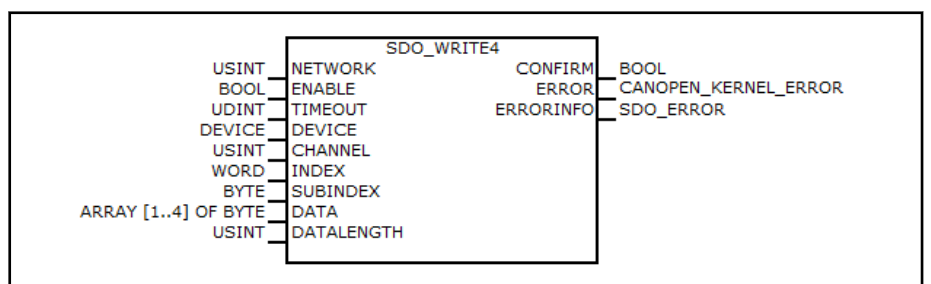


Fig. 7-4: Interface of the SDO_WRITE4 function block

CAA_CiA405.library

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	Refer to CIA405BASE, page 48
	ENABLE	BOOL	Refer to CIA405BASE, page 48
	TIMEOUT	UDINT	Refer to CIA405BASE, page 48
	DEVICE	DEVICE	Addressed device; 0 means local device DEVICE, page 42
	CHANNEL	USINT	SDO channel number
	INDEX	WORD	Object index
	SUBINDEX	BYTE	Object subindex
	DATA	ARRAY [1..4] OF BYTE	Object data
	DATALength	USINT	Length of the valid data in bytes
VAR_OUTPUT	CONFIRM	BOOL	Refer to CIA405BASE, page 48
	ERROR	CANOPEN_KERNEL_ERROR	Refer to CIA405BASE, page 48
	ERRORINFO	SDO_ERROR	SDO abort code 0 ... $2^{32}-1$ SDO_ERROR, page 43

Tab. 7-10: Interface of the SDO_WRITE4 function block

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NETWORKS	1	0/1 at ENABLE
TIMEOUT	UDINT	0	...	0	0/1 at ENABLE
DEVICE	DEVICE	0	127	0	0/1 at ENABLE
CHANNEL	USINT	1	128	1	0/1 at ENABLE
INDEX	WORD	16#0000	16#FFFF	16#0000	0/1 at ENABLE
SUBINDEX	BYTE	16#00	16#FF	16#00	0/1 at ENABLE
DATA	ARRAY [1..4] OF BYTE	4(16#00)	4(16#FF)	4(16#00)	0/1 at ENABLE
DATALength	USINT	1	4	1	0/1 at ENABLE

Tab. 7-11: Input behavior

Functional description

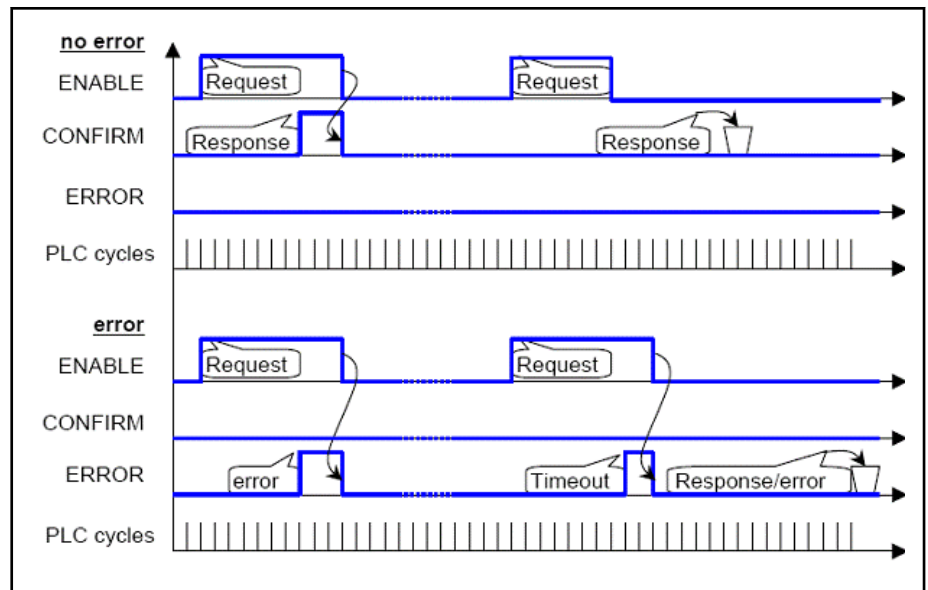


Fig. 7-5: Request/response diagram, time diagram of the SDO_WRITE4 function block

After all data has been provided for the inputs, "ENABLE" is set to TRUE (1).

The SDO is sent and if the CANopen software reports success to the function block, the output CONFIRM becomes TRUE (2).

This is observed by the caller and the caller will set ENABLE to FALSE (3 and the output CONFIRM becomes FALSE (4).

- At a rising edge at the ENABLE input, the function block starts collecting the inputs and initiates the transfer of the SDO specified in DATA and DATALENGTH to the receiver specified with DEVICE, INDEX and SUB-INDEX. The value of DEVICE is restricted to the range from 1 to 127. To access the local object dictionary, it is allowed to use 0 for DEVICE.
- The function block is allowed to continue its execution when the ENABLE input is TRUE. If a result from a request is called by a subordinate CANopen software, the outputs CONFIRM, ERROR and ERROR-INFO are set correspondingly.
- The function block is exited at a falling edge of the ENABLE input. If the transfer has not yet been completed, it is aborted if possible. The outputs CONFIRM and ERROR are set to FALSE and "no error".
- In case of a FALSE at the ENABLE input, the function block returns the control immediately and does not execute any action.
- For a more detailed description of the responses of ENABLE, CONFIRM and ERROR, refer to [Control mechanism, page 46](#).
- For information on the inputs NETWORK and TIMEOUT, refer to [Additional parameters, page 45](#).
- The input "DATA" defined as ARRAY[1..4] OF BYTE is arranged as it can be seen in the following:

Example

The value 16#01020304 (hexadecimal) is assigned as ARRAY[1..4] OF BYTE := 04, 03, 02, 01.

Result and troubleshooting

The result of a writing process is transmitted to the CONFIRM, ERROR and ERRORINFO outputs (immediately after the writing call or at a later point) with a rising edge either to CONFIRM or to ERROR.

CAA_CiA405.library

In case ERROR is equal to 1 (different error), ERRORINFO provides more specific information regarding the reason. This is especially the case if an SDO abort occurs. ERRORINFO contains the abort code.

In general, it cannot be assumed that this function is completed synchronously; however, it should be possible to continue executing this function while the call is returned by "SDO_WRITE4" and the PLC program is continued. Thus, the result of a call might only be available after multiple cycles after creating a rising edge at ENABLE.

SDO_WRITE_DATA

Brief description

The function block is used to send SDOs (Service Data Objects).

