Driving efficiency and sustainability

PLC500 MOTION CONTROLLER

PLC500MC

Application Note

Application Note

PLC500MC

Document: 10010339513

Revision: 00

Publication Date: 05/2023

The information below describes the reviews made in this manual.

Version	Revision	Description
-	R00	First Edition.

CONTENTS

1 INTRODUCTION

This Application Note describes the main characteristics and necessary information for the configuration and use of the PLC500MC together with the SCA06 servo drive.

For motion control, the correct settings of the network and devices involved is essential. Please, follow the steps described in this document for the proper setup.

For further information about the product hardware, interfaces and communication protocols, refer to the PLC500 User Manual, available at http://www.weg.net.

1.1 ABBREVIATIONS AND DEFINITIONS

CNC: Computer Numerical Control—it is a method that controls the motion of machines by direct interpretation of coded instructions in the form of numbers and letters.

Codesys: Programming platform that allows developing, configuring and monitoring solutions for industrial automation and system integration.

CoE: CANopen over EtherCAT.

EDS: Configuration file that contains information about objects, services and settings of a network slave.

EEPROM: Electrically Erasable Programmable Read-Only Memory

Ethernet: Interconnection architecture for local area networks (IEEE 802.3).

EtherCAT: Technology for Ethernet-based real-time communication (Ethernet for Control Automation Technology).

FB: Function block.

MC: Motion controller.

PDO: Process data.

PLC: Programmable logic controller.

PLCopen: Organization that promotes industrial control based on the IEC61131-3 standard.

POU: Program organization unit.

SCA06: WEG Servo drive - SCA06.

SoftMotion: Smooth motion control.

SDO: Service data.

u: Application unit.

XML: Configuration file that contains information about objects, services and settings of an EtherCAT slave.

1.2 ABOUT THE PLC500MC

The PLC500 Motion Controller (PLC500MC) is a Programmable Logic Controller with the SoftMotion function that enables the control of up to **32 real or virtual axes**, allowing a great variety of motion controls, such as positioning of single axis, synchronization of multiple axes (electronic cams and electronic gears), interpolation of multiple axes (linear, circular and helical), speed control, torque control, G-Code reading and interpretation, control of CNC machines, control of cutting machines and industrial robots, among other functionalities.

It is developed to solve medium and large applications. It features high processing speed due to its CPU, consisting of a Dual-core ARM Cortex-A7 processor running at 1 GHz, a 200 MHz Real-time ARM Cortex-M4 coprocessor, 1 GByte RAM memory and 4 GByte Flash memory.

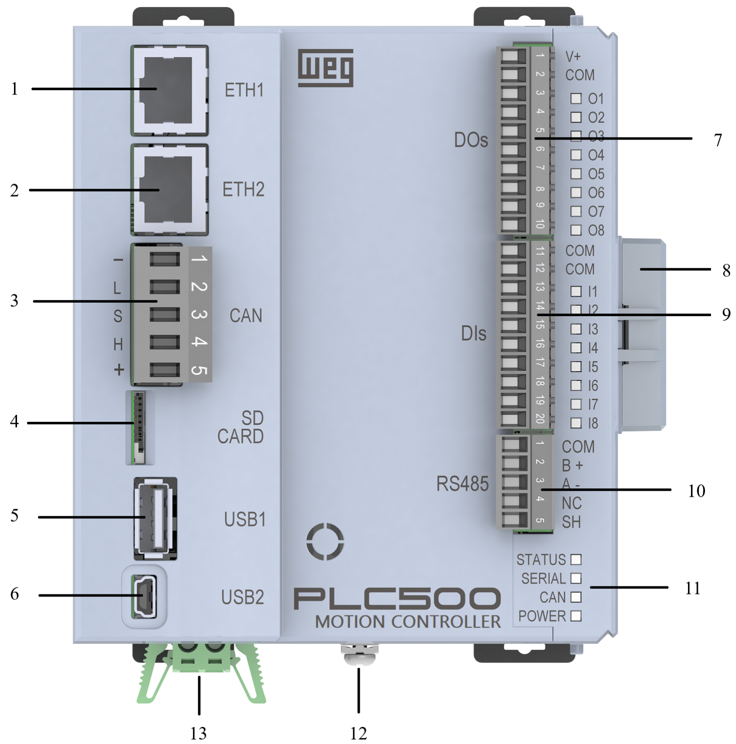
It has eight digital outputs, three of which have PWM functionality up to 300 kHz, and eight digital inputs, four of which can operate up to 150 kHz.

As communication interfaces, it has two independent Ethernet ports, CAN port, serial RS485, USB OTG, USB device and micro SD card.

Built-in supercapacitors are used for real time clock (RTC) and for saving retentive data to flash memory during power off, eliminating the need for batteries.

The PLC500MC allows the connection of expansion cards for digital, analogue, thermocouple, PT100, PT1000, load cell, relay and other inputs and outputs, providing more flexibility for applications. It has plug-in connectors and can be mounted on a DIN 35 rail or directly on the panel.

The PLC500MC is programmed via CODESYS software, widely used in the industrial environment, allowing the use of numerous applications and functions already developed on the market, as well as the import of applications from other products.



Nº	Information
1	Ethernet connector 1
2	Ethernet connector 2
3	CAN Connector
4	Memory Card
5	USB1 Host
6	USB2 Device
7	Digital Output Connector
8	Closure of Expansions
9	Digital Input Connector
10	Serial RS485 Connector
11	Indication LEDs
12	Grounding Screw
13	Product Power Supply (24 V)

Figure 1.1: *PLC500 Motion Controller.*

The PLC500MC has a large memory area available to the user. The memory usage of an application can be viewed through Codesys at: View->View memory usage.

The PLC500MC memory is divided according to the table below.

Memory	Capacity	Description
Area 0 (DATA)	128M Bytes	Stores all local and global data (variables, function blocks, instances etc.).
Area 1 (CODE)	32M Bytes	Stores all codes generated by the application and also the constant data.
Area 2 (RETAIN)	64k Bytes	Stores retain-type variables (keeps the value after the controller reboot).
Area 3 (PERSISTENT)	16k Bytes	Stores persistent-type variables (keeps the value after reboot and after download if their layout remains identical).

Table 1.1: *Memory areas.*

1.3 ETHERCAT TECHNOLOGY

EtherCAT (**Ether**net for **C**ontrol **A**utomation **T**echnology) is a powerful Ethernet-based real-time communication technology. With its short cycle times, low jitter values and different network topologies, the system is standard in many industrial automation applications today.

1.3.1 EtherCAT PLC500MC Interfaces

The PLC500MC has two independent interfaces (**ETH1** and **ETH2**) that can be used for EtherCAT communication. Figure 1.2 shows the PLC500MC and its two possible interfaces for EtherCAT communication.

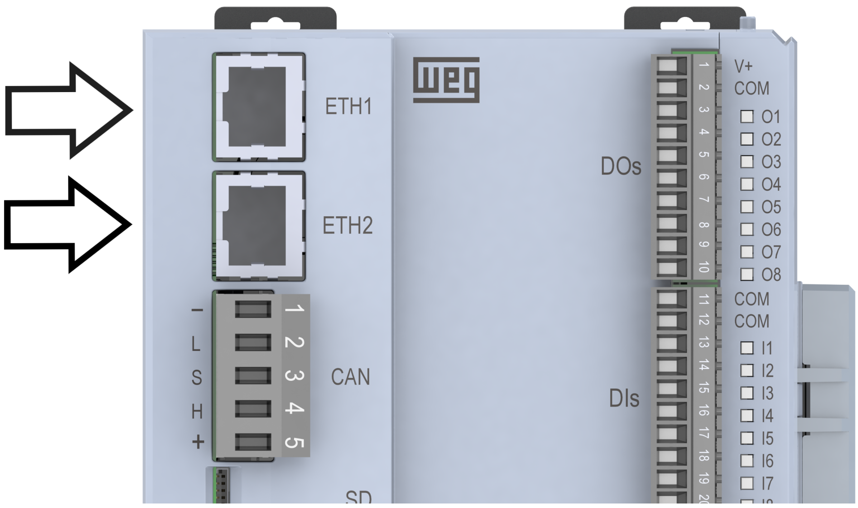


Figure 1.2: *EtherCAT Interfaces*

1.3.2 EtherCAT Scope

Functionalities supported by the EtherCAT protocol available on the PLC500MC include:

- Different bus topologies (line and star).
- Great flexibility with hot connection.
- Distributed clocks.
- Bus diagnostics: by the editor and by the application.
- Network scan: recognize and insert connected slaves.
- EtherCAT redundancy.
- Supported protocol layers:
  - CoE (CANopen over EtherCAT) / SDO Communication.
  - EoE (Ethernet over EtherCAT).
  - SoE (Servodrive over EtherCAT).

– FoE (File over EtherCAT).
– VoE (Vendor over EtherCAT).

- Support for MDP slaves (Modular Device Profile).
- Various function blocks for use in the application.

The PLC500MC has an interface that simplifies the settings of the EtherCAT network master and its slaves. Through this interface it is possible to:

- Configure the network automatically or use the expert mode.
- Add and configure slaves using XML EtherCAT (ESI) files.
- Configure sync units. (Sync Unit)
- Configure PDOs (process data).
- Configure boot parameters for CoE and SoE.
- Configure EoE slaves.
- View CoE objects online and support SDOinfo upload.
- View network diagnostic history online.
- Read and write to the device EEPROM memory.

1.4 MOTION CONTROL

The PLC500MC enables motion control for single axis and multiple synchronized axes (electronic cams and electronic gears) and allows control of CNC machines and industrial robots.

Servo drives compatible with CiA402 can be easily operated by the PLC500MC without users worrying about status word, control word, operation mode and other parameters needed for motion control.

The PLC500MC has several specific functionalities for motion control, including:

- Extensive library of blocks for axis control, handling and processing of CNC paths, axis groups and popular kinematic transformations.
- Integrated cam editor.
- Integrated 3D CNC editor according to DIN 66025 (G-Code).
- Configurator of axis groups for different kinematics (customizable).
- Easy commissioning of axes (using the **Online Configuration Mode**).
- Function blocks certified according to **PLCopen MotionControl, Part 1 (V20)**.
- **G-code** decoder, including support for subprograms and expressions in **G-code**.
- Function blocks for testing transition speeds.
- Function blocks for reading and processing CNC paths from files (for externally created and processed paths).
- Function blocks certified according to **PLCopen MotionControl Part 4 (coordinated motion)**.

