Machine Expert Application Note

Product Family
Integrated Drive
NLi080QX with CANBUS
NLi120QX with CANBUS

Index

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 EcoStruxure Machine Expert V2.1 are, what the programmer should do with them and expect from them, as well as describing how to configure the PacDrive.

For the following images a LMC101C has been used as the CANopen Master, but the description apply for all Schneider Electrics PacDrives in EcoStruxure Machine Expert V2.1.

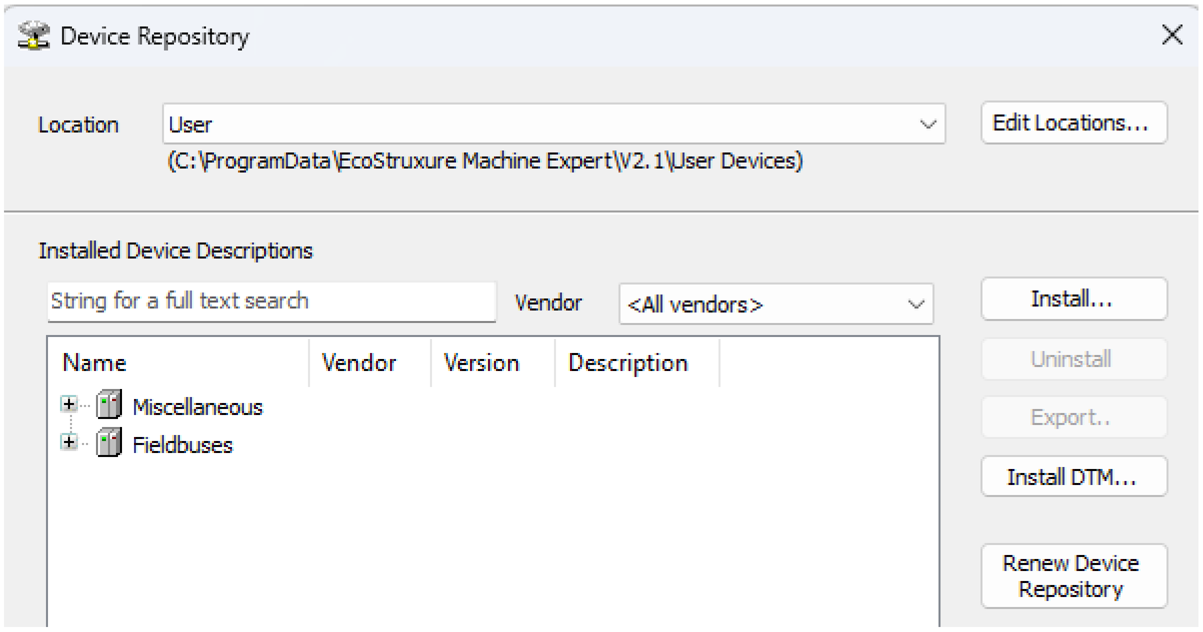
2 Components/Software used

Type/Name	Version Software/Firmware
EcoStruxure Machine Expert	V2.1x64
LMC101C	V01.72.19.01

3 EcoStruxure Machine Expert - 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   EcoStruxure Machine Expert V2.1 environment.

If the PacDrive is not present in the device repository and .eds file must be imported into EcoStruxure Machine Expert V2.1. Search for the "Tools" serction in the upper menu and select "Device Repository". Click on "Install" and search the computer resources for the .eds file to import in EcoStruxure Machine Expert V2.1.


Device Repository

Location User
(C:\ProgramData\EcoStruxure Machine Expert\V2.1\User Devices)

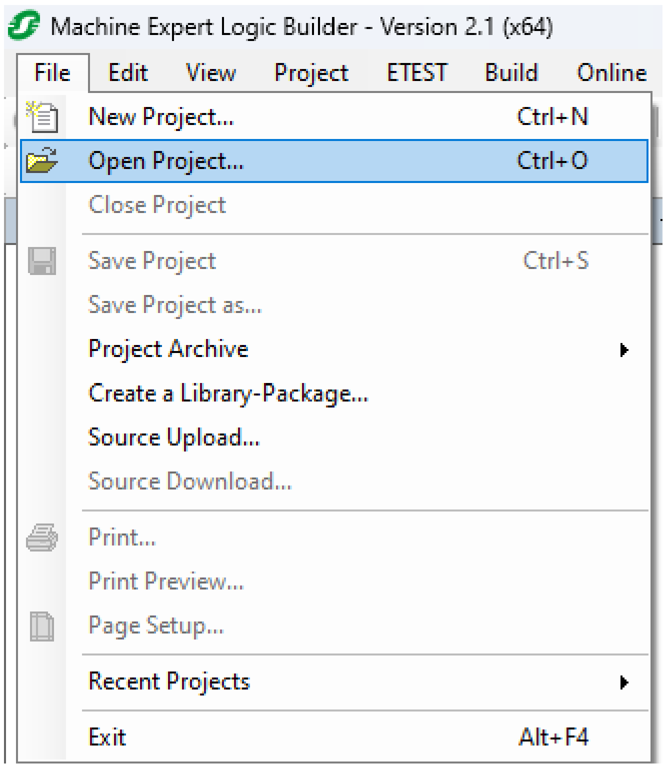
Installed Device Descriptions

String for a full text search   Vendor <All vendors>

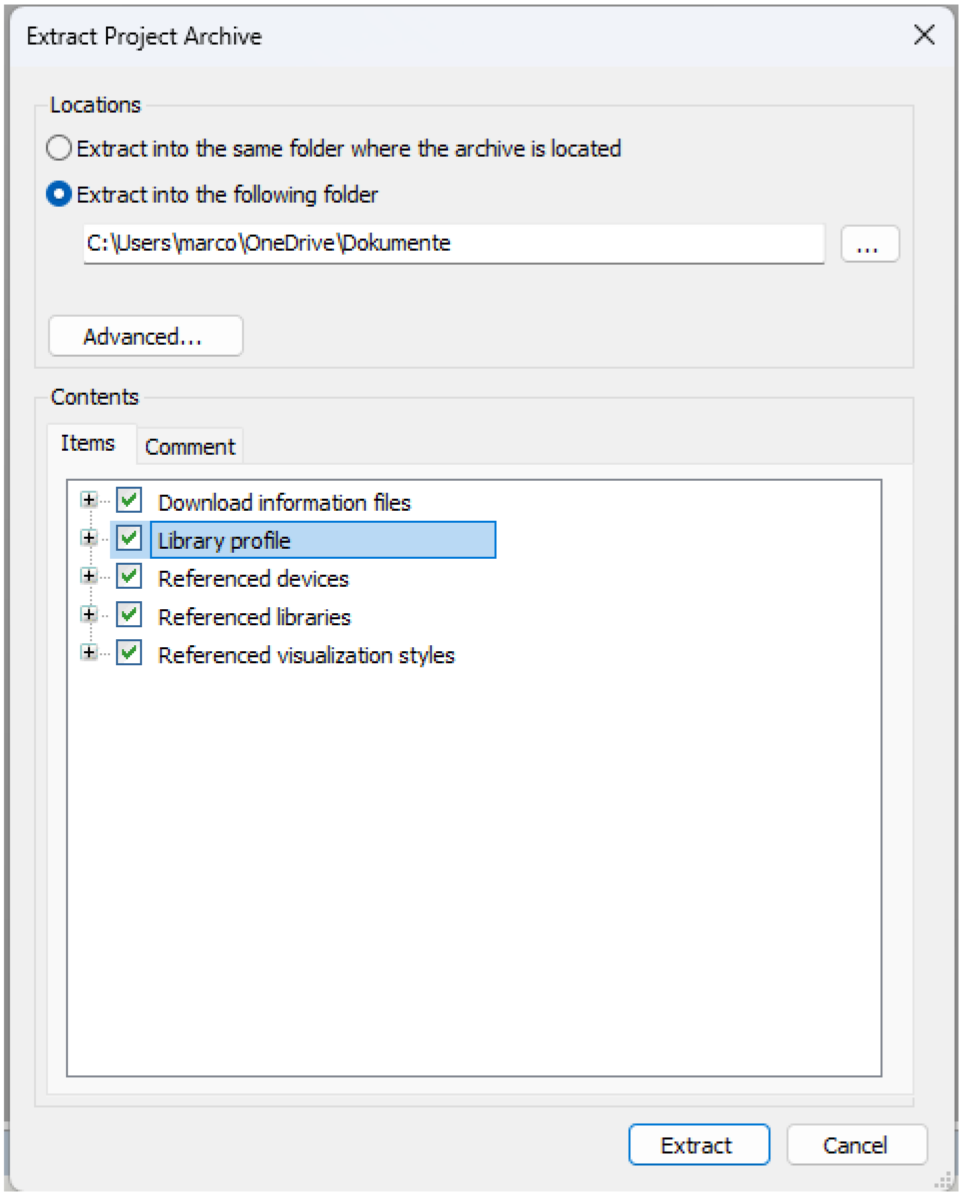
Name	Vendor	Version	Description
Miscellaneous			
Fieldbuses			

