

# Codesys Application Note

Product Family  
Integrated Drive  
**NLi080QX with CANBUS**  
**NLi120QX with CANBUS**

## Index

<b>1</b>	<b>Objective.....</b>	<b>3</b>
<b>2</b>	<b>Components/Software used.....</b>	<b>4</b>
<b>3</b>	<b>CODESYS V3 - Setting up a CANopen Network.....</b>	<b>5</b>
3.1	Create a CODESYS V3 project with CANbus.....	5
3.2	Adding CANopen Manager and CANopen devices.....	8
3.3	Configuration of CANopen Master.....	11
3.4	Configuration of CANopen slave device.....	12
3.5	Adding and configure a SM_Drive_GenericDSP402.....	16
<b>4</b>	<b>CODESYS V3 Function Blocks.....</b>	<b>19</b>
4.1	MC_Power.....	21
4.2	MC_MoveAbsolute.....	22
4.3	SDO_READ4.....	24
4.4	SDO_WRITE4.....	25

# 1 Objective

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

It contains great details on what the CANopen parameters in CODESYS V3 are, what the programmer should do with them and expect from them, as well as describing the CANopen Function Blocks available in CODESYS V3 for further Network Management and Diagnostic within the running Task.

For the following images a CMZ-FCT640 has been used as the CANopen Master, but the description apply for all CMZ CANopen Masters in CODESYS V3.

## 2 Components/Software used

CODESYS V3.5	SP17 Patch 2
FCT640 (CMZ Sistemi Elettronici)	3.5.11.13

**Type/Name**

**Version Software/Firmware**

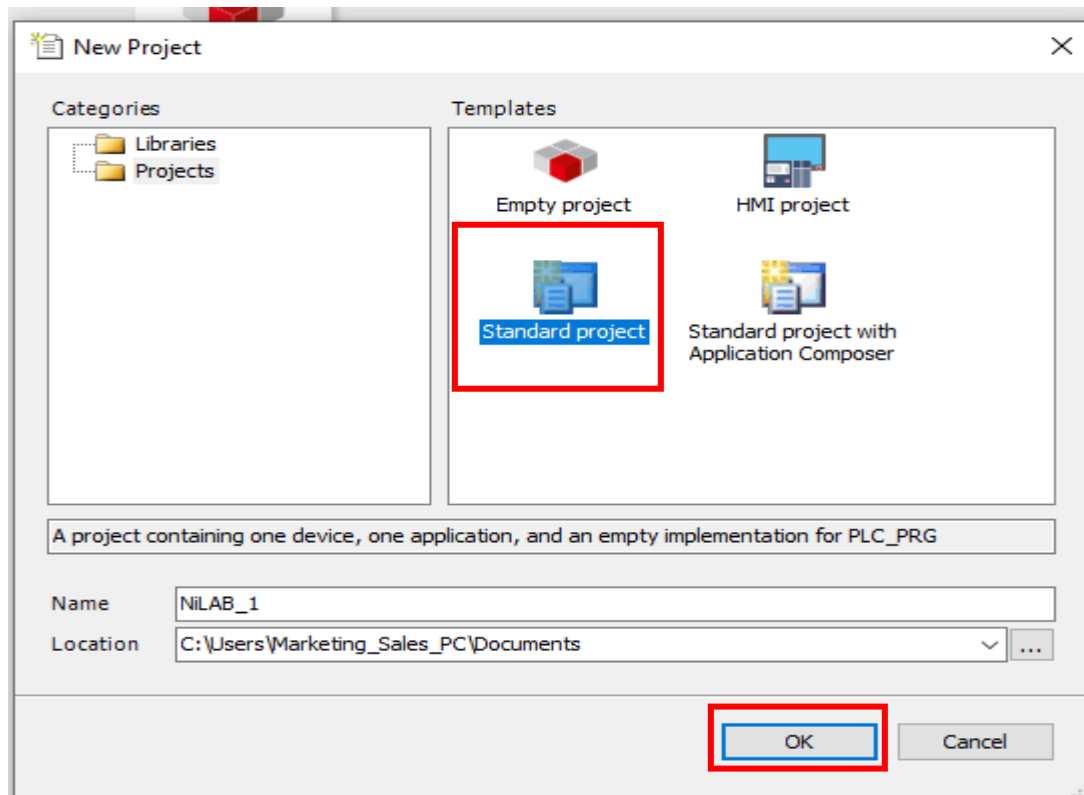
## 3 CODESYS V3 - Setting up a CANopen Network

This chapter will show how to create a CANopen Network, starting from adding the CANbus, CANopen Manager and slave devices in the CODESYS V3 environment, as well as giving the instructions on how to install a .eds file if the slave device is missing from the Device Repository of CODESYS V3.

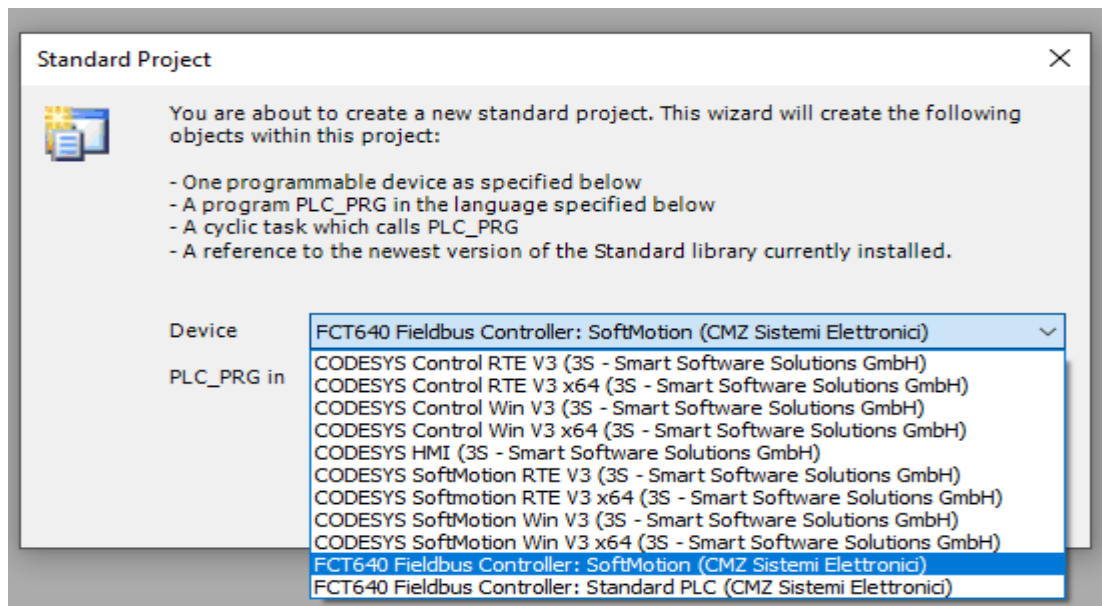
Remember that to use CMZ PLCs, the CMZ CODESYS Library installation is needed before the starting of a new project. On the CMZ website ([cmz.it](http://cmz.it)), login with the credentials provided by CMZ and search for the "Controllers" section, click on it and select the FCT640 folder. In the "Manuals and documents" folder search for the file "FCT CODESYS series User Guide Software" (there is 2 version of it, one in English and one in Italian) and open it. Follow the procedure to configure the controller for CODESYS usage.

### 3.1 Create a CODESYS V3 project with CANbus

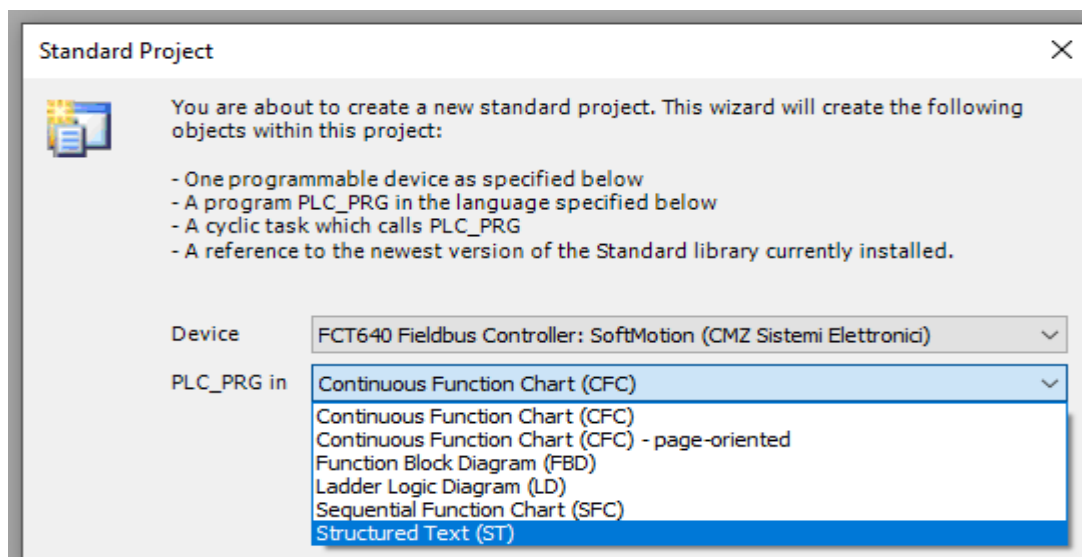
To create a new project on CODESYS V3 select "New Project", then select "Standard Project" from the templates menu. Give a name to the file and click "Ok".



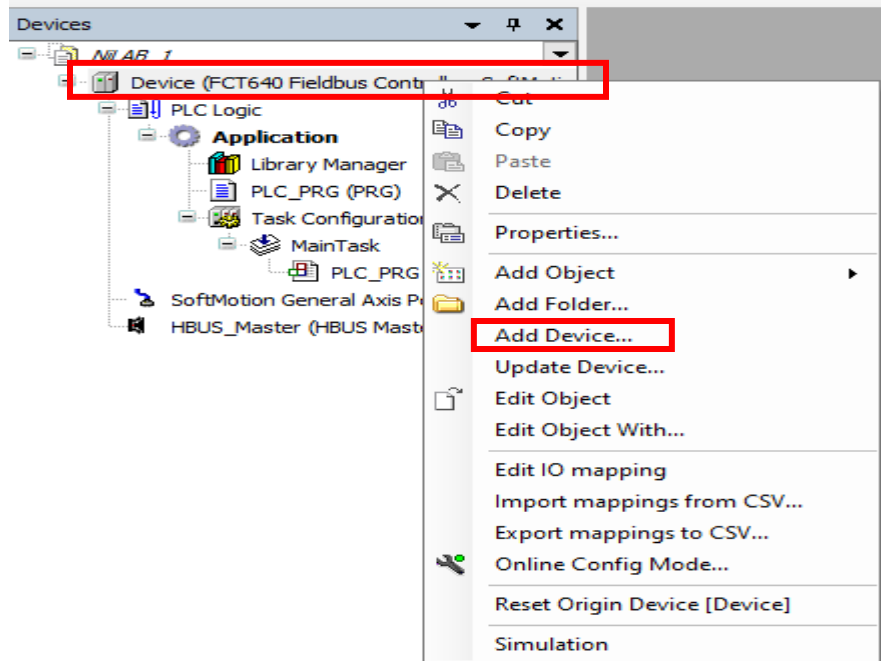
When clicking on "Ok", this other window will appear, select the PLC from "Device" (in this example we will use the FCT640 Fieldbus Controller: Soft Motion from CMZ).