This function block can be used as an alternative to the SDO_WRITE_DATAx function blocks. It supports any data length.

The response and timing correspond to that of [SDO_WRITE4, page 49](#). The only difference is in the data type of the variables DATA and DATALENGTH.

The function block extends [CIA405BASE, page 48](#).

Interface description

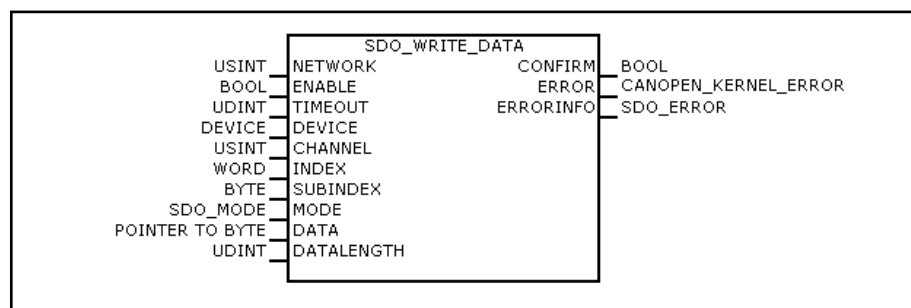


Fig. 7-6: Interface of the SDO_WRITE_DATA function block

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	Refer to CIA405BASE, page 48
	ENABLE	BOOL	Refer to CIA405BASE, page 48
	TIMEOUT	UDINT	Refer to CIA405BASE, page 48
	DEVICE	DEVICE	Addressed device; 0 means local device DEVICE, page 42
	CHANNEL	USINT	SDO channel number
	INDEX	WORD	Object index
	SUBINDEX	BYTE	Object subindex
	MODE	SDO_MODE	Specifies the SDO mode to be used SDO_MODE, page 41
	DATA	POINTER TO BYTE	Pointer to data to be written, (data in little endian byte order)
	DATALENGTH	UDINT	Length of the valid data in bytes

	Name	Type	Description
VAR OUTPUT	CONFIRM	BOOL	Refer to CIA405BASE , page 48
	ERROR	CANOPEN_KERNEL_ERROR	Refer to CIA405BASE , page 48
	ERRORINFO	SDO_ERROR	SDO abort code 0 ... $2^{32}-1$ SDO_ERROR , page 43

Tab. 7-12: Interface of the SDO_WRITE_DATA function block

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NETWORKS	1	0/1 at ENABLE
TIMEOUT	UDINT	0	...	0	0/1 at ENABLE
DEVICE	DEVICE	0	127	0	0/1 at ENABLE
CHANNEL	USINT	1	128	1	0/1 at ENABLE
INDEX	WORD	16#0000	16#FFFF	16#0000	0/1 at ENABLE
SUBINDEX	BYTE	16#00	16#FF	16#00	0/1 at ENABLE
MODE	SDO_MODE	1	...	1	0/1 at ENABLE
DATA	POINTER TO BYTE				0/1 at ENABLE
DATALength	UINT	1	...	1	0/1 at ENABLE

Tab. 7-13: Input behavior

SDO_READ4

Brief description The function block is used to receive SDOs (Service Data Objects).
 The function block extends [CIA405BASE](#), page 48.

Interface description

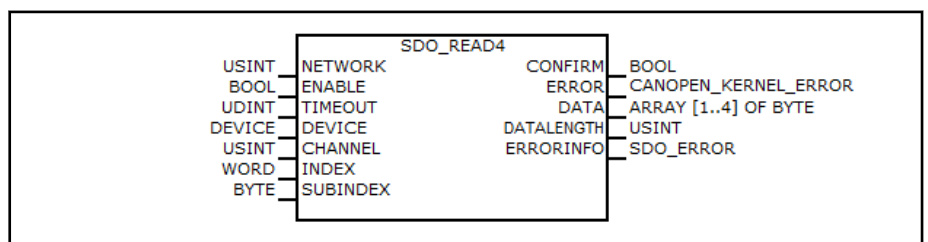


Fig. 7-7: Interface of the SDO_READ4 function block

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	Refer to CIA405BASE , page 48
	ENABLE	BOOL	Refer to CIA405BASE , page 48
	TIMEOUT	UDINT	Refer to CIA405BASE , page 48
	DEVICE	DEVICE	Addressed device; 0 means local device DEVICE , page 42

CAA_CiA405.library

	Name	Type	Description
	CHANNEL	USINT	SDO channel number
	INDEX	WORD	Object index
	SUBINDEX	BYTE	Object subindex
VAR OUTPUT	CONFIRM	BOOL	Refer to CIA405BASE , page 48
	ERROR	CANOPEN_KERNEL_ERROR	Refer to CIA405BASE , page 48
	DATA	ARRAY [1..4] OF BYTE	BYTE ARRAY into which data is written. Data is stored in CANopen byte order (= little endian).
	DATALength	USINT	Length of the valid data in bytes
	ERRORINFO	SDO_ERROR	SDO abort code 0 ... 2 ³² -1 SDO_ERROR , page 43

Tab. 7-14: Interface of the SDO_READ4 function block

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NETWORKS	1	0/1 at ENABLE
TIMEOUT	UDINT	0	...	0	0/1 at ENABLE
DEVICE	DEVICE	0	127	0	0/1 at ENABLE
CHANNEL	USINT	1	128	1	0/1 at ENABLE
INDEX	WORD	16#0000	16#FFFF	16#0000	0/1 at ENABLE
SUBINDEX	BYTE	16#00	16#FF	16#00	0/1 at ENABLE

Tab. 7-15: Input behavior

Functional description

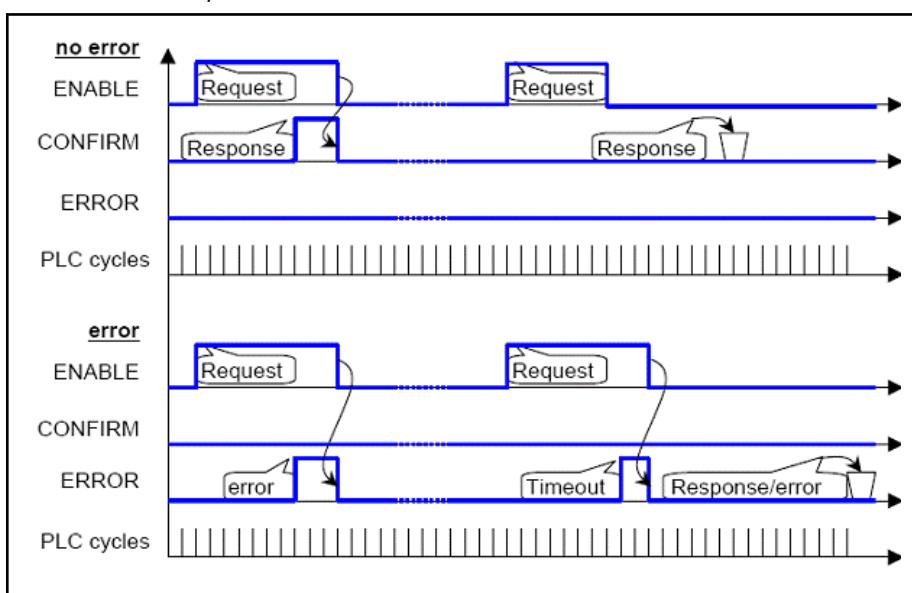


Fig. 7-8: Request/response diagram, time diagram of the SDO_READ4 function block

After all data has been provided for the inputs, "ENABLE" is set to TRUE (1).