3.1 Open the EcoStruxure Machine Expert V2.1 example project with CANbus

Open the program EcoStruxure Machine Expert V2.1 and tap on "File", then select "Open Project" and search for the provided example file.



If another version of EcoStruxure Machine Expert V2.1 is used, the program will automatically ask if the project want to be updated and the missing libraries installed, tap "Ok" because this components are needed for the project to work. A list of the missing libraries will appear, select all of them and tap on "Extract" (choose a fitting destination for the extracted files).


Extract Project Archive

Locations

( ) Extract into the same folder where the archive is located

[✓ Extract into the following folder]

C:\Users\marco\OneDrive\Dokumente

Contents

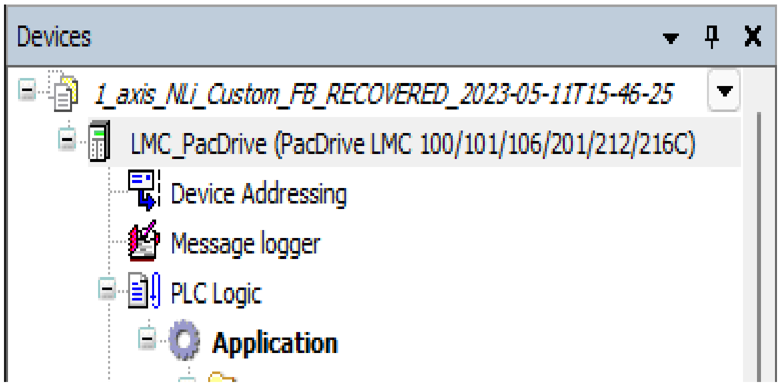
Items | Comment

- ✓ Download information files
- ✓ Library profile
- ✓ Referenced devices
- ✓ Referenced libraries
- ✓ Referenced visualization styles

After that the program will ask to update the project in the current version of EcoStruxure Machine Expert if the file is imported from a different version of the program. Tap on “Update” nad the project will open.

3.2 Configuration of the CANopen devices

The CANopen devices are located in the “Devices” section in the left part of the page, here select the LMC_PacDrive from the top of the section and tap on “Configuration Setting” in the LMC_PacDrive menu. Select the device to connect from the list of available devices.


Devices

1_axis_NLi_Custom_FB_RECOVERED_2023-05-11T15-46-25

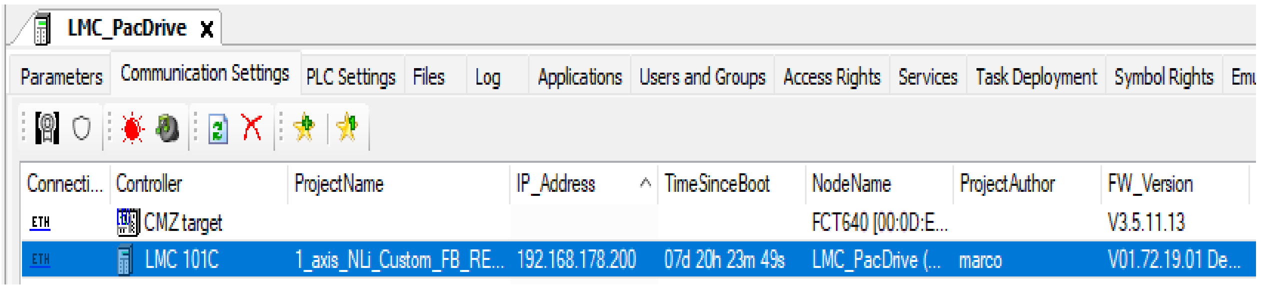
LMC_PacDrive (PacDrive LMC 100/101/106/201/212/216C)

