

### CANopen in CODESYS V3

This document describes what each CAN parameter in the CODESYS V3 programming environment does, as well as show the available Function Blocks to implement diagnostics or parameterization within the PLC's Task.

CODESYS V3,  
CPX-FB14,  
CTEU-CO,  
CMMP-...-M0/M3,  
EMCA-CO

Title ..... CANopen in CODESYS V3  
Version ..... 1.10  
Document no. .... 100150  
Original .....en  
Author ..... Festo  
  
Last saved ..... 01.08.2017

## Copyright Notice

This documentation is the intellectual property of Festo AG & Co. KG, which also has the exclusive copyright. Any modification of the content, duplication or reprinting of this documentation as well as distribution to third parties can only be made with the express consent of Festo AG & Co. KG.

Festo AG & Co KG reserves the right to make modifications to this document in whole or in part. All brand and product names are trademarks or registered trademarks of their respective owners.

## Legal Notice

Hardware, software, operating systems and drivers may only be used for the applications described and only in conjunction with components recommended by Festo AG & Co. KG.

Festo AG & Co. KG does not accept any liability for damages arising from the use of any incorrect or incomplete information contained in this documentation or any information missing therefrom.

Defects resulting from the improper handling of devices and modules are excluded from the warranty.

The data and information specified in this document should not be used for the implementation of safety functions relating to the protection of personnel and machinery.

No liability is accepted for claims for damages arising from a failure or functional defect. In other respects, the regulations with regard to liability from the terms and conditions of delivery, payment and use of software of Festo AG & Co. KG, which can be found at [www.festo.com](http://www.festo.com) and can be supplied on request, shall apply.

All data contained in this document do not represent guaranteed specifications, particularly with regard to functionality, condition or quality, in the legal sense.

The information in this document serves only as basic information for the implementation of a specific, hypothetical application and is in no way intended as a substitute for the operating instructions of the respective manufacturers and the design and testing of the respective application by the user.

The operating instructions for Festo products can be found at [www.festo.com/sp](http://www.festo.com/sp).

Users of this document (application note) must verify that all functions described here also work correctly in the application. By reading this document and adhering to the specifications contained therein, users are also solely responsible for their own application.

# Table of contents

<b>1</b>	<b>Objective</b>	<b>4</b>
<b>2</b>	<b>Components/Software used</b>	<b>5</b>
<b>3</b>	<b>CODESYS V3 – Setting up the CANopen Network</b>	<b>6</b>
3.1	Adding the CANbus and CANopen Manager	6
3.2	Adding CANopen devices	7
3.2.1	Installing the EDS File	8
<b>4</b>	<b>CANopen Network Parameters</b>	<b>11</b>
4.1	Basics	11
4.1.1	Baudrate or data transmission speed	11
4.1.2	Node ID	12
<b>5</b>	<b>CANbus Parameters</b>	<b>12</b>
5.1	General Tab	12
<b>6</b>	<b>CANopen Manager Parameters</b>	<b>13</b>
6.1	General Tab	13
6.1.1	General Parameters	13
6.1.2	Guarding Parameters	15
6.1.3	Sync Parameters	15
6.1.4	Time Parameters	16
<b>7</b>	<b>CANopen device parameters</b>	<b>17</b>
7.1	General Tab	17
7.1.1	General Parameters	17
7.1.2	Nodeguarding Parameters	19
7.1.3	Emergency	19
7.1.4	TIME Parameters	19
7.1.5	Checks at Startup	20
7.2	Process Data Objects Tab	20
7.2.1	Enabling PDOs	20
7.2.2	Modifying PDOs	23
7.2.3	PDO Properties	25
7.3	Service Data Objects Tab	28
7.3.1	Initial configuration via SDOs	29
7.3.2	CPX initial configuration via SDOs	30
<b>8</b>	<b>Network Management (NMT) and Diagnostics in CODESYS V3</b>	<b>34</b>
8.1	Reading the device state	34
8.1.1	Property <DeviceName>.CANopenState	34
8.1.2	GET_STATE Function Block	35
8.2	Network Management Function Block	36
8.3	EMCY handling Function Blocks	38
8.3.1	RECV_EMCY_DEV	38
8.3.2	RECV_EMCY	40
8.4	SDO handling Function Blocks	43
8.4.1	SDO_READ4/SDO_WRITE4	43
8.4.2	SDO_READ_DATA/SDO_WRITE_DATA	44

## **1 Objective**

This document is aimed towards people that have a medium to high knowledge of CANopen networks.

It goes through great detail on what the CANopen parameters in CODESYS V3 are, what the programmer is able to do with them and expect from them, as well as describing the CANopen Function Blocks available in CODESYS V3 for further Network Management and Diagnostics within the running Task.

For the following images a CECC-LK PLC has been used as a CANopen Master, but the description apply for all Festo CANopen Masters in CODESYS V3.

## 2 Components/Software used

Type/Name	Version Software/Firmware
CODESYS V3	SP7
CECC-LK	Rev. 4 / FW 2.3.8.0.9242 / TSP3.5.7.159
CPX-FB14	Rev. 20
CMMP-AS-C2-3A-M3	FW 4.01501.2.3
CTEU-CO	Rev 03
EMCA-CO	FW 1.2.0.8

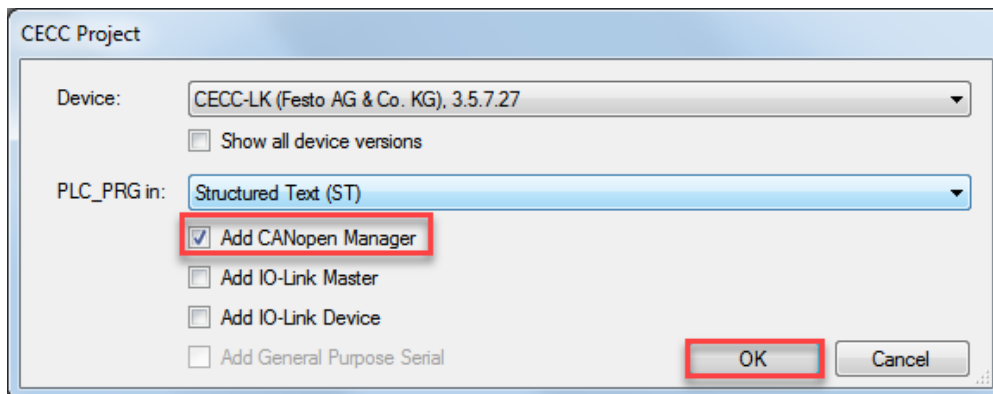
Table 2.1: 1      Components/Software used

### 3 CODESYS V3 – Setting up the CANopen Network

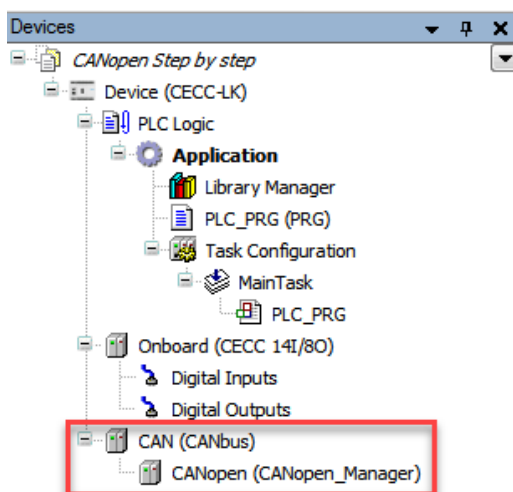
The following chapter will show how to add the CANbus, CANopen Manager and CANopen slave devices in the CODESYS V3 environment as well as give instructions on how to install the EDS file in case the slave device is missing from the Device Repository.

#### 3.1 Adding the CANbus and CANopen Manager

When starting a new project, after selecting a PLC, the Project options will pop up. Select the corresponding PLC FW version and select the Add CANopen Manager option.



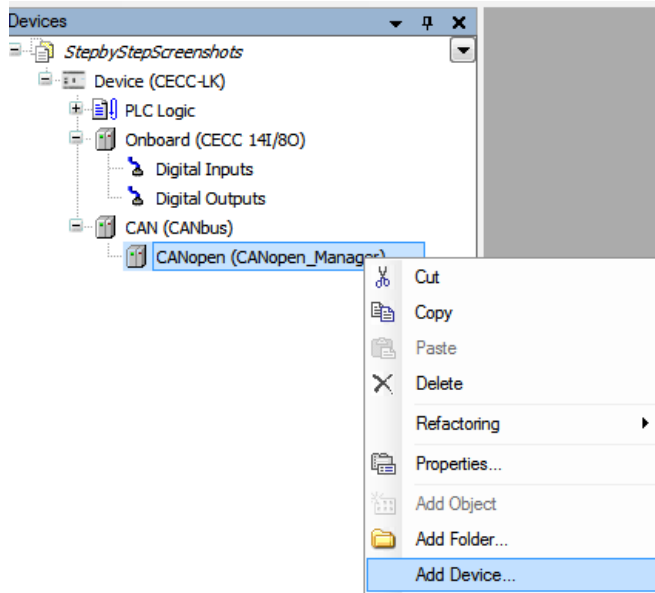
After clicking OK, the CANbus and CANopen Manager should be visible on the Devices window.



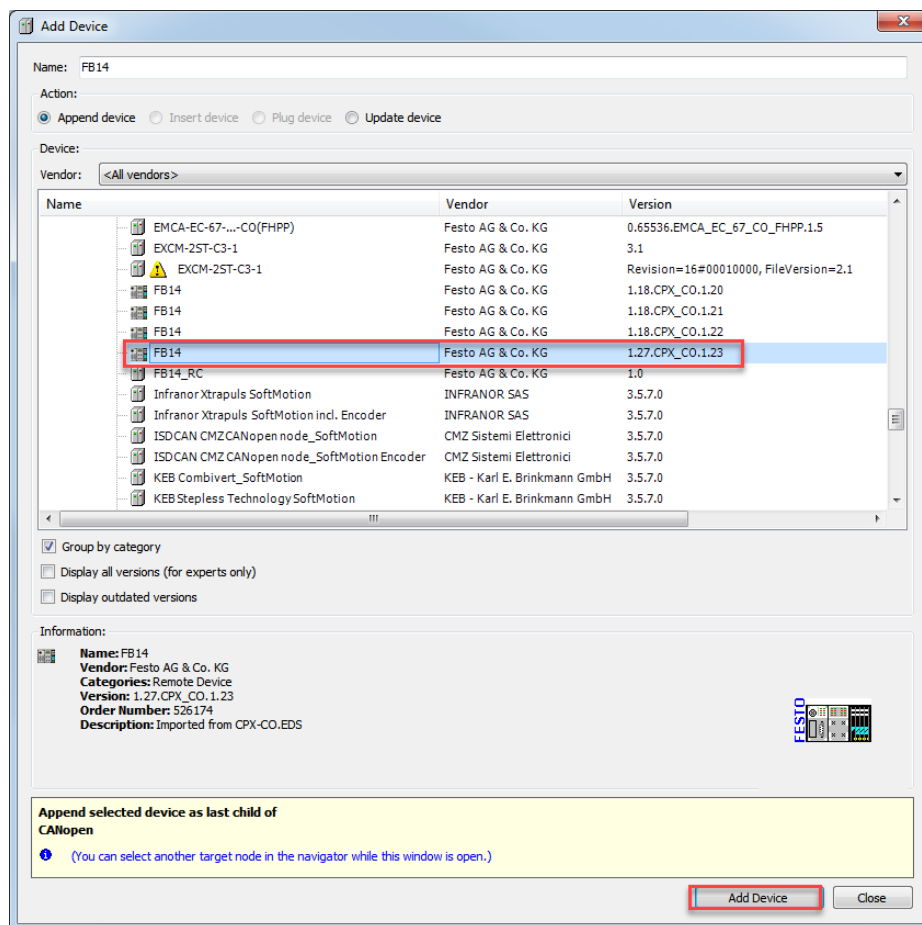
### 3.2 Adding CANopen devices

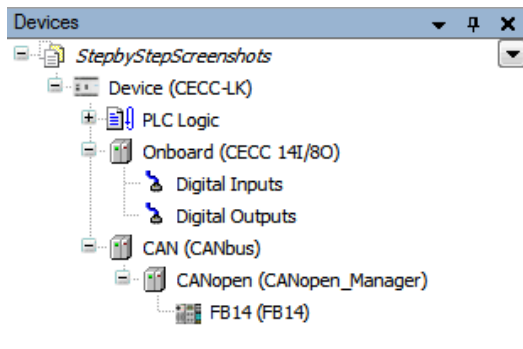
When working with CODESYS V3 provided by Festo, some CANopen devices are already integrated in the software.

To add an available device, right click on the CANopen Manager and select the Add device... option.



Highlight the needed device, and click on the Add Device button.





If the CANopen device is not located on this list, the EDS file must be installed in the software.

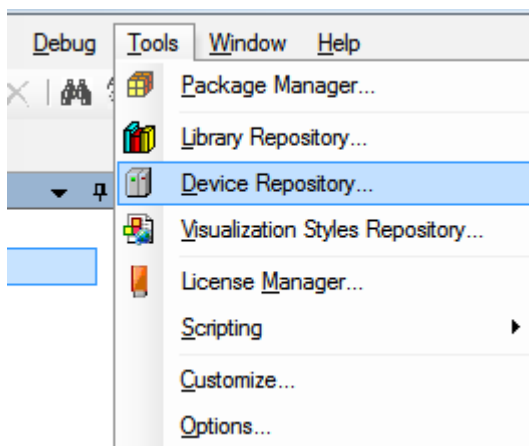
### 3.2.1 Installing the EDS File

The Electronic Data Sheet (EDS) describes the functionality of a CANopen device in a standardized manner. To add a CANopen slave in our network we have to install the EDS file in CODESYS V3.

The EDS file for Festo devices can be found in the Festo webpage, under the Software Tab.

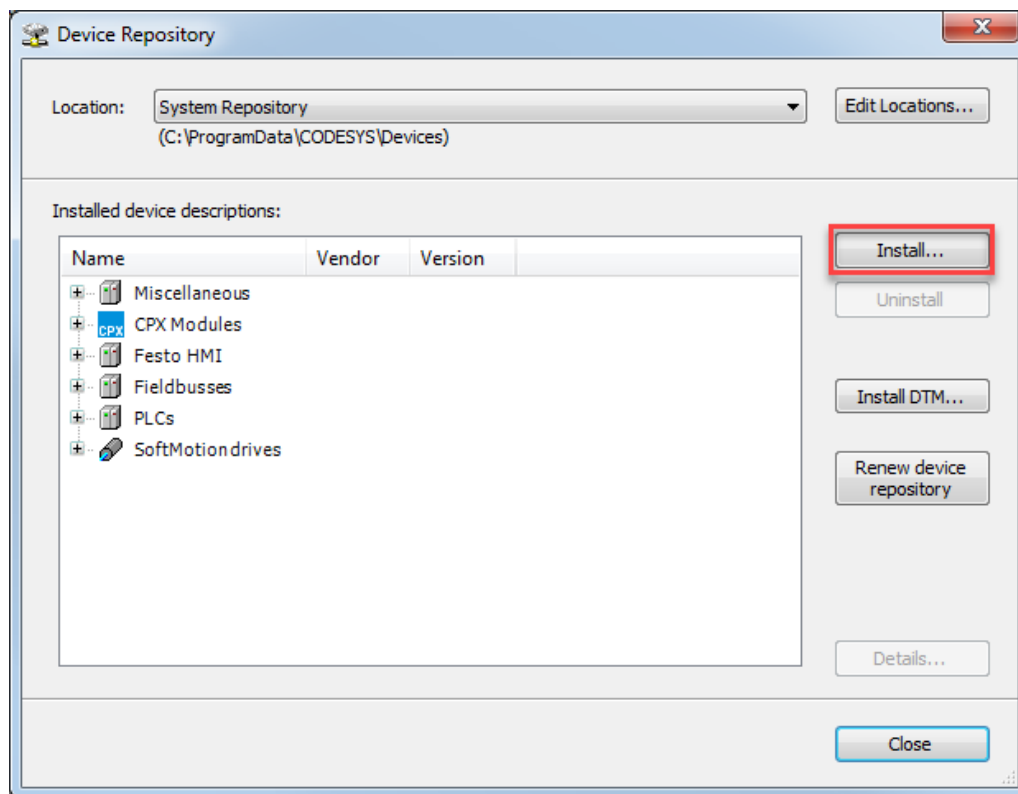
Top 3	Product information [25]	Technical documentation [2]	Certificates [3]	Software [3]	Expert knowledge [1]	Training [0]
Description		Version		Filter result		
<b>CANopen EDS</b>		3		→ Device Description Files		
<input type="checkbox"/> EDS file for the CTEU-CO CANopen bus module. Usable for valve terminals with the Festo I-Port interface as: e.g. VTUB-12, VTUG,...		7/4/2014		→ File and language versions		
<b>Supported systems:</b>		<ul style="list-style-type: none"> <li>Bus node CTEU-CO (570038) Revision 04</li> </ul>		★★★★★ (7)		

To install open CODESYS V3 and go to Tools -> Device Repository.