1.4.1 Cam editor

The PLC500MC has a cam table editor that simplifies visualization and implementation for this purpose.

Scope of cam editor:

- Graphical and numerical planning for the cam using any base representation of distance, speed, acceleration and jerk.
- Linear or polynomial interpolation (5th order polynomial).
- Configuration of the tappets and their switching behavior on the cam.
- Cam configuration relating to dimension, period and continuity requirements.
- Possibility to import and export cam tables.

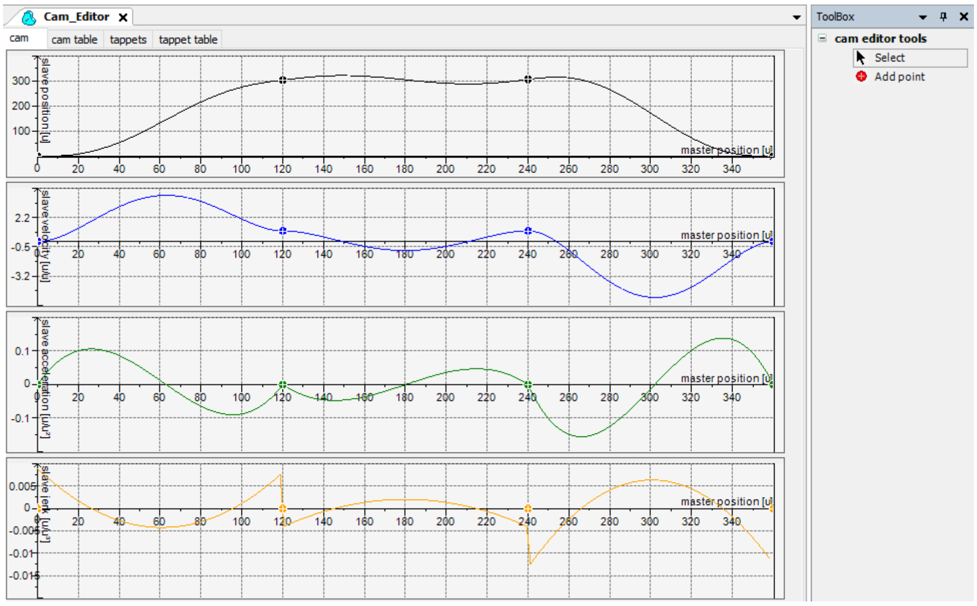


Figure 1.3: *Cam editor.*

Additional information regarding the Cam Editor can be found directly on the Codesys website, available at: https://help.codesys.com (Add-ons > CODESYS SoftMotion > Basic Motion > Cams).

1.4.2 3D CNC editor

The PLC500MC can interpret and execute G-Code programs in accordance with DIN 66025.

Scope of the 3D CNC editor according to DIN 66025 (G-Code):

- Simultaneous graphic and text editor.
- Path preprocessing (offline preview of effects, e.g., angle smoothing).
- Path pre-interpolation (preview (offline) of the resulting position, speed, acceleration and jerk curves of all supported axes).
- Importing DXF and ASCII files (.cnc, .gcode, .txt).
- Read and save in file.
- Program transformations (rotate, shift and resize the G code).

- Conversion to tables.
- Program information (path length, path duration, number of objects etc.)

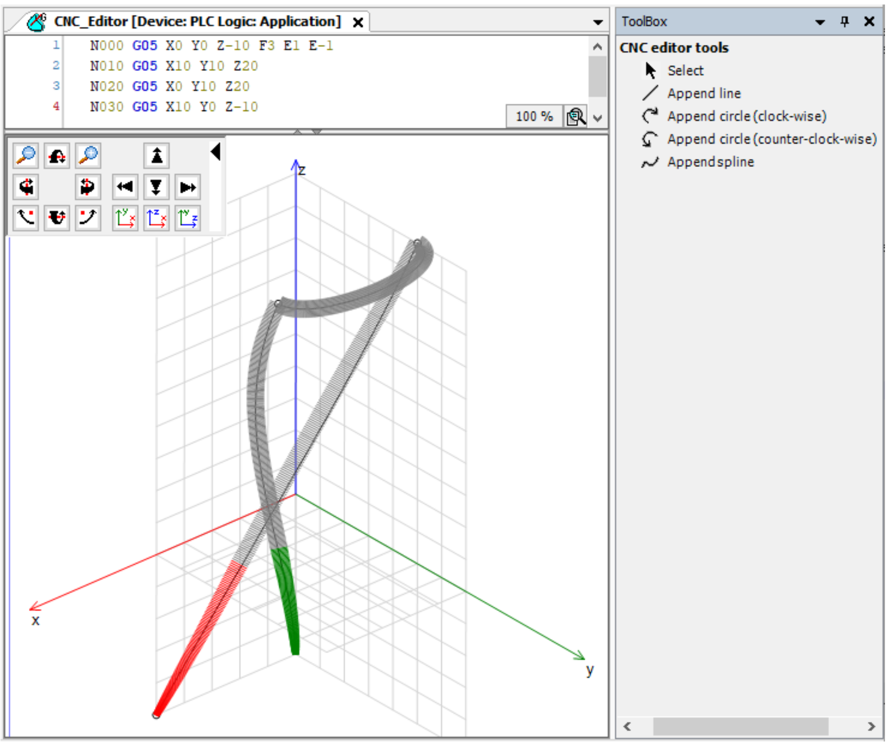


Figure 1.4: *3D CNC Editor.*

Additional information regarding the 3D CNC Editor can be found directly on the Codesys website, available at: https://help.codesys.com (Add-ons > CODESYS SoftMotion > CNC > Editor).

1.4.3 Scope of SoftMotion + CNC Robotic libraries

Motion control instructions are defined as function blocks (FB) and can be used during the application to perform a wide variety of motions. Instructions for motion control are developed based on the specifications of the PLCopen function blocks[1]. In addition to the PLCopen-based instructions, additional blocks are also available, simplifying the implementation of motion control.

- Function blocks certified according to PLCopen MotionControl, Part 1 (V20):
  - Absolute and relative positioning (**MC_MoveAbsolute**, **MC_MoveRelative**).
  - Overlapping positioning (**MC_MoveSuperimposed**).
  - Constant speed motion (**MC_MoveVelocity**).
  - Consistent support of profiles with jerk limitation (continuous acceleration for any kind of interruption of the current motion).
  - Drive guided return (**MC_Home**).
  - Blocking stop (**MC_Stop**).
  - Control release (**MC_Power**).
  - Reading and writing of parameters (**MC_Read/WriteParameter**).
  - Reading of the actual position (**MC_ReadActualPosition**).
  - Position, speed and acceleration profiles (**MC_*Profile**).
  - Set and move the position (**MC_SetPosition**).
  - Reading of actual speed and actual torque (**MC_ReadActualVelocity**, **MC_ReadActualTorque**).

[1] PLCopen is an organization that promotes industrial control based on the IEC61131-3 standard. For more information about PLCopen, refer to the official website at: http://www.plcopen.org/.

- – Cam switching (**MC_DigitalCamSwitch**).
- – Electronic gear with synchronization position (**MC_GearInPos**).
- – Full halt (**MC_Halt**).
- – Tracking of master signals respecting speed, acceleration and jerk limits (**SMC_TrackSetValues**).
- – Additional blocks.
- – Brake control and query.
- – Monitoring of dragging error, a position window or maximum values.
- – Measuring of distance traveled.
- – Management of errors in function blocks.
- – Controller-guided return (**MC_Homming**).
- – Device commissioning.
- – Absolute and relative positioning with transition speed (**SMC_MoveContinuousAbsolute** and **SMC_MoveContinuousRelative**)
- – Control mode settings (position, speed or torque).

- Visualization templates for the most important function blocks used for quick commissioning integrated into the Codesys software.
- G code decoder.
- Support for G-code subprograms and expressions.
- Limiter to restrict the values of speed and acceleration dynamics for one or more axes.
- Blocks for testing speeds in transitions.
- Interpolator to calculate CNC path points based on the speed profile.
- Blocks for coordinate transformation (**SMC_ScaleQueue3D** and **SMC_CoordinateTransformation3D**).
- Transformation blocks (including inverse) for popular kinematics:

  - – 2D / 3D gantry systems.
  - – Gantry systems with orientation axes and tool compensation.
  - – Belt-driven gantry systems (H-gantries and T-gantries).
  - – Polar transformation.
  - – SCARA with 2/3 arms.
  - – Bipod.
  - – Tripod with linear and articulated axes.
  - – 5-axis kinematics for 3-axis gantry with rotary and tilting tool.
  - – 4-axis kinematics for palletizing robots.
  - – 6-axis kinematics for articulated-arm robots.

- Blocks for reading and processing CNC paths from a file (for externally created and processed paths).
- Trapezoidal/sigmoidal/quadratic/ smooth quadratic path speed modes.
- Odometer function.
- Parameterizable 3D coordinate transformation (including inverse).
- Certified function library with function blocks according to PLCopen Motion, Part 4 (Coordinated motion).

  - – Administrative blocks: **MC_GroupEnable/Disable/Reset/ReadError**, etc.
  - – Motion commands: **MC_MoveDirectAbsolute**, **MC_MoveDirectRelative**, **MC_MoveCircular***, **MC_MoveLinear***, **MC_GroupHalt**, **MC_GroupStop**.
  - – Monitoring: **MC_TrackConveyorBelt**, **MC_TrackRotaryTable**, **MC_SetDynCoordTransform**.
  - – Jog mode in any coordinate system: **SMC_GroupJog2**.
  - – Support of different coordinate systems: world coordinates (WCS), machine coordinates (MCS), various product coordinates (PCS_1, PCS_2), tool coordinates (TCS) and axis coordinates (ACS).

- Support for wait on the path with holding time (**SMC_GroupWait**).
- Public and documented interface for creating user-specific kinematics in the IEC 61131-3 language.
- Additional orientation kinematics, which can be combined with the other kinematics.
- Tools with orientation and position shift.

2 CREATING AND CONFIGURING ETHERCAT NETWORK + SOFTMOTION

This section describes the necessary steps to perform EtherCAT communication between the PLC500MC and the SCA06 servo drive through the Codesys software. Additional information and advanced settings will be described in the remaining sections of this application note or can be found directly on the Codesys website, available at: https://help.codesys.com.

2.1 COMPONENTS USED

The components necessary for this application manual:

Component	FW version
PLC500MC	1.2.0 or above
Servoconversor SCA06	2.11 or above
Servomotor	Compatible with the servo drive
EtherCAT ECO4 accessory	Rev. 2436 or above

Table 2.1: *Necessary components.*

For passive network components (cables, connectors and power supply), use only components certified for industrial applications. Refer to the product documentation for more information about the proper installation of the SCA06 servo drive and the used servomotor.