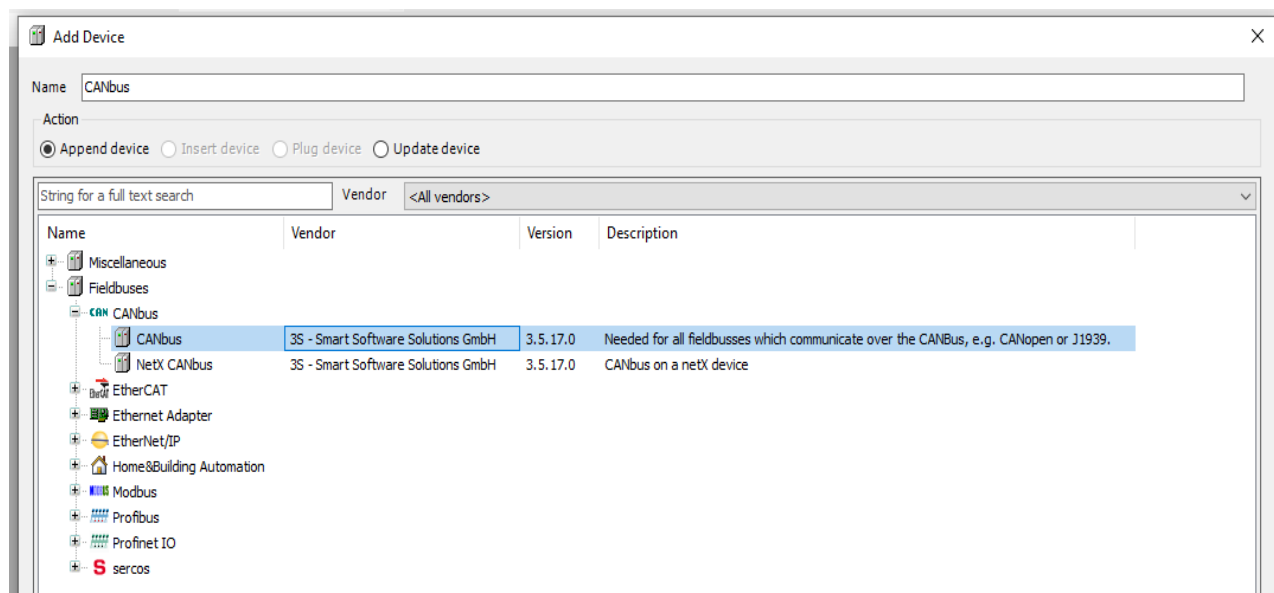
In the PLC\_PRG section select the program language to use into the CODESYS V3 project (we will use Structured Text on this example).



Now the CODESYS V3 project should open, search for the "Device" section in the top-left part of the page and right click on it.

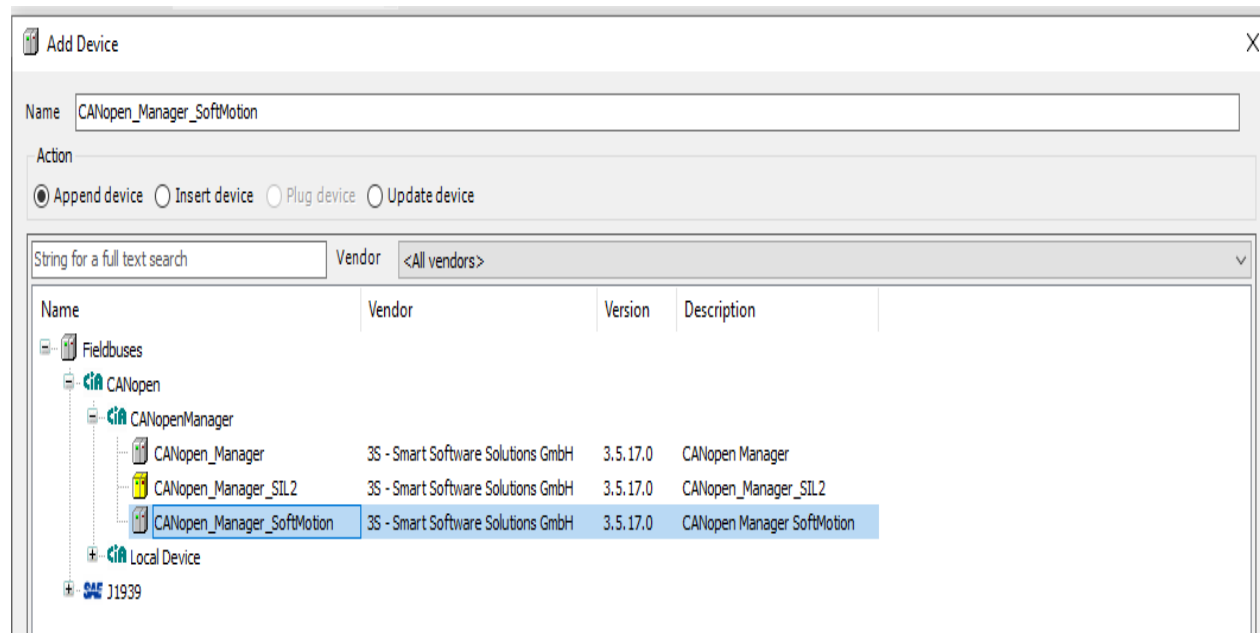


Select "Add device" and click on the "Fieldbuses" icon and a drop down menu will appear. From there select "CANbus" and click on the CANbus icon, then click on "Add Device" in the lower-right part of the page (double clicking on it will add it also).

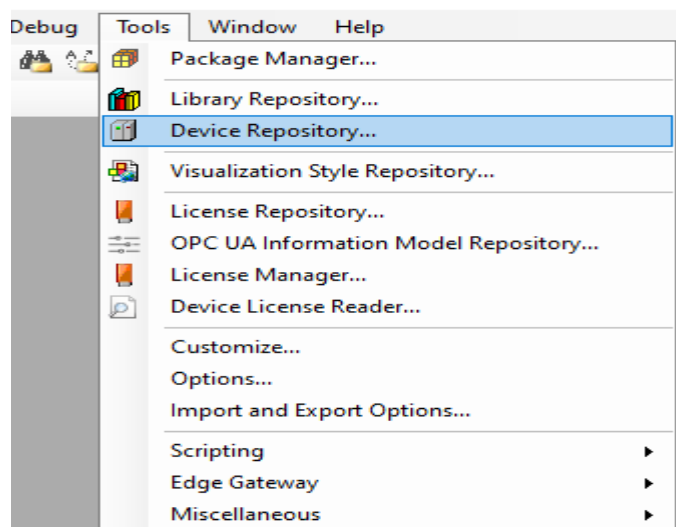


## 3.2 Adding CANopen Manager and CANopen devices

Now to add an available device to the CANbus, right click on its icon and select "Add Device". Select "CANopen" under the "Fieldbuses" icon, select then CANopenManager and add the "CANopen\_Manager\_SoftMotion" (double click on it will also work).

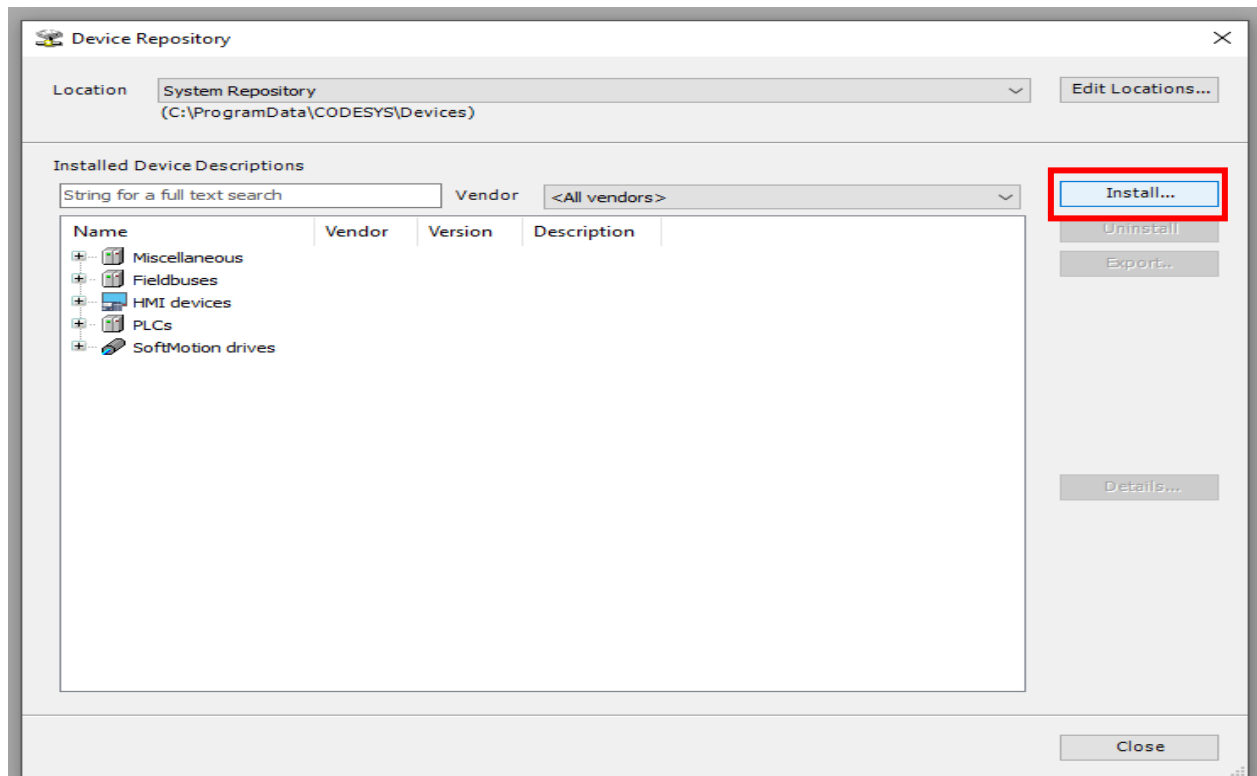


Once the CANopen Manager has been added, right click on it and select "Add Device" to add the CANopen slave devices. A drop down menu will appear showing all the devices available. To add a new device to the CODESYS V3 device repository a .eds file is needed. Find the "Tools" section in the upper part of the page and select "Device Repository".