Device Addressing

Message logger

PLC Logic

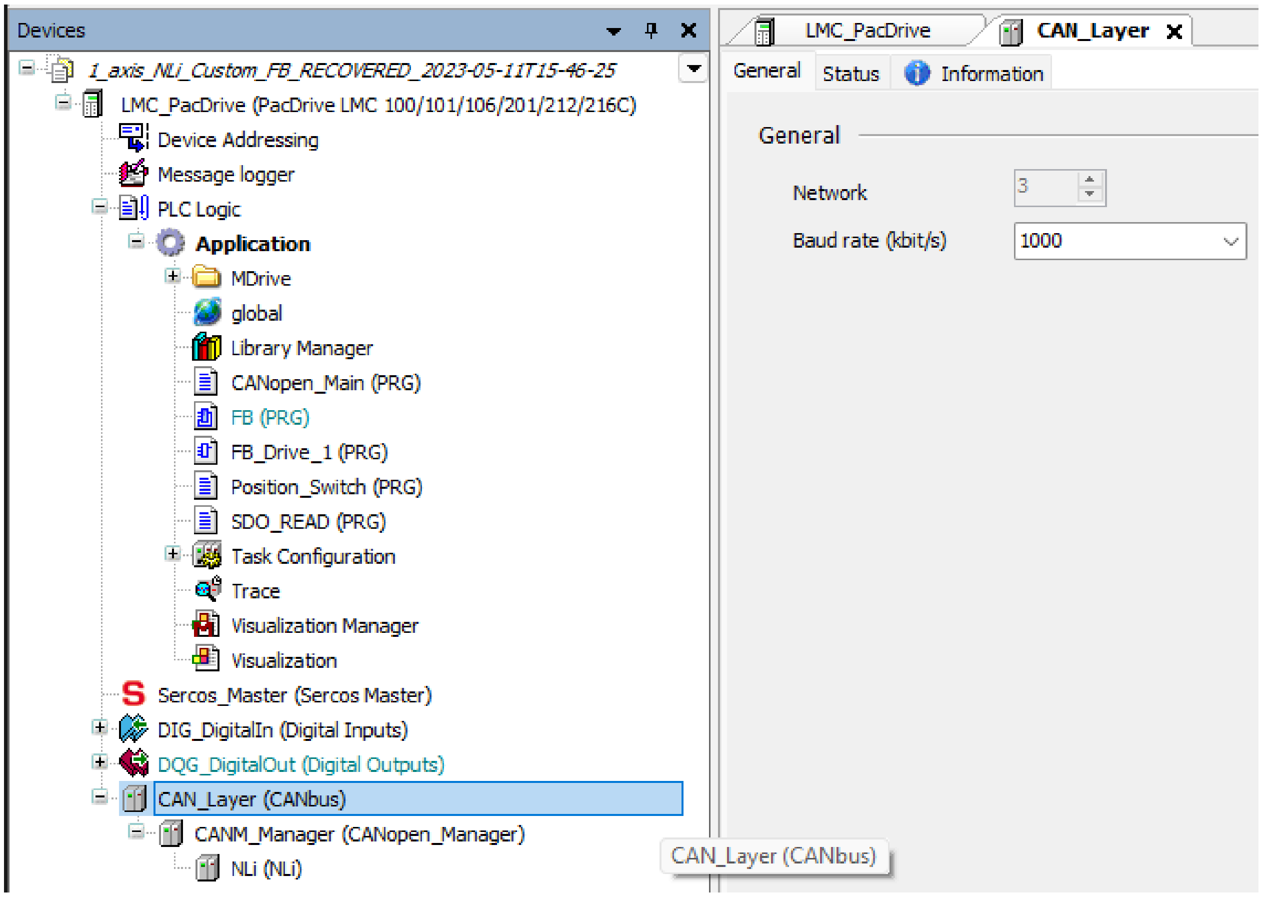
Application


LMC_PacDrive

Parameters | Communication Settings | PLC Settings | Files | Log | Applications | Users and Groups | Access Rights | Services | Task Deployment | Symbol Rights | Emu

Connecti...	Controller	ProjectName	IP_Address	TimeSinceBoot	NodeName	ProjectAuthor	FW_Version
ETH	CMZ target				FCT640 [00:0D:E...		V3.5.11.13
ETH	LMC 101C	1_axis_NLi_Custom_FB_RE...	192.168.178.200	07d 20h 23m 49s	LMC_PacDrive (...	marco	V01.72.19.01 De...

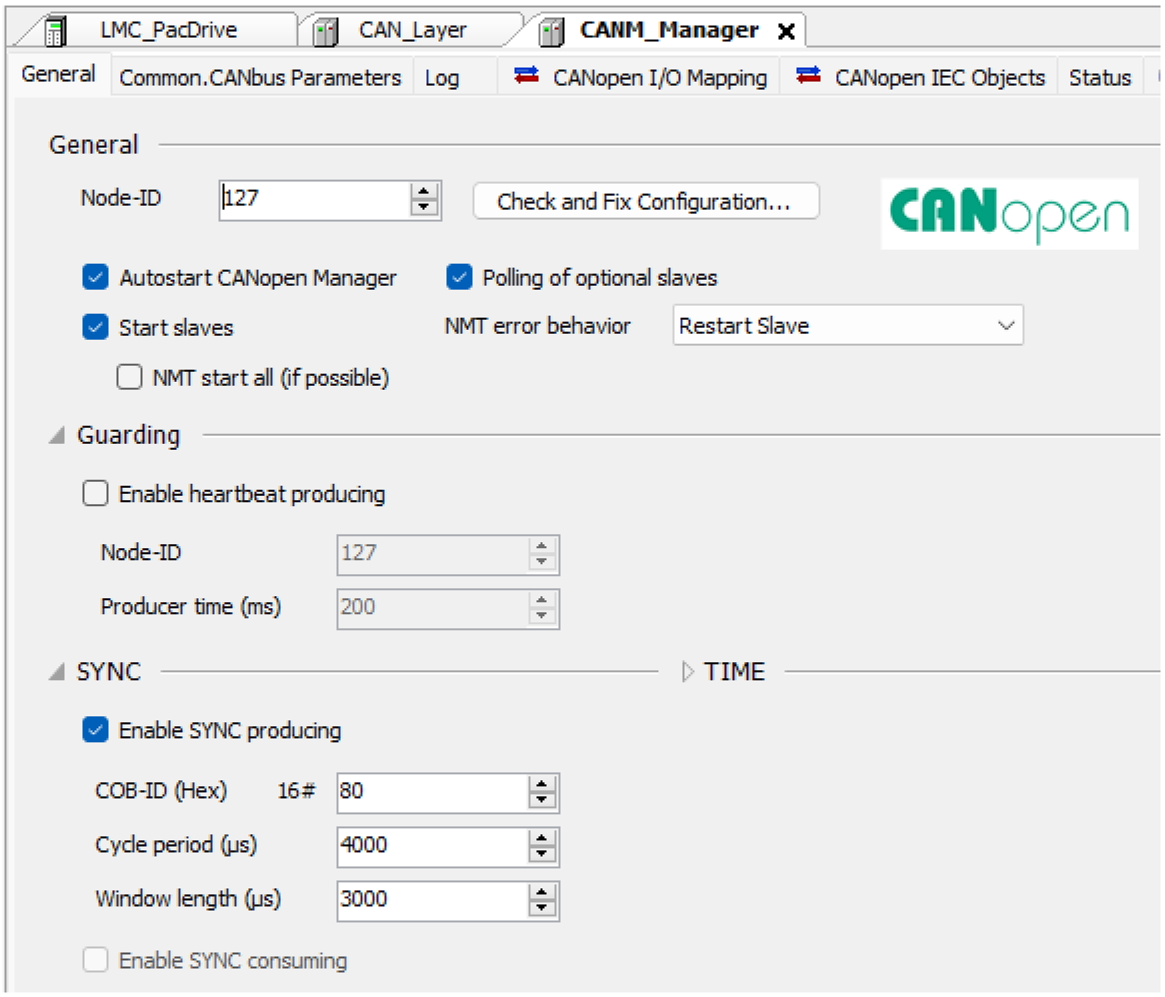
After this search in the bottom part of the devices tree the “CAN_Layer (CANbus)” device and open it. Go to the “General” section and change the baud rate to 1000 kbit/s if it’s setted to another value.



Network: 3

Baud rate (kbit/s): 1000

Move then to the “CANM_Manager (CANopen_Manager)” device just under the previous one in the devices tree and open it, in the “General” section tick the box “Enable SYNC producing” and use 80 as “COB-ID” , 4000 as “Cycle” period and 3000 as “Window length”.


LMC_PacDrive | CAN_Layer | CANM_Manager

General | Common.CANbus Parameters | Log | CANopen I/O Mapping | CANopen IEC Objects | Status

General

Node-ID: 127

Check and Fix Configuration...