2.2 NETWORK ARCHITECTURE

Figure 2.1 shows the topology of the network used—the computer must be connected to the PLC500MC through the ETH1 or USB2 interface. The EtherCAT communication with the SCA06 servo drive will use the ETH2 interface of the PLC500MC.

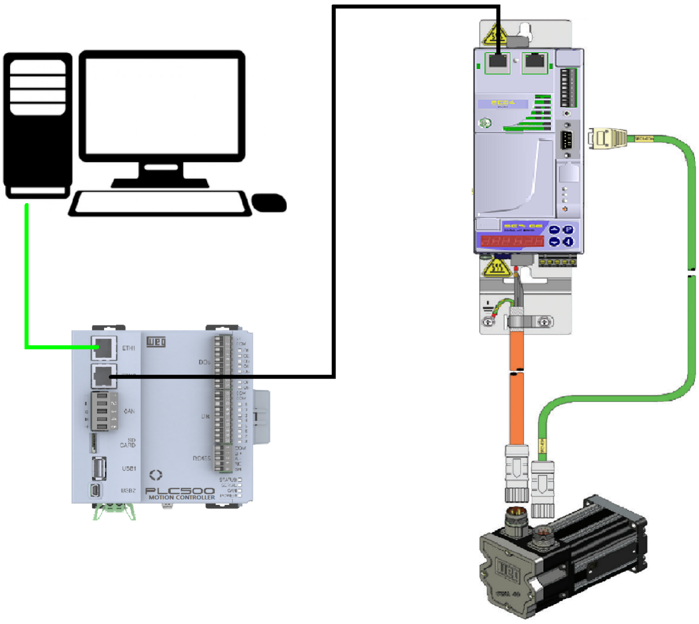


Figure 2.1: *Network architecture.*

2.3 SCA06 SERVO DRIVE SETTINGS

Correctly connect the EtherCAT ECO4 accessory and the servomotor to the SCA06 servo drive.

Starting from the SCA06 factory parameters:

- Change parameter **P0202** to **5** (control via CAN/EtherCAT network).
- Change parameter **P0385** to set the motor model according to its nameplate and motor table.

Follow the recommendations described in the SCA06 servo drive manual to set the device parameters related to the motor settings, desired functions for the I/O signals, etc.

For further explanations, refer to the SCA06 servo drive Programming Manual.

- Restart the servo drive.

By doing so, the SCA06 servo drive will be ready to be accessed through the EtherCAT network.

2.4 CREATING A PROJECT IN CODESYS

- Download the Codesys software and install the **WEG Package** according to the PLC500 manual.
- After installation, open Codesys and create a new project in **File > New Project**. Select **Standard Project**, define a directory and the application name. Select the **PLC500MC** Device and the desired programming language, as shown in Figure 2.2.

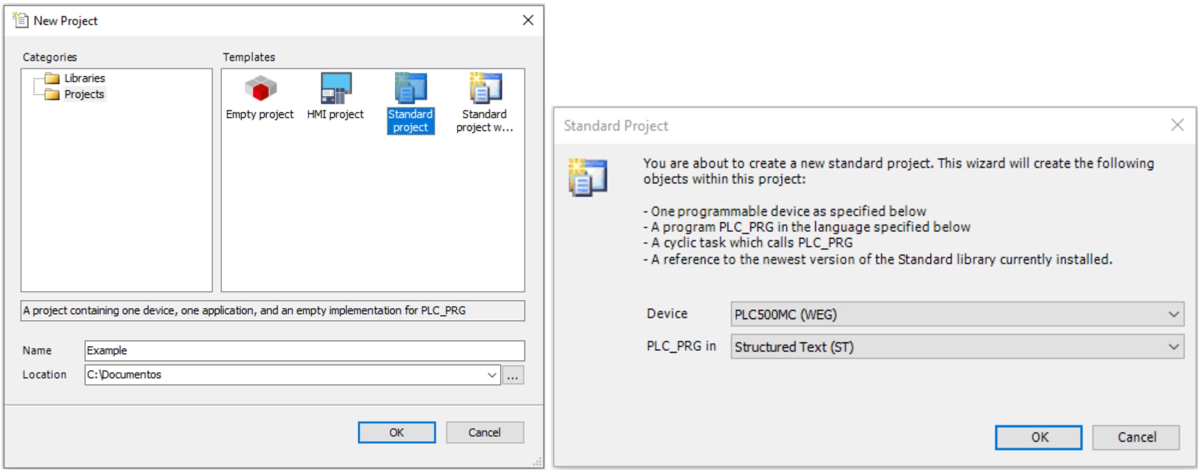


Figure 2.2: *Project configuration in Codesys.*

When creating an application for the Device PLC500MC, the standard network interfaces will be automatically pre-configured, as shown in Figure 2.3.

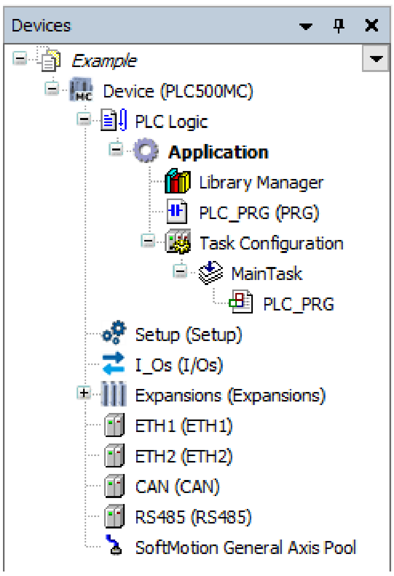


Figure 2.3: *PLC500MC Interfaces*

2.4.1 Add EtherCAT Master SoftMotion

- To add a new **EtherCAT Master SoftMotion** communication interface, right-click **Device (PLC500MC)** and click **Add Device**. In the dialog, select **Fieldbuses > EtherCAT > Master > EtherCAT Master SoftMotion** and click **Add Device** to add it to the device tree, as shown in Figure 2.4.

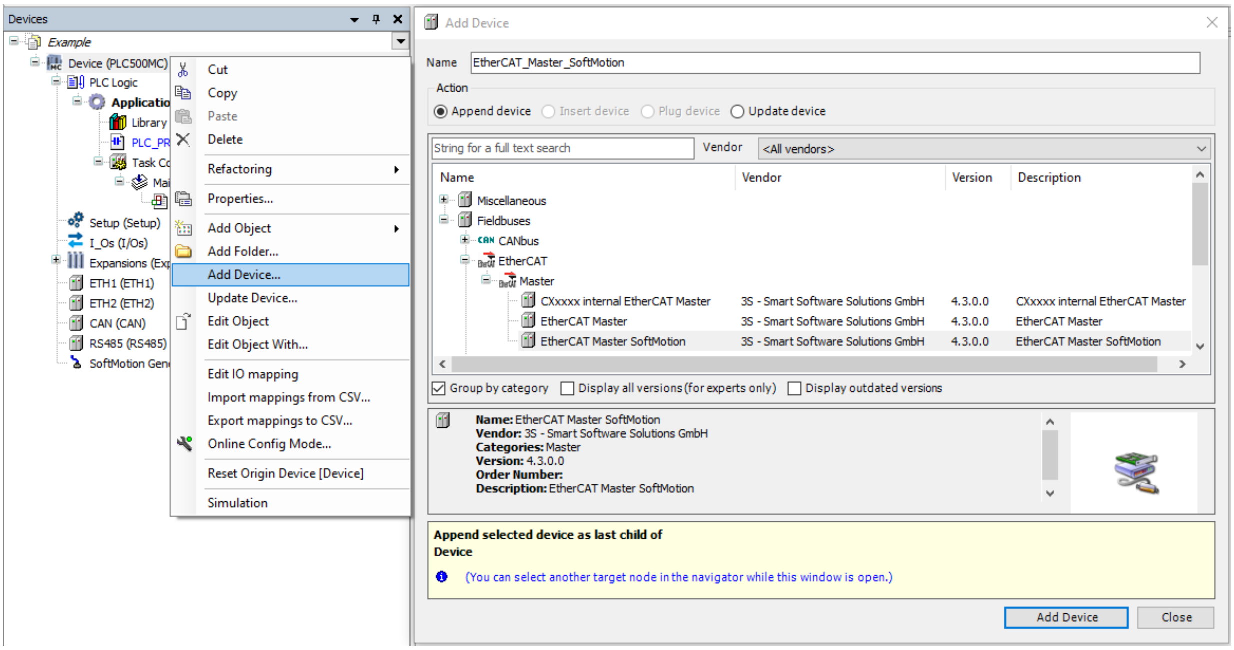


Figure 2.4: *Adding **EtherCAT Master SoftMotion** to the device tree.*

When adding the **EtherCAT Master SoftMotion** communication interface, a task called **EtherCAT_Task** will be created[2].

2.4.2 Adding SCA06_SoftMotion as slave to EtherCAT network

- To add the **SCA06_SoftMotion** device as a slave of the EtherCAT network, right-click the **EtherCAT Master Softmotion** device previously created and select **Add Device**.)
- In the **Action** section, from the opened dialog, make sure that the **Append device** option is checked. Search for the **SCA06_SoftMotion** device—it is located inside the **WEG > Servo Drives** folder.
- Click on **Add Device**.

Figure 2.5 shows the previous steps directly in the Codesys software.

[2]Task used for **SoftMotion** motion control commands.