The SDO is received and if the CANopen software reports success to the function block, the output CONFIRM becomes TRUE (2).

This is observed by the caller and the caller will set ENABLE to FALSE (3) and the output CONFIRM becomes FALSE (4).

- At a rising edge at the ENABLE input, the function block starts collecting the inputs and initiates the transfer of the SDO specified with DEVICE, INDEX and SUBINDEX. The value of DEVICE is restricted to the range from 1 to 127. To access the local object dictionary, it is allowed to use 0 for DEVICE.
- The function block may continue its execution if the ENABLE input is TRUE. If a result from a request is called by a subordinate CANopen software, the outputs CONFIRM, DATA and DATALENGTH (if successful) or ERROR and ERRORINFO (if failed) are set correspondingly.
- The function block is exited at a falling edge at the ENABLE input. If the transfer has not yet been completed, it is aborted if possible. The outputs CONFIRM and ERROR are set to FALSE and "no error".
- In case of a FALSE at the ENABLE input, the function block returns the control immediately and does not execute any actions.
- For a more detailed description of the responses of ENABLE, CONFIRM and ERROR, refer to [Control mechanism, page 46](#).
- For information on the inputs NETWORK and TIMEOUT, refer to [Additional parameters, page 45](#).
- The output "DATA" defined as ARRAY[1..4] OF BYTE is arranged as it can be seen in the following:

Example The value 16#01020304 (hexadecimal) is assigned as ARRAY[1..4] OF BYTE := 04, 03, 02, 01.

SDO_READ_DATA

Brief description

The function block is used to receive SDOs (Service Data Objects).

This function block is used as an alternative to the SDO_READx function blocks. It supports any data length. The response and timing correspond to that of [SDO_READ4, page 53](#). The only difference is in the data type of the variables DATA and DATALENGTH and the option of affecting the type of storing (MODE).

The function block extends [CIA405BASE, page 48](#).

Interface description

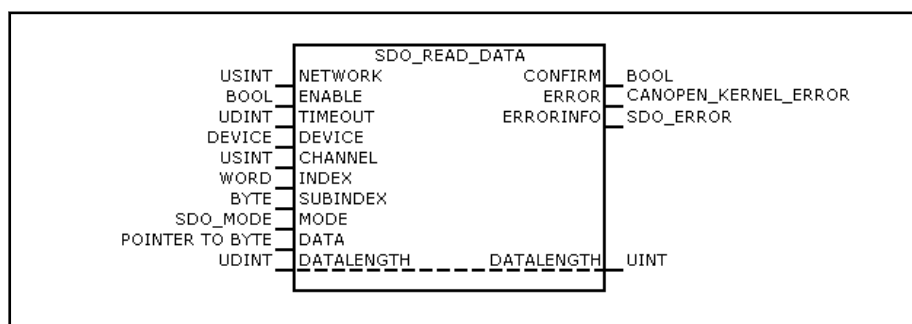


Fig. 7-9: Interface of the SDO_READ_DATA function block

CAA_CiA405.library

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	Refer to CIA405BASE, page 48
	ENABLE	BOOL	Refer to CIA405BASE, page 48
	TIMEOUT	UDINT	Refer to CIA405BASE, page 48
	DEVICE	DEVICE	Addressed device; 0 means local device DEVICE, page 42
	CHANNEL	USINT	SDO channel number
	INDEX	WORD	Object index
	SUBINDEX	BYTE	Object subindex
	MODE	SDO_MODE	Specifies the SDO mode to be used SDO_MODE, page 41
	DATA	POINTER TO BYTE	Pointer to the memory area to which the read data is to be written. Data is stored in CANopen byte order (= little endian).
VAR_OUTPUT	CONFIRM	BOOL	Refer to CIA405BASE, page 48
	ERROR	CANOPEN_KERNEL_ERROR	Refer to CIA405BASE, page 48
	ERRORINFO	SDO_ERROR	SDO abort code 0 ... $2^{32}-1$ SDO_ERROR, page 43
VAR_INOUT	DATALength	UDINT	Input: Buffer capacity Output: Length of the valid data in bytes Initialization value: 0

Tab. 7-16: Interface of the SDO_READ_DATA function block

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NETWORKS	1	0/1 at ENABLE
TIMEOUT	UDINT	0	...	0	0/1 at ENABLE
DEVICE	DEVICE	0	127	0	0/1 at ENABLE
CHANNEL	USINT	1	128	1	0/1 at ENABLE
INDEX	WORD	16#0000	16#FFFF	16#0000	0/1 at ENABLE
SUBINDEX	BYTE	16#00	16#FF	16#00	0/1 at ENABLE

Tab. 7-17: Input behavior

Obsolete function blocks

SDO_WRITE

Brief description



This function block is obsolete. It only supports data up to 64 kb.
 Please use the new function block instead [SDO_WRITE_DATA](#),
[page 52](#).

The function block is used to send SDOs (Service Data Objects).

This function block is used as an alternative to the SDO_WRITEx function blocks. It supports any data length.

The response and timing correspond to that of [SDO_WRITE4](#), [page 49](#). The only difference is the data type of the variables DATA and DATALENGTH.

The function block extends [CIA405BASE](#), [page 48](#).

Interface description

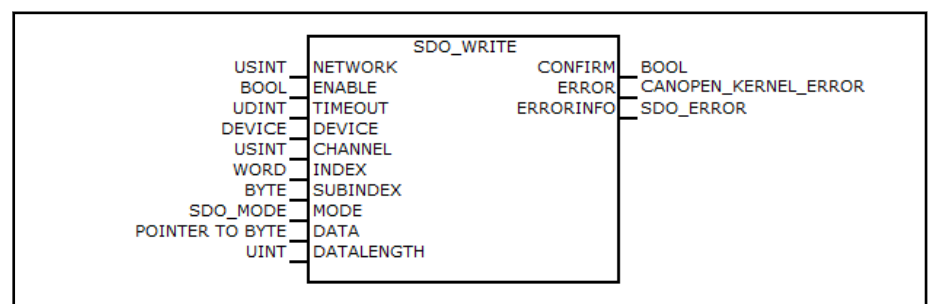


Fig. 7-10: Interface of the SDO_WRITE function block

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	Refer to CIA405BASE , page 48
	ENABLE	BOOL	Refer to CIA405BASE , page 48
	TIMEOUT	UDINT	Refer to CIA405BASE , page 48
	DEVICE	DEVICE	Addressed device; 0 means local device DEVICE , page 42
	CHANNEL	USINT	SDO channel number
	INDEX	WORD	Object index
	SUBINDEX	BYTE	Object subindex
	MODE	SDO_MODE	SDO_MODE , page 41
	DATA	POINTER TO BYTE	Object data
VAR OUTPUT	DATALENGTH	UINT	Length of the valid data in bytes
	CONFIRM	BOOL	Refer to CIA405BASE , page 48
	ERROR	CANOPEN_KERNEL_ERROR	Refer to CIA405BASE , page 48
	ERRORINFO	SDO_ERROR	SDO abort code 0 ... 2 ³² -1 SDO_ERROR , page 43

