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
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 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. 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

1  Objective .................................................................................................................. 4

2  Components/Software used ...................................................................................... 5

3  CODESYS V3 – Setting up the CANopen Network .................................................. 6

3.1 Adding the CANbus and CANopen Manager .......................................................... 6

3.2 Adding CANopen devices ....................................................................................... 7

3.2.1 Installing the EDS File ....................................................................................... 8

4  CANopen Network Parameters .............................................................................. 11

4.1 Basics ..................................................................................................................... 11

4.1.1 Baudrate or data transmission speed ................................................................... 11

4.1.2 Node ID ............................................................................................................... 12

5  CANbus Parameters ................................................................................................. 12

5.1 General Tab ........................................................................................................... 12

6  CANopen Manager Parameters ............................................................................. 13

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

7  CANopen device parameters .................................................................................. 17

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

8  Network Management (NMT) and Diagnostics in CODESYS V3 ...................... 34

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.
## Components/Software used

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

Table 2.1: 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.

To install 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**

<table>
<thead>
<tr>
<th>Bit rate (Kbit/s)</th>
<th>Bus length (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>20^</td>
</tr>
<tr>
<td>800</td>
<td>50</td>
</tr>
<tr>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>125</td>
<td>500</td>
</tr>
<tr>
<td>50</td>
<td>1000</td>
</tr>
<tr>
<td>20</td>
<td>2500</td>
</tr>
<tr>
<td>10</td>
<td>5000</td>
</tr>
</tbody>
</table>

*) 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.

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 CiA405 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:** 127
  - **Producer Timer (ms):** 200

**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):** 15 # 80
  - **Cycle Period (us):** 8000
  - **Window Length (us):** 0
  - **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 (us)**.

The **Cycle Period (us)** 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 (us)** must be set. The result of this is that Sync telegrams can only be transmitted within the **Window Length (us)**, 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

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.
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.

**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.

<table>
<thead>
<tr>
<th>Nodeguarding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Nodeguarding</td>
</tr>
<tr>
<td>Guard Time (ms):</td>
</tr>
<tr>
<td>Life Time Factor:</td>
</tr>
<tr>
<td>Enable Heartbeat Producing</td>
</tr>
<tr>
<td>Producer Time (ms):</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Emergency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Emergency</td>
</tr>
<tr>
<td>COB-ID:</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable TIME Producing</td>
</tr>
<tr>
<td>COB-ID (Hex):</td>
</tr>
<tr>
<td>Enable TIME Consuming</td>
</tr>
</tbody>
</table>

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

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.
### 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.

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

![Image of CANopen device parameters](image-url)

**Operation Parameters | Factor Group**

**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**

- **Use Parameter Channel**

**Receive PDOs (Master => Slave)**

**Transmit PDOs (Slave => Master)**

**Name**

- 16#1800: TPDO 1 Communication Parameter
  - SCON
  - SPOS
  - REC_NR/SDIR
  - RSB/ACT_VAL1
  - ACT_POS/ACT_VAL2

- 16#1801: TPDO 2 Communication Parameter
  - RES
  - SUBINDEX
  - RESCODE_PNU
  - PARVAL

- 16#1802: TPDO 3 Communication Parameter
  - FHPP+ _OByte01
  - FHPP+ _OByte02
  - FHPP+ _OByte03
  - FHPP+ _OByte04
  - FHPP+ _OByte05
  - FHPP+ _OByte06
  - FHPP+ _OByte07
  - FHPP+ _OByte08

- 16#1803: TPDO 4 Communication Parameter
  - FHPP+ _OByte09
  - FHPP+ _OByte10
  - FHPP+ _OByte11
  - FHPP+ _OByte12
  - FHPP+ _OByte13
  - FHPP+ _OByte14
  - FHPP+ _OByte15
  - FHPP+ _OByte16