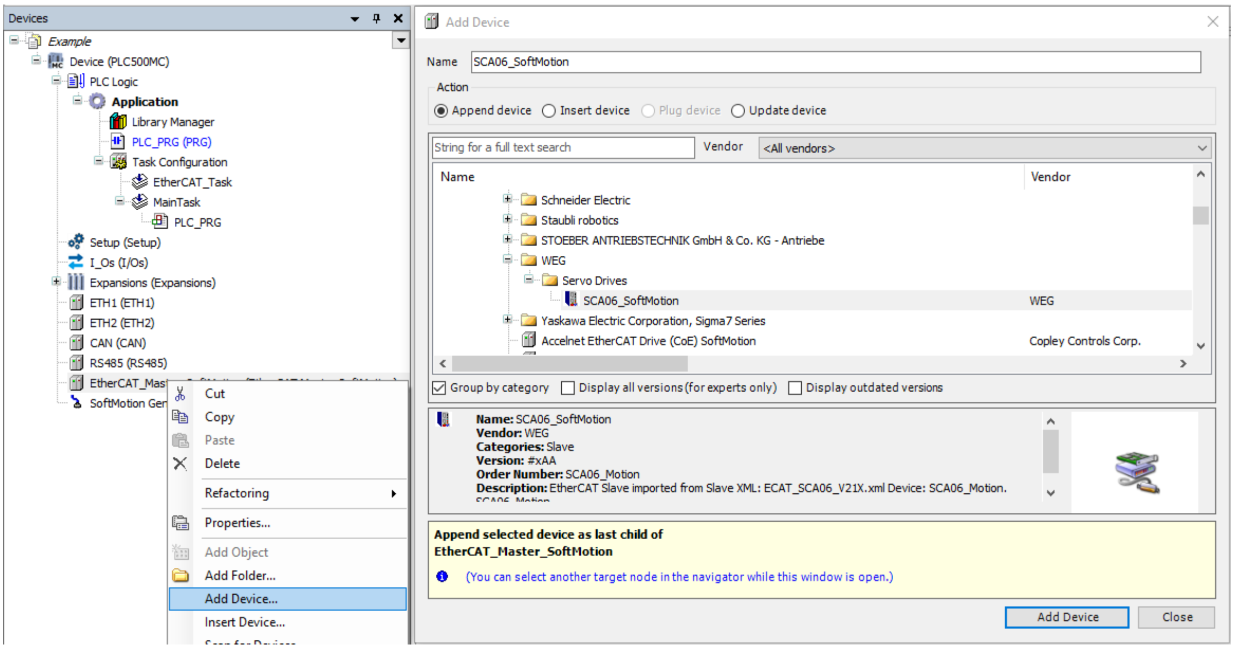


Figure 2.5: *Adding SCA06_SoftMotion as a slave to the EtherCAT network.*

> **NOTE!**
> It is important to add the SoftMotion unit, because the default unit, imported via XML, does not contain an associated SoftMotion axis (the **SCA06_SoftMotion** is installed along with the WEG Package).

After these settings, the device tree should contain the icons shown in Figure 2.6.

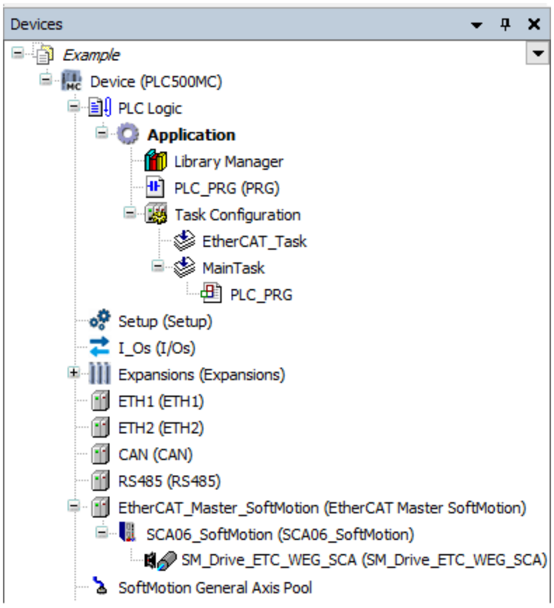


Figure 2.6: *Device tree for using SoftMotion.*

2.4.3 Configuring EtherCAT Master SoftMotion

- Open the **EtherCAT Master SoftMotion** device settings, and on the **General** tab, select **Autoconfig Master/Slave**. By doing so, the main master/slave settings will be done automatically based on the device description file.
- Configure the other options on the page as shown in Figure 2.7

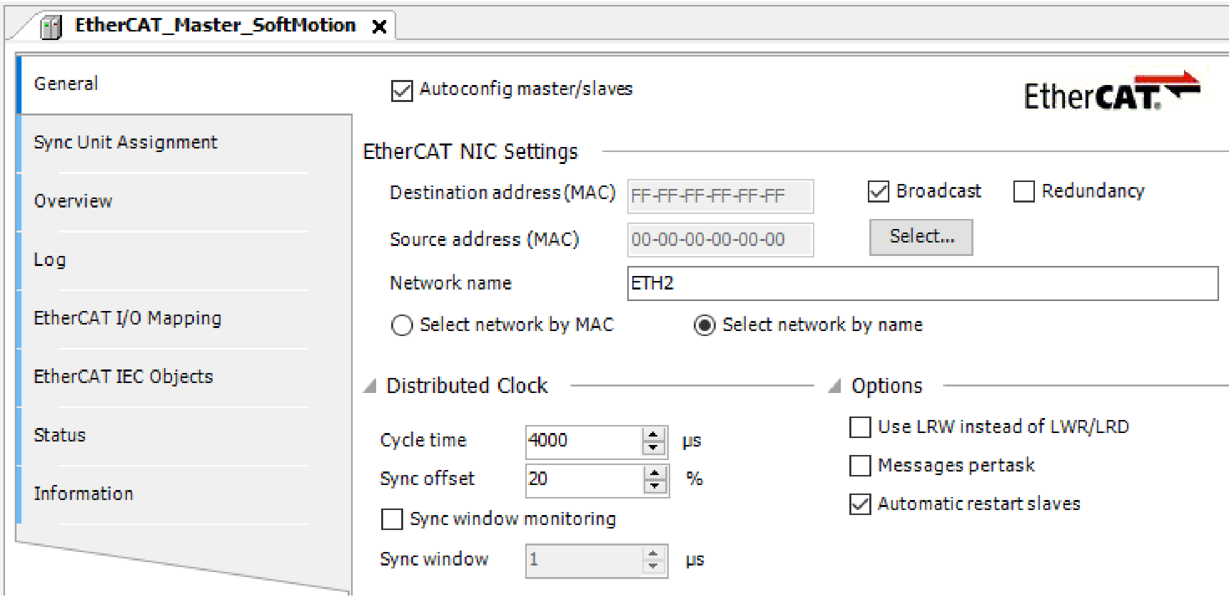


Figure 2.7: *Default EtherCAT Master SoftMotion settings.*

Information about advanced settings will be presented in Section 4, or it can be found directly on the Codesys website, available at: https://help.codesys.com (Fieldbus Support > EtherCAT > Configuration).

2.4.4 Configuring SCA06_SoftMotion

If the **Autoconfig Master/Slave** option is used in **EtherCAT Master Softmotion**, the **SCA06_SoftMotion** servo drive configuration will be done automatically.

Information about advanced settings will be presented in Section 4, or it can be found directly on the Codesys website, available at: https://help.codesys.com (Fieldbus Support > EtherCAT > Configuration).

2.4.5 Configuring SM_Drive_ETC_WEG_SCA

- Open the SM_Drive_ETC_WEG_SCA settings.

On the **General** tab are the settings referring to the type and limits of the axis, type of the speed ramp and drag supervision.

- Configure the page as shown in Figure 2.8.

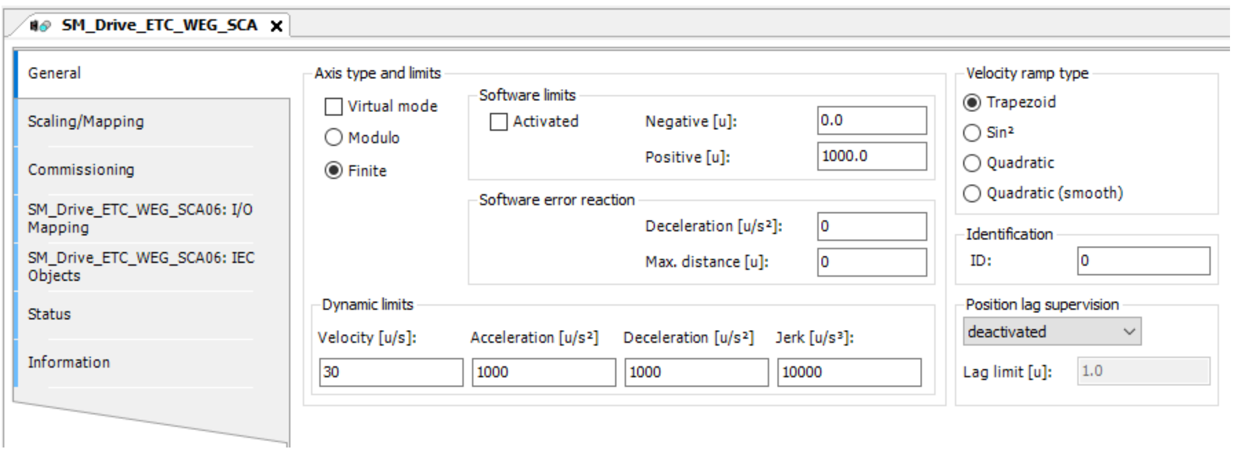


Figure 2.8: *SM_Drive_ETC_WEG_SCA default configuration.*

This configuration defines the axis as finite, limiting by software disabled, trapezoidal speed ramp, axis ID equal to 0, no drag supervision and dynamic limits[3] defined in the *Dynamic limits* field.

- Click the **Scaling/Mapping** tab.

On the **Scaling/Mapping** tab, you can define the relationship between the application units (e.g., millimeters or degrees) and the servo drive unit (pulses).

- Configure the page as shown in Figure 2.9.

NOTE!
It is possible to manually map the variables of the SM_Drive_ETC_WEG_SCA. To do so, in the **Mapping** field, uncheck **Automatic mapping**.