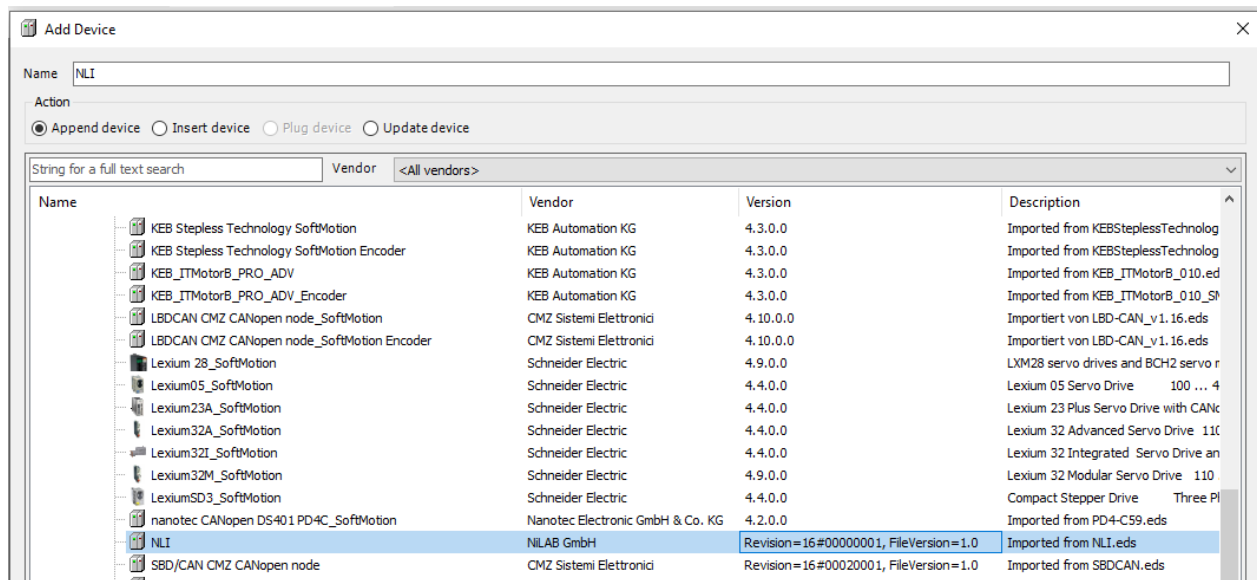
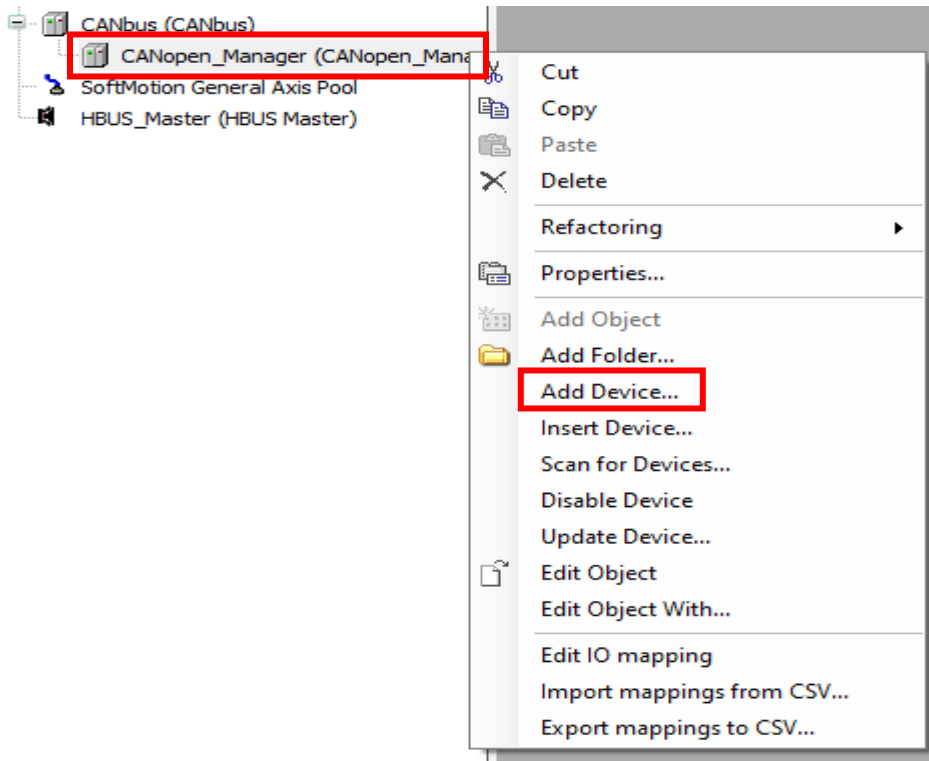




From there click on "Install" and search for the .eds file to import into CODESYS V3.



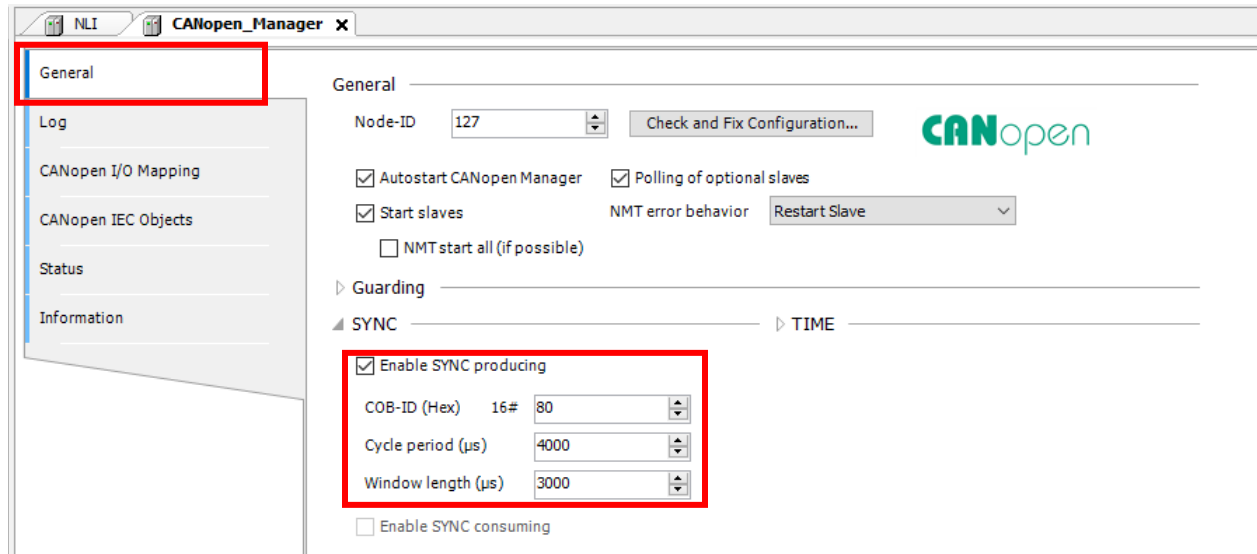
Installing the NLi.eds file into CODESYS V3 will allow to select "NLi" device. Right click on CANopen\_Manager and select "Add Device", then search for the "NLi" device and add it.



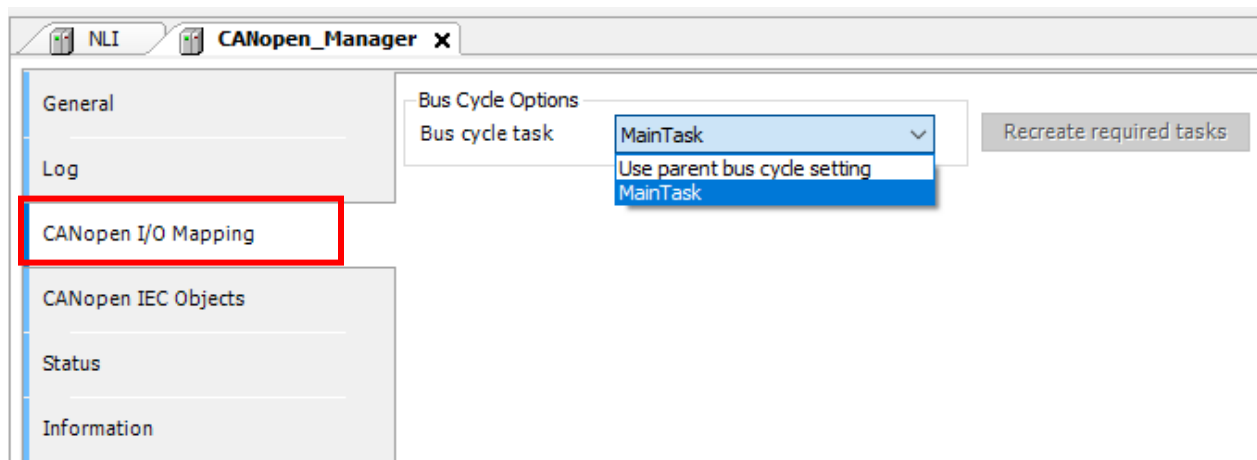
### 3.3 Configuration of CANopen Master

In this section we will explain how to configure the Master device in the CANopen Network. Select the CANopen Manager and double click on it, then go to the "General"

section and click on "Enable SYNC Producing" under the SYNC dropdown icon. We recommend to use a Cycle Period of 4000 us and a window length of 3000 us.



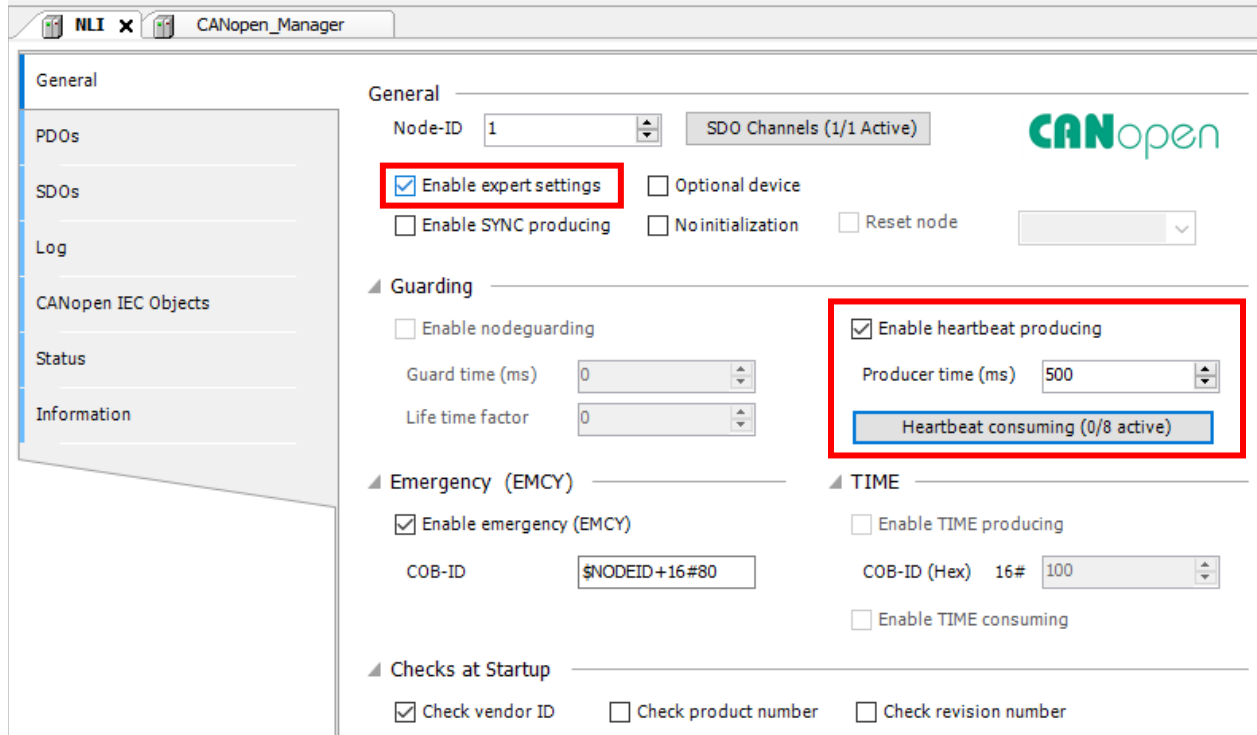
Then go in the "CANopen I/O Mapping" section and select "Main Task" as "Bus Cycle Options".