Tab. 7-18: Interface of the SDO_WRITE function block

CAA_CiA405.library

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NETWORKS	1	0/1 at ENABLE
TIMEOUT	UDINT	0	...	0	0/1 at ENABLE
DEVICE	DEVICE	0	127	0	0/1 at ENABLE
CHANNEL	USINT	1	128	1	0/1 at ENABLE
INDEX	WORD	16#0000	16#FFFF	16#0000	0/1 at ENABLE
SUBINDEX	BYTE	16#00	16#FF	16#00	0/1 at ENABLE
MODE	SDO_MODE	1	...	1	0/1 at ENABLE
DATA	POINTER TO BYTE				0/1 at ENABLE
DATALength	UINT	1	...	1	0/1 at ENABLE

Tab. 7-19: Input behavior

SDO_READ**Brief description**

This function block is obsolete. It only supports data up to 64 kb and no block transfer.

Please use the new function block instead [SDO_READ_DATA](#), [page 55](#).

The function block is used to receive SDOs (Service Data Objects).

This function block is used as an alternative to the SDO_READx function blocks. It supports any data length. The response and timing correspond to that of [SDO_READ4](#), [page 53](#). The only difference is the data type of the variables DATA and DATALength.

The function block extends [CIA405BASE](#), [page 48](#).

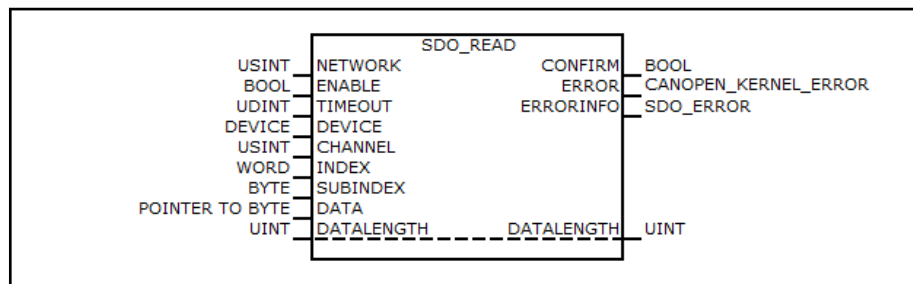
Interface description

Fig. 7-11: Interface of the SDO_READ function block

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	Refer to CIA405BASE , page 48
	ENABLE	BOOL	Refer to CIA405BASE , page 48
	TIMEOUT	UDINT	Refer to CIA405BASE , page 48
	DEVICE	DEVICE	Addressed device; 0 means local device DEVICE , page 42

	Name	Type	Description
	CHANNEL	USINT	SDO channel number
	INDEX	WORD	Object index
	SUBINDEX	BYTE	Object subindex
	DATA	POINTER TO BYTE	Pointer to the memory area to which the read data is to be written. Data is stored in CANopen byte order (= little endian).
VAR OUTPUT	CONFIRM	BOOL	Refer to CIA405BASE, page 48
	ERROR	CANOPEN_KERNEL_ERROR	Refer to CIA405BASE, page 48
	ERRORINFO	SDO_ERROR	SDO abort code 0 ... $2^{32}-1$ SDO_ERROR, page 43
VAR_INOUT	DATALength	UINT	Input: Buffer capacity Output: Length of the valid data in bytes Initialization value: 0

Tab. 7-20: Interface of the SDO_READ function block

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NETWORKS	1	0/1 at ENABLE
TIMEOUT	UDINT	0	...	0	0/1 at ENABLE
DEVICE	DEVICE	0	127	0	0/1 at ENABLE
CHANNEL	USINT	1	128	1	0/1 at ENABLE
INDEX	WORD	16#0000	16#FFFF	16#0000	0/1 at ENABLE
SUBINDEX	BYTE	16#00	16#FF	16#00	0/1 at ENABLE

Tab. 7-21: Input behavior

7.4.5 Function block "ID of own node" (GET_LOCAL_NODE_ID)

GET_LOCAL_NODE_ID

Brief description The function block GET_LOCAL_NODE_ID returns the own node ID.
 The function block extends [CIA405BASE, page 48](#).

Interface description

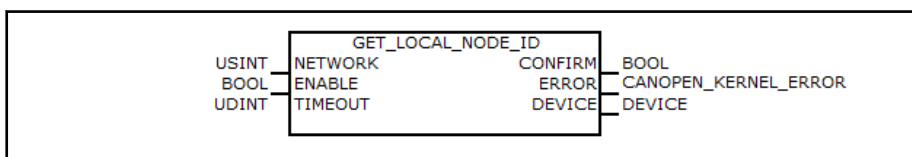


Fig. 7-12: Interface of the GET_LOCAL_NODE_ID function block

CAA_CiA405.library

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	Refer to CIA405BASE , page 48
	ENABLE	BOOL	Refer to CIA405BASE , page 48
	TIMEOUT	UDINT	Refer to CIA405BASE , page 48
VAR OUTPUT	CONFIRM	BOOL	Refer to CIA405BASE , page 48
	ERROR	CANOPEN_KERNEL_ERROR	Refer to CIA405BASE , page 48
	DEVICE	DEVICE	ID of local node; 0 means error DEVICE , page 42

Tab. 7-22: Interface of the GET_LOCAL_NODE_ID function block

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NETWORKS	1	0/1 at ENABLE
TIMEOUT	UDINT	0	...	0	0/1 at ENABLE

Tab. 7-23: Input behavior

Functional description

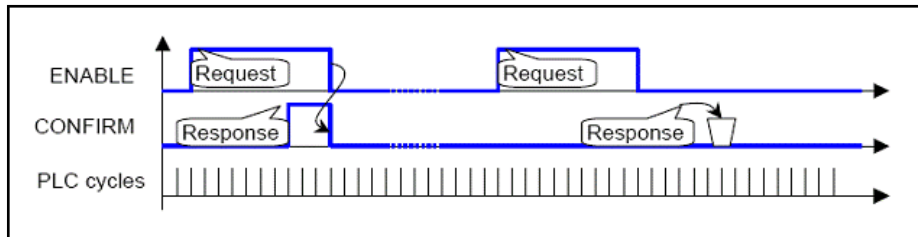


Fig. 7-13: Request/response diagram, time diagram of the GET_LOCAL_NODE_ID function block

- Data output DEVICE is only valid if CONFIRM is TRUE. If CONFIRM is FALSE, the data output is invalid and can contain any value.
- The data request is only initiated once at a rising edge at "ENABLE". Another request is not initiated before another rising edge was detected at "ENABLE".
- If there is an error while executing the function block, it has to output the invalid node number, DEVICE=0 and ERROR=0.
- For a more detailed description of the responses of ENABLE, CONFIRM and ERROR, refer to [Control mechanism](#), page 46.
- For information on the inputs NETWORK and TIMEOUT, refer to [Additional parameters](#), page 45.