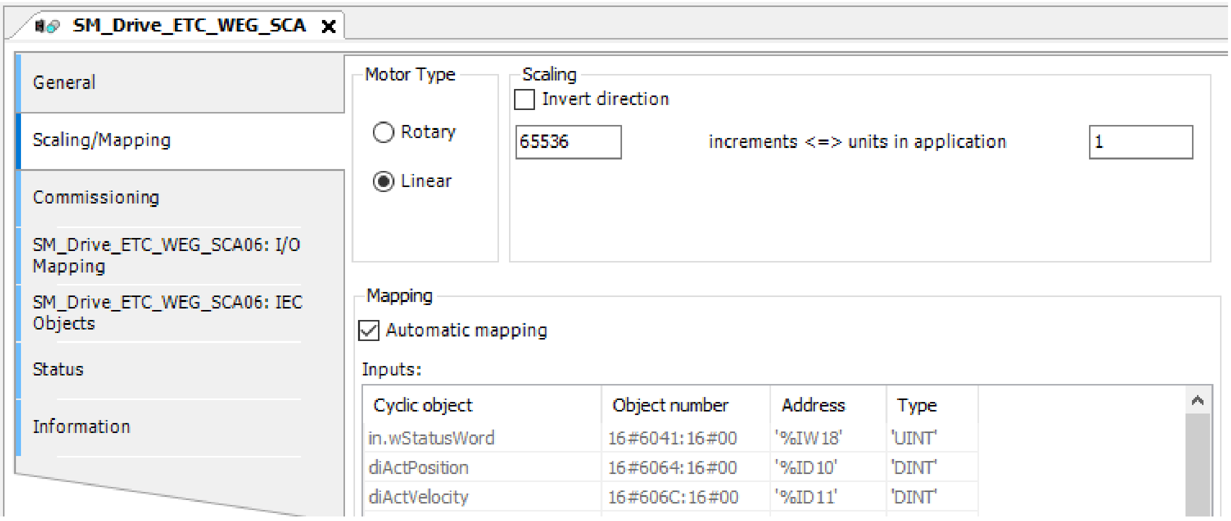


***Figure 2.9:** Default SM_Drive_ETC_WEG_SCA scale configuration.*

This configuration defines that 65536 servomotor pulses will be equivalent to one application unit, that is, each application unit will be exactly one revolution in the servomotor[4].

ATTENTION!
It is essential to correctly configure these values, as the SoftMotion function blocks will use the **application unit** as a parameter for the motion.

Information about advanced settings will be given in Section 5, or it can be found directly on the Codesys website, available at: https://help.codesys.com (Add-ons > CODESYS SoftMotion > Reference > User Interface > Objects > SoftMotion Drives).

- After applying the settings of this section, download the program to the **PLC500MC** and do the monitoring in the **Online** mode.

2.5 MONITORING

2.5.1 EtherCAT communication status

The status of the EtherCAT network can be monitored in Codesys **Online** mode. When encountering connection problems, as shown in Figure 2.10, check again if the cables are properly connected and review the settings in Section 2.

[3]These limits are taken into account when using a group of axes (PLCopen Part 4). In addition, they are used by **SMC_ControlAxisBy*** function blocks to detect leaps.

[4]The SCA06 servo drive has a resolution of 65536 pulses per revolution (see the SCA06 EtherCAT manual for more information).

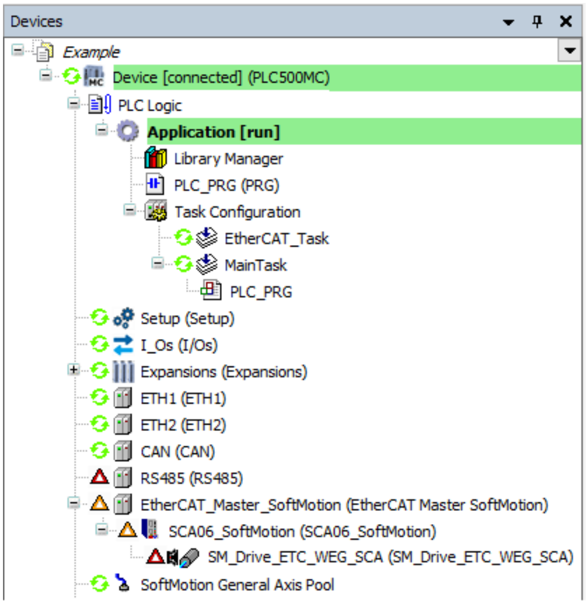


Figure 2.10: *Indication of EtherCAT communication error.*

When the settings are correct and the devices are communicating properly, all EtherCAT communication items will be green, as shown in Figure 2.11.

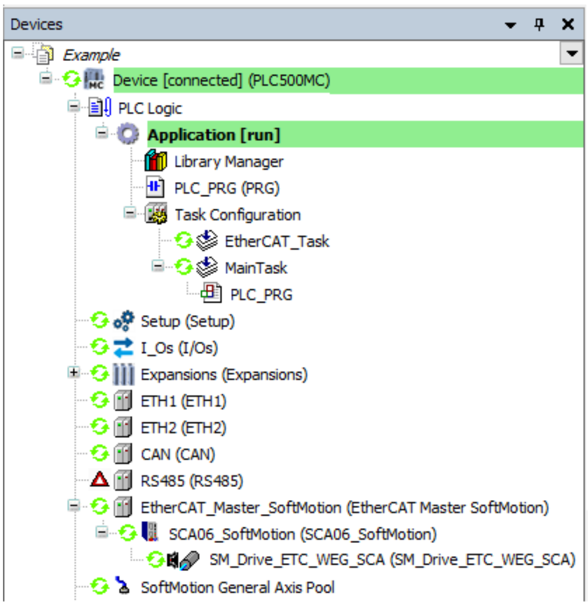


Figure 2.11: *Communication properly configured and devices communicating.*

2.5.2 Check variation in the current position of the servomotor

- After correctly setting the EtherCAT network and still in the **Online** mode, open the **SM_Drive_ETC_WEG_SCA** settings.

When the PLC is in the **Online** mode, on the **General** tab, a field will be enabled for viewing the axis, as shown in Figure 2.12.

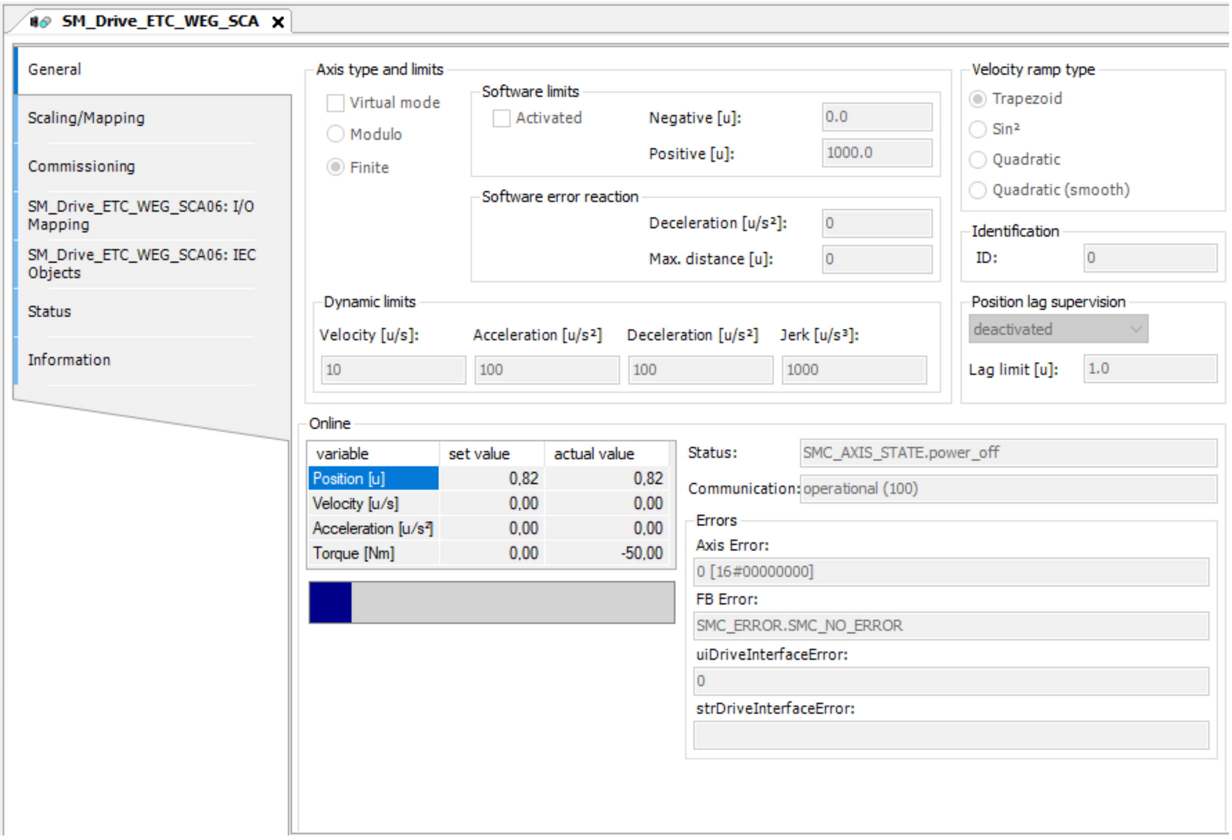


Figure 2.12: Monitoring the servomotor online.

In this field it is possible to observe the communication and axis status, position variables, speed, acceleration and torque, with their references and current values.

- Move the servomotor shaft manually and observe the position value changing in **Position [u] - actual value**.

2.6 COMMISSIONING

It is possible to test the settings applied to the SCA06 servo drive through the steps presented in this subsection.

- Exit the **Online** mode and enter into the PLC again using the **Online Config Mode** option. This is the PLC configuration mode, through which it is possible to test and validate the configurations applied to the servo drive.

> **ATTENTION!**
> When using the **Online Config Mode** option, the application in the PLC will be automatically deleted.

To use the **Online Config Mode** option, on the device tree, click **PLC500MC** and then click **Online Config Mode**, as shown in Figure 2.13

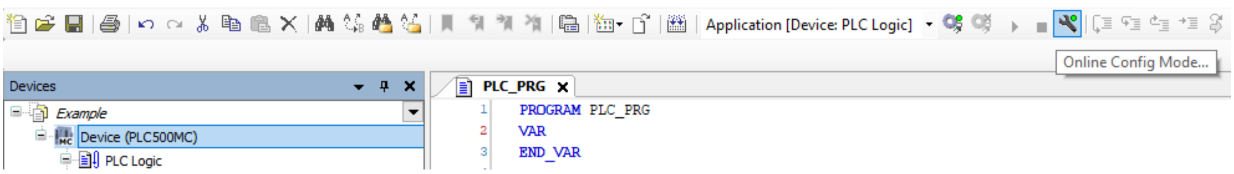


Figure 2.13: Online Config Mode.

- Open the servomotor settings (**SM_Drive_ETC_WEG_SCA**) and click the **Commissioning** tab. On this tab, in addition to the variables and status of the axis, some buttons are available for activating the servomotor, as shown in Figure 2.14.

NOTE!
This page is only enabled using the **Online Config Mode** option.