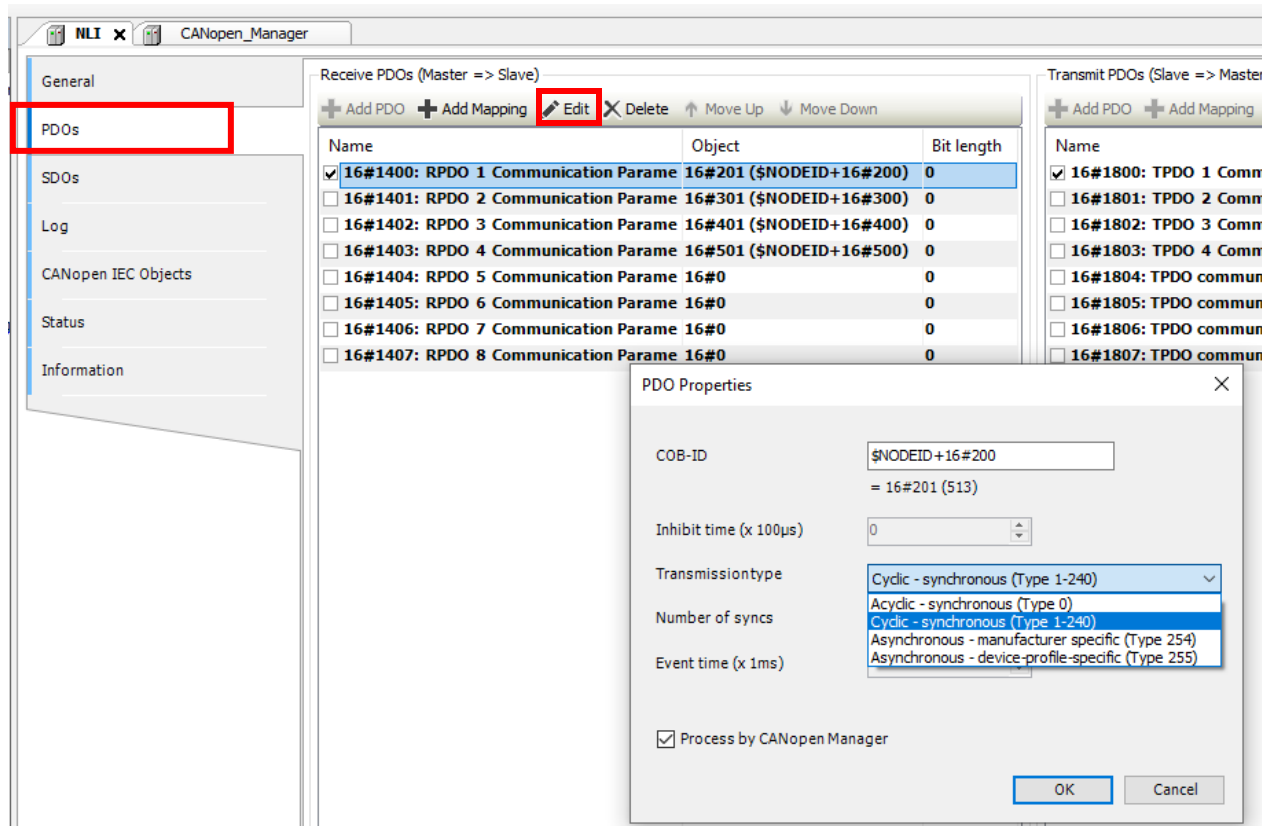
### 3.4 Configuration of CANopen slave device

In this section we will explain how to configure a CANopen slave device into the CANopen Network.

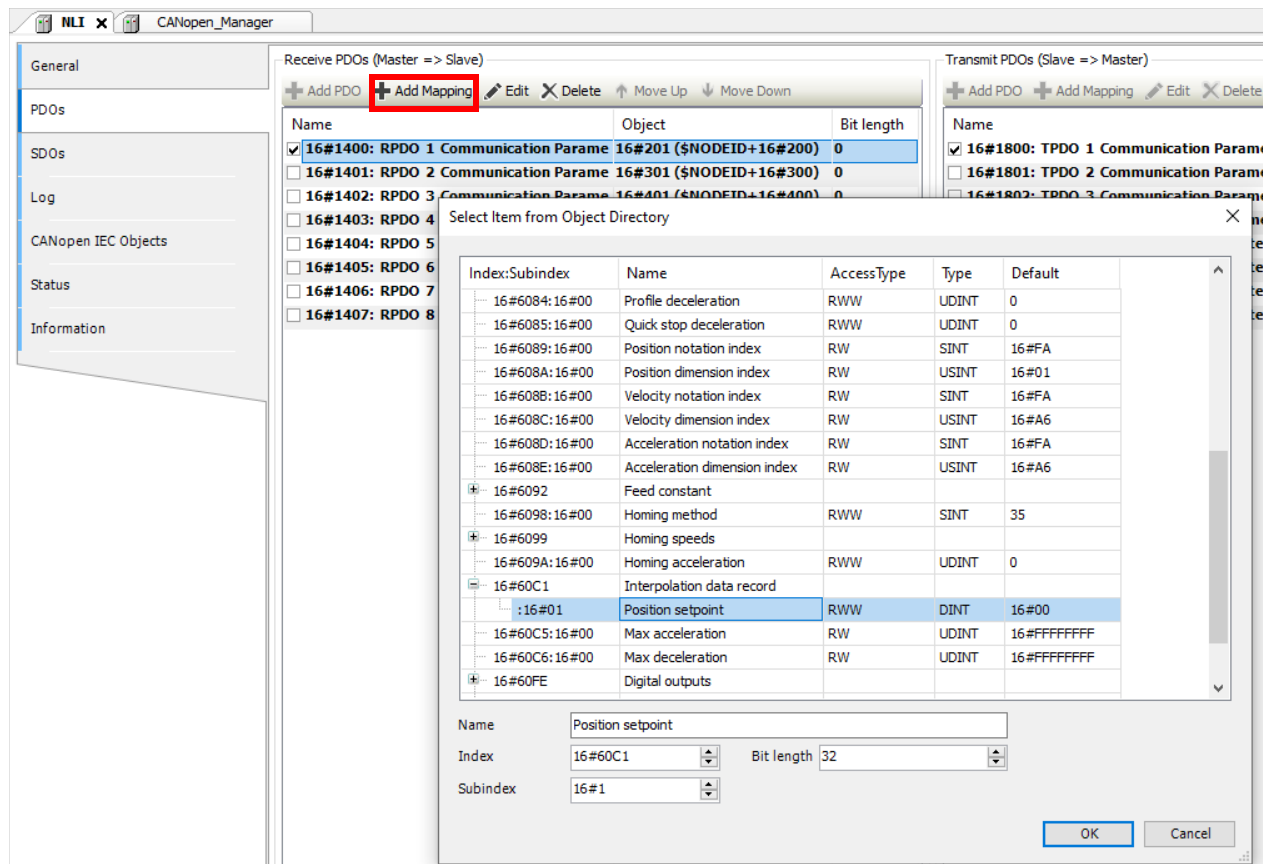
Double click on the "NLI" device and go to the "General" section, there activate "Enable expert settings" and "Enable heartbeat producing" with a producer time of 500 ms (recommended) and 0/8 "heartbeat consuming".



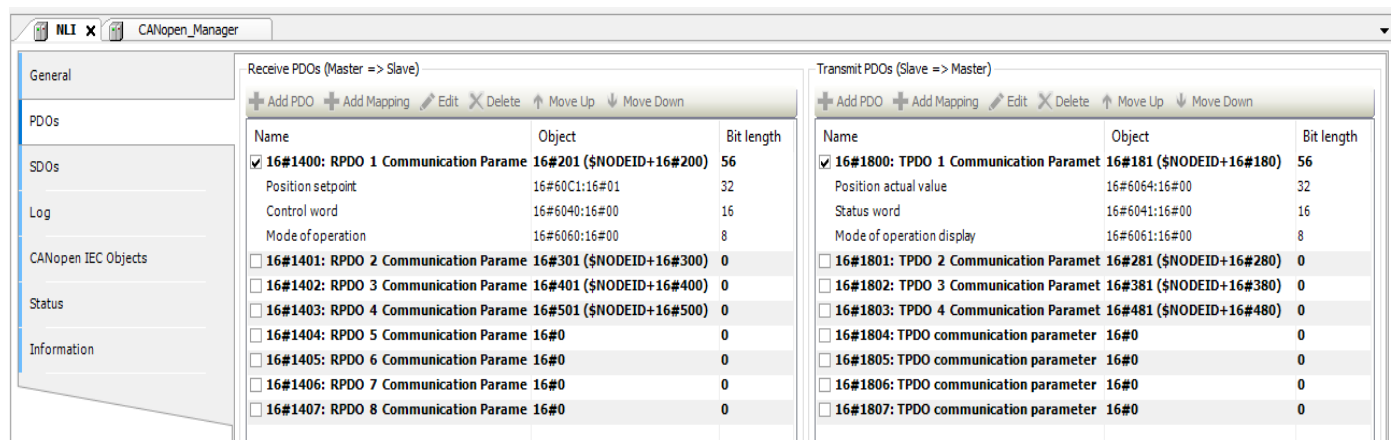
Then move to the "PDOs" section, enable only 1 RPDO and 1 TPDO from the table and right click on them and tap edit. For the RPDO select "Cyclic – synchronous (Type 1-240)" as "Transmissiontype".