**Name**

- 16#1400: RPDO 1 C
  - 16#202 ($NOD 64
  - CCON
  - CPOS
  - REC_NR/CDIR
  - RES/DEM_VAL1/PARA1
  - RES/DEM_VAL2/PARA2

- 16#1401: RPDO 2 C
  - 16#3016:16#00 8
  - RES
  - SUBINDEX
  - REQCODE_PNU
  - PARVAL

- 16#1402: RPDO 3 C
  - 16#400: ($NOD 64
  - FHPP+ _OByte01
  - FHPP+ _OByte02
  - FHPP+ _OByte03
  - FHPP+ _OByte04
  - FHPP+ _OByte05
  - FHPP+ _OByte06
  - FHPP+ _OByte07
  - FHPP+ _OByte08

- 16#1403: RPDO 4 C
  - 16#500: ($NOD 64
  - FHPP+ _OByte09
  - FHPP+ _OByte10
  - FHPP+ _OByte11
  - FHPP+ _OByte12
  - FHPP+ _OByte13
  - FHPP+ _OByte14
  - FHPP+ _OByte15
  - FHPP+ _OByte16
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.

The following properties are available:

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:

\[
\text{minimum Window Length} \approx \frac{\text{Cyclic Synchronous Bits}}{\text{Baudrate}}
\]
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.

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:

\[ \text{minimum Window Length} \approx \frac{\text{Cyclic Synchronous Bits}}{\text{Baudrate}} \]

\[ \text{minimum Window Length} \approx \frac{256 \text{ bit}}{125.000 \text{ bit/s}} \]

\[ \text{minimum Window Length} \approx 2,05 \text{ ms} \]

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 5 ms.

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

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.
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.

<table>
<thead>
<tr>
<th>Line</th>
<th>Index/Subindex</th>
<th>Name</th>
<th>Value</th>
<th>Bit length</th>
<th>Abort if error</th>
<th>Jump to line if error</th>
<th>Next line</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16#1005:16#00</td>
<td>Set COB-ID sync</td>
<td>16#00000800</td>
<td>32</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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**.

**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 downloaded 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:

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.

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.

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2400</td>
<td>System Parameter</td>
</tr>
<tr>
<td>0x2410</td>
<td>Parameter Modul 0</td>
</tr>
<tr>
<td>0x2411</td>
<td>Parameter Modul 1</td>
</tr>
<tr>
<td>0x2412</td>
<td>Parameter Modul 2</td>
</tr>
</tbody>
</table>

- :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.
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.
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.

<table>
<thead>
<tr>
<th>Inputs</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10: Diagnostic functions</td>
<td>Monitor parameters</td>
<td></td>
</tr>
<tr>
<td>11: Diagnostic functions</td>
<td>Monitor parameters</td>
<td></td>
</tr>
<tr>
<td>12: Diagnostic functions</td>
<td>Monitor parameters</td>
<td></td>
</tr>
<tr>
<td>13: Diagnostic functions</td>
<td>Monitor parameters</td>
<td></td>
</tr>
<tr>
<td>10: Signal range</td>
<td>No sensor connected</td>
<td></td>
</tr>
<tr>
<td>11: Signal range</td>
<td>0..10V</td>
<td></td>
</tr>
<tr>
<td>12: Signal range</td>
<td>4..20mA</td>
<td></td>
</tr>
<tr>
<td>13: Signal range</td>
<td>No sensor connected</td>
<td></td>
</tr>
</tbody>
</table>
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.

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.