- [x] Autostart CANopen Manager
- [x] Polling of optional slaves
- [x] Start slaves
- [ ] NMT start all (if possible)

NMT error behavior: Restart Slave

Guarding

- [ ] Enable heartbeat producing

Node-ID: 127

Producer time (ms): 200

SYNC

TIME

- [x] Enable SYNC producing

COB-ID (Hex) 16#: 80

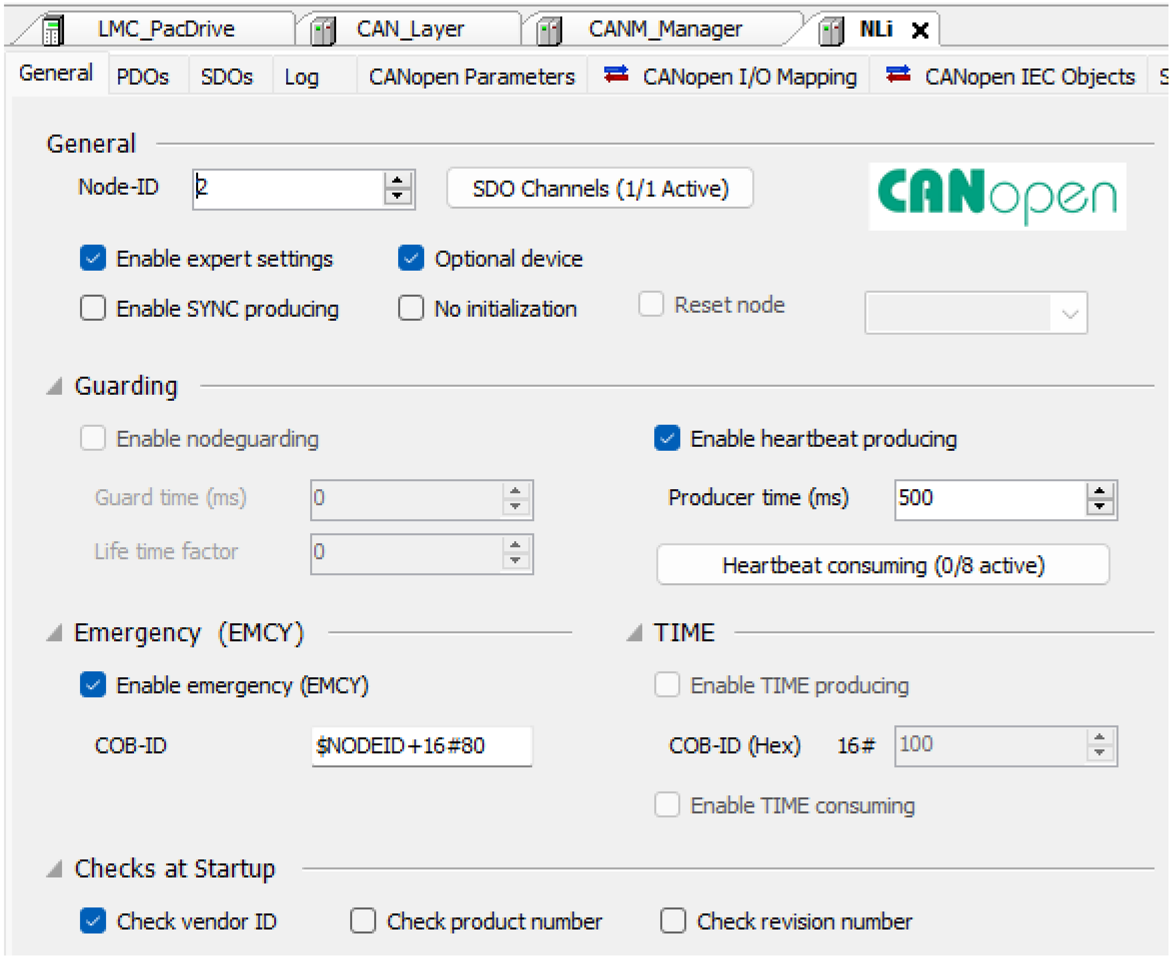
Cycle period (µs): 4000

Window length (µs): 3000

- [ ] Enable SYNC consuming

Open the n the “CANopen I/O Mapping” window and select “TASK_SR_Main” as “Bus cycle task”.

Once the “CANM_Manager” is configured tap on the “Nli” device in the devices tree and open the “General” section. First select the “Node-ID” of the device in use. Activate “Enable expert settings” and tick the box “Enable heartbeat producing” in the “Guarding” drop down menu and set the “Producer time” to 500 and 0/8 “Heartbeat consuming”.


LMC_PacDrive | CAN_Layer | CANM_Manager | NLi

General | PDOs | SDOs | Log | CANopen Parameters | CANopen I/O Mapping | CANopen IEC Objects

General

Node-ID: 2

SDO Channels (1/1 Active)

- [✓] Enable expert settings
- [✓] Optional device
- [ ] Enable SYNC producing
- [ ] No initialization
- [ ] Reset node

Guarding

- [ ] Enable nodeguarding
- Guard time (ms): 0
- Life time factor: 0
- [✓] Enable heartbeat producing
- Producer time (ms): 500

Heartbeat consuming (0/8 active)

Emergency (EMCY)

- [✓] Enable emergency (EMCY)
- COB-ID: $NODEID+16#80

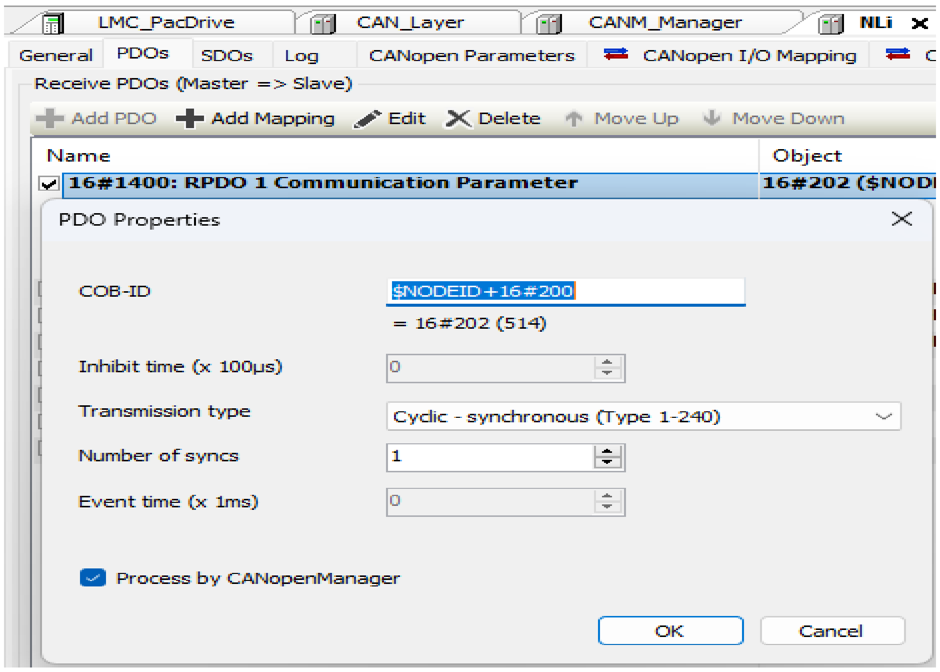
TIME

- [ ] Enable TIME producing
- COB-ID (Hex) 16#: 100
- [ ] Enable TIME consuming

Checks at Startup

- [✓] Check vendor ID
- [ ] Check product number
- [ ] Check revision number

Then move to the “PDOs” section of the Nli device and activate the first RPDO (16#14000), right click on it (double click also works) and select “Edit”. In “PDO Properties” select “Cyclic – synchronous (Type 1-240) as “Transmission type”.