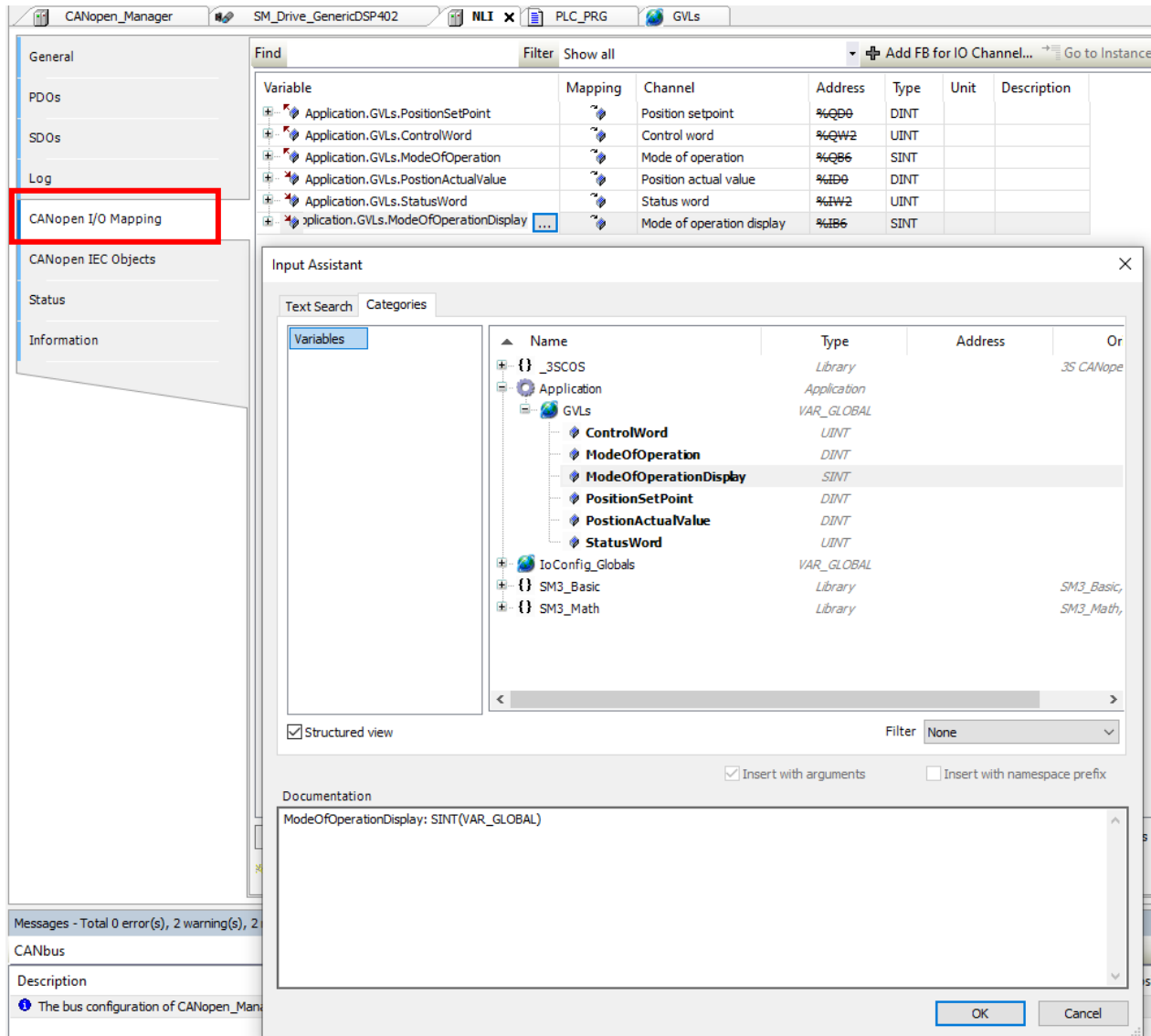
Now the RPDO must be configured, click on the enabled RPDO and tap on "+ Add Mapping". A window will open, containing the mapping table imported from the NLi.eds file. From there select these 3 parameters to insert into the RPDO : Position Setpoint (found under the "interpolation data record" dropdown menu), Control Word and Mode of Operation (NOTE: IT'S MANDATORY TO SELECT THESE PARAMETERS IN THE SHOWN ORDER).



The same thing must be done in the TPDOs section, there use "asynchronous (type 255)" as "Transmissiontype", click on "+ Add Mapping" and select these 3 parameters from the table: Position Actual Value, Status Word and Mode of Operation Display (AS FOR THE RPDOs, PARAMETERS MUST BE ADDED IN THIS ORDER).



Now that the RPDOs and the TPDOs are configured, tap on the "CANopen I/O Mapping" section and assign a variable for every parameter of the PDOs (is recommended to map global variables into the devices, to create a global variable list right click on "Application" and select "Global variable list" from the "Add Object" section).



The screenshot shows the CANopen Manager interface with the "CANopen I/O Mapping" section selected in the left sidebar. The main window displays a table of variables and their mappings. Below this, the "Input Assistant" dialog is open, showing a tree view of variables. The "ModeOfOperationDisplay" variable is selected, and its documentation is displayed at the bottom.

Variable	Mapping	Channel	Address	Type	Unit	Description
Application.GVLs.PositionSetPoint		Position setpoint	%QB0	DINT		
Application.GVLs.ControlWord		Control word	%QW2	UINT		
Application.GVLs.ModeOfOperation		Mode of operation	%QB6	SINT		
Application.GVLs.PositionActualValue		Position actual value	%ID0	DINT		
Application.GVLs.StatusWord		Status word	%IW2	UINT		
Application.GVLs.ModeOfOperationDisplay		Mode of operation display	%IB6	SINT		

**Input Assistant**

Text Search Categories

Variables

Name	Type	Address	Or
_3SCOS	Library		3S CANope
Application	Application		
GVLs	VAR_GLOBAL		
ControlWord	UINT		
ModeOfOperation	DINT		
ModeOfOperationDisplay	SINT		
PositionSetPoint	DINT		
PositionActualValue	DINT		
StatusWord	UINT		
IoConfig_Globals	VAR_GLOBAL		
SM3_Basic	Library		SM3_Basic,
SM3_Math	Library		SM3_Math,

☒ Structured view Filter: None

☒ Insert with arguments ☐ Insert with namespace prefix

Documentation

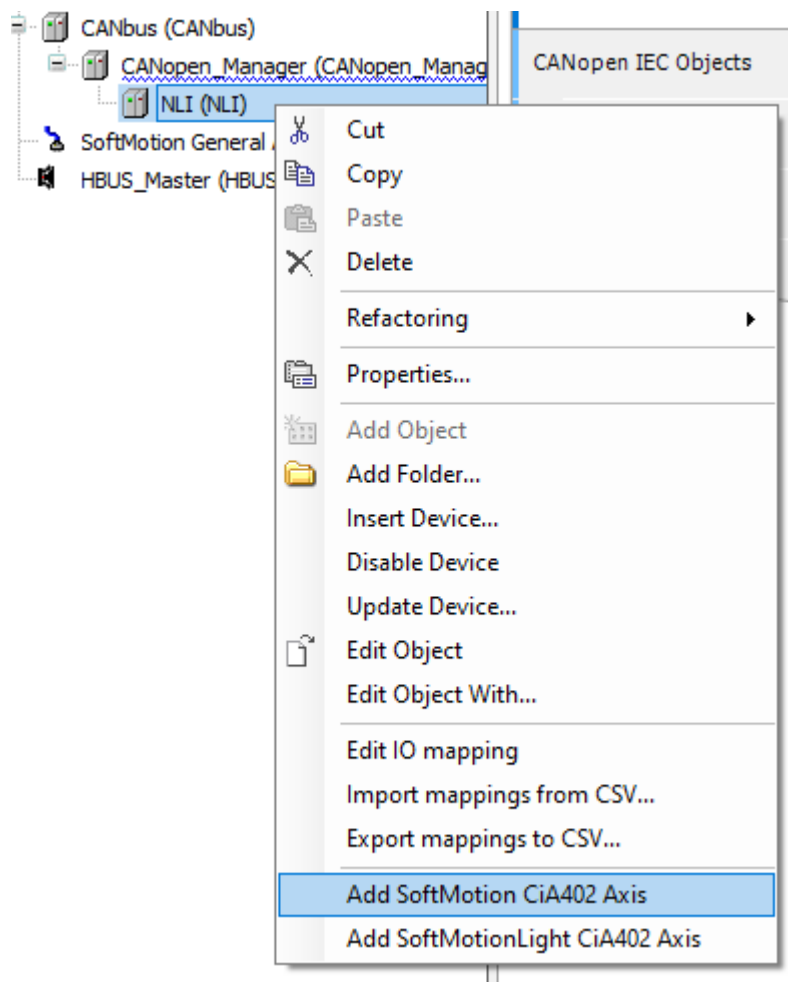
ModeOfOperationDisplay: SINT(VAR\_GLOBAL)

OK Cancel

### 3.5 Adding and configure a SM\_Drive\_GenericDSP402

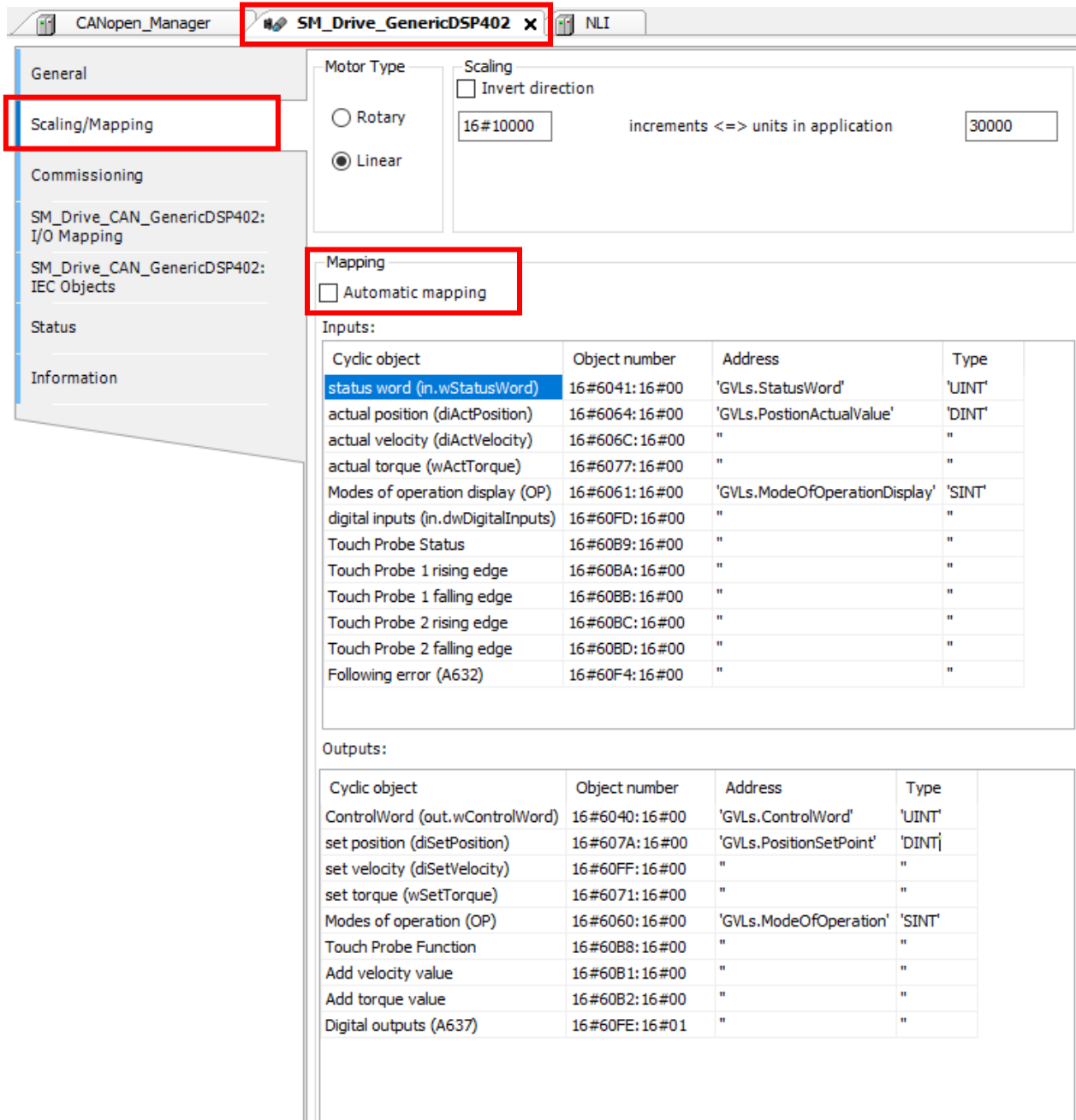
In this part is explained how to add a SM\_Drive to the "NLI" device and how to configure the axis for the cinematic.