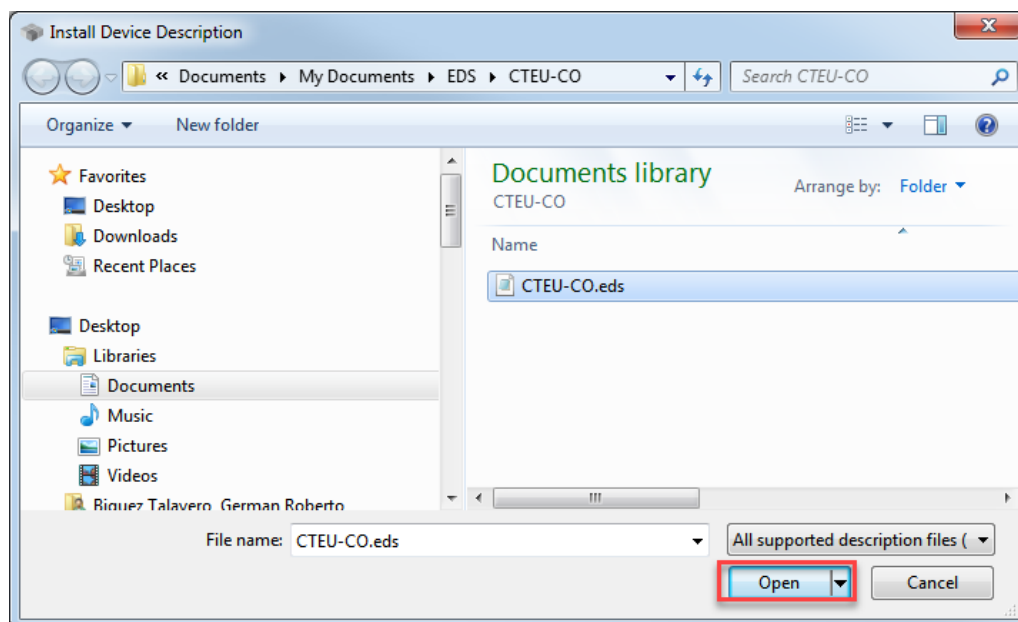


Click on the install button.

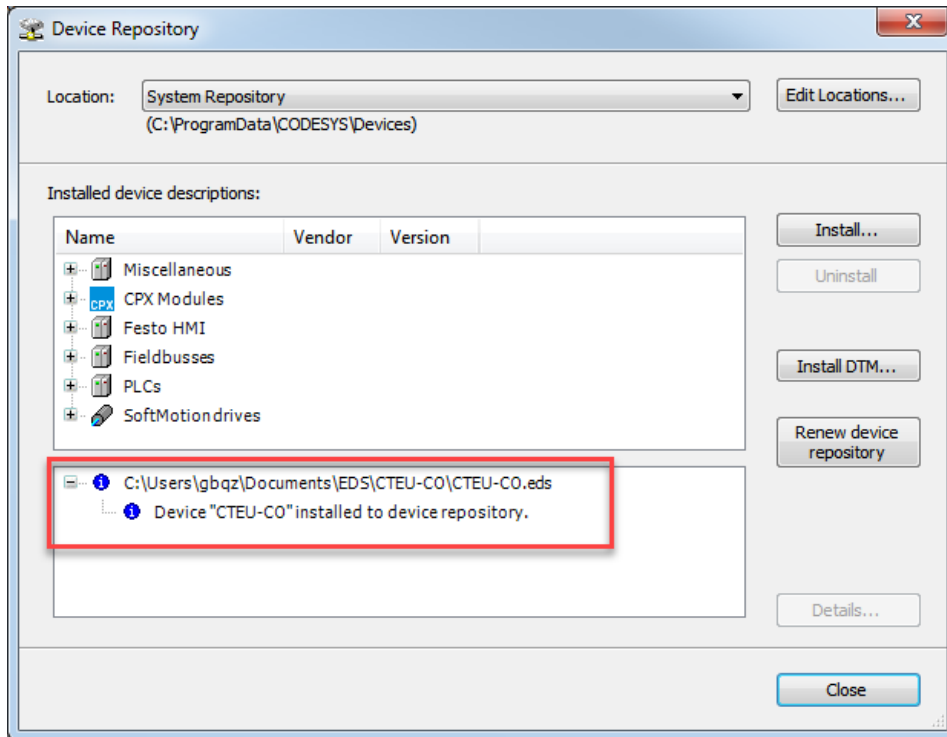




Look for the downloaded EDS file, select it and click on Open.



When the installation has been successful the following message will appear.



## 4 CANopen Network Parameters

Adding the devices into the project will display the parameters available for modification. These parameters are read from the EDS file that has been installed in CODESYS.

This chapter gives an overview on the 2 parameters that are considered basic for any CANopen network: the defined Baudrate and the Node ID of each device in the network. Also what can be parameterized in the CANbus element.

### 4.1 Basics

#### 4.1.1 Baudrate or data transmission speed

When selecting a baudrate the following criteria must be fulfilled:

- All network devices must be set to operate on the same baudrate
- The bus length must be taken into consideration

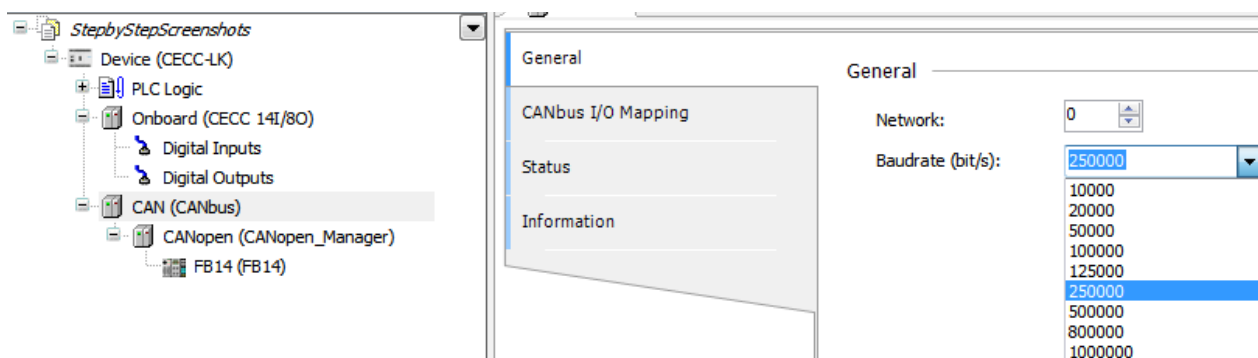
#### Speed/Cable length

Bit rate (Kbit/s)	Bus length (m)
1000	20*
800	50
500	100
250	250
125	500
50	1000
20	2500
10	5000

\*) A figure of 40m at 1 Mbit/s is often found in the CAN literature. This does not apply to networks with optically isolated CAN controllers.

**IMPORTANT:** A CANopen device doesn't have to support all Baud Rates!  
All supported Baud Rates of a Device are listed in its EDS file

On CODESYS V3, this parameter must be set on the CANbus element.



For the CANopen devices this speed can be set via switches or via software.

## 4.1.2 Node ID

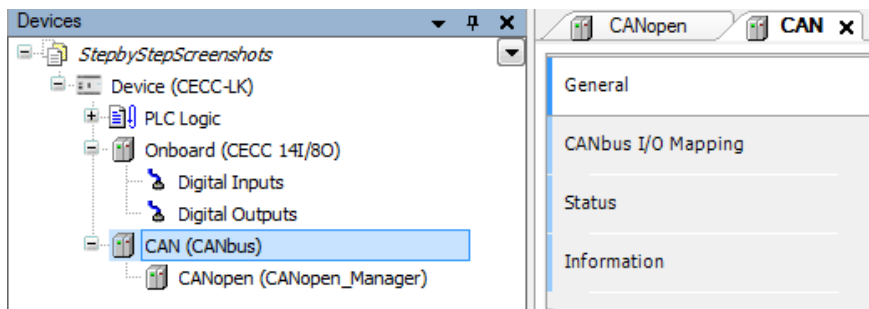
The Node ID is the identification number for each element in the network. It must be unique to each network participant and can't be repeated within the same network.

The number can be set to any value from 1 to 127, being 1 the node ID with the highest priority in the network and 127 the node ID with the lowest priority in the network.

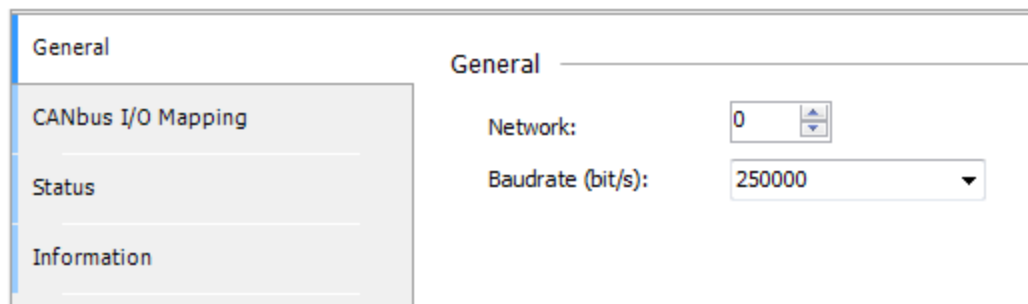
The Node ID is normally set in the slave device through some DIL switch combination or a parameterization software. This number must be matched on the CODESYS V3 project.

## 5 CANbus Parameters

This chapter shows which parameters are available in the CANbus element of CODESYS V3.



### 5.1 General Tab



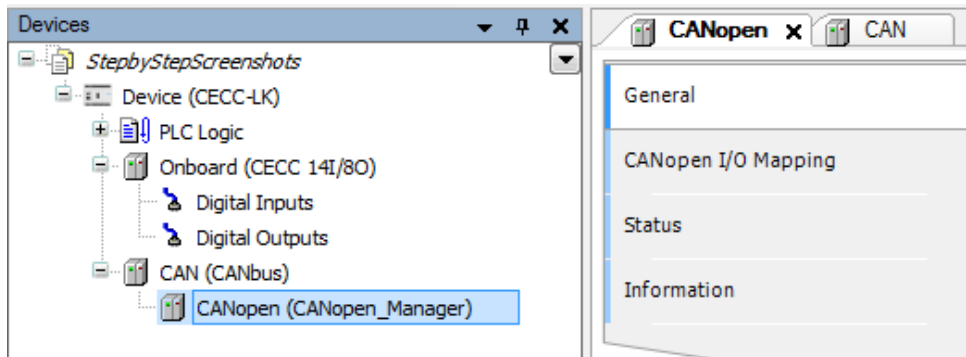
**Network** refers to the assigned CAN interface being used. If only one interface is available, network number must be 0. There are no current Festo Devices within CODESYS V3 that have 2 CAN cards, so it should always be set to 0.

**Baudrate (bit/s)** refers to the data transmission speed in which the CAN network will operate. See section 3.1.1

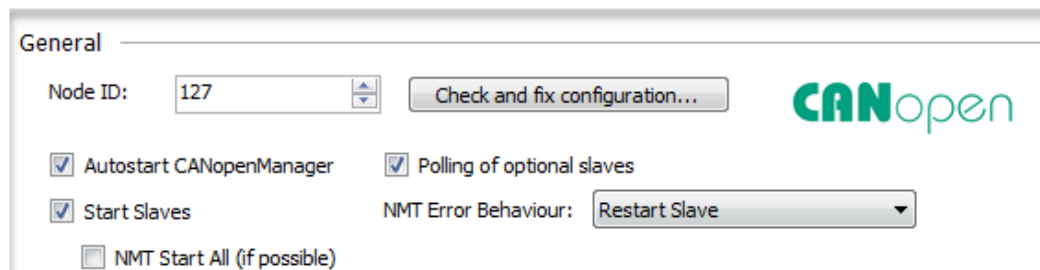
## 6 CANopen Manager Parameters

This chapter will explain what each parameter located in the CANopen Manager does and what reaction can be expected from having each one selected.

### 6.1 General Tab



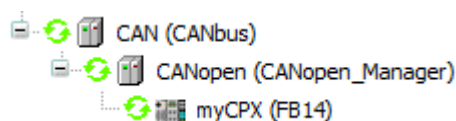
#### 6.1.1 General Parameters



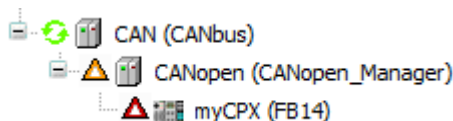
**Node ID** is the identification number of the CANopen Master in the network. It must be set between a value of 1 to 127.

**Check and fix configuration** will verify if there are conflicts between the Node ID's or COB-ID's in the network. If Sync is enabled it will also check that the Cycle Period and Window length times are achievable within the Task where the CAN network is being executed.

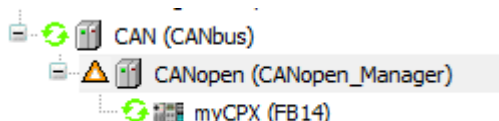
**Autostart CANopenManager** sets the CANopen Manager to OPERATIONAL if all mandatory slaves are ready.



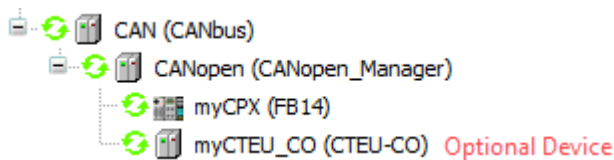
If mandatory slaves are not ready, the CANopen Manager will remain in PRE-OPERATIONAL state.



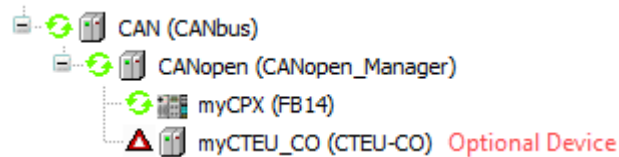
If deactivated, the CANopen Manager will remain in PRE-OPERATIONAL state until started by the application with the CiA405 NMT block. CANopen Slave devices will go to OPERATIONAL state if Start slaves or Polling for optional slaves is checked, but no PDOs will be transmitted.



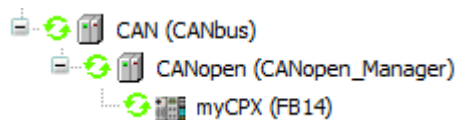
**Polling of optional slaves** a slave which did not respond during boot sequence is polled every second until a successful response is received.