Receive PDOs (Master => Slave)

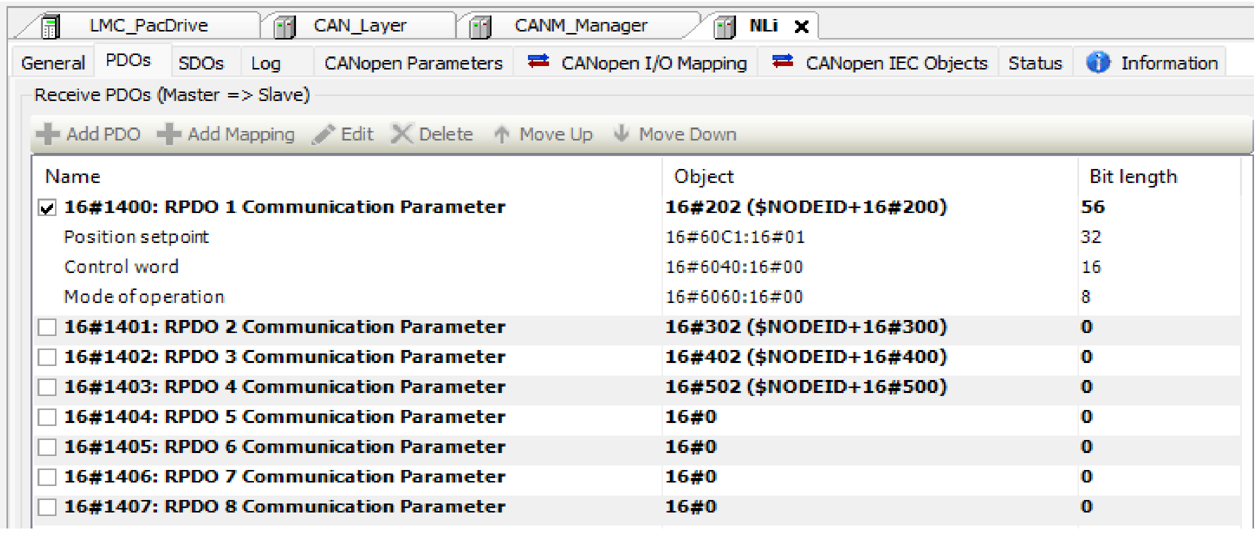
Name	Object
16#1400: RPDO 1 Communication Parameter	16#202 ($NODI

PDO Properties

COB-ID	$NODEID+16#200
	= 16#202 (514)
Inhibit time (x 100µs)	0
Transmission type	Cyclic - synchronous (Type 1-240)
Number of syncs	1
Event time (x 1ms)	0

✓ Process by CANopenManager

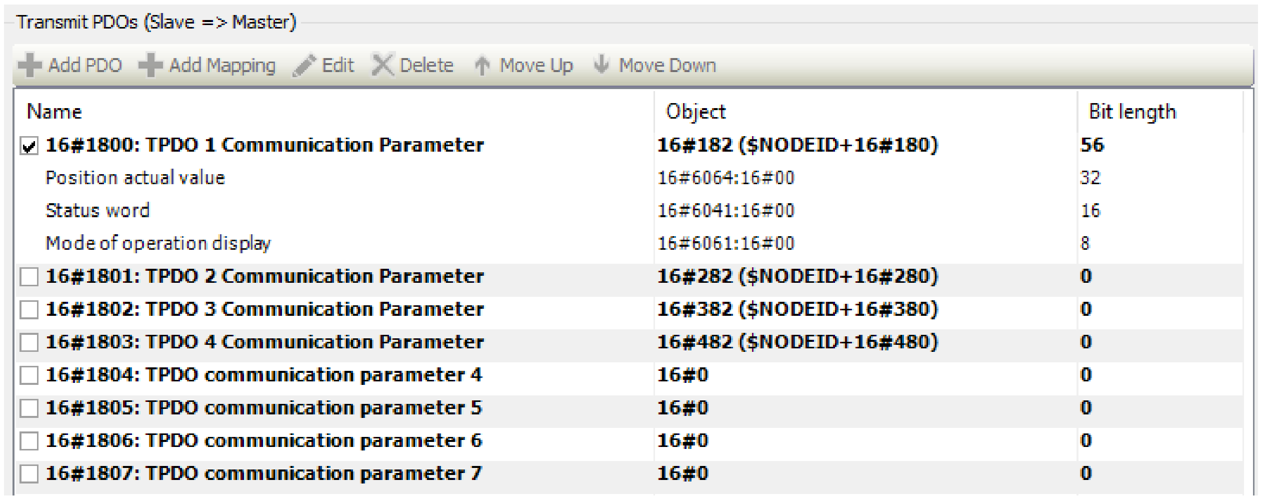
Tap on “Ok” to confirm the settings. Now select the RPDO and tap on “Add Mapping”, the list of all mapped parameters from the .eds file will open. Select “Position setpoint” (under the interpolation data record drop down menu), “Control word” and “Mode of operation”. Beware that the parameters must be setted in this specific order (32,16 and 8 bits) for the transmission to work.



Receive PDOs (Master => Slave)

Name	Object	Bit length
✓ 16#1400: RPDO 1 Communication Parameter	16#202 ($NODEID+16#200)	56
Position setpoint	16#60C1:16#01	32
Control word	16#6040:16#00	16
Mode of operation	16#6060:16#00	8
16#1401: RPDO 2 Communication Parameter	16#302 ($NODEID+16#300)	0
16#1402: RPDO 3 Communication Parameter	16#402 ($NODEID+16#400)	0
16#1403: RPDO 4 Communication Parameter	16#502 ($NODEID+16#500)	0
16#1404: RPDO 5 Communication Parameter	16#0	0
16#1405: RPDO 6 Communication Parameter	16#0	0
16#1406: RPDO 7 Communication Parameter	16#0	0
16#1407: RPDO 8 Communication Parameter	16#0	0

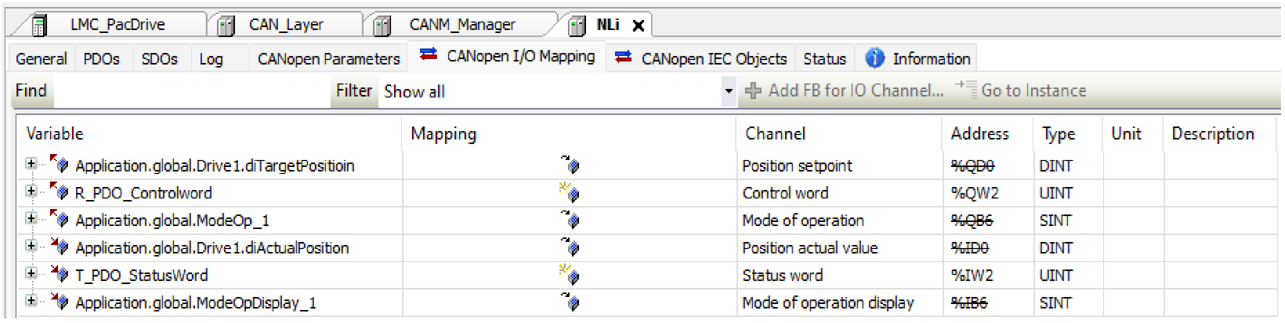
Now the same thing shall be done with the “TPDO” section. The TPDOs don’t require any synchronism in the transmission so as the “Transmission type” the default settings work fine. The parameters “Position actual value”, “Status word” and “Mode of operation display” must be mapped in the same way as the picture below.