7.4.6 Function blocks "Querying state" (GET_STATE)

GET_STATE

Brief description The function block returns the current state of a CANopen network device. The function block extends [CIA405BASE](#), page 48.

Interface description

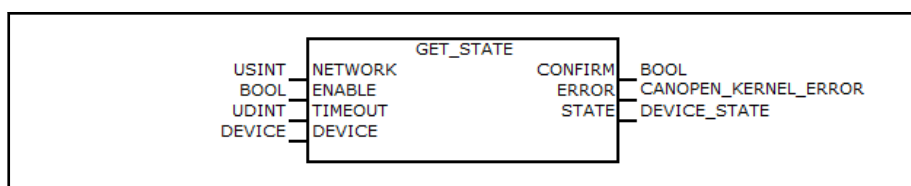


Fig. 7-14: Interface of the GET_STATE function block

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	Refer to CIA405BASE , page 48
	ENABLE	BOOL	Refer to CIA405BASE , page 48
	TIMEOUT	UDINT	Refer to CIA405BASE , page 48
	DEVICE	DEVICE	Addressed device; 0 means local device DEVICE , page 42
VAR OUTPUT	CONFIRM	BOOL	Refer to CIA405BASE , page 48
	ERROR	CANOPEN_KERNEL_ERROR	Refer to CIA405BASE , page 48
	STATE	DEVICE_STATE	Device state DEVICE_STATE , page 40

Tab. 7-24: Interface of the GET_STATE function block

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NETWORKS	1	0/1 at ENABLE
TIMEOUT	UDINT	0	...	0	0/1 at ENABLE
DEVICE	DEVICE	0	127	0	0/1 at ENABLE

Tab. 7-25: Input behavior

Functional description

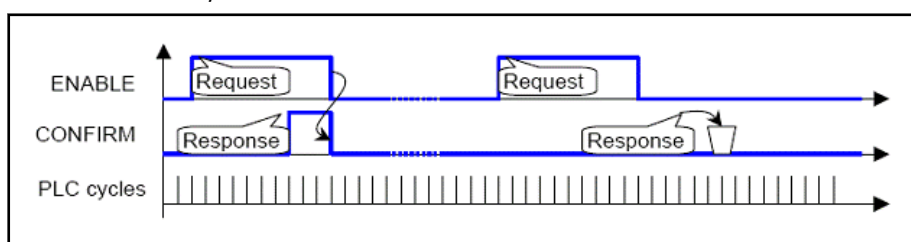


Fig. 7-15: Request/response diagram, time diagram of the GET_STATE function block

If the state is not known, the function block returns the control with the state UNKNOWN. This is the case if the device is not monitored and the heartbeat is not running.

If there is monitoring (guarding) and the device does not respond or if it is known that the device is not available, the function returns the state NOT_AVAILABLE.

If the DEVICE is 0 or equal to the own node ID, the function block returns the state of the local communication process.

CAA_CiA405.library

- Data output STATE is only valid if CONFIRM is TRUE. If CONFIRM is FALSE, the data output is invalid and can contain any value.
- The data request is only initiated once at a rising edge at “ENABLE”. Another request is not initiated before another rising edge was detected at “ENABLE”.
- For a more detailed description of the responses of ENABLE, CONFIRM and ERROR, refer to [Control mechanism, page 46](#).
- For information on the inputs NETWORK and TIMEOUT, refer to [Additional parameters, page 45](#).
- If there is an error while executing the function block, it has to output STATE = UNKNOWN (6) with CONFIRM = TRUE and ERROR = 0.

GET_CANOPEN_KERNEL_STATE

Brief description The function block returns the current state of a CANopen kernel.

The function block extends [CIA405BASE, page 48](#).

Interface description

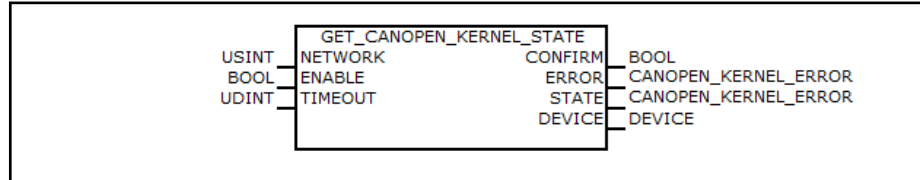


Fig. 7-16: Interface of the GET_CANOPEN_KERNEL_STATE function block

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	Refer to CIA405BASE, page 48
	ENABLE	BOOL	Refer to CIA405BASE, page 48
	TIMEOUT	UDINT	Refer to CIA405BASE, page 48
VAR OUTPUT	CONFIRM	BOOL	Refer to CIA405BASE, page 48
	ERROR	CANOPEN_KERNEL_ERROR	Refer to CIA405BASE, page 48
	STATE	CANOPEN_KERNEL_ERROR	Kernel state CANOPEN_KERNEL_ERROR, page 42
	DEVICE	DEVICE	Further device information linked to STATE; 0 means local device DEVICE, page 42

Tab. 7-26: Interface of the GET_CANOPEN_KERNEL_STATE function block

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NETWORKS	1	0/1 at ENABLE
TIMEOUT	UDINT	0	...	0	0/1 at ENABLE

Tab. 7-27: Input behavior

Functional description

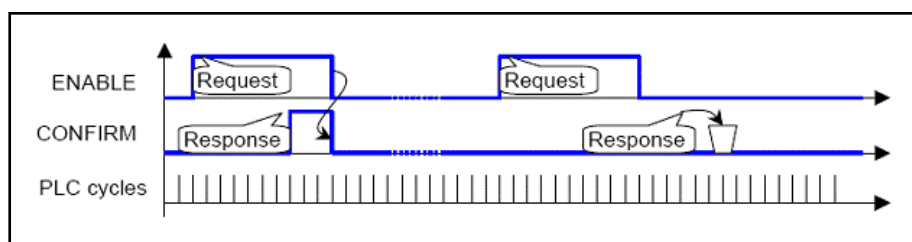


Fig. 7-17: Request/response diagram, time diagram of the GET_CANOPEN_KERNEL_STATE function block

- Data output STATE is only valid if CONFIRM is TRUE.
If CONFIRM is FALSE, the data output is invalid and can contain any value.
- The data request is only initiated once at a rising edge at "ENABLE". Another request is not initiated before another rising edge was detected at "ENABLE".
- For a more detailed description of the responses of ENABLE, CONFIRM and ERROR, refer to [Control mechanism](#), page 46.
- For information on the inputs NETWORK and TIMEOUT, refer to [Additional parameters](#), page 45.
- If there is an error while executing the function block, it has to output DEVICE = 0 (local device) with STATE <> 0 and ERROR = 0.
- The function block reports every error, except for
 - a) SDO abort information created by an SDO client and
 - b) EMCY messages received by a CAN device.