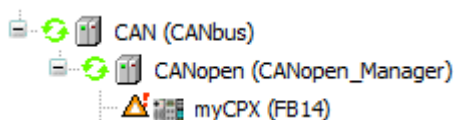
If the option is deactivated, an optional slave will only be detected if it sends a boot-up telegram. The Optional slave can be initialized via the CiA405 NMT block.



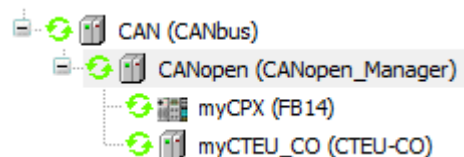
**Start slaves** means the CANopen Manager is responsible for starting the slaves. When the CANopen slave devices are in PRE-OPERATIONAL state, the CANopen Manager will send a “Start-Remote-Node” request.



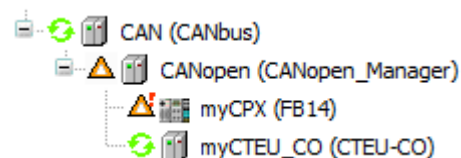
If deactivated, CANopen Slave devices will remain in PRE-OPERATIONAL state until a “Start-Remote-Node” request is sent via the CiA 405 NMT block.



**NMT Start All (if possible).** If active the CANopen Manager will start all available slaves with a single “Start All” request. This results in all mandatory slaves going to OPERATIONAL mode at practically the same time.



If deactivated, the CANopen Manager will send an individual “Start-Remote-Node” request as each slave becomes available. Devices become available at different times depending on how long they take to power up and set their internal parameters. Adding to this time are the number of SDOs that must be transmitted during startup.



NOTE – The picture above displays myCTEU\_CO has already received the “Start-Remote-Node” request and is in OPERATIONAL state, and myCPX that is in PRE-OPERATIONAL state waiting for its “Start-Remote-Node” request.

**NMT Error Behaviour** defines the reaction to a guard event. Guard event means that either the Node Guarding or Heartbeat times have expired.

- **Reset Slave** after a guard event the slave will be automatically restarted (NMT Reset + SDO configuration + NMT start) by the stack. This means that as soon as the Slave device becomes available, communication will be re-established.
- **Stop Slave** after a guard event the slave will be stopped. In this case, even if the slave device becomes available, he will remain in PRE-OPERATIONAL state until a “Start-Remote-Node” request is sent with the CiA405 NMT block.

### 6.1.2 Guarding Parameters

The CANopen Manager is only able to produce Heartbeat pulses. Node Guarding can only be implemented in slave devices.

▲ Guarding

☒ Enable Heartbeat Producing

Node ID:

Producer Time (ms):

**Enable Heartbeat Producing.** If enabled, the master will send heartbeats according to the defined **Producer Time** interval.

**Node ID** Identifier of the heartbeat producer. This number must match the Node ID of the heartbeat producing element, in this case, the CANopen Master Node ID.

**Producer Timer (ms)** Heartbeat interval in milliseconds.

### 6.1.3 Sync Parameters

▲ Sync

☒ Enable Sync Producing

COB-ID (Hex): 16#

Cycle Period (µs):

Window Length (µs):

☐ Enable Sync Consuming

**Enable Sync Producing.** When enabled the CANopen Manager will send SYNC telegrams for the Sync-Consumers in the network. Synchronous tasks are then executed after the SYNC telegram has been received.

The SYNC telegram is sent with the defined **COB-ID** once every **Cycle Period (µs)**.

The **Cycle Period (µs)** must have a value large enough to allow all synchronous data transmission to take place between SYNC telegrams.

This value must also be a number divisible by the Task Cycle Time of the PLC where the Bus Task is being executed. If my Task Cycle Time is 10ms, my **Cycle Period** must be any multiple of 10ms.

Different telegrams have different priorities according to their type and the Node ID of the participant in the network. High priority messages will most likely be sent when available, but low priority messages could be delayed due to this.

To ensure that low priority asynchronous telegrams can be transmitted, a **Window Length (µs)** must be set. The result of this is that Sync telegrams can only be transmitted within the **Window Length (µs)**, leaving some space for the lower priority asynchronous messages to be transmitted.

For Festo Motor Controllers it is recommended to use a **Window Length** of approx. 75% of the **Cycle Period**. The window length time can never be bigger than the Cycle Period.

**Enable Sync Consuming** means a device other than the CANopen Manager will generate the SYNC messages. The SYNC telegram parameter must always be set in the CANopen Manager window, independent of who will be generating them.

NOTE – As of now there are no Festo devices that are able to produce SYNC telegrams, other than the Master PLCs.

### 6.1.4 Time Parameters

▲ TIME

☒ Enable TIME Producing

COB-ID (Hex): 16#

Producer Time (ms):

If supported, TIME telegrams can be used to keep devices clocks synchronized under the TIME producer's clock. This means that the TIME telegram will be used by the slave device instead of its internal clock.

**Enable TIME Producing.** The CANopen Manager will send TIME telegrams.

**COB-ID (Hex):** Communication Object Identifier which defines the TIME telegram.

**Producer Time (ms)** defines the interval in milliseconds during which the TIME telegram is sent. Must be a multiple of the task cycle time.

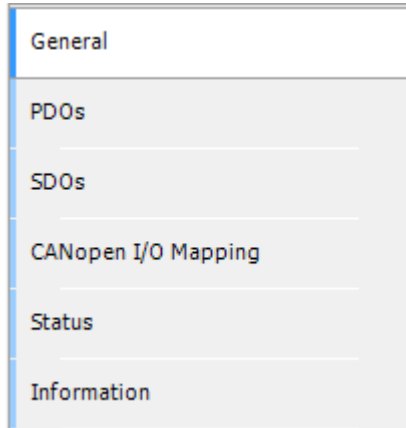
NOTE - As of now there is no Festo CANopen device that supports the TIME functions for clock synchronization.



## 7 CANopen device parameters

This chapter will explain what each parameter located in the CANopen Slave Device does and what reaction can be expected from having each one selected. The chapter will also describe how to parameterize PDO and SDO objects in CODESYS V3, and provide some examples on when this can be useful.

### 7.1 General Tab



#### 7.1.1 General Parameters



If **Enable Expert Settings** is not selected, we will only be able to configure the Node ID on the General Parameters area.

If selected, the parameterization of SDO Channels, Optional Device, No initialisation and Reset Node will be made available.



**Node ID** is the identification number of the CANopen node in the network. It must be set between a value of 1 to 127, matching the settings of the Node.

**SDO Channels** This button opens a dialog where Service Data Objects (SDO) channels can be defined. An SDO establishes a peer-to-peer communication channel between two devices (SDO server or client channel) to allow entry in the CANopen object dictionary.

The first SDO server channel is predefined and must be supported by all CANopen devices.

**NOTE** - As of now there are no Festo CANopen devices that support more than 1 SDO Channel. This means the only one that can access SDOs is the CANopen Manager.

**Enable Sync Producing** means the selected CANopen device will be responsible for sending the SYNC telegrams.

SYNC parameters COB-ID, Cycle Period and Window Length must be configured on the CANopen Manager tab and the “Enable Sync Consuming” option must be selected on the CANopen Manager.

**General**

Node ID: 10

☐ Enable Expert Settings

☒ Enable Sync Producing

**Sync**

☐ Enable Sync Producing

COB-ID (Hex): 16# 80

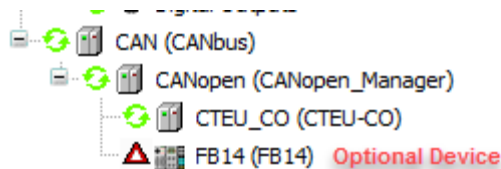
Cycle Period (µs): 50000

Window Length (µs): 20000

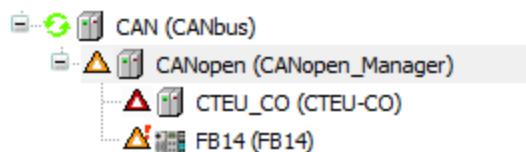
☒ Enable Sync Consuming

NOTE - As of now there are no Festo Devices which are able to be SYNC producers other than the CANopen Masters, so the Enable Sync Consuming option should NEVER be selected when only using Festo CANopen devices.

**Optional Device.** If selected, the CANopen Manager and the mandatory slaves will go to OPERATIONAL state independent of the state of the optional device.



If **Optional Device** is not selected, the device is considered mandatory in the network. The CANopen Manager will remain in PRE-OPERATIONAL and will only move to OPERATIONAL when all configured mandatory slaves are available.



NOTE – The CANopen Device’s reaction may vary according to the CANopen Manager Settings. If the option NMT Start All is selected, all available devices will remain in PRE-OPERATIONAL until all mandatory slaves in the network are available.

**No initialization** For CANopen devices that start with a valid configuration. If enabled, no SDOs and no NMT start command will be sent to the slave. The slave can only be started with a “Start-Remote-Node” request using the CiA405 NMT block.

NOTE - This option is NOT recommended for Festo devices, as most of them require some SDO parameters to be downloaded.

**Reset Node** if available, the CANopen Device communication parameters are reset to their default values. Which parameters are reset is dependent on the selected subindex:

- Sub:001: All parameters are restored.
- Sub:002: Communication-related parameters (Index 1000h - 1FFFh manufacturer-specific communication parameters) are restored.
- Sub:003: Application-related parameters (Index 6000h - 9FFFh manufacturer specific application parameters) are restored.
- Sub:004 - Sub:127: Manufacturer-specific choice of parameters is restored.
- Sub:128 - Sub254: Reserved for future use.

NOTE – Never set this option for Festo Devices, as they will lose their complete configuration. In CMMP’s case, the FCT project must be downloaded again.

### 7.1.2 Nodeguarding Parameters

Setting the guard parameters is very important if we wish to monitor the actual device state at any given moment.

It is important to note that one device can't handle Nodeguarding and Heartbeat consuming/producing at the same time, because both functions use the same COB-ID. In a network, some devices can use Nodeguarding and others Heartbeat.

NOTE - If no Guard function is implemented, after the slaves initialization, the state "Unknown" will be displayed by the master. This doesn't mean there is no communication. Simply that the state is not being monitored.

▲ Nodeguarding

☐ Enable Nodeguarding
 ☐ Enable Heartbeat Producing

Guard Time (ms): 
 Producer Time (ms):

Life Time Factor: 
 Heartbeat Consuming (0/1 active)

If **Enable Nodeguarding** is selected, a message is sent by the master every **Guard Time (ms)** to the slave device requesting the current device state, repeating it as many times as defined in the **Life Time Factor** or until there is a response from the module. If no answer is received, this module is marked as "not available".

If **Enable Heartbeat Producing** is set, the device will send its current state every **Producer Time (ms)**. This heartbeat info can be used by a **Heartbeat Consumer** to monitor the producing device.

NOTE - If allowed by the EDS file, CANopen slaves can be made Heartbeat Consumers of one another, in order to establish a certain dependency between them.

### 7.1.3 Emergency

▲ Emergency

☒ Enable Emergency

COB-ID:

**Enable Emergency** allows the use of EMCY messages with the set **COB-ID** to be reported whenever an internal failure is detected.

This messages can be retrieved by using the RECV\_EMCY\_DEF, RECV\_EMCY FB's located in the CiA405 Library. Check section 7.3 for application examples.

### 7.1.4 TIME Parameters

▲ TIME

☐ Enable TIME Producing

16#  
 COB-ID (Hex):

☐ Enable TIME Consuming

**Enable TIME Producing.** The CANopen Device will send TIME messages

**COB-ID (Hex):** Communication Object Identifier which defines the TIME stamp message.

**Enable TIME Consuming:** The selected device will synchronize its clock with the TIME producer's clock.

NOTE – There is no current Festo Device that can work with TIME telegrams.

#### 7.1.5 Checks at Startup

##### ▲ Checks at Startup

☒ Check Vendor ID    ☒ Check Product Number    ☒ Check Revision Number

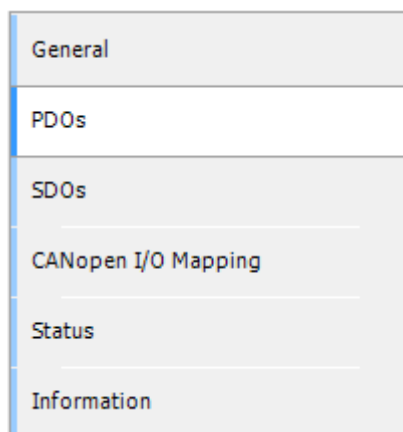
The selected information is retrieved from the CANopen Slave's object 0x1018 and compared with the EDS file information. In case of a mismatch, the configuration is stopped and the slave is not started.

This can be used to make sure a device can only be replaced by another one from the same Vendor, with the same product number, with the same revision number or any combination of the three.

NOTE - If the module reports an error during the check phase, it could be that the EDS information is not matching the actual hardware. If multiple EDS versions are available, change the EDS version to match the actual hardware or make a FW upgrade/downgrade where possible.

## 7.2 Process Data Objects Tab

The Process Data Objects (PDOs) information is read from the devices EDS file.



#### 7.2.1 Enabling PDOs

The PDOs can be enabled as required by the application. Only enabled PDOs will be transmitted.

For larger applications, where Bus traffic must be optimal and as small as possible for each element, it is recommended to only set the PDO's that are being used by simply marking out the checkmark on the left.

Name	Object	Bit length
<input checked="" type="checkbox"/> <b>16#1400: Receive PDO 1 C</b>	<b>16#206 (\$NODE)</b>	<b>64</b>
8-Bit Digital Output 1-8	16#6200:16#01	8
8-Bit Digital Output 9-16	16#6200:16#02	8
8-Bit Digital Output 17-24	16#6200:16#03	8
8-Bit Digital Output 25-32	16#6200:16#04	8
8-Bit Digital Output 33-40	16#6200:16#05	8
8-Bit Digital Output 41-48	16#6200:16#06	8
8-Bit Digital Output 49-56	16#6200:16#07	8
8-Bit Digital Output 57-64	16#6200:16#08	8
<input type="checkbox"/> <b>16#1401: Receive PDO 2 C</b>	<b>16#306 (\$NODE)</b>	<b>64</b>
Analogue Output Channel 1	16#6411:16#01	16
Analogue Output Channel 2	16#6411:16#02	16
Analogue Output Channel 3	16#6411:16#03	16
Analogue Output Channel 4	16#6411:16#04	16

### 7.2.1.1 Example with Motor Controllers

When adding a Festo Motor controller, such as CMMP or EMCA-CO, the PDO's containing FHPP+ are set ON by default.