Right click on the "NLI" device icon and select "Add SoftMotion CIA402 axis".





Under the "NLI" icon the SM\_Drive will appear, double click on it and go in the "Scaling/Mapping" section. There select "linear" as "Motor Type" and use 30000 as increments for NLI80 series engines or 60000 for NLI120 series engines. In the lower part of the page check if all the variables are right configured in the inputs and outputs section. If there are any variables missing in the list, disable the "Automatic Mapping" checkbox and insert manually the missing variables.



**General**

**Scaling/Mapping**

**Commissioning**

SM\_Drive\_CAN\_GenericDSP402: I/O Mapping

SM\_Drive\_CAN\_GenericDSP402: IEC Objects

Status

Information

**Motor Type**

☐ Rotary

☒ Linear

**Scaling**

☐ Invert direction

16#10000 increments <=> units in application 30000

**Mapping**

☐ Automatic mapping

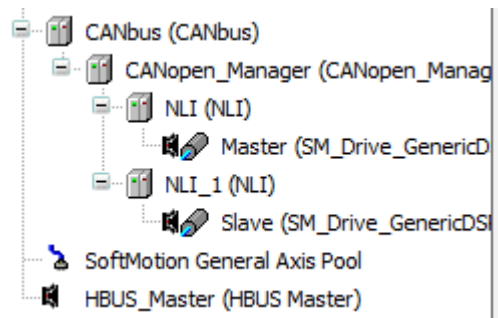
**Inputs:**

Cyclic object	Object number	Address	Type
status word (in.wStatusWord)	16#6041: 16#00	'GVLS.StatusWord'	'UINT'
actual position (diActPosition)	16#6064: 16#00	'GVLS.PositionActualValue'	'DINT'
actual velocity (diActVelocity)	16#606C: 16#00	"	"
actual torque (wActTorque)	16#6077: 16#00	"	"
Modes of operation display (OP)	16#6061: 16#00	'GVLS.ModeOfOperationDisplay'	'SINT'
digital inputs (in.dwDigitalInputs)	16#60FD: 16#00	"	"
Touch Probe Status	16#60B9: 16#00	"	"
Touch Probe 1 rising edge	16#60BA: 16#00	"	"
Touch Probe 1 falling edge	16#60BB: 16#00	"	"
Touch Probe 2 rising edge	16#60BC: 16#00	"	"
Touch Probe 2 falling edge	16#60BD: 16#00	"	"
Following error (A632)	16#60F4: 16#00	"	"

**Outputs:**

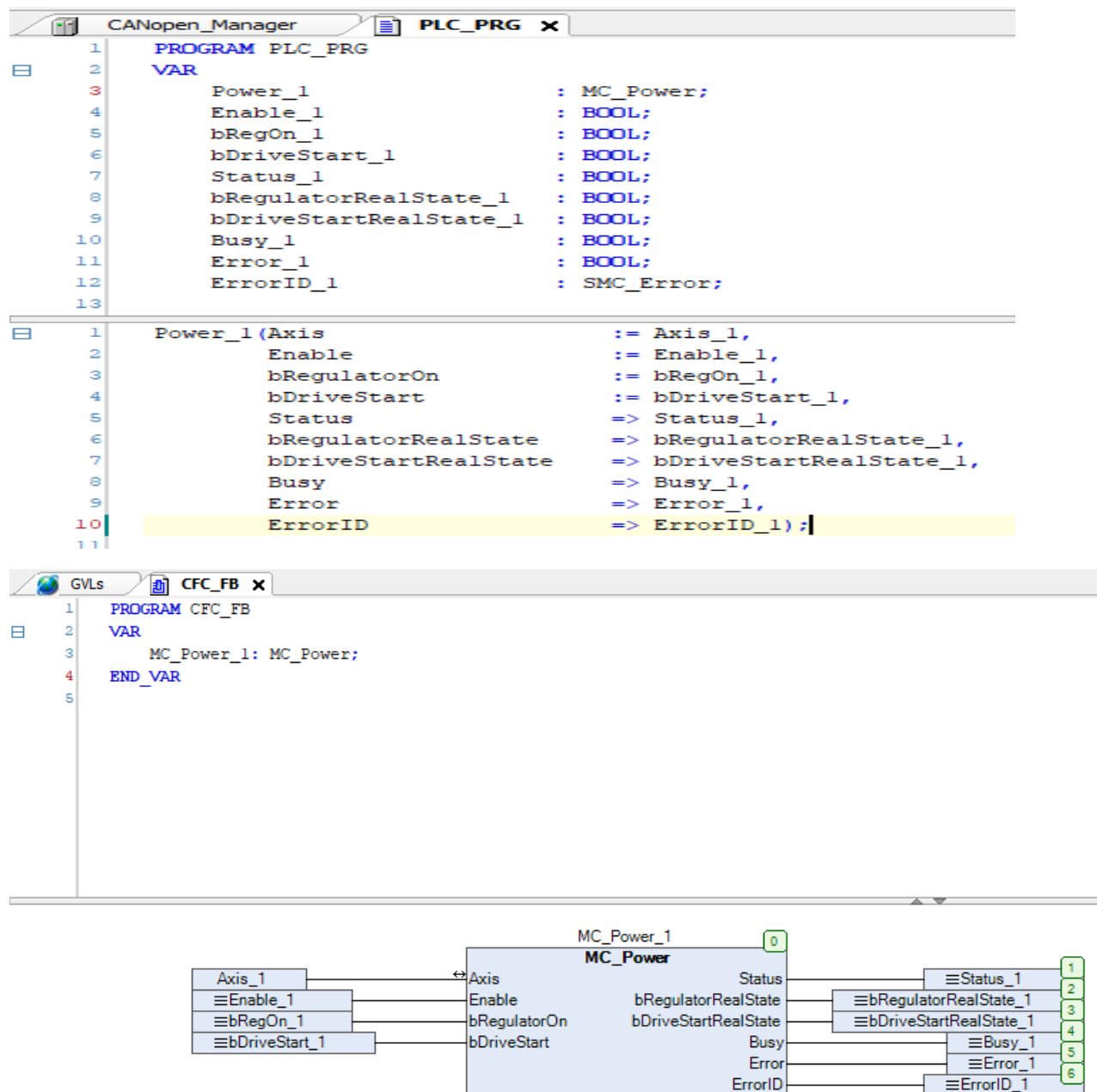
Cyclic object	Object number	Address	Type
ControlWord (out.wControlWord)	16#6040: 16#00	'GVLS.ControlWord'	'UINT'
set position (diSetPosition)	16#607A: 16#00	'GVLS.PositionSetPoint'	'DINT'
set velocity (diSetVelocity)	16#60FF: 16#00	"	"
set torque (wSetTorque)	16#6071: 16#00	"	"
Modes of operation (OP)	16#6060: 16#00	'GVLS.ModeOfOperation'	'SINT'
Touch Probe Function	16#60B8: 16#00	"	"
Add velocity value	16#60B1: 16#00	"	"
Add torque value	16#60B2: 16#00	"	"
Digital outputs (A637)	16#60FE: 16#01	"	"

To instantiate 2 or more SM\_Drives in the same project, is necessary to add a "NLI" device for each drive in the same way of the first one, but it will need a different nominative and different variables mapped in.

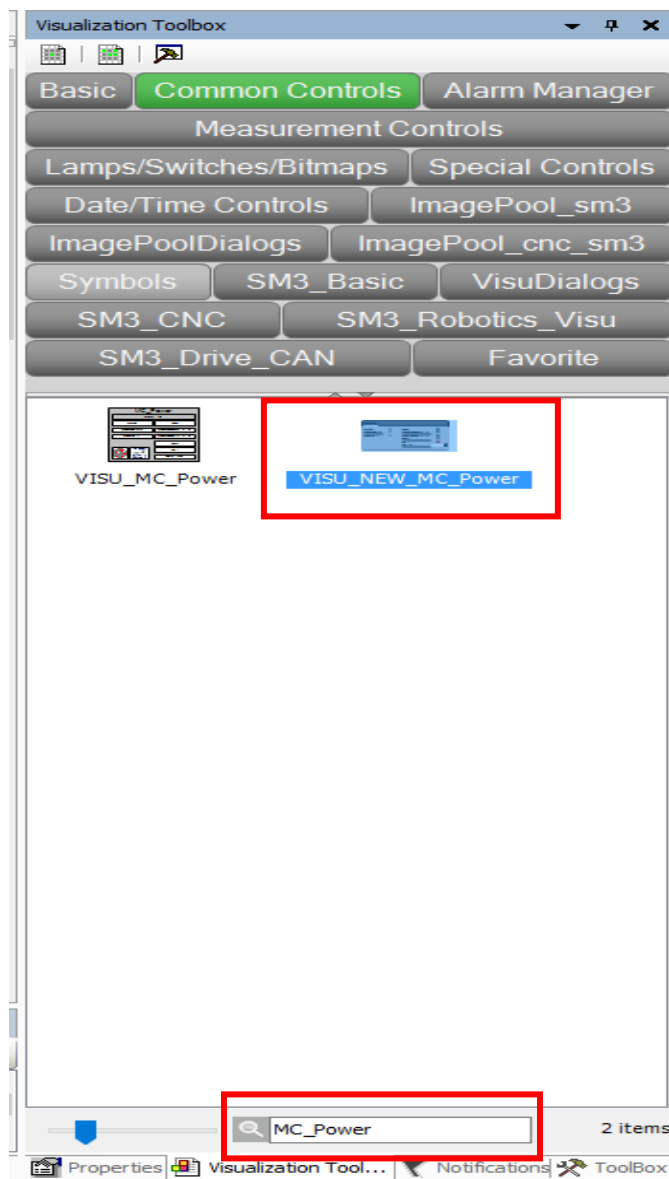


## 4 CODESYS V3 Function Blocks

This section will explain how to use certain CODESYS V3 Function Blocks to do simple operation on the drives like the power on/off function or simple movements as well as reading/writing SDOs into the drives. Function Blocks in CODESYS V3 can be used in different type of programming language, but , depending on witch type of programming is used they have different format rules. Below there's an example of the MC\_Power Function Block used both on ST (Structured Text) and on CFC (Continuous Function Chart).