SDO abort specifications are handled by the function blocks [SDO_READ](#), page 58, and [SDO_WRITE](#), page 57.

EMCY messages are handled by the function blocks [RECV_EMCY](#), page 67, and [RECV_EMCY_DEV](#), page 65.
- As soon as STATE is reported, this is confirmed to error handling of the subordinate level. This means, for example, to delete the error message in the memory to release the input of another error message reported upon the next data query.

7.4.7 Function blocks "Network management" (NMT)

NMT

Brief description The function block controls the network management functions for one or for all CANopen nodes.

The function block extends [CIA405BASE](#), page 48.

Interface description

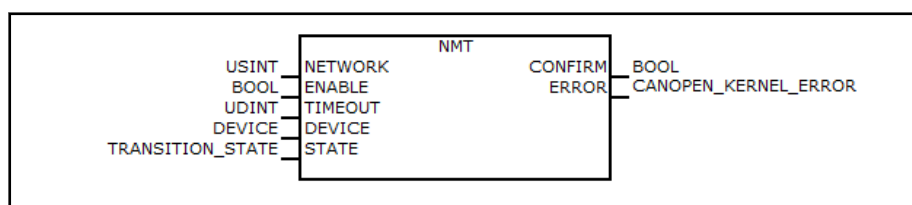


Fig. 7-18: Interface of the NMT function block

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	Refer to CIA405BASE , page 48
	ENABLE	BOOL	Refer to CIA405BASE , page 48

CAA_CiA405.library

	Name	Type	Description
	TIMEOUT	UDINT	Refer to CIA405BASE , page 48
	DEVICE	DEVICE	Addressed device; 0 means local device DEVICE , page 42
	STATE	TRANSITION_STATE	State transitions of the CANopen network layer TRANSITION_STATE , page 42
VAR OUTPUT	CONFIRM	BOOL	Refer to CIA405BASE , page 48
	ERROR	CANOPEN NEL_ERROR	Refer to CIA405BASE , page 48

Tab. 7-28: Interface of the NMT function block

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NETWORKS	1	0/1 at ENABLE
TIMEOUT	UDINT	0	...	0	0/1 at ENABLE
DEVICE	DEVICE	0	127	0	0/1 at ENABLE
STATE	TRANSITION_STATE	16#00	16#8FF	16#00	0/1 at ENABLE

Tab. 7-29: Input behavior

Functional description

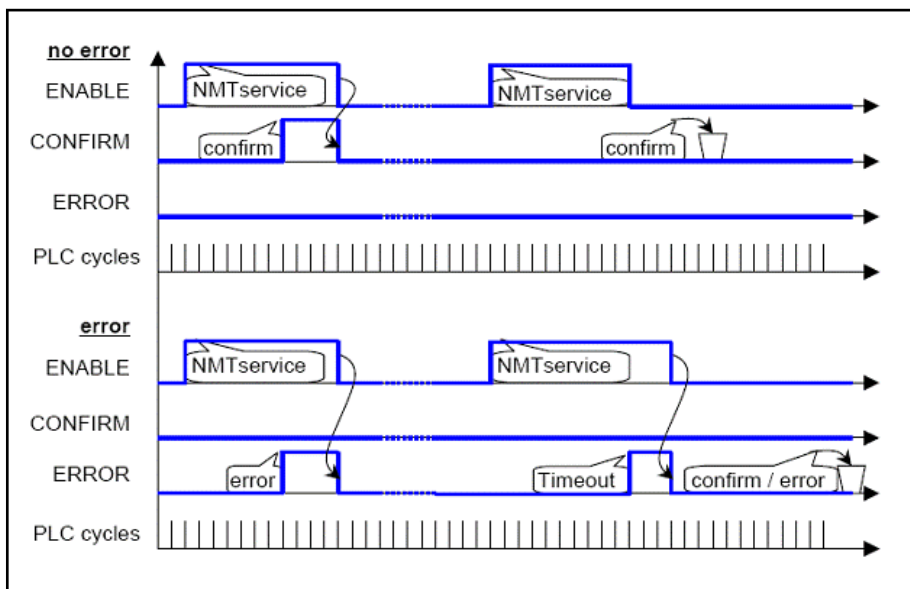


Fig. 7-19: NMT service diagram, time diagram of the "NMT" function block

- For a more detailed description of the responses of ENABLE, CONFIRM and ERROR, refer to [Control mechanism](#), page 46.
- For information on the inputs NETWORK and TIMEOUT, refer to [Additional parameters](#), page 45.
- **NMTservice:** The NMT service result is the query to the addressed node (DEVICE=0 indicates: all nodes) to make the desired state changes.

When using **DEVICE = 0**, an error can occur if an NMT master or device does not support all commands. In this case, the flag **TRANSITION_STATE.ALL_EXCEPT-NMT-AND-SENDER** can be set for the **STATE** input if the NMT command should not address the NMT master and the device to be sent.

- **confirm:** The event “confirm” positively confirms that the service was processed without errors, e.g. the CANopen kernel returns no_error to the function block.
- **error:** The event “error” negatively confirms that the service was not processed without errors, e.g. the CANopen kernel returns an error to the function block.
- **Timeout:** The time control starts at a rising edge at “ENABLE”. A timeout event occurs if not confirmation was returned to the function block before the time control reached the timeout value. In case of a timeout, **ERROR = 3**.

RECV_EMCY_DEV

Brief description

The function block checks whether an emergency object has been received from **DEVICE**.

The function block extends [CIA405BASE](#), page 48.