Receive PDOs (Master => Slave)		Transmit PDOs (Slave => Master)	
<div><div><div><div>+</div><div>Add PDO</div></div><div><div>+</div><div>Add Mapping</div></div><div><div><div></div><div>Edit</div></div></div></div></div>		<div><div><div><div>+</div><div>Add PDO</div></div><div><div>+</div><div>Add Mapping</div></div><div><div><div></div><div>Edit</div></div><div><div><div>X</div><div>Delete</div></div></div></div></div></div>	
<div><div>Name</div><div><div><div><div><input checked="" type="checkbox"/></div><div>16#1400: RPDO 1 Communication Parameter</div></div><div>CCON</div><div>CPOS</div><div>REC_NR/CDIR</div><div>RES/DEM_VAL1/PARA1</div><div>RES/DEM_VAL2/PARA2</div></div><div><div><div><input checked="" type="checkbox"/></div><div>16#1401: RPDO 2 Communication Parameter</div></div><div>RES</div><div>SUBINDEX</div><div>REQCODE_PNU</div><div>PARAVAL</div></div><div><div><div><input checked="" type="checkbox"/></div><div>16#1402: RPDO 3 Communication Parameter</div></div><div>FHPP+_O_Byte01</div><div>FHPP+_O_Byte02</div><div>FHPP+_O_Byte03</div><div>FHPP+_O_Byte04</div><div>FHPP+_O_Byte05</div><div>FHPP+_O_Byte06</div><div>FHPP+_O_Byte07</div><div>FHPP+_O_Byte08</div></div><div><div><div><input checked="" type="checkbox"/></div><div>16#1403: RPDO 4 Communication Parameter</div></div><div>FHPP+_O_Byte09</div><div>FHPP+_O_Byte10</div><div>FHPP+_O_Byte11</div><div>FHPP+_O_Byte12</div><div>FHPP+_O_Byte13</div><div>FHPP+_O_Byte14</div><div>FHPP+_O_Byte15</div><div>FHPP+_O_Byte16</div></div></div></div>		<div><div>Name</div><div><div><div><div><input checked="" type="checkbox"/></div><div>16#1800: TPDO 1 Communication Parameter</div></div><div>SCON</div><div>SPOS</div><div>REC_NR/SDIR</div><div>RSB/ACT_VAL1</div><div>ACT_POS/ACT_VAL2</div></div><div><div><div><input checked="" type="checkbox"/></div><div>16#1801: TPDO 2 Communication Parameter</div></div><div>RES</div><div>SUBINDEX</div><div>RESPCODE_PNU</div><div>PARAVAL</div></div><div><div><div><input checked="" type="checkbox"/></div><div>16#1802: TPDO 3 Communication Parameter</div></div><div>FHPP+_I_Byte01</div><div>FHPP+_I_Byte02</div><div>FHPP+_I_Byte03</div><div>FHPP+_I_Byte04</div><div>FHPP+_I_Byte05</div><div>FHPP+_I_Byte06</div><div>FHPP+_I_Byte07</div><div>FHPP+_I_Byte08</div></div><div><div><div><input checked="" type="checkbox"/></div><div>16#1803: TPDO 4 Communication Parameter</div></div><div>FHPP+_I_Byte09</div><div>FHPP+_I_Byte10</div><div>FHPP+_I_Byte11</div><div>FHPP+_I_Byte12</div><div>FHPP+_I_Byte13</div><div>FHPP+_I_Byte14</div><div>FHPP+_I_Byte15</div><div>FHPP+_I_Byte16</div></div></div></div>	

If FHPP+ will not be used in the project, it is recommended to disable the PDO's, to reduce busload.

Operation Parameters | Factor Group **FHPP+ Editor** |  
 Integrated Drive **Axis**  
**EMCA-EC-67-M-1TM-CO** **User Defined Rotative Axis (unlimited)**

**Message from PLC** | Answer to PLC |

Message Options

Control Data	Parameter Channel		
8	16	24	32

☒ Use Parameter Channel

Operation Parameters | Factor Group **FHPP+ Editor** |  
 Integrated Drive **Axis**  
**EMCA-EC-67-M-1TM-CO** **User Defined Rotative Axis (unlimited)**

Message from PLC | **Answer to PLC** |

Message Options

Control Data	Parameter Channel		
8	16	24	32

☒ Use Parameter Channel

Receive PDOs (Master => Slave)			Transmit PDOs (Slave => Master)		
Name	Object	Bit len	Name	Object	Bit len
<input checked="" type="checkbox"/> <b>16#1400: RPDO 1 C</b>	<b>16#202 (\$NOD 64</b>	<b>64</b>	<input checked="" type="checkbox"/> <b>16#1800: TPDO 1 Communication Parameter</b>		
CCON	16#3000:16#00	8	SCON		
CPOS	16#3001:16#00	8	SPOS		
REC_NR/CDIR	16#3002:16#00	8	REC_NR/SDIR		
RES/DEM_VAL1/PARA1	16#3003:16#00	8	RSB/ACT_VAL1		
RES/DEM_VAL2/PARA2	16#3004:16#00	32	ACT_POS/ACT_VAL2		
<input checked="" type="checkbox"/> <b>16#1401: RPDO 2 C</b>	<b>16#302 (\$NOD 64</b>	<b>64</b>	<input checked="" type="checkbox"/> <b>16#1801: TPDO 2 Communication Parameter</b>		
RES	16#3010:16#00	8	RES		
SUBINDEX	16#3011:16#00	8	SUBINDEX		
REQCODE_PNU	16#3012:16#00	16	RESPCODE_PNU		
PARAVAL	16#3013:16#00	32	PARAVAL		
<input type="checkbox"/> <b>16#1402: RPDO 3 C</b>	<b>16#402 (\$NOD 64</b>	<b>64</b>	<input type="checkbox"/> <b>16#1802: TPDO 3 Communication Parameter</b>		
FHPP+_O_Byte01	16#3101:16#00	8	FHPP+_I_Byte01		
FHPP+_O_Byte02	16#3102:16#00	8	FHPP+_I_Byte02		
FHPP+_O_Byte03	16#3103:16#00	8	FHPP+_I_Byte03		
FHPP+_O_Byte04	16#3104:16#00	8	FHPP+_I_Byte04		
FHPP+_O_Byte05	16#3105:16#00	8	FHPP+_I_Byte05		
FHPP+_O_Byte06	16#3106:16#00	8	FHPP+_I_Byte06		
FHPP+_O_Byte07	16#3107:16#00	8	FHPP+_I_Byte07		
FHPP+_O_Byte08	16#3108:16#00	8	FHPP+_I_Byte08		
<input type="checkbox"/> <b>16#1403: RPDO 4 C</b>	<b>16#502 (\$NOD 64</b>	<b>64</b>	<input type="checkbox"/> <b>16#1803: TPDO 4 Communication Parameter</b>		
FHPP+_O_Byte09	16#3109:16#00	8	FHPP+_I_Byte09		
FHPP+_O_Byte10	16#3110:16#00	8	FHPP+_I_Byte10		
FHPP+_O_Byte11	16#3111:16#00	8	FHPP+_I_Byte11		
FHPP+_O_Byte12	16#3112:16#00	8	FHPP+_I_Byte12		
FHPP+_O_Byte13	16#3113:16#00	8	FHPP+_I_Byte13		
FHPP+_O_Byte14	16#3114:16#00	8	FHPP+_I_Byte14		
FHPP+_O_Byte15	16#3115:16#00	8	FHPP+_I_Byte15		
FHPP+_O_Byte16	16#3116:16#00	8	FHPP+_I_Byte16		

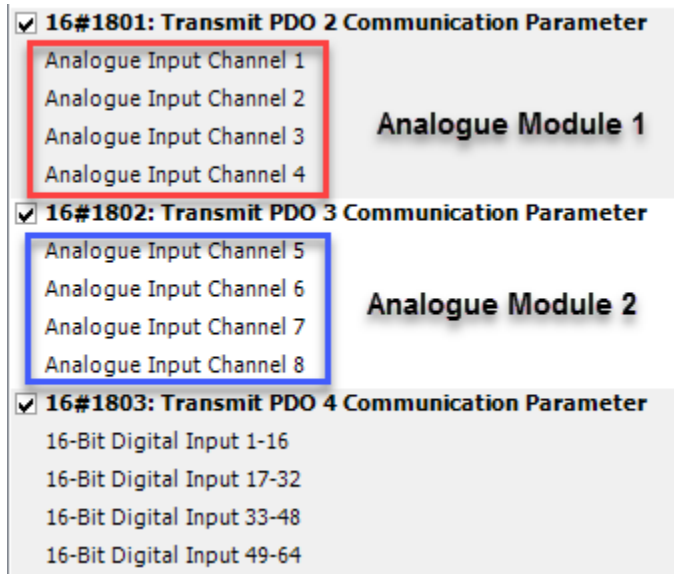
## 7.2.2 Modifying PDOs

Each PDO can transmit up to 8 Bytes and there can only be 4 PDOs per device. Both of these rules must be kept in mind when doing the following procedure. Check the modules manual to verify if other rules apply to that particular device.

Modifying the PDO content can be helpful in applications where different I/O objects must be transmitted than the ones set by default.

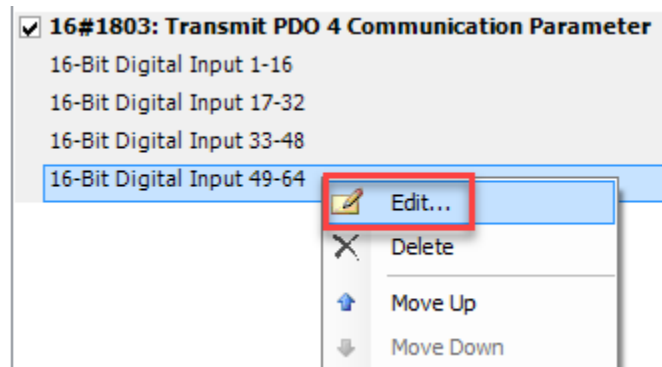
### 7.2.2.1 Example with CPX-FB14

In an application where my CPX system has 12 Analogue Input Channels (3 Analogue Input Modules with 4 analogue inputs each), the 8 Analogue Input Channels mapped by default are insufficient.



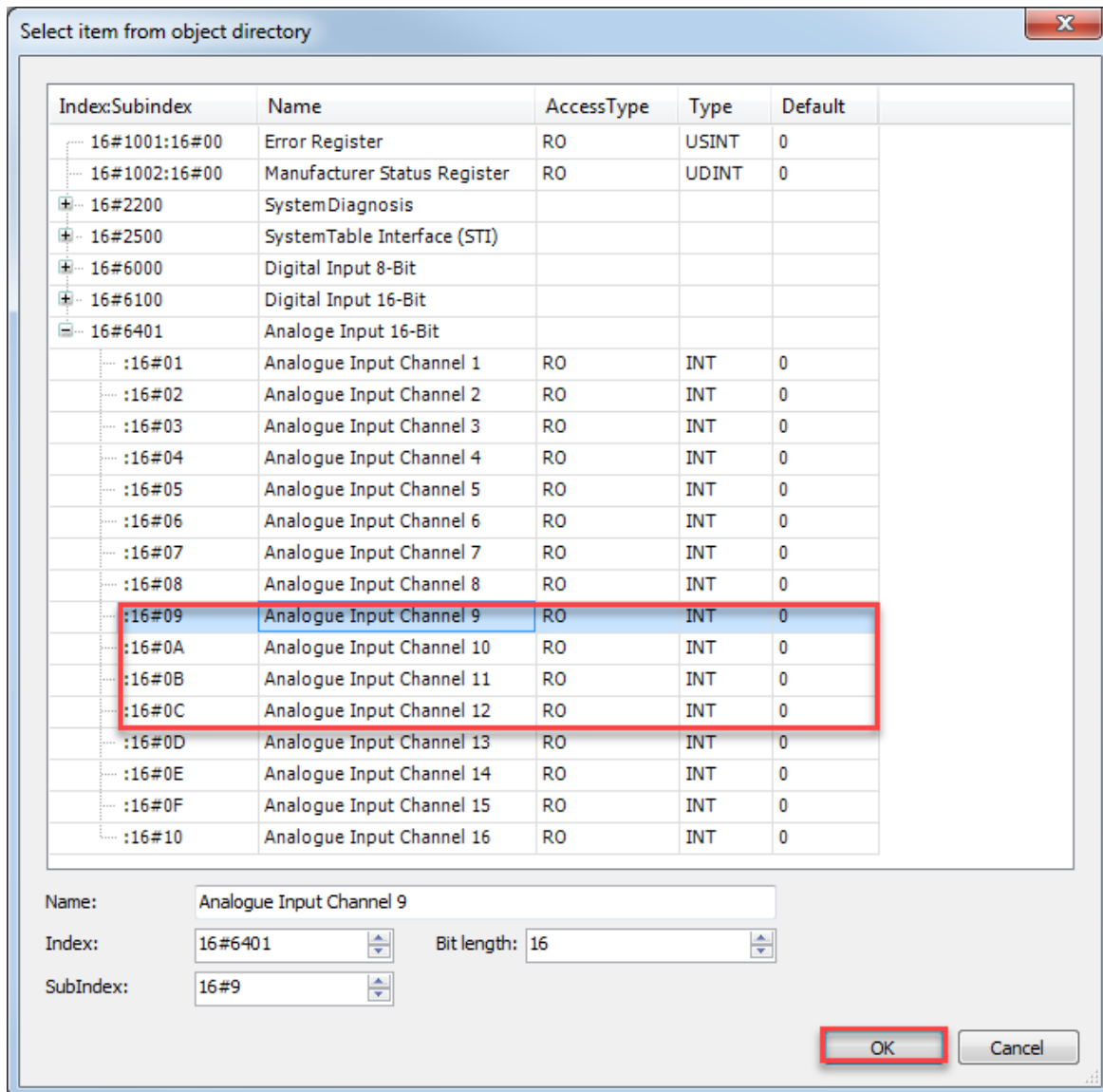
Depending on what is needed in the CPX configuration, some of the other elements can be replaced for more analogue inputs.

This can be done by right clicking on the element that will be replaced and clicking on Edit.

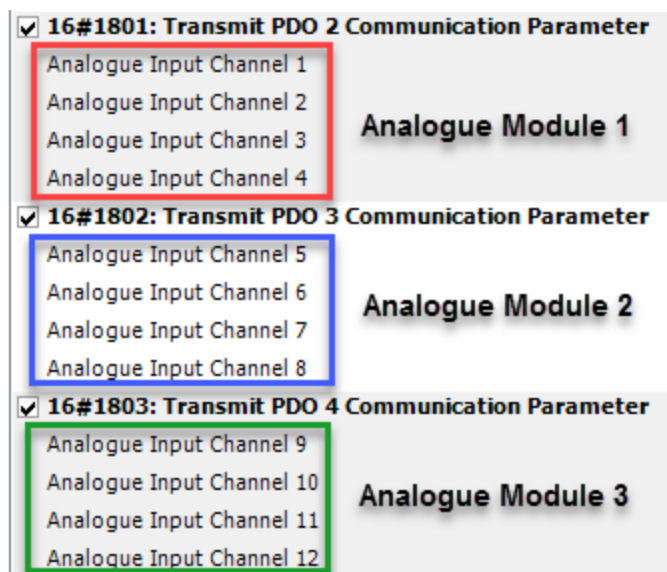


NOTE – The content of a PDO can also be manually deleted, and afterwards filled as the application requires.

The object directory should be displayed with the information available from the EDS file. From here the next group of analogue inputs can be selected.



Now 3 of the PDOs are used to transmit the information from the Analogue input modules and 1 PDO with up to 8 Bytes is left for other inputs.





### 7.2.3 PDO Properties

The PDO properties can be set by selecting the PDO that wants to be modified and clicking on the Edit button.

+ Add PDO   + Add Mapping <b>Edit</b> X Delete   ↑ Move Up   ↓ Move Down		
Name	Object	Bit length
<input checked="" type="checkbox"/> 16#1400: receive_pdo_parameter_rpdo1	16#20A (\$NODEID+16#200)	64
CCON	16#3000:16#00	8
CPOS	16#3001:16#00	8
REC_NR/CDIR	16#3002:16#00	8
RES/DEM_VAL1/PARA1	16#3003:16#00	8
RES/DEM_VAL2/PARA2	16#3004:16#00	32

The following properties are available:

PDO Properties

COB-ID:

\$NODEID+16#200

= 16#206 (518)

Inhibit time (x 100µs):

0

Transmission Type:

asynchronous - device profile specific (Type 255)

Number of Syncs:

1

Event Time (x 1ms):

0

OK

Cancel

**COB-ID** is the communication object identifier for the PDO message.

The PDO messages can be set to work with the following **Transmission Types**:

- Acyclic – synchronous (Type 0) – The PDO is transmitted once after a change of state within the SYNC time window.
- Cyclic – synchronous (Type 1-240) – the PDO is transmitted periodically within the SYNC window each defined Number of Syncs.
- Synchronous – RTR only (Type 252) – available only for Transmit PDOs. The information is updated after the SYNC telegram but is only transmitted after an explicit request.
- Asynchronous – RTR only (Type 253) – available only for Transmit PDOs. The information is updated and transmitted after an explicit request.
- Asynchronous – manufacturer specific (Type 254) – The PDO is transmitted after special events.(for Festo devices this one works in the same way as Asynchronous Type 255)
- Asynchronous – device profile specific (Type 255) – The PDO is transmitted in accordance to the CiA device Profile.