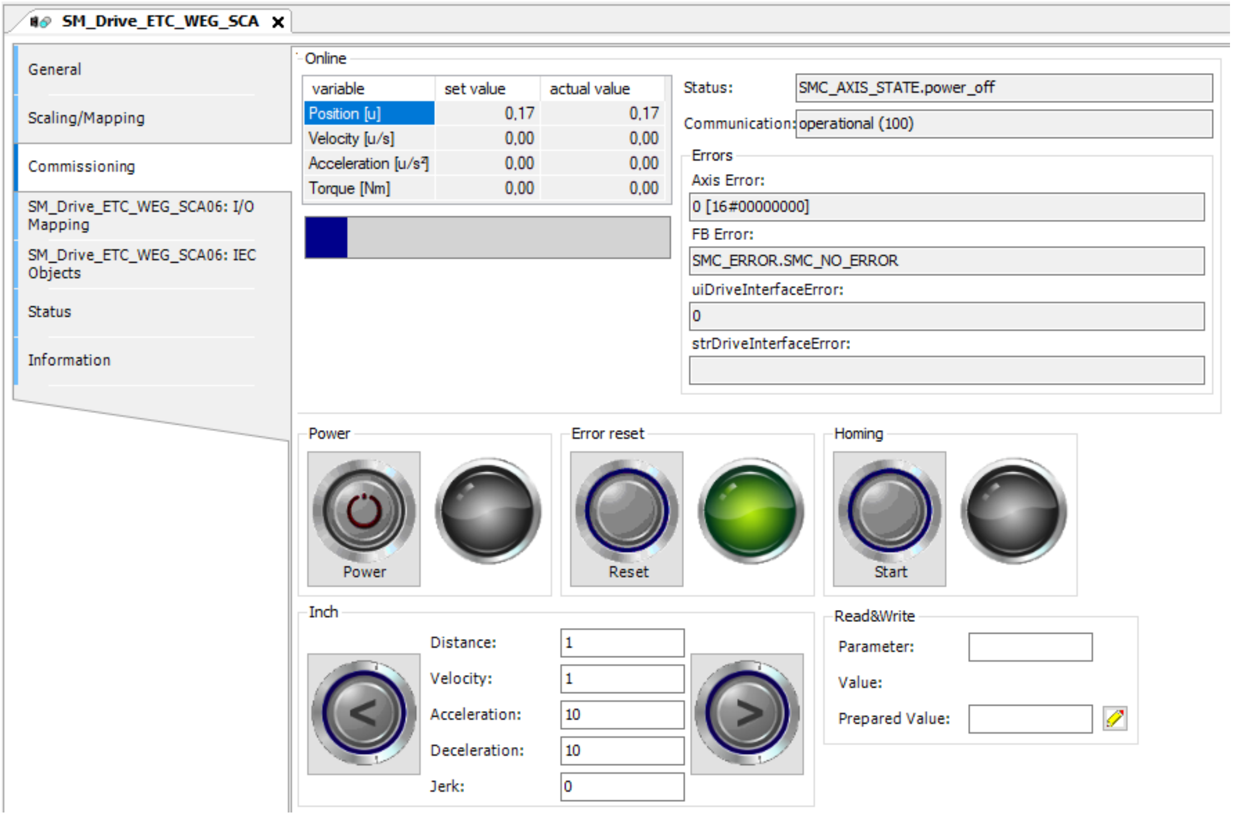


Figure 2.14: *Commissioning the SCA06_SoftMotion.*

ATTENTION!
You can move the axis using the buttons on this page. The axis can perform unexpected motions if the settings are not correct. Therefore, take all necessary safety precautions.

Operating elements	Description
Power	The Drive is enabled (equivalent to the **MC_Power**) function block.
Error reset	Restarts the Drive after an error (equivalent to **MC_Reset**) function block.
Start homing	The Drive executes the *homing* command with the parameters defined internally in the servo drive (equivalent to the **MC_Home** function block)*.
Jogging mode	Using the **<** and **>** buttons, you can move the axis back and forth according to the values specified for **Distance**, **Velocity**, **Acceleration**, **Deceleration** and **Jerk** (equivalent to **MC_Inch** function block).
ReadWrite	For the specified drive parameter, the current value (**Value**) is read by the PLC and displayed. In **Prepared value**, you can specify a new value and write it to the drive parameter (equivalent to **MC_ReadParameter** and **MC_WriteParameter** function blocks).

Table 2.2: *Commissioning elements.*

- Click the **Power** button to enable the servo drive, then hold down the **>** button. The servomotor must make a complete turn and stop.
- If you wish, test some more commands and exit the **Online Config Mode**.

3 SOFTMOTION APPLICATION

This section describes the necessary steps to create a SoftMotion application to control a simple axis.

3.1 CREATING AN APPLICATION

For a SoftMotion application, it is necessary to create a specific **POU** that will be used for the axis motion.

- Use the settings presented in Section 2.4 as a basis.
- On the device tree, right-click **Application > Add Object > POU...**
- Create a **Program POU** with the name **MyMotion**.
- In the **Implementation language** field, select **Structured Text (ST)**.
- Click **Add**.

Figure 3.1 shows the previous steps directly through Codesys.

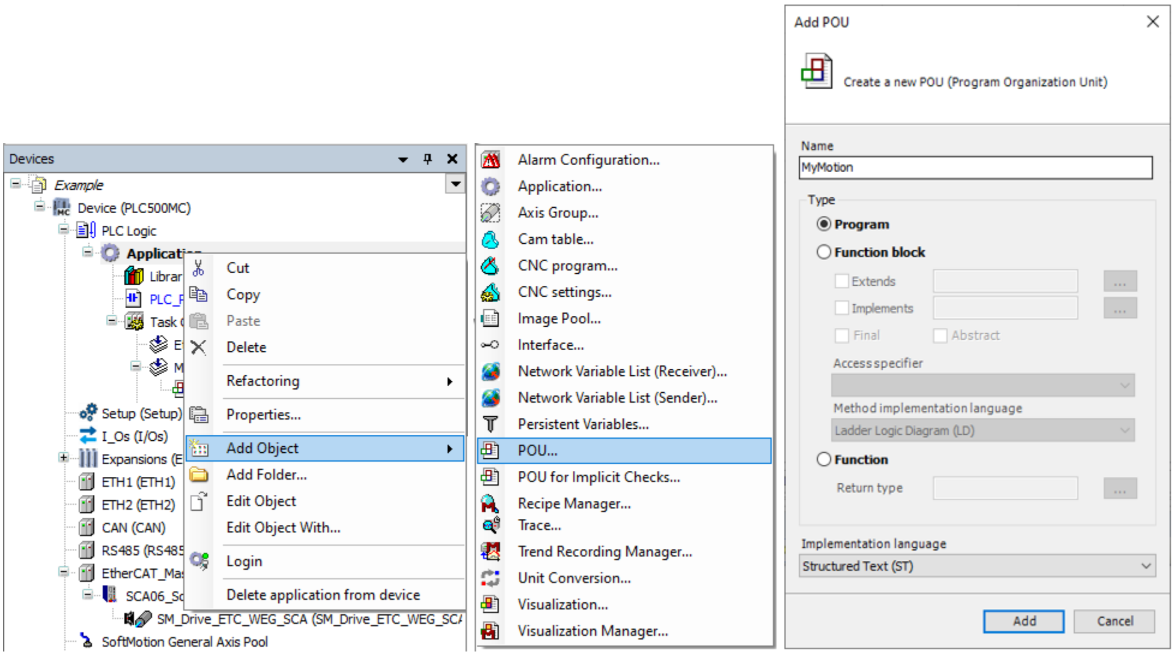


Figure 3.1: *Create SoftMotion POU.*

This **POU** must be called under the **EtherCAT_Task** task.

NOTE!
All function blocks related to axis motion must be declared and called in *EtherCAT_Task*. Other functionalities must be used in different tasks, with lower priority.

- Drag **POU MyMotion** under the **EtherCAT_Task**, as shown in Figure 3.2 task.

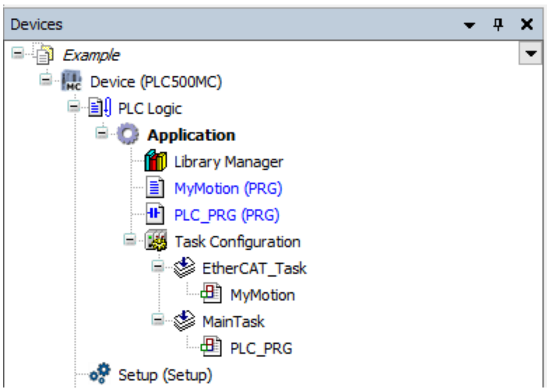


Figure 3.2: *Add MyMotion POU to the* ***EtherCAT_Task*** *task.*

- Open **POU MyMotion**.
- Create a **MC_Power** instance and another **MC_MoveRelative** and reference the **Axis** input of the function blocks to the name of the axis created, as shown in Figure 3.3.

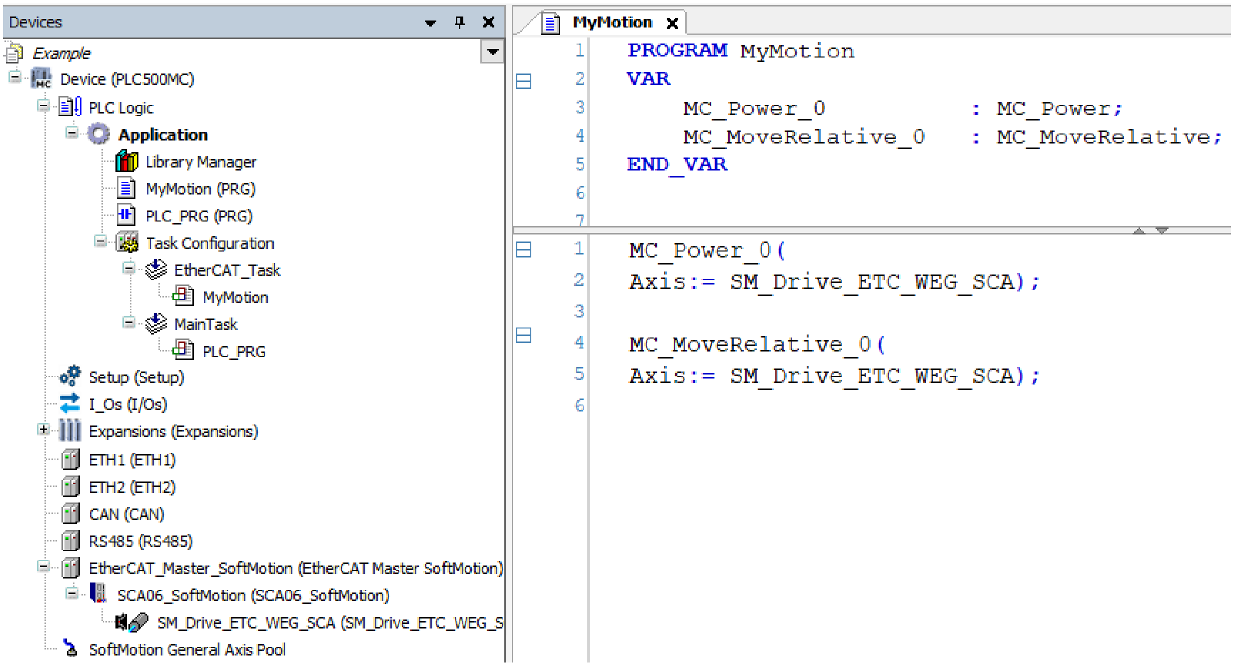


Figure 3.3: *Add the* ***MyMotion*** *POU to the* ***EtherCAT_Task*** *task.*

MyMotion Application:

```
PROGRAM MyMotion
VAR
   MC_Power_0            : MC_Power;
   MC_MoveRelative_0     : MC_MoveRelative;
END_VAR
```

```
MC_Power_0(
   Axis:= SM_Drive_ETC_WEG_SCA);

MC_MoveRelative_0(
   Axis:= SM_Drive_ETC_WEG_SCA);
```

3.2 CREATE VISUALIZATION

> **NOTE!**
> The **SM3_Basic** library has several built-in visualization templates that can be used to test the functionality of a function block in a simplified way.