Transmit PDOs (Slave => Master)

Add PDO | Add Mapping | Edit | Delete | Move Up | Move Down

Name	Object	Bit length
✓ **16#1800: TPDO 1 Communication Parameter**	**16#182 ($NODEID+16#180)**	**56**
Position actual value	16#6064:16#00	32
Status word	16#6041:16#00	16
Mode of operation display	16#6061:16#00	8
☐ **16#1801: TPDO 2 Communication Parameter**	**16#282 ($NODEID+16#280)**	**0**
☐ **16#1802: TPDO 3 Communication Parameter**	**16#382 ($NODEID+16#380)**	**0**
☐ **16#1803: TPDO 4 Communication Parameter**	**16#482 ($NODEID+16#480)**	**0**
☐ **16#1804: TPDO communication parameter 4**	**16#0**	**0**
☐ **16#1805: TPDO communication parameter 5**	**16#0**	**0**
☐ **16#1806: TPDO communication parameter 6**	**16#0**	**0**
☐ **16#1807: TPDO communication parameter 7**	**16#0**	**0**

After the RPDO and the TPDO are configured correctly, search for the “CANopen I/O Mapping” window and map the variables required for the parameters chosen in the PDO section.


LMC_PacDrive | CAN_Layer | CANM_Manager | NLi

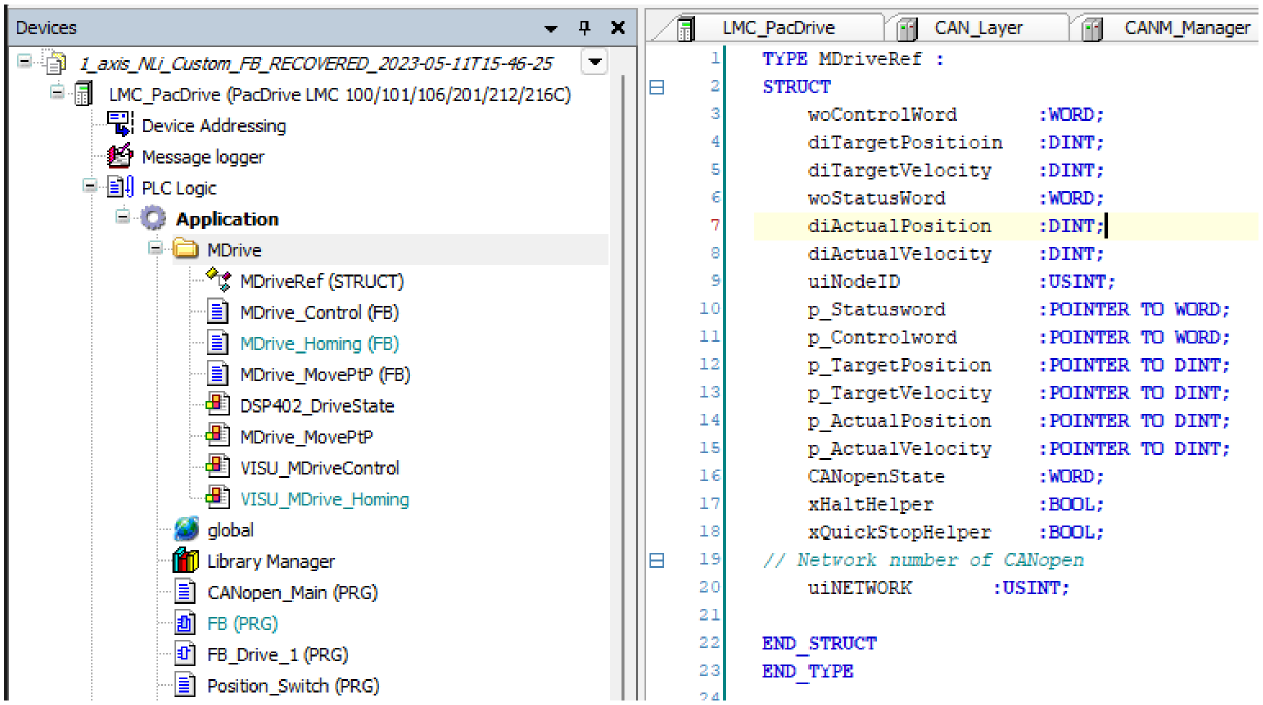
General | PDOs | SDOs | Log | CANopen Parameters | CANopen I/O Mapping | CANopen IEC Objects | Status | Information

Find | Filter Show all | Add FB for IO Channel... | Go to Instance

Variable	Mapping	Channel	Address	Type	Unit	Description
Application.global.Drive1.diTargetPositioin		Position setpoint	~~%QD0~~	DINT		
R_PDO_Controlword		Control word	%QW2	UINT		
Application.global.ModeOp_1		Mode of operation	~~%QB6~~	SINT		
Application.global.Drive1.diActualPosition		Position actual value	~~%ID0~~	DINT		
T_PDO_StatusWord		Status word	%IW2	UINT		
Application.global.ModeOpDisplay_1		Mode of operation display	~~%IB6~~	SINT		

3.3 Creating an Axis reference for CANopen Network

In this section is described how to create an axis reference in the EcoStruxure Machine Expert V2.1 environment. Right click on “Application” in the devices tree and add a new folder. Tap on the new folder and select “Add Object” → “DUT” (Type “Structure”). In this structure the axis parameters will be defined. To work with the Nli example project the variables must be the same ones from the picture below for the program to work without any change. (The structure is already defined in the example project).



```
TYPE MDriveRef :
STRUCT
    woControlWord          :WORD;
    diTargetPositioin      :DINT;
    diTargetVelocity       :DINT;
    woStatusWord           :WORD;
    diActualPosition       :DINT;
    diActualVelocity       :DINT;
    uiNodeID               :USINT;
    p_Statusword           :POINTER TO WORD;
    p_Controlword          :POINTER TO WORD;
    p_TargetPosition       :POINTER TO DINT;
    p_TargetVelocity       :POINTER TO DINT;
    p_ActualPosition       :POINTER TO DINT;
    p_ActualVelocity       :POINTER TO DINT;
    CANopenState           :WORD;
    xHaltHelper            :BOOL;
    xQuickStopHelper       :BOOL;
// Network number of CANopen
    uiNETWORK        :USINT;

END_STRUCT
END_TYPE
```

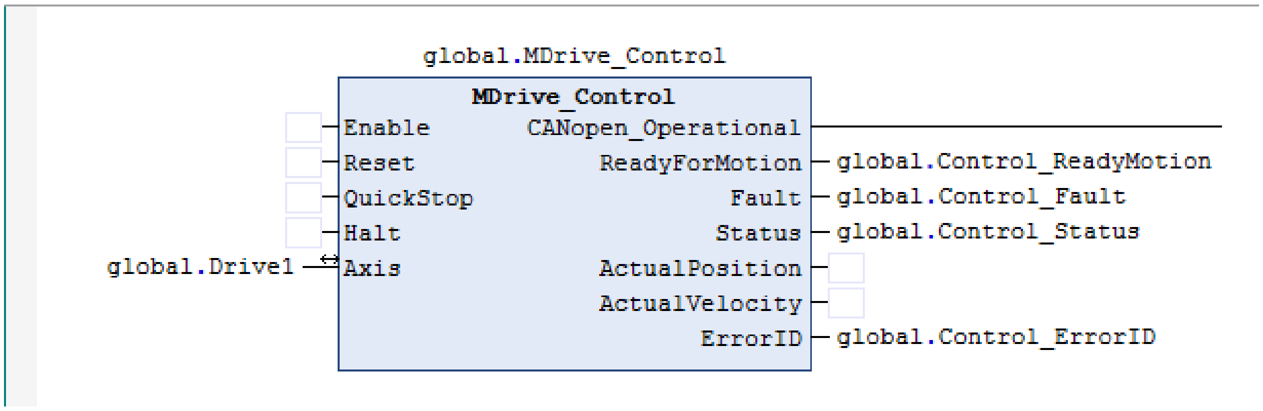
This structure will work as the axis reference for the Function Blocks of the project, so every reference needed for the input of the functions must be defined here. For a simple use is suggested to add a global variable that directly recall to the structure (ex. Drive1 : MdriveRef;)

4 EcoStruxure Machine Expert V2.1 Function Blocks

In the EcoStruxure Machine Expert V2.1 the classic MC_Function Blocks are not supported, so these functions must be instantiated in a different way from standard Codesys. The example project already contains an example of how the Enable/Disable axis and MovePtP Function Blocks will work and how they are built.