Interface description

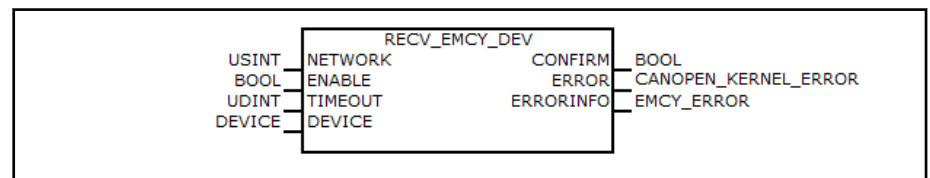


Fig. 7-20: Interface of the RECV_EMCY_DEV function block

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	Refer to CIA405BASE , page 48
	ENABLE	BOOL	Refer to CIA405BASE , page 48
	TIMEOUT	UDINT	Refer to CIA405BASE , page 48
	DEVICE	DEVICE	Addressed device; 0 means local device DEVICE , page 42
VAR OUTPUT	CONFIRM	BOOL	Refer to CIA405BASE , page 48
	ERROR	CANOPEN_KERNEL_ERROR	Refer to CIA405BASE , page 48
	ERRORINFO	EMCY_ERROR	EMCY code EMCY_ERROR , page 42

Tab. 7-30: Interface of the RECV_EMCY_DEV function block

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NETWORKS	1	0/1 at ENABLE

CAA_CiA405.library

Name	Type	Min. value	Max. value	Default value	Effective
TIMEOUT	UDINT	0	...	0	0/1 at ENABLE
DEVICE	DEVICE	0	127	0	0/1 at ENABLE

Tab. 7-31: Input behavior

Functional description

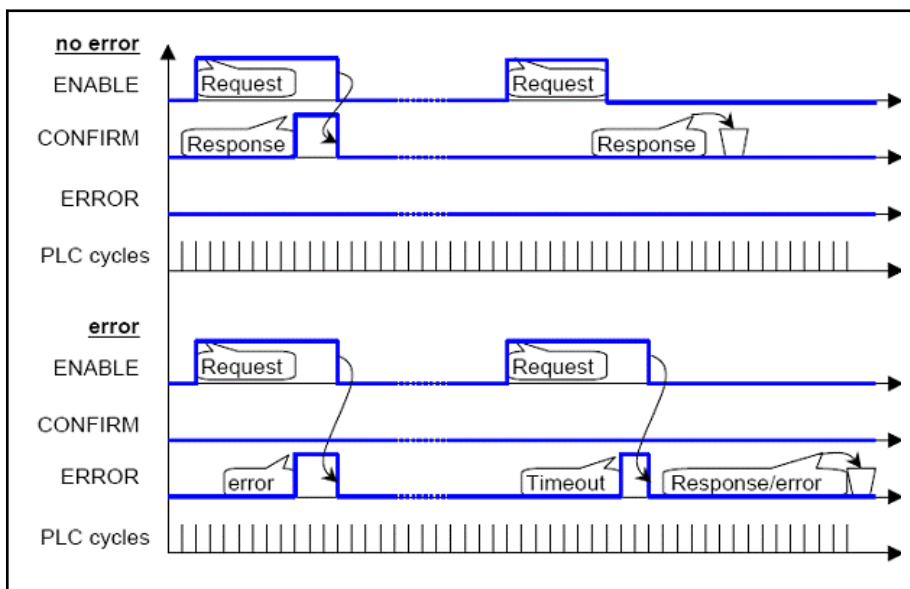


Fig. 7-21: Request/response diagram, time diagram of the RECV_EMCY_DEV function block

The function block checks whether an emergency object (/ DS-301/) has been received from DEVICE.

If the function block executed its action without any errors, the output CONFIRM is set to TRUE and ERROR is set to FALSE.

ERRORINFO contains information on the received emergency.

An emergency was received if the error code of ERRORINFO is not 0. An error code with a value of 0 means that there is currently no emergency in the device.

A received emergency is not deleted and is transmitted upon each function block call until the device has sent an emergency recovery.

If an error occurs while checking an emergency, CONFIRM is set to FALSE and ERROR is set to the respective error value.

- For a more detailed description of the responses of ENABLE, CONFIRM and ERROR, refer to [Control mechanism, page 46](#).
- For information on the inputs NETWORK and TIMEOUT, refer to [Additional parameters, page 45](#).
- **Response:** This result is a positive confirmation that the query has been processed successfully, e.g. the EMCY value is available at the ERRORINFO output.

If no EMCY has been received, ERRORINFO = 0, i.e. all elements of the structure [EMCY_ERROR, page 42](#), are zero.

- **error:** This event is a negative confirmation and means that the query cannot be processed without error, e.g. the CANopen kernel returns an error when accessing the value of EMCY. In this case, ERRORINFO = 0.

Also refer to [CANOPEN_KERNEL_ERROR, page 42](#)

- **Timeout:** The time control starts at a rising edge at "ENABLE". A timeout event occurs if not confirmation was returned to the function block before the time control reached the timeout value.

In case of a timeout, ERROR = 3 and ERRORINFO = 0.



The function block should transmit the EMCY of the checked device upon each call.

Often, the earliest EMCY contains the original error information; later EMCYs often result from sequential errors.

In a subordinate software level, a receiving EMCY does not overwrite the EMCYs different from no_error (equal to "Reset Error").

But an incoming no_error EMCY always overwrites the saved EMCY.

RECV_EMCY

Brief description

The function block checks whether an emergency object has been received from any DEVICE.

The function block extends [CIA405BASE](#), page 48.

Interface description

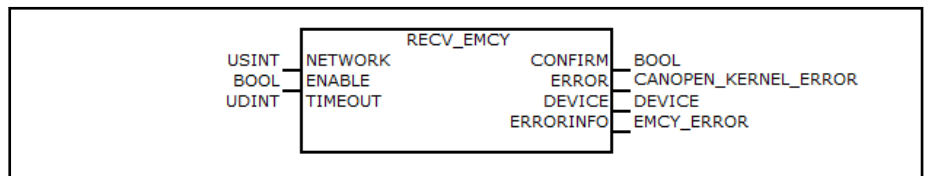


Fig. 7-22: Interface of the RECV_EMCY function block

	Name	Type	Description
VAR_INPUT	NETWORK	USINT	Refer to CIA405BASE , page 48
	ENABLE	BOOL	Refer to CIA405BASE , page 48
	TIMEOUT	UDINT	Refer to CIA405BASE , page 48
VAR OUTPUT	CONFIRM	BOOL	Refer to CIA405BASE , page 48
	ERROR	CANOPEN_KERNEL_ERROR	Refer to CIA405BASE , page 48
	DEVICE	DEVICE	Device receiving the emergency object 0 means local device DEVICE , page 42
	ERRORINFO	EMCY_ERROR	EMCY code EMCY_ERROR , page 42

Tab. 7-32: Interface of the RECV_EMCY function block

Name	Type	Min. value	Max. value	Default value	Effective
ENABLE	BOOL	FALSE	TRUE	FALSE	
NETWORK	USINT	1	MAX_NETWORKS	1	0/1 at ENABLE
TIMEOUT	UDINT	0	...	0	0/1 at ENABLE