- Add a **Visualization** object to the device tree, as shown in Figure 3.4.

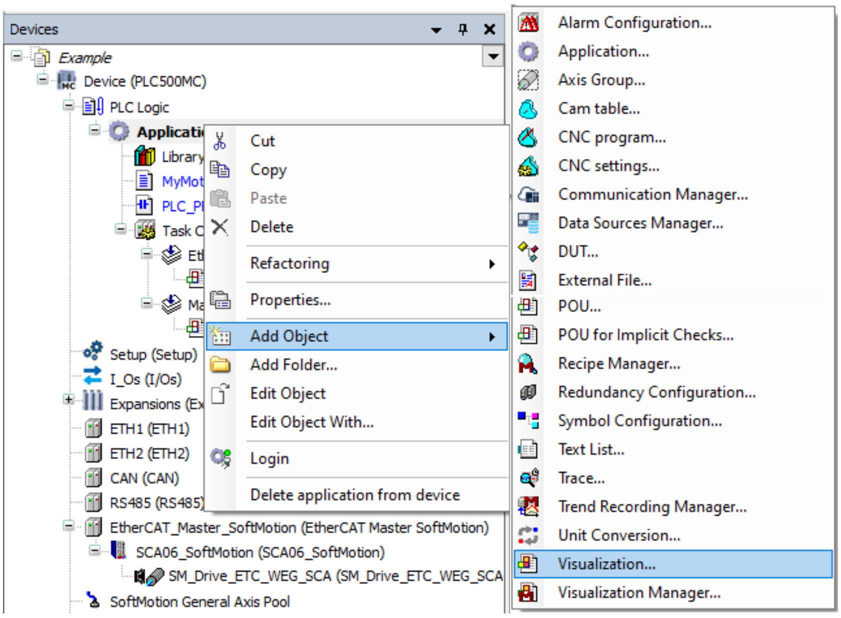


Figure 3.4: *Add a* ***Visualization*** *object.*

When adding a **Visualization** object, a dialog will pop up, as shown in Figure 3.5.

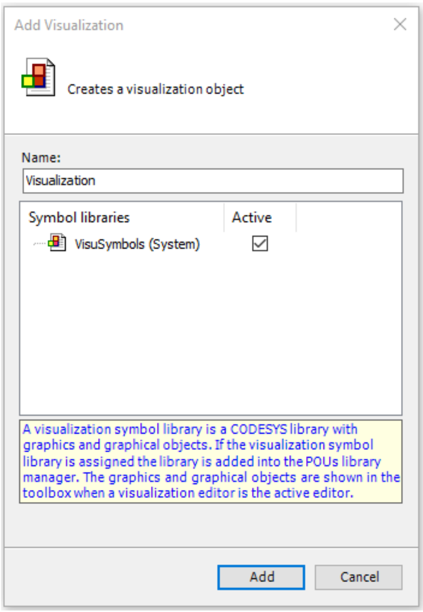


Figure 3.5: *Dialog opened when adding a* ***Visualization*** *object.*

- Check **Active** and click **OK**.
- Open the **Visualization** object created.
- In the **Visualization Toolbox**, located on the right side of the screen, select the **SM3_Basic** tab. In the search field, enter **MC_Power** and select the **VISU_NEW_MC_Power**, as shown in Figure 3.6.

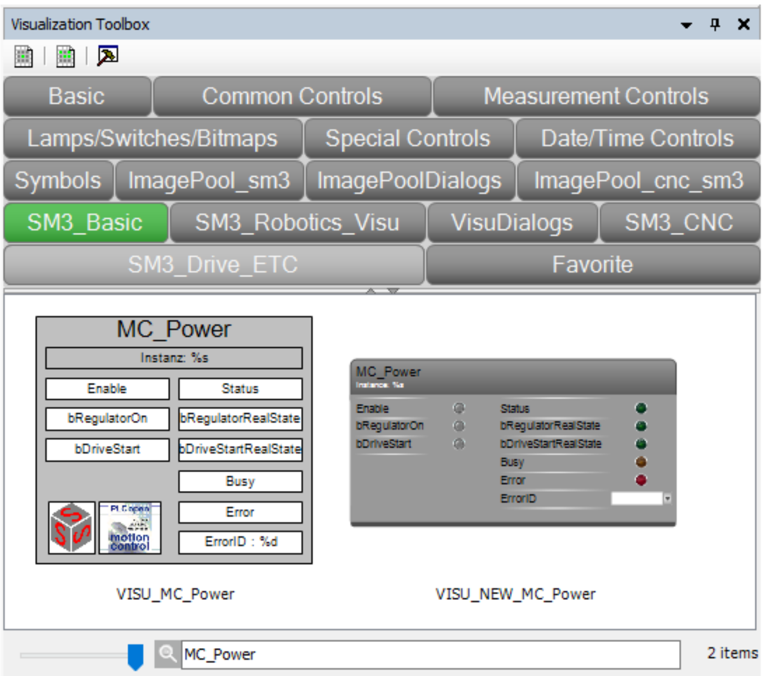


Figure 3.6: *Search for the **MC_Power** visualization template.*

- Drag and drop the template onto the visualization.

When dropping the object, the **Assign parameters** dialog will pop up for the visualization template.

- Double-click **Value** and click **...**.

By doing so, a new dialog, **Input Assistant**, will pop up.

- Search for the **MC_Power_0** function block instance created in the **MyMotion** POU and click **OK**.

Figure 3.7 shows the previous steps directly through Codesys.

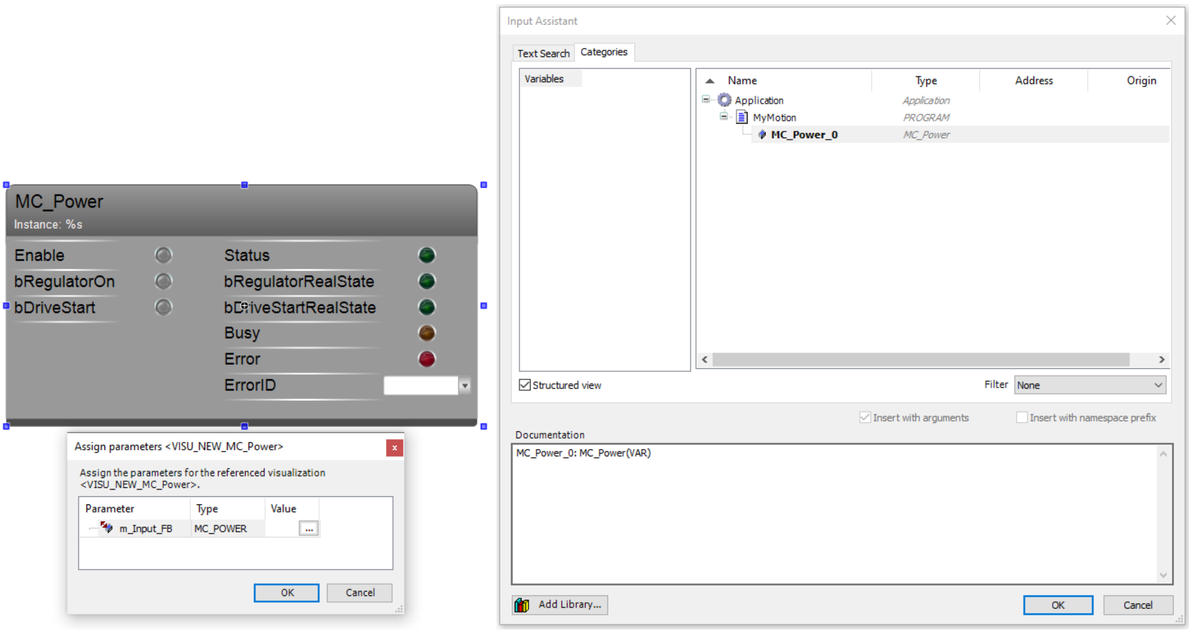


Figure 3.7: *Add a **Visualization** object.*

Hence, the inputs and outputs of the visualization template are automatically mapped to the function block instance.

NOTE!
Another way to map the visualization template to the created function block is to select the template and use the **Properties > References > m_Imput_FB** tab.

- Follow the same procedure now using the **VISU_NEW_MC_MoveRelative** template, referencing the **MC_MoveRelative_0** function block instance.

After the configuration, the visualization page should contain these two visualization templates mapped in the previously created function blocks, as in Figure 3.8.