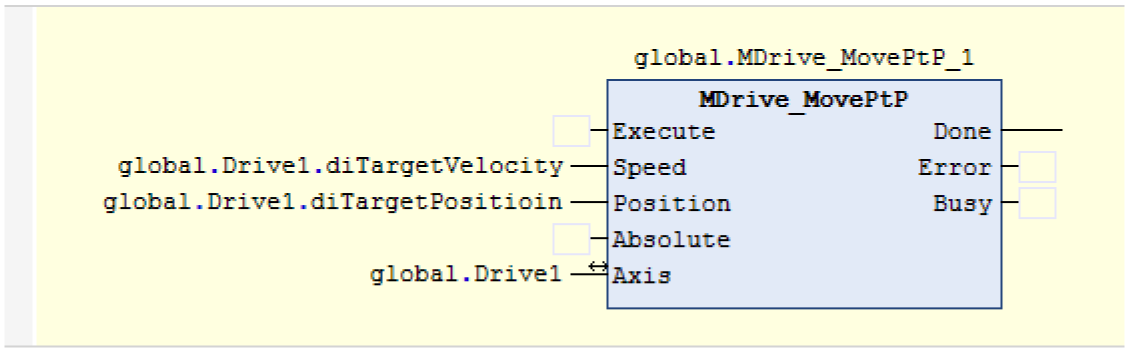
4.1 MDrive_Control

This function block works in the same way of the MC_Power function block used in Codesys, but with a different logic. This block will read the CANopen State via the CAN_GET_STATE function, if the result of the reading is the "Operational" state and no errors occurred the axis will be enabled, in any different case an error flag will be activated.



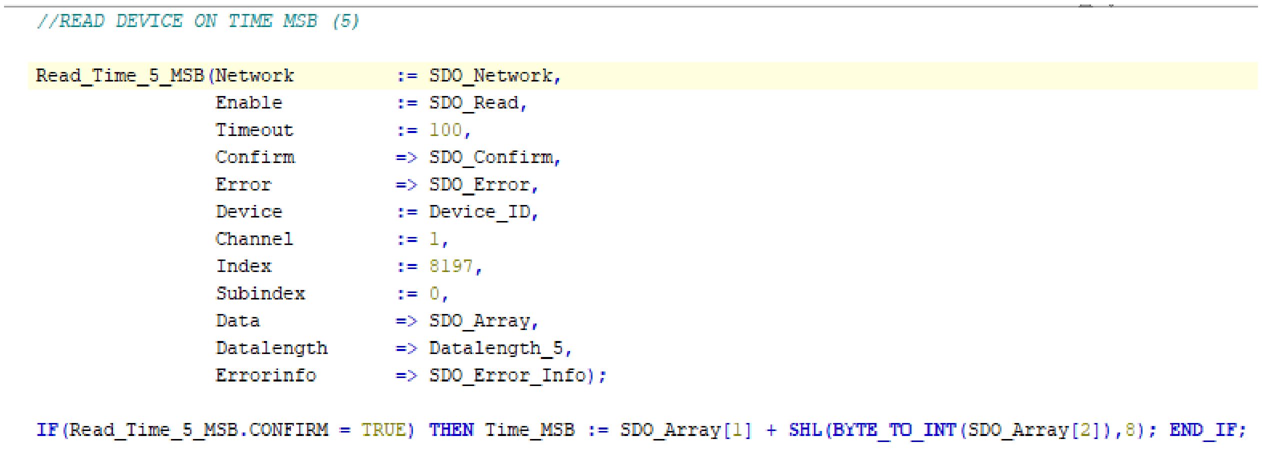
4.2 Mdrive_MovePtP

This function block work similar to the MC_MoveAbsolute function from the standard Codesys. This block allows to chose between absolute and relative movement by the setting of a flag input. In the provided example the function block is set to work only in the absolute mode. The logic behind this block is to set some of the bits of the Control Word and read the consequential changes in the Status Word bits to make the axis move. The axis must be enabled first for this function block to work. In the example project the block receive a different target position once the first one is reached to create a 2 point moving loop. The block requires a set speed, acceleration and deceleration in order to create the trajectory of the movement.



4.3 SDO_READ4

This is the exact same function block of the standard Codesys, it allows to read parameters up to 4 Bytes from the object dictionary of the selected device. In the provided example there's a visualization interface from where is possible to read multiple motor parameters with the button "SDO READ". This function block requires an enable button, the Network number (:=4 in the example project), the device ID, the number of the channel used (:=1 in the example), the index and subindex and the datalength. The parameter value is stored into an array of Byte [1..4] after the reading is done.



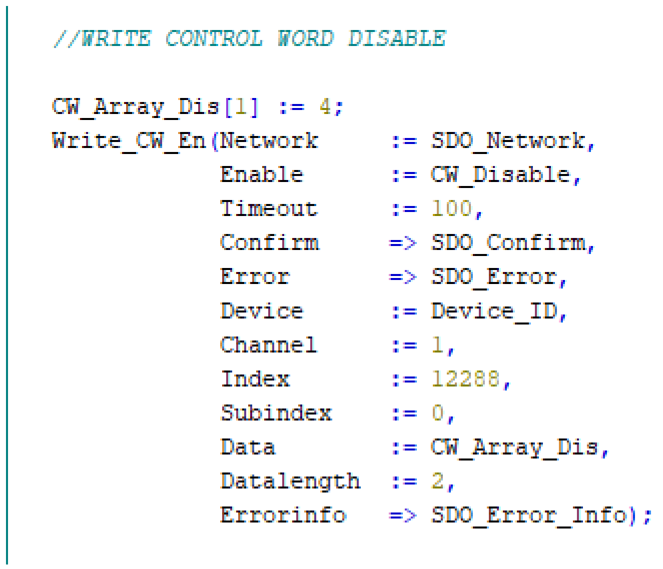
```
//READ DEVICE ON TIME MSB (5)

Read_Time_5_MSB(Network          := SDO_Network,
                Enable           := SDO_Read,
                Timeout          := 100,
                Confirm          => SDO_Confirm,
                Error            => SDO_Error,
                Device           := Device_ID,
                Channel          := 1,
                Index            := 8197,
                Subindex         := 0,
                Data             => SDO_Array,
                Datalength       => Datalength_5,
                Errorinfo        => SDO_Error_Info);

IF(Read_Time_5_MSB.CONFIRM = TRUE) THEN Time_MSB := SDO_Array[1] + SHL(BYTE_TO_INT(SDO_Array[2]),8); END_IF;
```

4.4 SDO_WRITE4

This is the exact same function block of the standard Codesys. It allows to write parameters up to 4 Bytes into the object dictionary of the selected device. Similar to SDO_READ4, this function block requires a boolean flag for enable, the Network number (:=4 in the example), the device ID, the number of the channel used (:=1 in the example), the index and subindex and the datalength. The array [1..4] of Bytes is the dta that the function is going to write in the selected index.