NOTE – The CiA device profiles are the following: CiA 401 for IO Terminals, CiA402 for Motor Controllers

Depending on the transmission type and if the functions are supported by the device, the following parameters can be set.

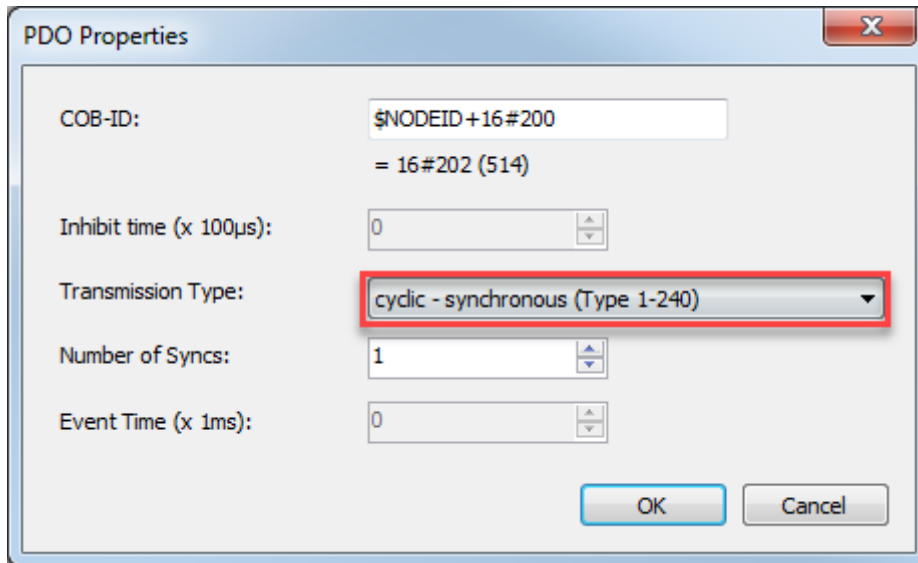
**Inhibit time (x 100us):** Minimal time between two messages of the PDO. Can be used to reduce the update rate of PDOs that are constantly being updated and transmitted.

**Number of syncs:** Can only be set for cyclic – synchronous PDOs. The PDO will be sent once each n number of SYNC telegrams.

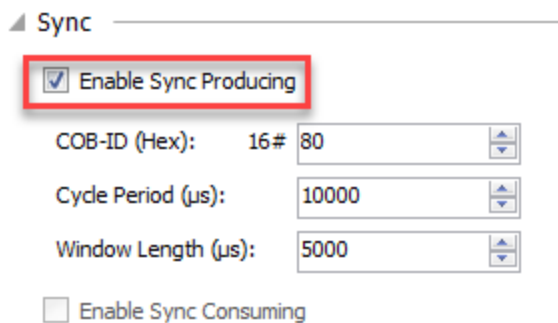
**Event Time (x 1ms):** Can only be set for asynchronous PDOs. The PDO will be sent after the data has been updated and once when the Event Time expires.

#### 7.2.3.1 PDO Properties for motor controllers

For Festo Motor Controllers it is recommended to set the communication type of all Receive and Transmit PDO's to Cyclic – Synchronous.



Make sure that Enable Sync Producing has been checked on the CANopen Manager.

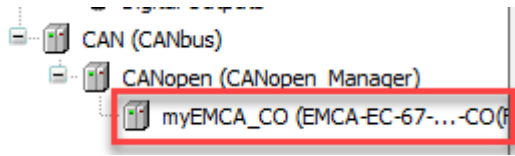


To know what the value of the Cycle Period and Window length should be, the following rule can be used:

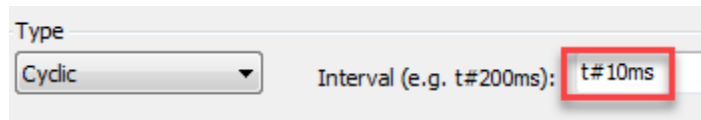
$$\text{minimum Window Length} \cong \frac{\text{Cyclic Synchronous Bits}}{\text{Baudrate}}$$

### 7.2.3.2 Cycle Period and Window Length estimate with EMCA

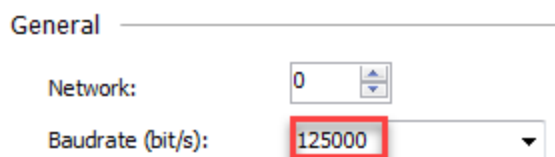
We have an EMCA Motor Controller transmitting FHPP and FPC.



The PLC's Cycle Task is 10ms



And the set network baudrate is 125 kbit/s



Each PDO transmits 8 Bytes or 64 bits.

Receive PDOs (Master => Slave)		Transmit PDOs (Slave => Master)	
Name	Bit length	Name	Bit length
<input checked="" type="checkbox"/> 16#1400: RPDO 1 Communication Parameter	64	<input checked="" type="checkbox"/> 16#1800: TPDO 1 Communication Parameter	64
CCON	8	SCON	8
CPOS	8	SPOS	8
REC_NR/CDIR	8	REC_NR/SDIR	8
RES/DEM_VAL1/PARA1	8	RSB/ACT_VAL1	8
RES/DEM_VAL2/PARA2	32	ACT_POS/ACT_VAL2	32
<input checked="" type="checkbox"/> 16#1401: RPDO 2 Communication Parameter	64	<input checked="" type="checkbox"/> 16#1801: TPDO 2 Communication Parameter	64
RES	8	RES	8
SUBINDEX	8	SUBINDEX	8
REQCODE_PNU	16	RESPCODE_PNU	16
PARAVAL	32	PARAVAL	32

The total number of bits that need to be transmitted are:

$$64 \text{ bits} \times 4 \text{ PDO's} = 256 \text{ bits}$$

Using the before mentioned formula:

$$\text{minimum Window Length} \cong \frac{\text{Cyclic Synchronous Bits}}{\text{Baudrate}}$$

$$\text{minimum Window Length} \cong \frac{256 \text{ bit}}{125.000 \text{ bit/s}}$$

$$\text{minimum Window Length} \cong 2,05 \text{ ms}$$

Because Cyclic Synchronous PDOs can be interrupted by higher priority telegrams, such as NMT telegrams, it is NOT recommended to set the minimum Window Length on the Sync parameters.

The minimum Window Length gives us an estimate of how long the times need to be and makes it easier to define the Cycle Period and Window Length values which will be used.

For the example, I would set the Cycle Period to 10 ms and Window Length to 5ms.

▲ Sync

☒ Enable Sync Producing

COB-ID (Hex): 16# 80

Cycle Period (µs): 10000

Window Length (µs): 5000

The Cycle Period being 10ms comes from the PLC's Cycle Task being 10ms. This means a SYNC telegram will be sent once every PLC cycle.

Window Length of 5ms means my Cyclic Synchronous PDOs have 5ms to be transmitted. This gives more than double the time needed, according to the calculated minimum Window Length.

As a result, I have 10ms for high priority messages (which can be sent at any time), 5ms for Cyclic Synchronous Telegrams, giving me a buffer in case they need to be interrupted, and 5ms for low priority messages.

NOTE – If the baudrate is increased, the number of bits that can be transmitted within the same Window Length will also increase. Follow the rules described in chapter 4.1.1 to verify up to which value the baudrate can be increased.

### 7.3 Service Data Objects Tab

General

PDOs

SDOs

CANopen I/O Mapping

Status

Information

The Service Data Objects (SDOs) Tab specifies which configuration values will be sent during the CANbus initialization.

The EDS file specifies which SDOs must be downloaded during initialization.

The SDOs will be sent in the order specified by the Line number.

Line	Index:Subindex	Name	Value	Bit length	Abort if error	Jump to line if error	Next line	Comment
1	16#1005:16#00	Set COB-ID sync	16#00000080	32	<input type="checkbox"/>	<input type="checkbox"/>	0	
2	16#100C:16#00	Set Guardtime	16#00000000	16	<input type="checkbox"/>	<input type="checkbox"/>	0	
3	16#100D:16#00	Set Lifetime	16#00000000	8	<input type="checkbox"/>	<input type="checkbox"/>	0	
4	16#1014:16#00	Disable Emcy CobID	16#80000086	32	<input type="checkbox"/>	<input type="checkbox"/>	0	
5	16#1014:16#00	Set Emcy CobID	16#00000086	32	<input type="checkbox"/>	<input type="checkbox"/>	0	
6	16#1016:16#01	Set Heartbeat Consumer	0	32	<input type="checkbox"/>	<input type="checkbox"/>	0	
7	16#1016:16#02	Set Heartbeat Consumer	0	32	<input type="checkbox"/>	<input type="checkbox"/>	0	
8	16#1016:16#03	Set Heartbeat Consumer	0	32	<input type="checkbox"/>	<input type="checkbox"/>	0	
9	16#1016:16#04	Set Heartbeat Consumer	0	32	<input type="checkbox"/>	<input type="checkbox"/>	0	
10	16#1016:16#05	Set Heartbeat Consumer	0	32	<input type="checkbox"/>	<input type="checkbox"/>	0	
11	16#1016:16#06	Set Heartbeat Consumer	0	32	<input type="checkbox"/>	<input type="checkbox"/>	0	

The ones that appear with a greyed font are the communication parameters. The modification of these have been done through the General Tab and PDO Tab.

If an SDO is considered critical for the application, the **Abort if error** mark can be set. This way, if there is an error while sending that SDO, the configuration will be stopped and the device will remain in PRE-OPERATIONAL state.

Line	Index:Subindex	Name	Value	Bit length	Abort if error	Jump to line if error	Next line
1	16#1005:16#00	Set COB-ID sync	16#00000080	32	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0

If an SDO is required for the following SDOs in the line, but not critical for the application, the **Jump to line if error** can be set to skip the SDOs and continue on the number set in **Next Line**.

Line	Index:Subindex	Name	Value	Bit length	Abort if error	Jump to line if error	Next line
1	16#1005:16#00	Set COB-ID sync	16#00000080	32	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0

SDO Timeout (ms):  ☐ Create all SDOs ☐ Write complete PDO configuration

**SDO Timeout (ms)** Is the time that the SDO Client will expect an answer from the SDO Server. If no answer comes after the Time set expires, the transmission will be aborted.

**Create all SDOs** will fill the SDO page with all SDOs from Index 16#2000 onwards with the default value described in the EDS file. For Festo devices, this option should ALWAYS be set to off, as all the SDO parameters for our devices have a default value of 0.






**Write complete PDO configuration** will force PDO configuration to be download if checked.

If not checked, the PDO configuration will still be downloaded. In order for PDOs to be set to default values, the Reset Node option in the CANopen slave must also be enabled. After the Reset Node, the device should start with the default PDO configuration. This PDO configuration must match what is stated as Default in the EDS file for communication to take place. If the configuration does not match, communication will not be established.

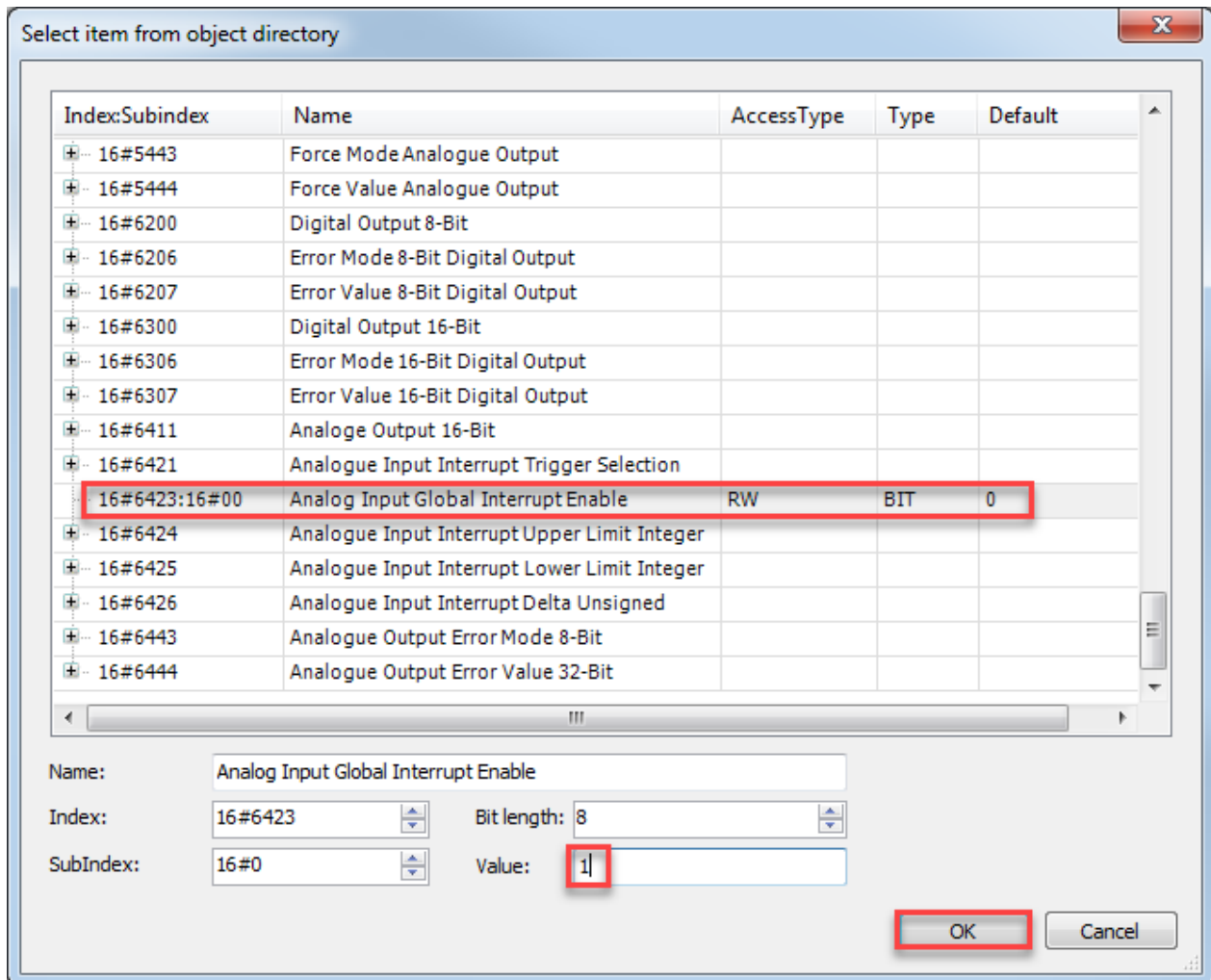
### 7.3.1 Initial configuration via SDOs

More SDOs can be added to download additional parameters.

An example of when this could be useful: When using CPX Analogue Inputs, Object Index 16#6423 MUST be set to 1 in order for the Analogue Inputs to be updated periodically. This SDO can be added by selecting the Add SDO Button:

General	<div> <div> Add SDO</div> <div> Edit</div> <div> Delete</div> <div> Move Up</div> <div> Move Down</div> </div>				
PDOs	Line	Index:Subindex	Name	Value	Bit length
SDOs	51	16#1603:16#00	Set number of mappings	16#04	8
	52	16#1403:16#01	Set and enable COB-ID	16#00000506	32
	53	16#1800:16#01	Disable PDO	16#80000186	32
CANopen I/O Mapping	54	16#1800:16#02	Set transmission type	16#FF	8
	55	16#1800:16#03	Set inhibit time	16#0000	16
Status	56	16#1800:16#05	Set event time	16#0000	16
	57	16#1A00:16#00	Clear pdo mapping	16#0	8
Information	58	16#1A00:16#01	Set Mapping	16#60000108	32

Looking for Index 0x6423 – Analogue Input Global Interrupt Enable, setting a value of 1 and clicking on OK.