- **INIT**
- **RESET_COMM**
- **RESET_APP**
- **PRE_OPERATIONAL**
- **STOPPED**
- **OPERATIONAL**
- **UNKNOWN**
- **NOT_AVAIL**

### 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.

**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.
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Initial</th>
</tr>
</thead>
<tbody>
<tr>
<td>START_REMOTE_NODE</td>
<td>16#5</td>
</tr>
<tr>
<td>STOP_REMOTE_NODE</td>
<td>16#4</td>
</tr>
<tr>
<td>ENTER_PRE_OPERATIONAL</td>
<td>16#7F</td>
</tr>
<tr>
<td>RESET_NODE</td>
<td>16#6</td>
</tr>
<tr>
<td>RESET_COMMUNICATION</td>
<td>16#7</td>
</tr>
<tr>
<td>ALL_EXCEPT_NMT_AND_SENDER</td>
<td>16#800</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>COB-ID</th>
<th>Byte-0</th>
<th>Byte-1</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>CS</td>
<td>Node-ID</td>
<td>bei Node-ID = 0 sind alle im Netz befindlichen Nodes adressiert</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>START Remote Node</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>STOP Remote Node</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0x80</td>
<td>Enter Pre-operational state</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0x81</td>
<td>Reset Node</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0x82</td>
<td>Reset Communication</td>
</tr>
</tbody>
</table>

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

```
PROGRAM PLC_PRG
VAR
CPXF14_DeviceState_CiA405: CIA405.DEVICE_STATE;
//CPXF14_DeviceState_3SCOS: 3SCOS.DEVICE_STATE;
Send_NMT_Messages: CIA405.NMT;
NMT_Device_State_Request: CIA405.TRANSITION_STATE;
*ToggleNMT: BOOL;
END_VAR

CPXF14_DeviceState_CiA405:= myCPX.CANopenState;
//CPXF14_DeviceState_3SCOS:= myCPX.CANopenState;

Send_NMT_Messages(
    NETWORK:= 1,
    ENABLE:= *ToggleNMT,
    TIMEOUT:= 0,
    CONFIRM=>, 
    ERROR=>, 
    DEVICE:= 6,
    STATE:= NMT_Device_State_Request);
```

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

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.

```
PROGRAM PLC_PRG
VAR
  EMCY_DEV_CPX: CIA405.RECV_EMCY_DEV;
  xRequest_EMCYCPX: BOOL;
  xConfirm_EMCYCPX: BOOL;
  CPX_EMCY_INFO: CIA405.EMCY_ERROR;
END_VAR

EMCY_DEV_CPX(
  NETWORK:= 1,
  ENABLE:= xRequest_EMCYCPX,
  TIMEOUT:= 0,
  CONFIRM:= xConfirm_EMCYCPX,
  ERROR:= ,
  DEVICE:= 6,
  ERRORINFO:= CPX_EMCY_INFO);
```

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

The implementation of the RECV_EMCY FB in structured text should look like this:

```
PROGRAM PLC_PRG

VAR
  EMCY_NW: CIA405.RECV_EMCY;
  xRequest_EMCY: BOOL;
  xConfirm_EMCY: BOOL;
  EMCY_INFO: CIA405.EMCY_ERROR;
  EMCY_NodeID: USINT;

END_VAR

EMCY_NW(
  NETWORK:= 1,
  ENABLE:= xRequest_EMCY,
  TIMEOUT:= 0,
  CONFIRM=> xConfirm_EMCY,
  ERROR=> ,
  DEVICE=> EMCY_NodeID,
  ERRORINFO=> EMCY_INFO);
```

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.
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.

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

```
<table>
<thead>
<tr>
<th>NETWORK</th>
<th>USINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLE</td>
<td>BOOL</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>USINT</td>
</tr>
<tr>
<td>DEVICE</td>
<td>DEVICE</td>
</tr>
<tr>
<td>CHANNEL</td>
<td>USINT</td>
</tr>
<tr>
<td>INDEX</td>
<td>WORD</td>
</tr>
<tr>
<td>SUBINDEX</td>
<td>BYTE</td>
</tr>
<tr>
<td>MODE</td>
<td>SDM_MODE</td>
</tr>
<tr>
<td>DATA</td>
<td>POINTER TO BYTE</td>
</tr>
<tr>
<td>DATALength</td>
<td>USINT</td>
</tr>
</tbody>
</table>
```

**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.