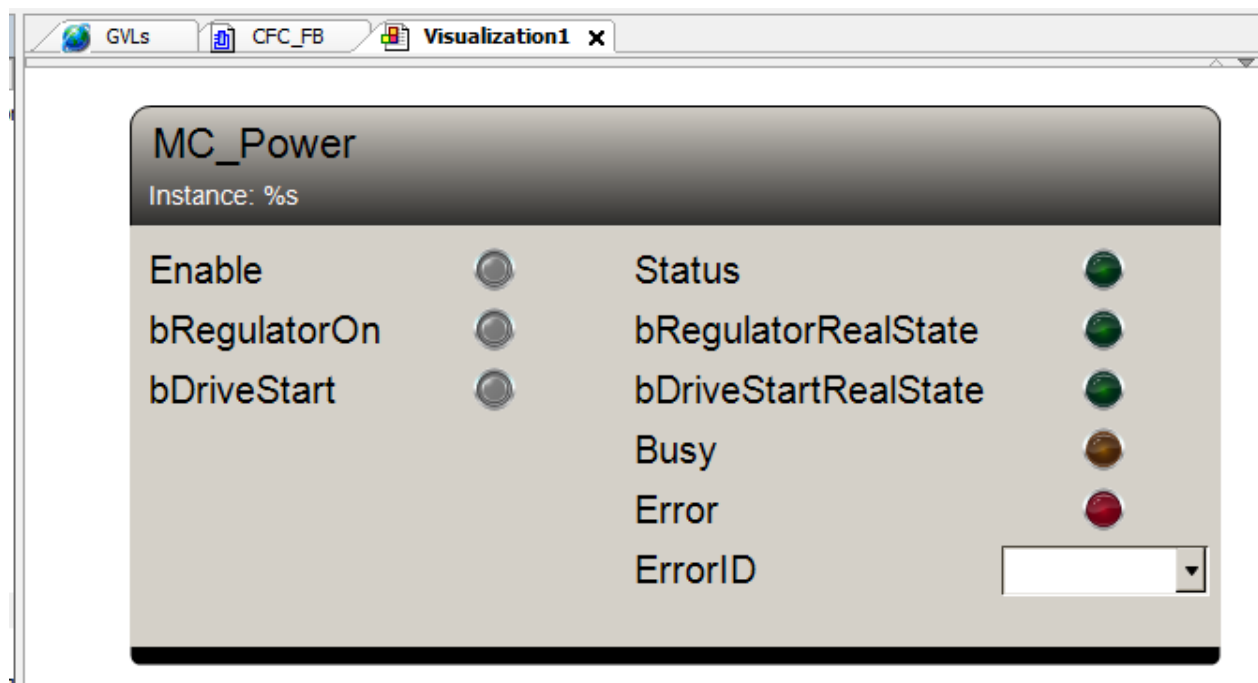


It's once again recommended to use a global variable list for the inputs and outputs of the CFC Function Block. To use Function Blocks in a really simple way it's possible to visualize them into the visualization CODESYS V3 environment (to add a Visualization Task right click on "Application" -> "Add Object" -> "Visualization"). In the visualization page search for "Visualization Toolbox" in the right-lower part of the screen and write into the search folder the name of the Function Block and drag it into the visualization page, then select the Function Block from the program variables to link it to the visualization tool. Now the Function Block can be managed directly from the visualization task in CODESYS V3 (recommended to use the VISU\_NEW versions of them).



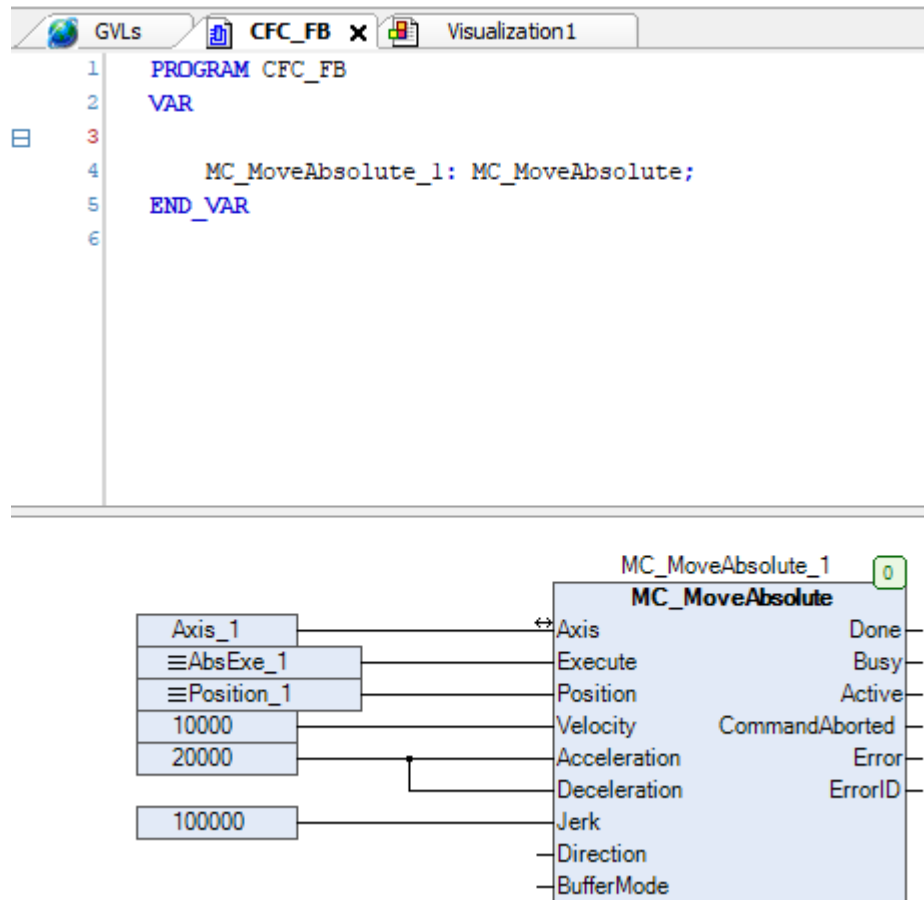
## 4.1 MC\_Power

This Function Block allow to control the power stage of the selected axis (chosen by the axis input into the Function Block). The "Enable" input must be "TRUE" for the Function Block to work, so it's recommended to assign directly in the variable list the value to "TRUE" and use the "bRegulatorOn" input as the on/off button (if no errors occur the axis will go in "standstill" status when "bRegulatorOn" is "TRUE" and in "power\_off" status when is "FALSE"). The input "bDriveStart" can also be assigned to "TRUE" directly in the variable list for an easy use of the Function Block, as here quickstop mechanisms are not considered. Various outputs can be assigned to variables to monitor the execution of the Function Block. The "status" output is set to "TRUE" when the axis is ready to move, "bReagulatorRealState" is set to "TRUE" when the power stage is switched on, "bDriveStartRealState" is set to "TRUE" when the drive is not blocked by a quickstop mechanism, "Busy" is "TRUE" when the function block is still in execution and not finished yet, "Error" is flagged as "TRUE" when an error occurs and "ErrorID" will show the errors identifications if there are any.



## 4.2 MC\_MoveAbsolute

This Function Block is used to set an absolute position into the axis and execute the movement to that target position. The Function Block allows to monitor the movement and displays if some errors occur.

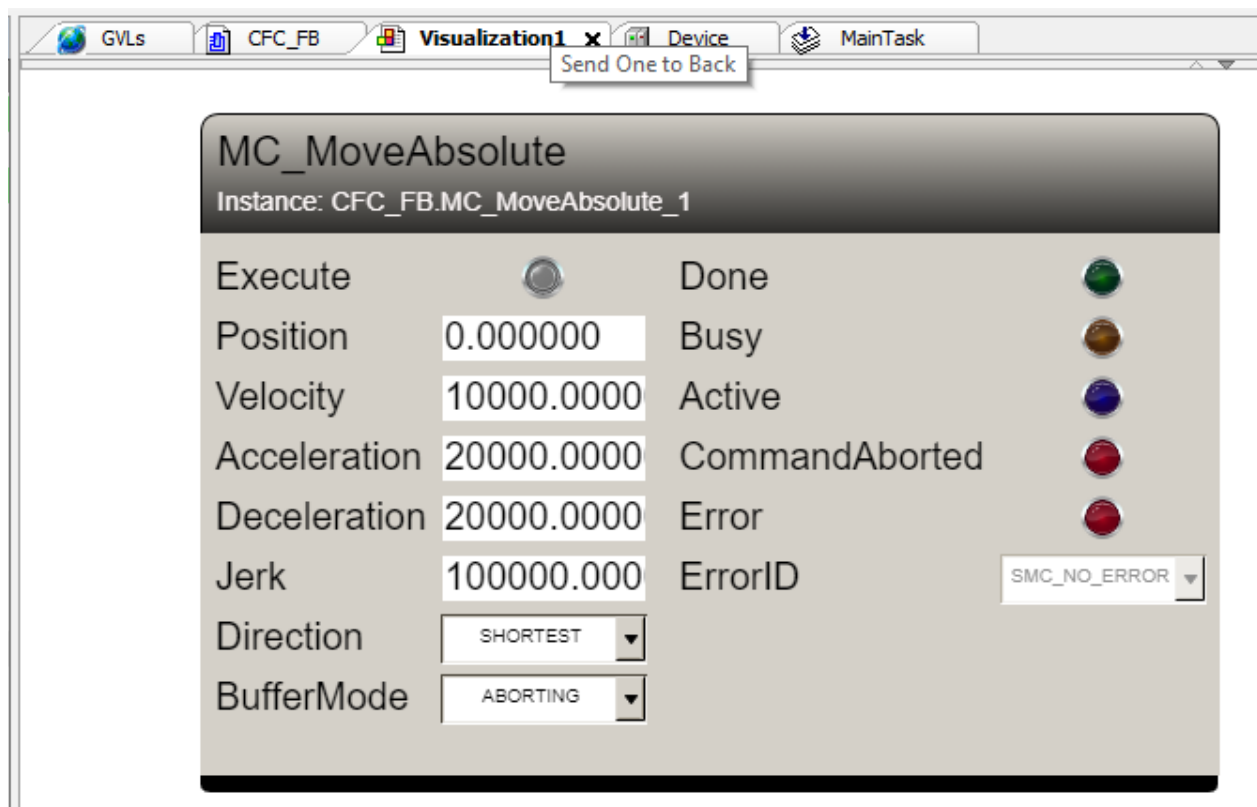


The input "Axis" is used to refer to an axis into the project, the "Execute" input must be flagged to "TRUE" to start the movement of the axis, the "Position" input refer to the target position of the axis (LREAL), "Velocity", "Acceleration" and "Deceleration" inputs all refer to the movement parameters (LREAL), the "Jerk" input refers to the rate of the acceleration changing with respect of time during the set trajectory execution (these parameters are calculated in counts into the Function Block), "Direction" and "BufferMode" input can be assigned in the global variable list (a specific type of variable like MC\_Direction and MC\_BUFFER\_MODE must be assigned if using variables for them)

or not assigned and selected from the dropdown menu in the visualization Function Block refer.