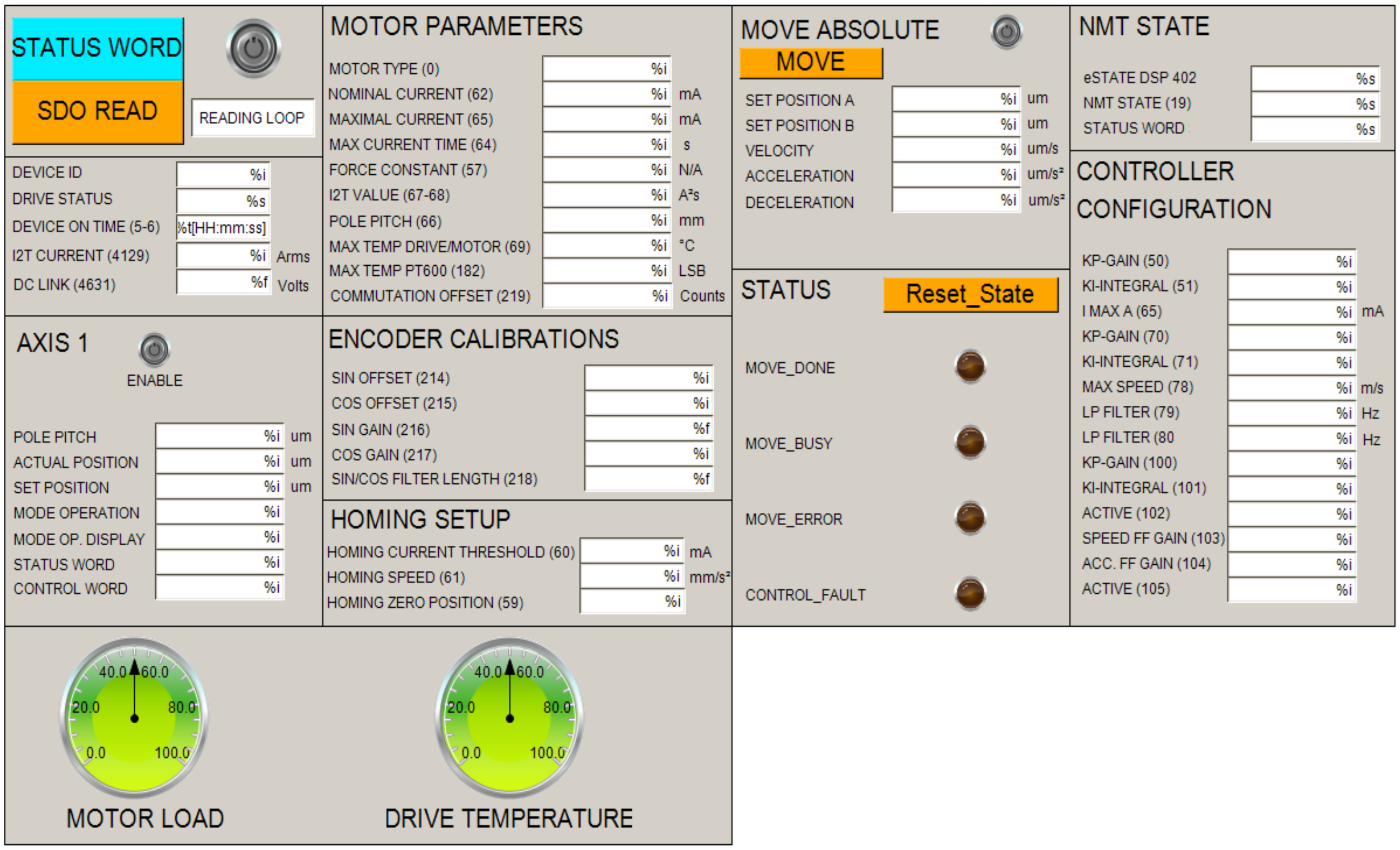


```
//WRITE CONTROL WORD DISABLE

CW_Array_Dis[1] := 4;
Write_CW_En(Network     := SDO_Network,
            Enable      := CW_Disable,
            Timeout     := 100,
            Confirm     => SDO_Confirm,
            Error       => SDO_Error,
            Device      := Device_ID,
            Channel     := 1,
            Index       := 12288,
            Subindex    := 0,
            Data        := CW_Array_Dis,
            Datalength  := 2,
            Errorinfo   => SDO_Error_Info);
```

5 Visualization Manager

The Nli example project contains a visualization interface with the buttons to enable function blocks and various motor paramaters (similar to the NiLAB Starter application). From this window is possible to enable the axis linked to the Nli device by the Node-ID and to activate the MovePtP function block, as well with the possibility of change the positions, speed, acceleration and deceleration values directly from the interface. There is a button for the reading of all motor parameters (SDO READ) and one for the reading of the Status Word, and a function to loop the readings in time as well. The "Status" section refers to the drive status during the activation of the function blocks and the "Reset_State" can be used to reset the state machine status inside the MovePtP block. The orange "MOVE" button will enable just one movement to the set position, the other button will start a moving loop between the 2 set positions.



STATUS WORD

SDO READ

READING LOOP

DEVICE ID	%i	
DRIVE STATUS	%s	
DEVICE ON TIME (5-6)	%t[HH:mm:ss]	
I2T CURRENT (4129)	%i	Arms
DC LINK (4631)	%f	Volts

AXIS 1

ENABLE

POLE PITCH	%i	um
ACTUAL POSITION	%i	um
SET POSITION	%i	um
MODE OPERATION	%i	
MODE OP. DISPLAY	%i	
STATUS WORD	%i	
CONTROL WORD	%i	

MOTOR PARAMETERS

MOTOR TYPE (0)	%i	
NOMINAL CURRENT (62)	%i	mA
MAXIMAL CURRENT (65)	%i	mA
MAX CURRENT TIME (64)	%i	s
FORCE CONSTANT (57)	%i	N/A
I2T VALUE (67-68)	%i	A²s
POLE PITCH (66)	%i	mm
MAX TEMP DRIVE/MOTOR (69)	%i	°C
MAX TEMP PT600 (182)	%i	LSB
COMMUTATION OFFSET (219)	%i	Counts

ENCODER CALIBRATIONS

SIN OFFSET (214)	%i
COS OFFSET (215)	%i
SIN GAIN (216)	%f
COS GAIN (217)	%i
SIN/COS FILTER LENGTH (218)	%f

HOMING SETUP

HOMING CURRENT THRESHOLD (60)	%i	mA
HOMING SPEED (61)	%i	mm/s²
HOMING ZERO POSITION (59)	%i	

MOVE ABSOLUTE

MOVE

SET POSITION A	%i	um
SET POSITION B	%i	um
VELOCITY	%i	um/s
ACCELERATION	%i	um/s²
DECELERATION	%i	um/s²

STATUS

Reset_State

MOVE_DONE

MOVE_BUSY

MOVE_ERROR

CONTROL_FAULT

NMT STATE

eSTATE DSP 402	%s
NMT STATE (19)	%s
STATUS WORD	%s

CONTROLLER CONFIGURATION

KP-GAIN (50)	%i	
KI-INTEGRAL (51)	%i	
I MAX A (65)	%i	mA
KP-GAIN (70)	%i	
KI-INTEGRAL (71)	%i	
MAX SPEED (78)	%i	m/s
LP FILTER (79)	%i	Hz
LP FILTER (80	%i	Hz
KP-GAIN (100)	%i	
KI-INTEGRAL (101)	%i	
ACTIVE (102)	%i	
SPEED FF GAIN (103)	%i	
ACC. FF GAIN (104)	%i	
ACTIVE (105)	%i	