Tab. 7-33: Input behavior

CAA_CiA405.library

Functional description

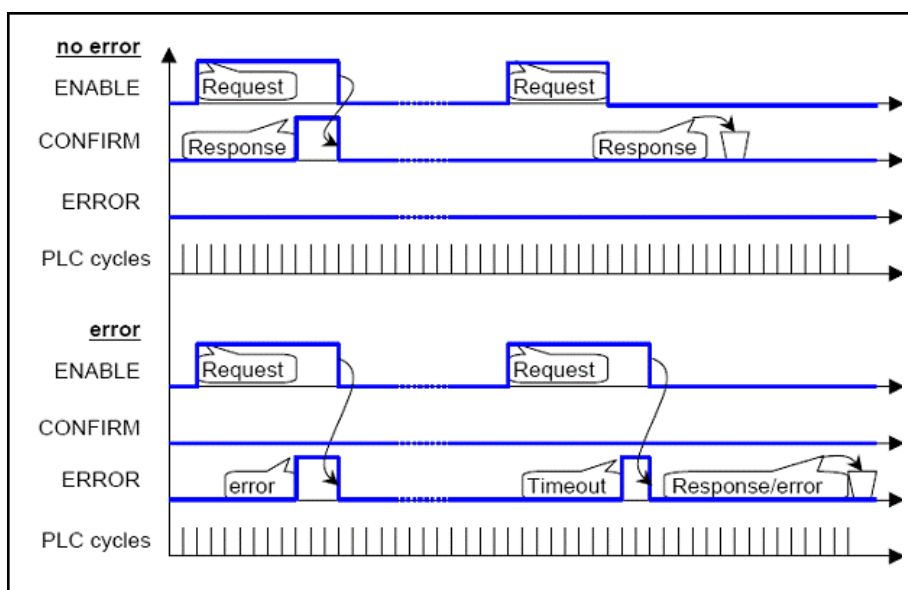


Fig. 7-23: Request/response diagram, time diagram of the RECV_EMCY function block

The function block checks whether an emergency object (/ DS-301/) has been received from any DEVICE.

If the function block executed its action without any error, the output CONFIRM is set to TRUE and ERROR is set to FALSE.

ERRORINFO contains information on the received emergency and DEVICE is set to the ID of the device sending the emergency object.

An emergency was received if DEVICE and the error code of ERRORINFO are not 0.

If an error occurs while checking an emergency, CONFIRM is set to FALSE and ERROR is set to the respective error value.

- For a more detailed description of the responses of ENABLE, CONFIRM and ERROR, refer to [Control mechanism, page 46](#).
- For information on the inputs NETWORK and TIMEOUT, refer to [Additional parameters, page 45](#).
-

Response: This result is a positive confirmation that the query has been processed successfully, e.g. the EMCY value received by the DEVICE is available at the ERRORINFO output.

If no EMCY has been received, ERRORINFO = 0, i.e. all elements of the structure [EMCY_ERROR, page 42](#), are zero.

error: This event is a negative confirmation and means that the query cannot be processed without error, e.g. the CANopen kernel returns an error when accessing the value of EMCY. In this case, ERRORINFO = 0.

Timeout: The time control starts at a rising edge at "ENABLE". A timeout event occurs if not confirmation was returned to the function block before the time control reached the timeout value.

In case of a timeout, ERROR = 3 and ERRORINFO = 0.



To transmit all EMCY messages in a loop upon each function block call and to report at least once a "Reset Error" EMCY, one memory can be administered for each device for received EMCY messages in a subordinate software level, as described in the recommendation for [RECV_EMCY_DEV](#), page 65, .

The RECV_EMCY function block scans the EMCY memory in a loop across all existing devices.

If, during that scan, a device was found that included an EMCY deviating from no_error in its EMCY memory, EMCY is transferred.

If a device is found which received an EMCY deviating from no_error during the last scan and its EMCY memory now includes no_error, this no_error EMCY is transferred.

Each device, a no_error has been transmitted for is skipped during the scan so that a no_error EMCY can only be transmitted once.

The search is continued upon the next execution. This "Reset Error" EMCY was only transmitted once, e.g. DEVICE > 0 with ERRORINFO = 0 to inform the user that a pending EMCY was deleted by the device through receiving a "Reset Error".

The function block transmits DEVICE = 0 and ERRORINFO = 0 after a complete scan of all EMCY memories of all devices to inform the user about the error-free state of all devices.

8 Service and support

Our worldwide service network provides an optimized and efficient support. Our experts offer you advice and assistance should you have any queries. You can contact us **24/7**.

Service Germany Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Hotline** and **Service Helpdesk** under:

Phone: **+49 9352 40 5060**
Fax: **+49 9352 18 4941**
E-mail: service.svc@boschrexroth.de
Internet: <http://www.boschrexroth.com/>

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

Index

A

Abbreviations.....	8
About this documentation.....	5
Validity of the documentation.....	5
ANSI Z535.6-2006.....	6

B

Bus coupler.....	8
------------------	---

C

CAA_CiA405.library.....	39
CAN bus configuration editor.....	13
CANopen	
Information.....	26
CANopen local device.....	32
General.....	32
I/O image.....	34
Information.....	34
Parameters.....	34
Status.....	34
CANopen manager	
Information.....	20
Status.....	20
CANopen master.....	15
CAN bus parameters.....	16
Information.....	16
Status.....	16
CANopen module.....	26
I/O image.....	26
Information.....	27
Parameters.....	26
Status.....	27
CANopen remote device.....	20
I/O image.....	25
Parameters.....	25
PDOs.....	22
SDOs.....	24
Status.....	26
CANopen slave.....	30
CAN bus parameters.....	31
Information.....	31
Status.....	31
CANOPEN_KERNEL_ERROR.....	42
CANOPEN_KERNEL_ERROR_CODES.....	43
CIA405BASE.....	48
Communication and field buses	
Features of the field bus.....	9

D

DEVICE.....	42
Device database	
Add devices.....	36
Device database dialog.....	35
Device database, overview.....	35
Manage device database.....	35

Remove devices.....	37
DEVICE_STATE.....	40
Documentation structure.....	5

E

EDS file.....	8
EMCY_ERROR.....	42

G

GET_CANOPEN_KERNEL_STATE.....	62
GET_LOCAL_NODE_ID.....	59
GET_STATE.....	60

H

Hazard warning.....	6
Helpdesk.....	71
Hotline.....	71

I

Information representation	
Terms and abbreviations.....	8

L

Libraries	
CAA_CiA405.library.....	39

M

Module.....	8
-------------	---

N

NMT.....	63
----------	----

P

Postinstalling	
CAN device descriptions.....	11

R

RECV_EMCY.....	67
RECV_EMCY_DEV.....	65

S

Safety alert symbol.....	6
Safety instructions.....	6
SDO_ERROR.....	43
SDO_MODE.....	41
SDO_READ.....	58
SDO_READ_DATA.....	55
SDO_READ4.....	53
SDO_WRITE.....	57
SDO_WRITE_DATA.....	52
SDO_WRITE4.....	49
Service hotline.....	71
Signal words.....	6
Support.....	71

Index

Symbols used..... 7

T

Terminal..... 8
TRANSITION_STATE..... 42

W

Warnings..... 6

Notes

Notes

Notes

Bosch Rexroth AG

Electric Drives and Controls

P.O. Box 13 57

97803 Lohr, Germany

Bgm.-Dr.-Nebel-Str. 2

97816 Lohr, Germany

Phone +49 9352 18 0

Fax +49 9352 18 8400

www.boschrexroth.com/electrics



R911344488