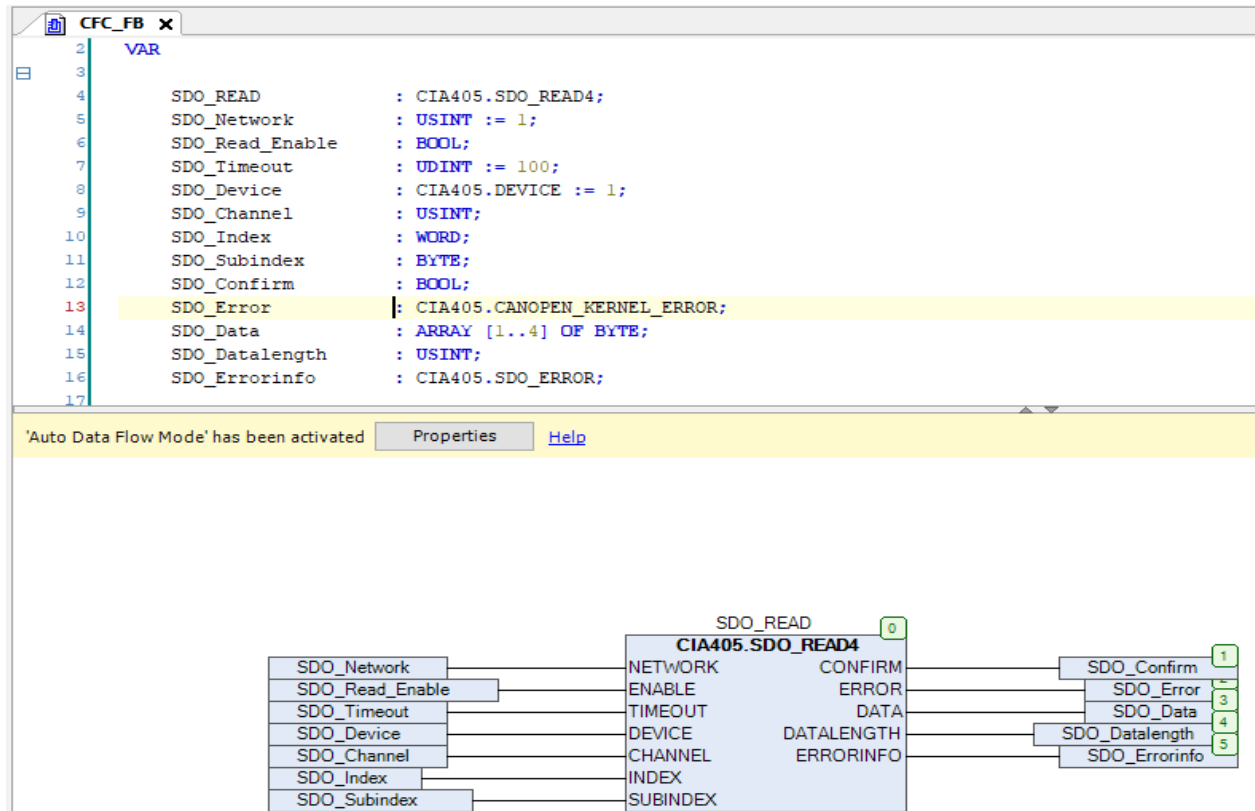
The output "done" is set to "TRUE" when the movement is completed without any errors and the target position is reached by the axis, the "Busy" output is flagged "TRUE" when the Function Block is still in execution until the movement is completed. The "active" output is "TRUE" when the command is executed, "CommandAborted" is flagged "TRUE" when the movement cannot be executed completely because of another command replaced the first one. "Error" is set to "TRUE" if any errors occur, "ErrorID" will show the error identifier.

Assigning a mutable variable in the "position" input and making the "Execute" command to toggle from "TRUE" to "FALSE" continuously will generate a series of movement to the assigned target positions.



## 4.3 SDO\_READ4

This Function Block is used to read specific objects into the objects dictionary of a device. Up to 4 Bytes can be read by this Function Block, the data are then written into an Array of Byte



The "Network" input (USINT) refer to the CAN network number the Function Block will operate on (NOT THE SAME NETWORK NUMBER FOUND IN CANBUS CONFIGURATION), Network number can be obtained by NetID + 1 (es. CAN0 => Network = 1).

"Enable" input when flagged "TRUE" will activate the Function Block on a rising edge.

"Timeout" refers to the actual timeout in ms (UDINT), "Device" input is the NodeID of the destination device (=0 for local device CANopen Manager, needs a CIA405.DEVICE variable as shown above). The "Channel" input (USINT) is the SDO channel used by the Function Block (=0 will choose the first available channel, =1..n for a specific numbered channel). "Index" (WORD) and "Subindex" (BYTE) refer to the actual index and subindex of the object the Function Block is reading.



"Confirm" output is flagged "TRUE" when the execution of the Function Block is completed without errors. "Error" represent the error code if any occurs (must use a CIA405.CANOPEN\_KERNEL\_ERROR variable as shown in the picture above).

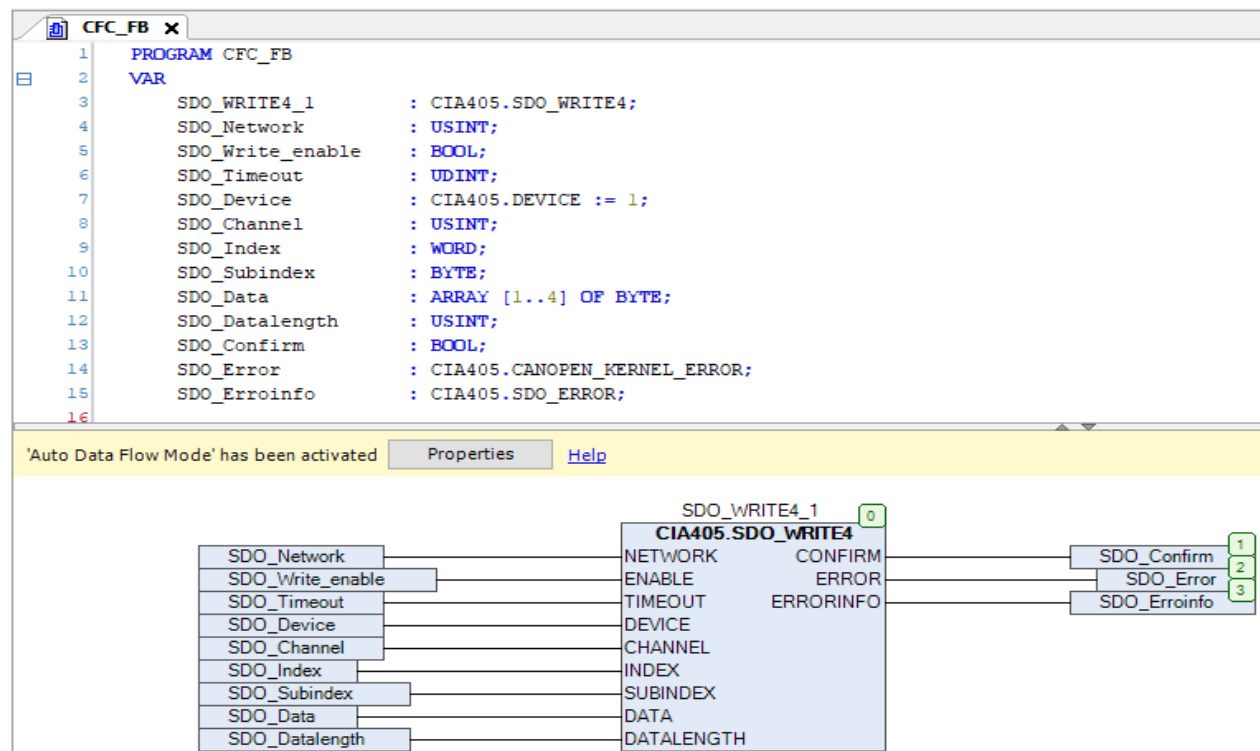
"Data" output is an array of 4 Bytes where the data of the chosen object are written to, in CANopen Byte order (Little Endian). "Datalength" refers to the actual amount of data in Bytes written into the "Data" output (USINT).

"Errorinfo" is a SDO\_ERROR type variable that contains the abort code in little endian in case of "Error" is in CANOPEN\_KERNEL\_ERROR.CANOPEN\_KERNEL\_OTHER\_ERROR state.

Inputs can be assigned to constant values to create a Function Block that reads a specific object, or writable variables can be used as inputs to create a Function Block that read any object depending on the values written in the inputs variables.

## 4.4 SDO\_WRITE4

This Function Block is used to write a specific object into the object dictionary of the selected device. Up to 4 Bytes can be written using this Function Block. The data to write in is an array of 4 Bytes.



The inputs of this Function Block are very similar to the ones used into the SDO\_READ4.

"Network" input (USINT) refer to the CAN network number the Function Block will operate on (NOT THE SAME NETWORK NUMBER FOUND IN CANBUS CONFIGURATION), Network number can be obtained by NetID + 1 (es. CAN0 => Network =1).

"Enable" input when flagged "TRUE" will activate the Function Block on a rising edge.

"Timeout" refers to the actual timeout in ms (UDINT), "Device" input is the NodeID of the destination device (=0 for local device CANopen Manager, needs a CIA405.DEVICE variable as shown above). The "Channel" input (USINT) is the SDO channel used by the Function Block (=0 will choose the first available channel, =1..n for a specific numbered channel). "Index" (WORD) and "Subindex" (BYTE) inputs refer to the actual index and subindex of the object written by the Function Block. "Data" input is an array of 4 Bytes containing the data that should be written into the selected object. "Datalength" (USINT) is the number of bytes that should be written into the selected object.

"Confirm" output is flagged "TRUE" when the execution of the Function Block is completed without errors, "Error" output must be assigned to a CIA405.CANOPEN\_KERNEL\_ERROR variable and will display the error code is any occurs, "Errorinfo" " is a SDO\_ERROR type variable that contains the abort code in little endian in case of "Error" is in CANOPEN\_KERNEL\_ERROR.CANOPEN\_KERNEL\_OTHER\_ERROR state.

## Copyright:

NiLAB GmbH  
Hans-Sachs-Straße 16  
A-9020 Klagenfurt am Wörthersee – Österreich

+43 720 513 258 (VoIP)

[www.nilab.at](http://www.nilab.at)  
[katharina.pirker@nilab.at](mailto:katharina.pirker@nilab.at)

1. Auflage  
April 2023