The SDO will be added at the end of the line.

99	16#1A03:16#00	Set number of mappings	16#04	8
100	16#1803:16#01	Set and enable COB-ID	16#00000486	32
101	16#6423:16#00	Analog Input Global Inter...	1	8

NOTE - If another SDO was highlighted before clicking on Add SDO, the new SDO will be added after the highlighted one.

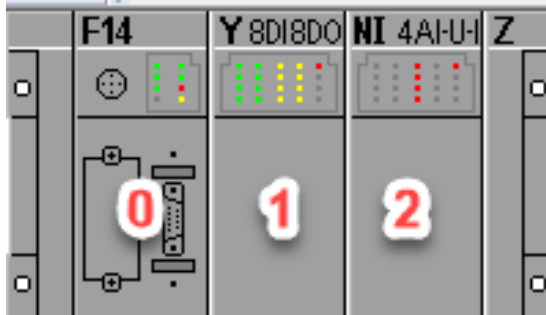
NOTE - The SDOs that are defined in the EDS file are visible when clicking on the Add SDO option. If a particular SDO is not there by default, but is stated in the devices documentation to be available, the Index, Subindex, Bit Length and Value can be manually filled to address that particular SDO. A reason for this could be that a different EDS file is being used.

### 7.3.2 CPX initial configuration via SDOs

CPX-FB14 modules can be parameterized by using SDO objects starting on Index 0x2400.

As an example we will set the CPX-4AE-U-I module of the following CPX configuration to set Input Channel 1 to work with 0 to 10 V and Input Channel 2 to work with 4 to 20 mA.

The displayed red numbers refer to the module number.



When clicking on Add SDO and going to the Index 0x2400 we see the parameters for module 2 are located on Index 0x2412.

+ 16#2400	System Parameter
+ 16#2410	Parameter Modul 0
+ 16#2411	Parameter Modul 1
- 16#2412	Parameter Modul 2
:16#01	P0: Monitoring (active/inactive)
:16#02	P1: Error Behavior, Debounce, Signal Stretch...
:16#03	P2: Reserved
:16#04	P3: Format of Analogue Value

The subindex must be consulted in the CPX-4AE-U-I documentation.

Channel parameters: Signal range channel x																																																																																																																																																																																																																																																																									
Function no.	4828 + m * 64 + 13 (channel 0 ... 1) 4828 + m * 64 + 14 (channel 2 ... 3) <span style="float:right">m = module number (0 ... 47)</span>																																																																																																																																																																																																																																																																								
Description	For the individual channels of the analogue input modules, the signal ranges of the analogue inputs can be set independently of each other. Channel 0: Function number 13 Bit 0 ... 3 Channel 1: Function number 13 Bit 4 ... 7 Channel 2: Function number 14 Bit 0 ... 3 Channel 3: Function number 14 Bit 4 ... 7																																																																																																																																																																																																																																																																								
Bit	Bit 0 ... 3: Signal range channel 0 or channel 2 Bit 4 ... 7: Signal range channel 1 or channel 3																																																																																																																																																																																																																																																																								
Values	<table><tr><th>Bit</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>Channel 0</th><th>channel 2</th></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>No sensor connected (presetting)</td><td></td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0 ... 10 V</td><td></td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>-10 ... +10 V</td><td></td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>-5 ... +5 V</td><td></td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1 ... 5 V</td><td></td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0 ... 20 mA</td><td></td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>4 ... 20 mA</td><td></td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>-20 ... +20 mA</td><td></td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0 ... 10 V (negative values suppressed)</td><td></td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0 ... 20 mA (negative values suppressed)</td><td></td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>4 ... 20 mA (negative values suppressed)</td><td></td></tr></table> <table><tr><th>Bit</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>Channel 1</th><th>channel 3</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>-</td><td>-</td><td>-</td><td>-</td><td>No sensor connected (presetting)</td><td></td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>0 ... 10 V</td><td></td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-10 ... +10 V</td><td></td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-5 ... +5 V</td><td></td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>1 ... 5 V</td><td></td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>0 ... 20 mA</td><td></td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>4 ... 20 mA</td><td></td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-20 ... +20 mA</td><td></td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>0 ... 10 V (negative values suppressed)</td><td></td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>0 ... 20 mA (negative values suppressed)</td><td></td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>4 ... 20 mA (negative values suppressed)</td><td></td></tr></table>	Bit	7	6	5	4	3	2	1	0	Channel 0	channel 2	-	-	-	-	0	0	0	0	0	No sensor connected (presetting)		-	-	-	-	0	0	0	1	0	0 ... 10 V		-	-	-	-	0	0	1	0	0	-10 ... +10 V		-	-	-	-	0	0	1	1	0	-5 ... +5 V		-	-	-	-	0	1	0	0	0	1 ... 5 V		-	-	-	-	0	1	0	1	0	0 ... 20 mA		-	-	-	-	0	1	1	0	0	4 ... 20 mA		-	-	-	-	0	1	1	1	0	-20 ... +20 mA		-	-	-	-	1	0	0	0	0	0 ... 10 V (negative values suppressed)		-	-	-	-	1	0	0	1	0	0 ... 20 mA (negative values suppressed)		-	-	-	-	1	0	1	0	0	4 ... 20 mA (negative values suppressed)		Bit	7	6	5	4	3	2	1	0	Channel 1	channel 3	0	0	0	0	0	-	-	-	-	No sensor connected (presetting)		0	0	0	1	-	-	-	-	-	0 ... 10 V		0	0	1	0	-	-	-	-	-	-10 ... +10 V		0	0	1	1	-	-	-	-	-	-5 ... +5 V		0	1	0	0	-	-	-	-	-	1 ... 5 V		0	1	0	1	-	-	-	-	-	0 ... 20 mA		0	1	1	0	-	-	-	-	-	4 ... 20 mA		0	1	1	1	-	-	-	-	-	-20 ... +20 mA		1	0	0	0	-	-	-	-	-	0 ... 10 V (negative values suppressed)		1	0	0	1	-	-	-	-	-	0 ... 20 mA (negative values suppressed)		1	0	1	0	-	-	-	-	-	4 ... 20 mA (negative values suppressed)	
Bit	7	6	5	4	3	2	1	0	Channel 0	channel 2																																																																																																																																																																																																																																																															
-	-	-	-	0	0	0	0	0	No sensor connected (presetting)																																																																																																																																																																																																																																																																
-	-	-	-	0	0	0	1	0	0 ... 10 V																																																																																																																																																																																																																																																																
-	-	-	-	0	0	1	0	0	-10 ... +10 V																																																																																																																																																																																																																																																																
-	-	-	-	0	0	1	1	0	-5 ... +5 V																																																																																																																																																																																																																																																																
-	-	-	-	0	1	0	0	0	1 ... 5 V																																																																																																																																																																																																																																																																
-	-	-	-	0	1	0	1	0	0 ... 20 mA																																																																																																																																																																																																																																																																
-	-	-	-	0	1	1	0	0	4 ... 20 mA																																																																																																																																																																																																																																																																
-	-	-	-	0	1	1	1	0	-20 ... +20 mA																																																																																																																																																																																																																																																																
-	-	-	-	1	0	0	0	0	0 ... 10 V (negative values suppressed)																																																																																																																																																																																																																																																																
-	-	-	-	1	0	0	1	0	0 ... 20 mA (negative values suppressed)																																																																																																																																																																																																																																																																
-	-	-	-	1	0	1	0	0	4 ... 20 mA (negative values suppressed)																																																																																																																																																																																																																																																																
Bit	7	6	5	4	3	2	1	0	Channel 1	channel 3																																																																																																																																																																																																																																																															
0	0	0	0	0	-	-	-	-	No sensor connected (presetting)																																																																																																																																																																																																																																																																
0	0	0	1	-	-	-	-	-	0 ... 10 V																																																																																																																																																																																																																																																																
0	0	1	0	-	-	-	-	-	-10 ... +10 V																																																																																																																																																																																																																																																																
0	0	1	1	-	-	-	-	-	-5 ... +5 V																																																																																																																																																																																																																																																																
0	1	0	0	-	-	-	-	-	1 ... 5 V																																																																																																																																																																																																																																																																
0	1	0	1	-	-	-	-	-	0 ... 20 mA																																																																																																																																																																																																																																																																
0	1	1	0	-	-	-	-	-	4 ... 20 mA																																																																																																																																																																																																																																																																
0	1	1	1	-	-	-	-	-	-20 ... +20 mA																																																																																																																																																																																																																																																																
1	0	0	0	-	-	-	-	-	0 ... 10 V (negative values suppressed)																																																																																																																																																																																																																																																																
1	0	0	1	-	-	-	-	-	0 ... 20 mA (negative values suppressed)																																																																																																																																																																																																																																																																
1	0	1	0	-	-	-	-	-	4 ... 20 mA (negative values suppressed)																																																																																																																																																																																																																																																																

The Bytes that set the analogue input modules signal ranges are highlighted in red. With the information above we know, for the Channel 1 configuration, Byte 13 must be set to binary value 2# 0001 0000.

For Channel 2, Byte 14 must be set to binary value 2# 0000 0110.

CPX Byte 13 and Byte 14 are labelled as P13 and P14 respectively in the SDO name column.

Select item from object directory

Index/Subindex	Name	AccessType	Type	Default
16#2412	Parameter Modul 2			
:16#01	P0: Monitoring (active/inactive)	RW	USINT	0
:16#02	P1: Error Behavior, Debounce, Signal Stretch...	RW	USINT	0
:16#03	P2: Reserved	RW	USINT	0
:16#04	P3: Format of Analogue Value	RW	USINT	0
:16#05	P4: Reserved	RW	USINT	0
:16#06	P5: Reserved	RW	USINT	0
:16#07	P6: (module specific)	RW	USINT	0
:16#08	P7: (module specific)	RW	USINT	0
:16#09	P8: (module specific)	RW	USINT	0
:16#0A	P9: (module specific)	RW	USINT	0
:16#0B	P10: (module specific)	RW	USINT	0
:16#0C	P11: (module specific)	RW	USINT	0
:16#0D	P12: (module specific)	RW	USINT	0
:16#0E	P13: (module specific)	RW	USINT	0
:16#0F	P14: (module specific)	RW	USINT	0

Name: P13: (module specific)

Index: 16#2412 Bit length: 8

SubIndex: 16#E Value: 2#00010000

OK Cancel

Select item from object directory

Index/Subindex	Name	AccessType	Type	Default
16#2412	Parameter Modul 2			
:16#01	P0: Monitoring (active/inactive)	RW	USINT	0
:16#02	P1: Error Behavior, Debounce, Signal Stretch...	RW	USINT	0
:16#03	P2: Reserved	RW	USINT	0
:16#04	P3: Format of Analogue Value	RW	USINT	0
:16#05	P4: Reserved	RW	USINT	0
:16#06	P5: Reserved	RW	USINT	0
:16#07	P6: (module specific)	RW	USINT	0
:16#08	P7: (module specific)	RW	USINT	0
:16#09	P8: (module specific)	RW	USINT	0
:16#0A	P9: (module specific)	RW	USINT	0
:16#0B	P10: (module specific)	RW	USINT	0
:16#0C	P11: (module specific)	RW	USINT	0
:16#0D	P12: (module specific)	RW	USINT	0
:16#0E	P13: (module specific)	RW	USINT	0
:16#0F	P14: (module specific)	RW	USINT	0

Name: P14: (module specific)

Index: 16#2412 Bit length: 8

SubIndex: 16#F Value: 2#00000110

OK Cancel

101	16#6423:16#00	Analog Input Global Inter...	1	8
102	16#2412:16#0E	P13: (module specific)	2#00010000	8
103	16#2412:16#0F	P14: (module specific)	2#00000110	8



NOTE – It is not a must to use Binary coding to set parameters. The equivalent decimal or hexadecimal value can also be used.




The successful data transfer can be verified with CPX-FMT

Inputs	
I0: Diagnostic functions	Monitor parameters
I1: Diagnostic functions	Monitor parameters
I2: Diagnostic functions	Monitor parameters
I3: Diagnostic functions	Monitor parameters
I0: Signal range	No sensor connected
I1: Signal range	0..10V
I2: Signal range	4..20mA
I3: Signal range	No sensor connected

## 8 Network Management (NMT) and Diagnostics in CODESYS V3

The following chapter will describe the available function blocks in CODESYS V3 for Network Management and Diagnostic Handling.

When adding the CANbus, CANopen Manager and CANopen Device, the following libraries will be automatically loaded to the project.

 CANbusDevice = CANbusDevice, 3.5.7.0 (3S - Smart Software Solutions GmbH)	CANbusDevice	3.5.7.0
 3S CANopenStack = 3S CANopenStack, 3.5.7.0 (3S - Smart Software Solutions GmbH)	_3SCOS	3.5.7.0
 CAA CiA405 = CAA CiA 405, 3.5.7.0 (CAA Technical Workgroup)	CIA405	3.5.7.0

The following functions are dependent that the libraries mentioned above are present in the project.

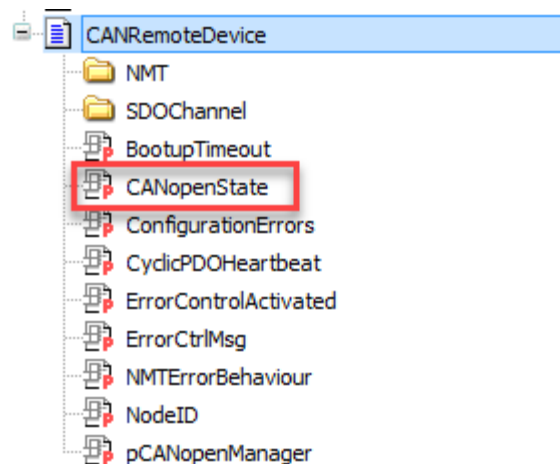
### 8.1 Reading the device state

There are two ways to read the device state in CODESYS V3.

#### 8.1.1 Property <DeviceName>.CANopenState

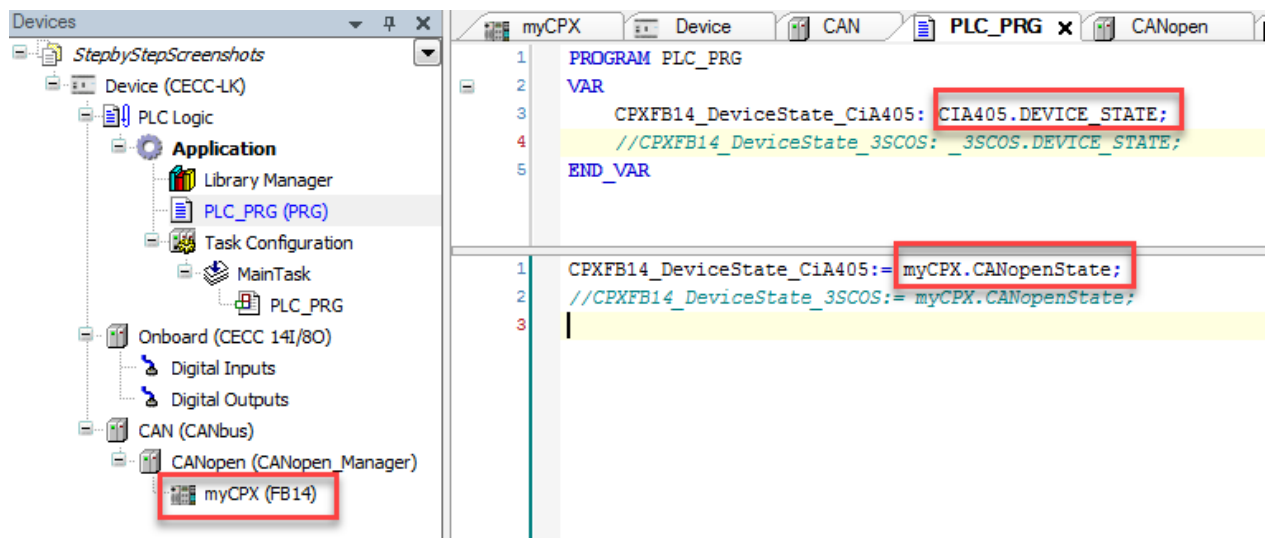
CANopen Devices are based on the CANRemoteDevice FB available in the 3S CANopenStack library. This FB contains a Property labelled CANopenState (Get).

Properties are a tool for object-oriented programming that consist of the accessor methods Get and Set. The method is automatically called as soon as a read or write access takes place to the FB implementing that property.



By typing <DeviceName>.CANopenState in a cyclical POU, the property will be executed once every time it is implemented in the code, meaning the property will give us the most recent device status changes.

The CANopenState can be transferred to a variable of enumeration type DEVICE\_STATE available in either the CAA CiA405 Library (CiA405) or the 3S CANopenStack library (\_3SCOS).



State values are one of the following.

Name
INIT
RESET_COMM
RESET_APP
PRE_OPERATIONAL
STOPPED
OPERATIONAL
UNKNOWN
NOT_AVAIL

```

1 CPXFB14_DeviceState_CiA405 INIT := myCPX.CANopenState;
2 //CPXFB14_DeviceState_3SCOS:= myCPX.CANopenState;
3 RETURN

```

```

1 CPXFB14_DeviceState_CiA405 PRE_OPERAT := myCPX.CANopenState;
2 //CPXFB14_DeviceState_3SCOS:= myCPX.CANopenState;
3 RETURN

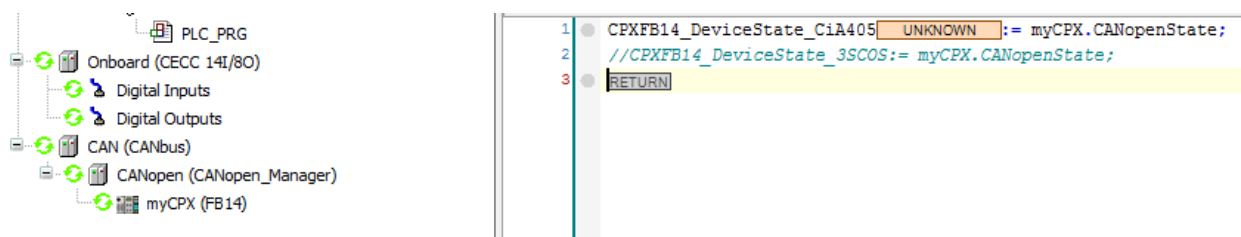
```

```

1 CPXFB14_DeviceState_CiA405 OPERATIONA := myCPX.CANopenState;
2 //CPXFB14_DeviceState_3SCOS:= myCPX.CANopenState;
3 RETURN

```

NOTE – State UNKNOWN is displayed when no Guarding function (Node Guarding or Heartbeat) has been implemented in the device. This doesn't mean there is no communication.



### 8.1.2 GET\_STATE Function Block

Using the GET\_STATE FB located in the CAA CiA 405 Library, the device state can be requested once every time the block is executed.



**Network** – In which CANbus network will the GET\_STATE request be executed. This number is the CANbus Network Number (0 for Festo PLCs) + 1 as default offset.

**Enable** – Enables FB on rising edge. Aborts request on falling edge.

**Timeout** – Set timeout in ms. 0 means no timeout.

**Device** – Set the CANopen device Node ID whose state will be requested. 0 can be used to request the CANopen Manager state.

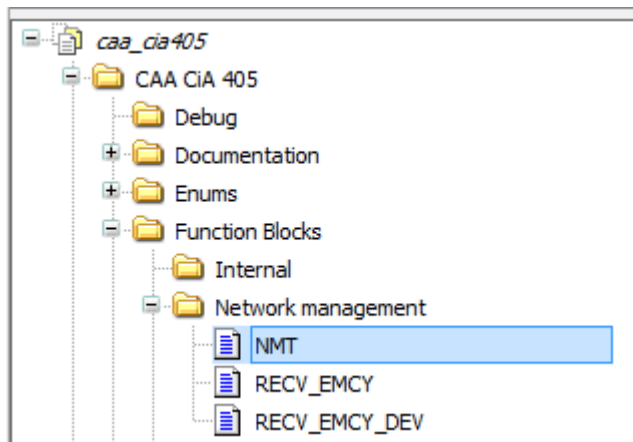
**Confirm** – Request finished with no error.

**Error** – Error Code while executing FB. Consult CANOPEN\_KERNEL\_ERROR enumeration of the CAA CiA 405 library for further details.

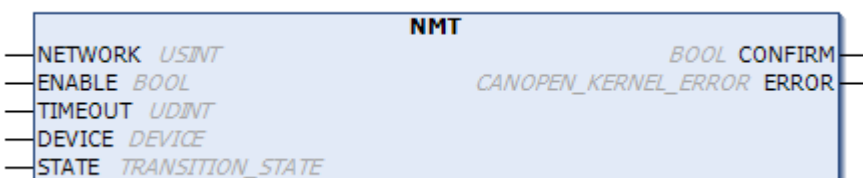
**State** – NMT state for requested Device. Consult DEVICE\_STATE enumeration of the CAA CiA 405 library for details.

## 8.2 Network Management Function Block

This block is located on the CAA CiA 405 Library, labelled as NMT.



The NMT FB is used to handle Network Management (NMT) services which are used to set a device in a specific Device State.



**Network** – In which CANbus network will the NMT request be executed. This number is the CANbus Network Number (0 for Festo PLCs) + 1 as default offset.

**Enable** – Enables FB on rising edge. Aborts request on falling edge.

**Timeout** – Set timeout in ms. 0 means no timeout.

**Confirm** – Request finished with no error.

**Error** – Error Code while executing FB. Consult CANOPEN\_KERNEL\_ERROR enumeration for further details.

**Device** – Node ID of destination device. 0 is used to address all devices in the network.

**State** – Send state according to the TRANSITION\_STATE enumeration available in the CAA CiA 405 library. Available states are the following:

Name	Initial
START_REMOTE_NODE	16#5
STOP_REMOTE_NODE	16#4
ENTER_PRE_OPERATIONAL	16#7F
RESET_NODE	16#6
RESET_COMMUNICATION	16#7
ALL_EXCEPT_NMT_AND_SENDER	16#800

NOTE – The above displayed values under the Initial column do not display the CiA 405 specification values. If using special hardware to monitor the data transmission of the NMT messages the following values will be displayed.

COB-ID	Byte-0	Byte-1	Remark
0x000	CS	Node-ID	bei Node-ID = 0 sind alle im Netz befindlichen Nodes adressiert
0	0	1	START Remote Node
0	0	2	STOP Remote Node
0	0	0x80	Enter Pre-operational state
0	0	0x81	Reset Node
0	0	0x82	Reset Communication

The implementation of the NMT FB in Structured Text should look like this.

```

1  PROGRAM PLC_PRG
2  VAR
3      CPXFB14_DeviceState_CiA405: CIA405.DEVICE_STATE;
4      //CPXFB14_DeviceState_3SCOS: 3SCOS.DEVICE_STATE;
5      Send_NMT_Messages: CIA405.NMT;
6      NMT_Device_State_Request: CIA405.TRANSITION_STATE;
7      xToggleNMT: BOOL;
8  END_VAR

1  CPXFB14_DeviceState_CiA405:= myCPX.CANopenState;
2  //CPXFB14_DeviceState_3SCOS:= myCPX.CANopenState;
3
4  Send_NMT_Messages(
5      NETWORK:= 1,
6      ENABLE:= xToggleNMT,
7      TIMEOUT:= 0,
8      CONFIRM=> ,
9      ERROR=> , |
10     DEVICE:= 6,
11     STATE:= NMT_Device_State_Request);

```

An example of the FB setting a CAN Device to STOP state.

```

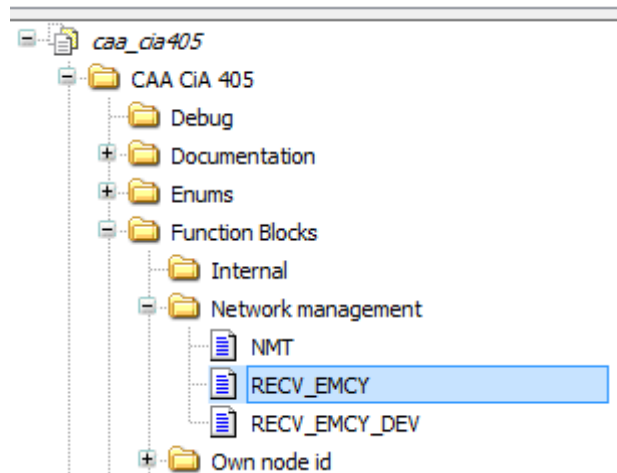
1  CPXFB14_DeviceState_Cia405 STOPPED = myCPX.CANopenState;
2  //CPXFB14_DeviceState_3SCOS:= myCPX.CANopenState;
3
4  Send_NMT_Messages(
5      NETWORK 16#01 := 1,
6      ENABLE TRUE := xToggleNMT TRUE,
7      TIMEOUT 16#00000000 := 0,
8      CONFIRM=> ,
9      ERROR=> ,
10     DEVICE 16#06 := 6,
11     STATE STOP_REMOT := NMT_Device_State_Request STOP_REMOT );

```

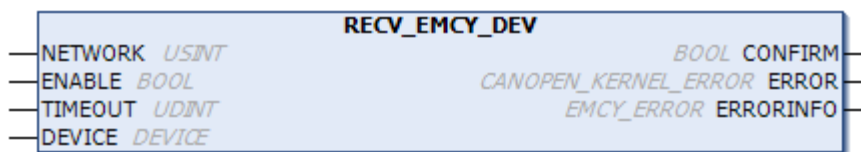
NOTE – ALL\_EXCEPT\_NMT\_AND\_SENDER (16#800) can be used in combination with any of the above mentioned DEVICE\_STATES and DEVICE Node ID 0 so all CAN devices are addressed except the CANopen Manager.

### 8.3 EMCY handling Function Blocks

There are two function blocks that can be used to handle EMCY messages being sent by any of the CAN devices. Receive Emergency (RECV\_EMCY) and Receive Emergency Device (RECV\_EMCY\_DEV).



#### 8.3.1 RECV\_EMCY\_DEV



The RECV\_EMCY\_DEV FB verifies if an EMCY telegram has been received from one specific Device in the network.

**Network** - In which CANbus network will the RECV\_EMCY\_DEV request be executed. This number is the CANbus Network Number (0 for Festo PLCs) + 1 as default offset.

**Enable** – Enables FB in rising edge. Aborts request on falling edge.

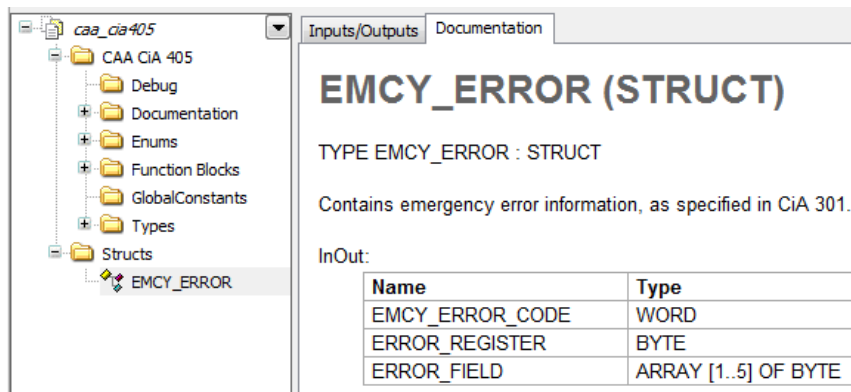
**Timeout** – Set timeout in ms. 0 means no timeout.

**Device** – NodeID of the device whose status will be requested.

**Confirm** – If set to true, function block finished without error.

**Error** – Error Code while executing FB. Consult CANOPEN\_KERNEL\_ERROR enumeration for further details. This error has nothing to do with the EMCY telegram, but with the execution of the FB itself.

**ErrorInfo** – with a successful execution, the EMCY information will be displayed in this output. The EMCY\_ERROR structure located in the CAA CiA405 library is used to arrange the EMCY telegram.



The screenshot shows the CODESYS IDE interface. On the left, the project tree displays the 'caa\_cia405' library with subfolders: CAA CiA 405, Debug, Documentation, Enums, Function Blocks, GlobalConstants, Types, and Structs. The 'EMCY\_ERROR' structure is selected under 'Structs'. On the right, the 'Documentation' tab is active, showing the title 'EMCY\_ERROR (STRUCT)', the type 'TYPE EMCY\_ERROR : STRUCT', and a description: 'Contains emergency error information, as specified in CiA 301.' Below this, the 'InOut:' section contains a table with the structure's fields and their types.

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

Consult the CAN slave device's manual for the error interpretation.

The implementation of the RECV\_EMCY\_DEV in structured text should look like this.

```

1  PROGRAM PLC_PRG
2  VAR
3      EMCY_DEV_CPX: CIA405.RECV_EMCY_DEV;
4      xRequest_EMCYCPX: BOOL;
5      xConfirm_EMCYCPX: BOOL;
6      CPX_EMCY_INFO: CIA405.EMCY_ERROR;
7
8  END_VAR
9
10
11 EMCY_DEV_CPX (
12     NETWORK:= 1,
13     ENABLE:= xRequest_EMCYCPX,
14     TIMEOUT:= 0,
15     CONFIRM=> xConfirm_EMCYCPX,
16     ERROR=> ,
17     DEVICE:= 6,
18     ERRORINFO=> CPX_EMCY_INFO);

```

An example of the FB requesting the EMCY telegram from Node ID 6, which is a CPX-FB14 device.

Expression	Type	Value
EMCY_DEV_CPX	CIA405.RECV_EMCY_DEV	
xRequest_EMCYCPX	BOOL	TRUE
xConfirm_EMCYCPX	BOOL	TRUE
CPX_EMCY_INFO	CIA405.EMCY_ERROR	
EMCY_ERROR_CODE	WORD	16#3320
ERROR_REGISTER	BYTE	16#05
ERROR_FIELD	ARRAY [1..5] OF BYTE	
ERROR_FIELD[1]	BYTE	16#12
ERROR_FIELD[2]	BYTE	16#01
ERROR_FIELD[3]	BYTE	16#05
ERROR_FIELD[4]	BYTE	16#00
ERROR_FIELD[5]	BYTE	16#00

```

1  EMCY_DEV_CPX (
2      NETWORK 16#01 := 1,
3      ENABLE TRUE := xRequest_EMCYCPX TRUE,
4      TIMEOUT 16#00000000 := 0,
5      CONFIRM TRUE => xConfirm_EMCYCPX TRUE,
6      ERROR=> ,
7      DEVICE 16#06 := 6,
8      ERRORINFO=> CPX_EMCY_INFO);RETURN

```

For the above shown example, the EMCY telegram interpretation can be made using the CPX-FB14's Manual.

### 8.3.2 RECV\_EMCY



The RECV\_EMCY FB verifies if an EMCY telegram has been received from any slave device in the network.

With the Enable Input set to TRUE, the FB will go through the storages of all existing devices and display any available EMCY telegram with the Node ID it is linked to.

If there are multiple EMCY telegrams, the one with the highest priority (lowest Node ID) will be displayed with the first execution of the RECV\_EMCY FB. To display the next EMCY telegram, the RECV\_EMCY FB must be executed again by toggling the ENABLE input OFF and ON.

If there is no EMCY telegram in any device, Confirm will be set to TRUE, indicating a successful FB execution and Device (Node ID) will display 0.

**Network** - In which CANbus network will the RECV\_EMCY request be executed. This number is the CANbus Network Number (0 for Festo PLCs) + 1 as default offset.

**Enable** – Enables FB in rising edge. Aborts request on falling edge.

**Timeout** – Set timeout in ms. 0 means no timeout.

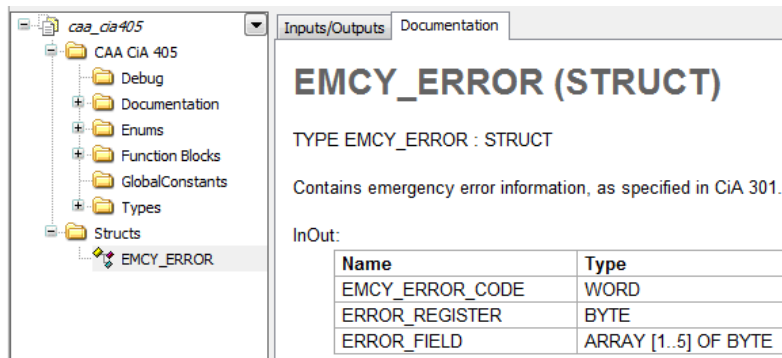
**Confirm** – If set to true, function block finished without error.

**Error** – Error Code while executing FB. Consult CANOPEN\_KERNEL\_ERROR enumeration for further details. This error has nothing to do with the EMCY telegram, but with the execution of the FB itself.



**Device** – If an EMCY telegram is available, this output will display the Node ID of the module sending the EMCY telegram.

**ErrorInfo** – with a successful execution, the EMCY information will be displayed in this output. The EMCY\_ERROR structure located in the CAA CiA405 library is used to arrange the EMCY telegram.



**EMCY\_ERROR (STRUCT)**

TYPE EMCY\_ERROR : STRUCT

Contains emergency error information, as specified in CiA 301.

InOut:

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

The implementation of the RECV\_EMCI FB in structured text should look like this:

```

1  PROGRAM PLC_PRG
2  VAR
3      EMCY_NW: CIA405.RECV_EMCI;
4      xRequest_EMCI: BOOL;
5      xConfirm_EMCI: BOOL;
6      EMCY_INFO: CIA405.EMCY_ERROR;
7      EMCY_NodeID: USINT;
8  END_VAR
9
10
11  EMCY_NW(
12      NETWORK:= 1,
13      ENABLE:= xRequest_EMCI,
14      TIMEOUT:= 0,
15      CONFIRM=> xConfirm_EMCI,
16      ERROR=> ,
17      DEVICE=> EMCY_NodeID,
18      ERRORINFO=> EMCY_INFO);
19

```

For the following screenshots, Node ID's 0x06 (CPX-FB14) and 0x0F (CTEU-CO) both are sending an EMCY telegram.

The following is the result after the first execution of the FB.

Expression	Type	Value
EMCY_NW	CIA405.RECV_EMCY	
xRequest_EMCY	BOOL	TRUE
xConfirm_EMCY	BOOL	TRUE
EMCY_NodeID	USINT	16#06
EMCY_INFO	CIA405.EMCY_ERROR	
EMCY_ERROR_CODE	WORD	16#3320
ERROR_REGISTER	BYTE	16#05
ERROR_FIELD	ARRAY [1..5] OF BYTE	
ERROR_FIELD[1]	BYTE	16#12
ERROR_FIELD[2]	BYTE	16#01
ERROR_FIELD[3]	BYTE	16#05
ERROR_FIELD[4]	BYTE	16#00
ERROR_FIELD[5]	BYTE	16#00

```

1  EMCY_NW (
2      NETWORK[16#01] := 1,
3      ENABLE TRUE := xRequest_EMCY TRUE,
4      TIMEOUT[16#00000000] := 0,
5      CONFIRM TRUE => xConfirm_EMCY TRUE,
6      ERROR=> ,
7      DEVICE[16#06] => EMCY_NodeID[16#06],
8      ERRORINFO=> EMCY_INFO);

```

Node ID 0x06 (CPX-FB14) has a higher priority, so his EMCY telegram is displayed first.

After toggling ENABLE OFF and ON the next EMCY telegram, in this case from 0x0F (CTEU-CO) is displayed.

Expression	Type	Value
EMCY_NW	CIA405.RECV_EMCY	
xRequest_EMCY	BOOL	TRUE
xConfirm_EMCY	BOOL	TRUE
EMCY_NodeID	USINT	16#0F
EMCY_INFO	CIA405.EMCY_ERROR	
EMCY_ERROR_CODE	WORD	16#3320
ERROR_REGISTER	BYTE	16#85
ERROR_FIELD	ARRAY [1..5] OF BYTE	
ERROR_FIELD[1]	BYTE	16#00
ERROR_FIELD[2]	BYTE	16#00
ERROR_FIELD[3]	BYTE	16#12
ERROR_FIELD[4]	BYTE	16#51
ERROR_FIELD[5]	BYTE	16#00

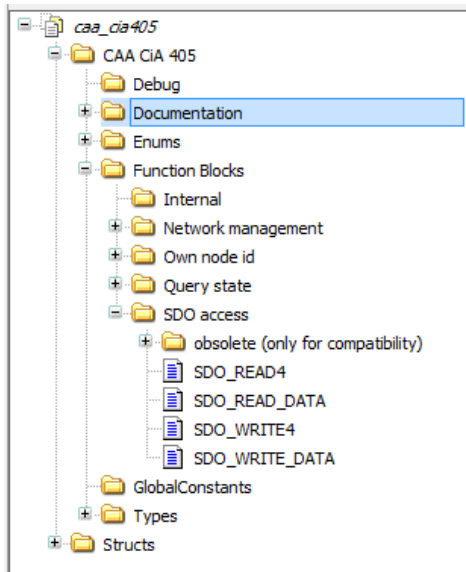
```

1  EMCY_NW (
2      NETWORK[16#01] := 1,
3      ENABLE TRUE := xRequest_EMCY TRUE,
4      TIMEOUT[16#00000000] := 0,
5      CONFIRM TRUE => xConfirm_EMCY TRUE,
6      ERROR=> ,
7      DEVICE[16#0F] => EMCY_NodeID[16#0F],
8      ERRORINFO=> EMCY_INFO);

```

The corresponding manual can be used to make the error interpretation.

## 8.4 SDO handling Function Blocks

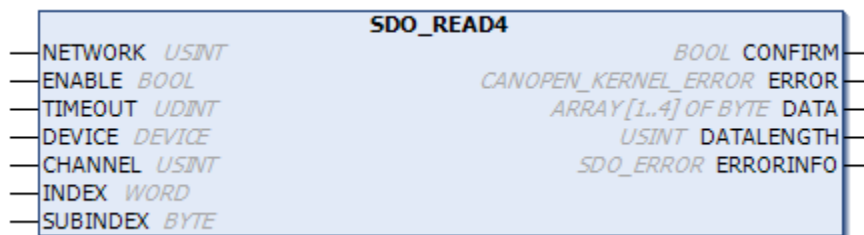


There are two possibilities for handling the SDO acyclic communication within the PLC's runtime:

### 8.4.1 SDO\_READ4/SDO\_WRITE4

SDO\_READ4/SDO\_WRITE4 can be used when the SDO has a size of 1 up to 4 bytes.

#### 8.4.1.1 SDO\_READ4



**Network** - In which CANbus network will the SDO\_READ4 request be executed. This number is the CANbus Network Number (0 for Festo PLCs) + 1 as default offset.

**Enable** – Enables FB in rising edge. Aborts request on falling edge.

**Timeout** – Set timeout in ms. 0 means no timeout.

**Device** – NodeID of the device whose SDO will be read.

**Channel** – Number of the available SDO Channel . Must be consulted in the Device SDO Channel configuration. The default SDO channel is 1.

**Index** –Index number of the SDO where the data will be read from. Consult the device manual for available Indexes with Read access.

**Subindex** – Subindex number of the SDO where the data will be read from. Consult the device manual for available Subindexes with Read access.

**Confirm** – Request finished with no error.

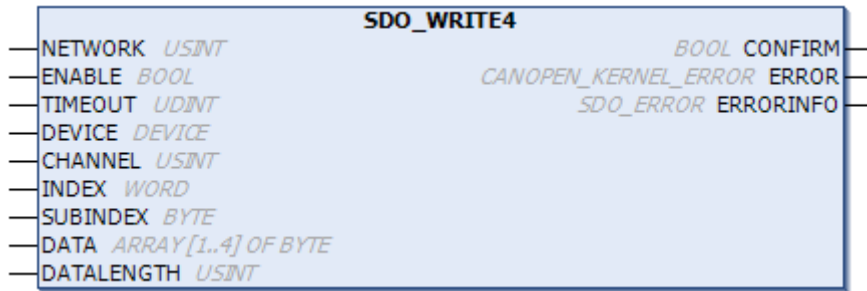
**Error** – Error Code while executing FB. Consult CANOPEN\_KERNEL\_ERROR enumeration for further details.

**Data** –Data values of the read SDO will be displayed in this array with a Little Endian byte order.

**Datalength** – The number of read bytes will be displayed in this output. It will take a value from 1 to 4.

**Errorinfo** – contains abort code in little endian in case of error.

#### 8.4.1.2 SDO\_WRITE4



**Network** - In which CANbus network will the SDO\_WRITE4 request be executed. This number is the CANbus Network Number (0 for Festo PLCs) + 1 as default offset.

**Enable** – Enables FB in rising edge. Aborts request on falling edge.

**Timeout** – Set timeout in ms. 0 means no timeout.

**Device** – NodeID of the device whose SDO will be written.

**Channel** – Number of the available SDO Channel . Must be consulted in the Device SDO Channel configuration. The default SDO channel is 1.

**Index** –Index number of the SDO where the data will be written to. Consult the device manual for available Indexes with Write access.

**Subindex** – Subindex number of the SDO where the data will be written to. Consult the device manual for available Subindexes with Write access.

**Data** –Data values of the SDO in Little Endian byte order. Array element [1] must always be filled with the Least Significant Byte.

**Datalength** – The number of bytes that will be written. It must be a value from 1 to 4.

**Confirm** – Request finished with no error.

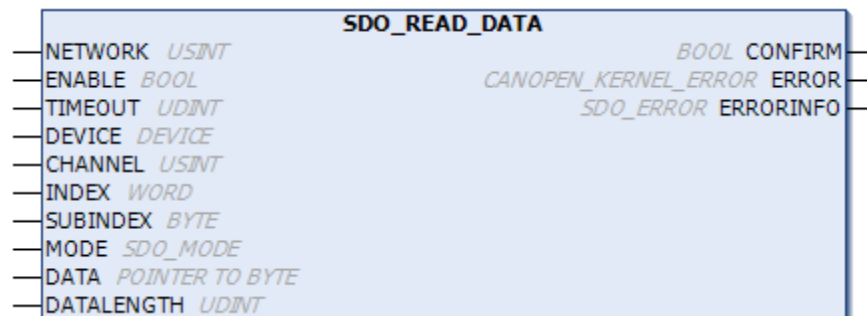
**Error** – Error Code while executing FB. Consult CANOPEN\_KERNEL\_ERROR enumeration for further details.

**Errorinfo** – contains abort code in little endian in case of error.

#### 8.4.2 SDO\_READ\_DATA/SDO\_WRITE\_DATA

SDOs that are larger than 4 Bytes must be handled with the SDO\_READ\_DATA/SDO\_WRITE\_DATA FB's. These blocks can also handle SDOs with a length between 1 and 4 Bytes.

##### 8.4.2.1 SDO\_READ\_DATA



**Network** - In which CANbus network will the SDO\_READ\_DATA request be executed. This number is the CANbus Network Number (0 for Festo PLCs) + 1 as default offset.

**Enable** – Enables FB in rising edge. Aborts request on falling edge.

**Timeout** – Set timeout in ms. 0 means no timeout.

**Device** – NodeID of the device whose SDO will be read.

**Channel** – Number of the available SDO Channel . Must be consulted in the Device SDO Channel configuration. The default SDO channel is 1.

**Index** –Index number of the SDO where the data will be read from. Consult the device manual for available Indexes with Read access.

**Subindex** – Subindex number of the SDO where the data will be read from. Consult the device manual for available Subindexes with Read access.

**Mode** – Input to specify which SDO mode will be used.

- **Auto** – SDO Client will select between the 3 modes.
- **Expedited** – For an SDO between 1 and 4 bytes of length.
- **Segmented** – For an SDO larger than 4 bytes of length.
- **Block** – For an SDO larger than 4 Bytes of length. Recommended if SDO length is bigger than 28 Bytes. Supported only by a few devices.

**Data** – Pointer to memory address where the read SDO data will be written to in Little Endian Byte order.

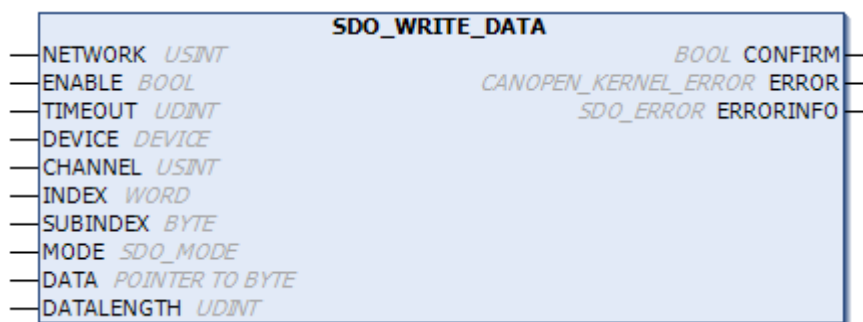
**Datalength** – Input: Size of memory DATA is pointing to. Must be different than 0.  
- Output: Amount of data written to DATA.

**Confirm** – Request finished with no error.

**Error** – Error Code while executing FB. Consult CANOPEN\_KERNEL\_ERROR enumeration for further details.

**Errorinfo** – contains abort code in little endian in case of error.

#### 8.4.2.2 SDO\_WRITE\_DATA



**Network** - In which CANbus network will the SDO\_WRITE\_DATA request be executed. This number is the CANbus Network Number (0 for Festo PLCs) + 1 as default offset.

**Enable** – Enables FB in rising edge. Aborts request on falling edge.

**Timeout** – Set timeout in ms. 0 means no timeout.

**Device** – NodeID of the device whose SDO will be written.

**Channel** – Number of the available SDO Channel . Must be consulted in the Device SDO Channel configuration. The default SDO channel is 1.

**Index** – Index number of the SDO where the data will be written to. Consult the device manual for available Indexes with Write access.

**Subindex** – Subindex number of the SDO where the data will be written to. Consult the device manual for available Subindexes with Write access.

**Mode** – Input to specify which SDO mode will be used.

- **Auto** – SDO Client will select between the 3 modes.
- **Expedited** – For an SDO between 1 and 4 bytes of length.
- **Segmented** – For an SDO larger than 4 bytes of length.
- **Block** – For an SDO larger than 4 Bytes of length. Recommended if SDO length is bigger than 28 Bytes. Supported only by a few devices.

**Data** – Pointer to memory address where the write SDO data will be read from in Little Endian Byte order.

**Datalength** – Length of data in Bytes.

**Confirm** – Request finished with no error.

**Error** – Error Code while executing FB. Consult CANOPEN\_KERNEL\_ERROR enumeration for further details.

**Errorinfo** – contains abort code in little endian in case of error.