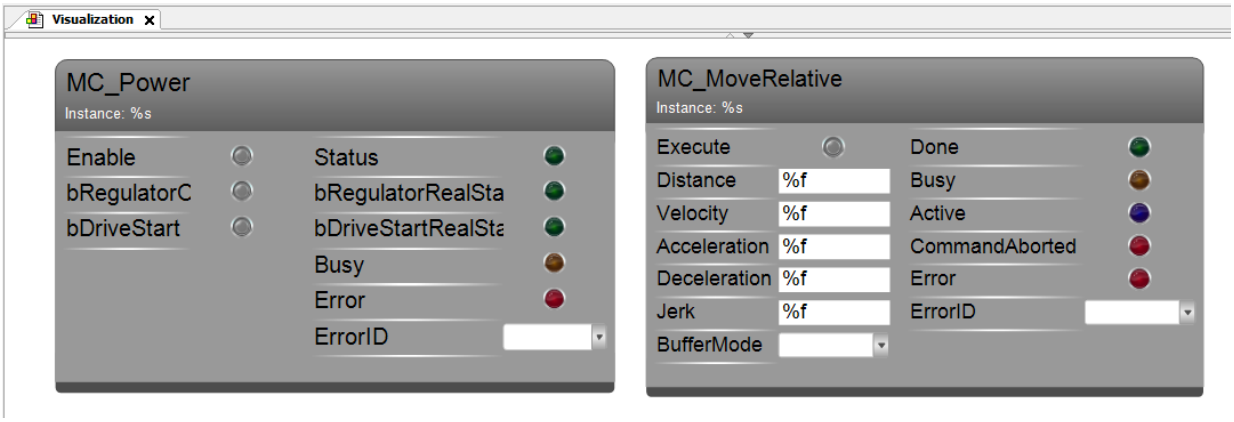


***Figure 3.8:** Add a **Visualization** object.*

Through these templates it will be possible to control the servomotor shaft.

- Download the program to the **PLC500MC**.
- In the **Online** monitoring mode, open the **Visualization** object.

ATTENTION!
You can move the shaft using buttons on this page. The shaft can perform unexpected motions if the settings are not correct. Therefore, take all necessary safety precautions.

- In the **MC_Power** template, click the **bDriveStart**, **bRegulatorOn** and **Enable** buttons, respectively.

Observe the block outputs; they will show the status of the servo drive. For the correct drive, the **Status**, **bRegulatorOnRealState** and **bDriveStartOnRealState** outputs should be green, as in Figure 3.9, indicating the **servo drive** is enabled for the motion.

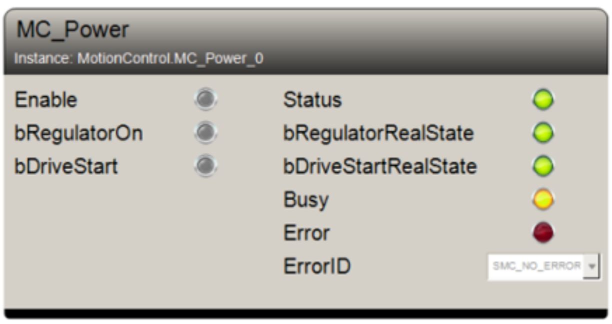


***Figure 3.9:** Servomotor enabled example.*

- In the **MC_MoveRelative** template, set the variables related to the motion (**Distance**, **Velocity**, **Acceleration**, **Deceleration** and **Jerk**), as shown in Figure 3.10. After that, click the **Execute** button to start the motion.

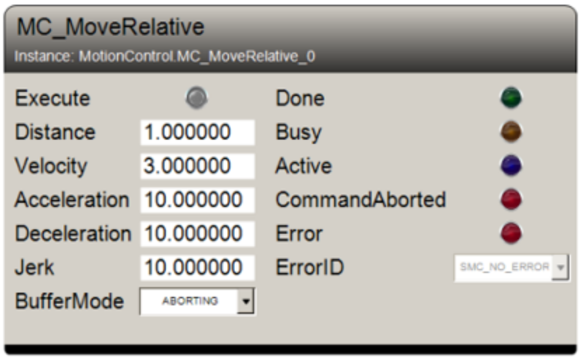


Figure 3.10: *Example of relative motion setting for the servomotor.*

If you wish, perform some more tests.

Other application examples can be found directly on the Codesys website, available at: https://help.codesys.com (Add-ons > CODESYS SoftMotion > Application Examples).

4 ADDITIONAL INFORMATION ON THE ETHERCAT NETWORK

This chapter contains some additional information and advanced settings used on the EtherCAT network.

4.1 ASSIGNING A STATIC ADDRESS TO THE SCA06 ON THE ETHERCAT NETWORK

It is possible to define a static address for the **SCA06** as a slave on the EtherCAT network using an internal EEPROM memory of the **ECO4** accessory.

- Use the settings presented in Section 2.4 as a basis.
- Open the **SCA06_Motion** settings, and on the **General** tab, enable the **Expert settings**. After that, several advanced settings will become available.
- In the **Identification** field, check **Configured statio alias (ADO 0x012)**, as shown in Figure 4.1.

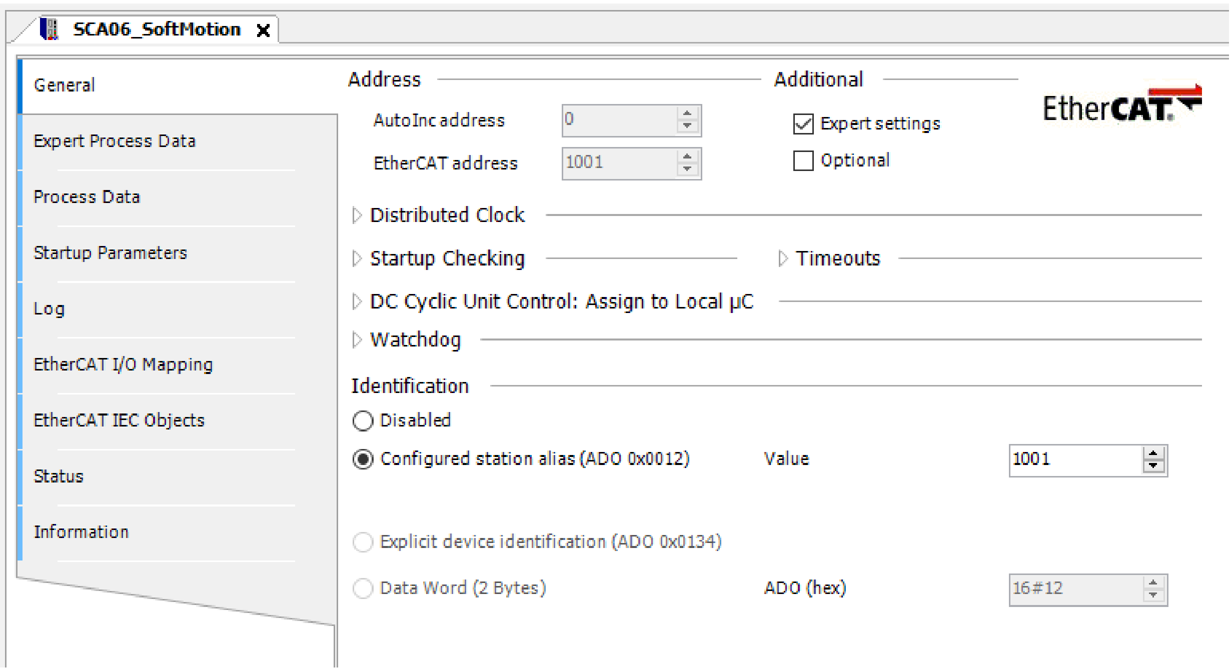


Figure 4.1: *Enabling identification.*

- After applying the settings from this section, download the program to the **PLC500MC** and do the monitoring in the **Online** mode.

In the **Online** mode, when establishing communication with the **SCA06_Motion**, in the **Identification** field, the **Actual address** variable will appear informing the actual value of the address. The **Write to EEprom** option will also be available, as in Figure 4.2.

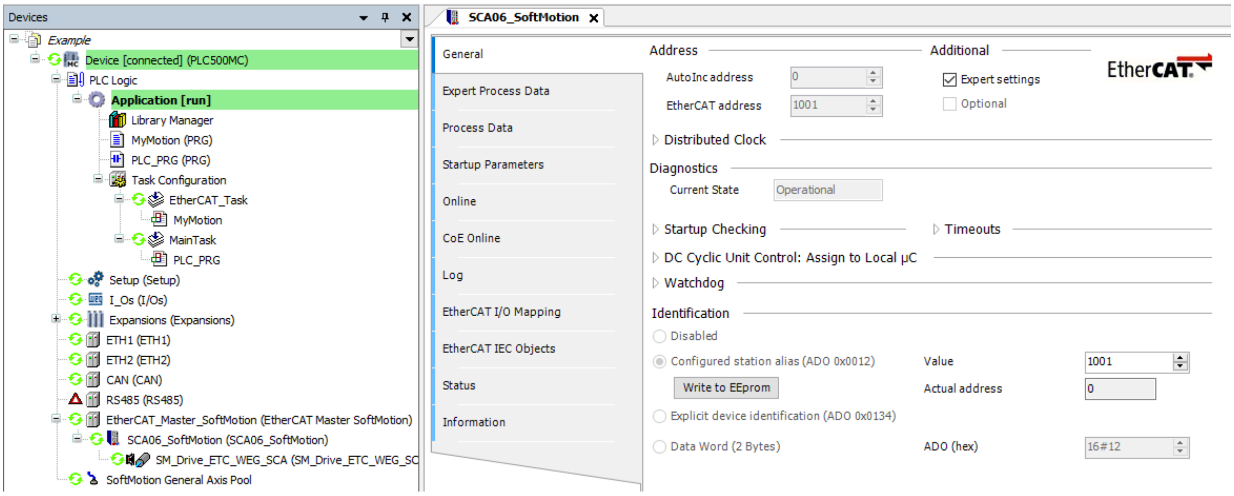


Figure 4.2: *Current address in EEPROM memory.*

- Enter the desired address into the **Value** field and click on **Write to EEprom**, as shown in Figure 4.3.

Figure 4.3: *Writing a new address to EEPROM memory.*

A message, as in Figure 4.4, will appear on the screen requesting that the servo drive be restarted to apply the new network address.

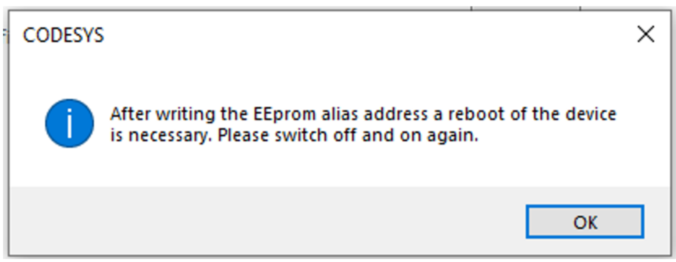


Figure 4.4: *Warning message to apply the EEPROM write.*

- Restart the servo drive.
- Exit the **Online** mode, and in the **Additional** field, select **Optional**. In the **Configured statio alias (ADO 0x012)** field, as shown in Figure 4.5. Make sure the address is the same as the one written to the EEPROM memory earlier.

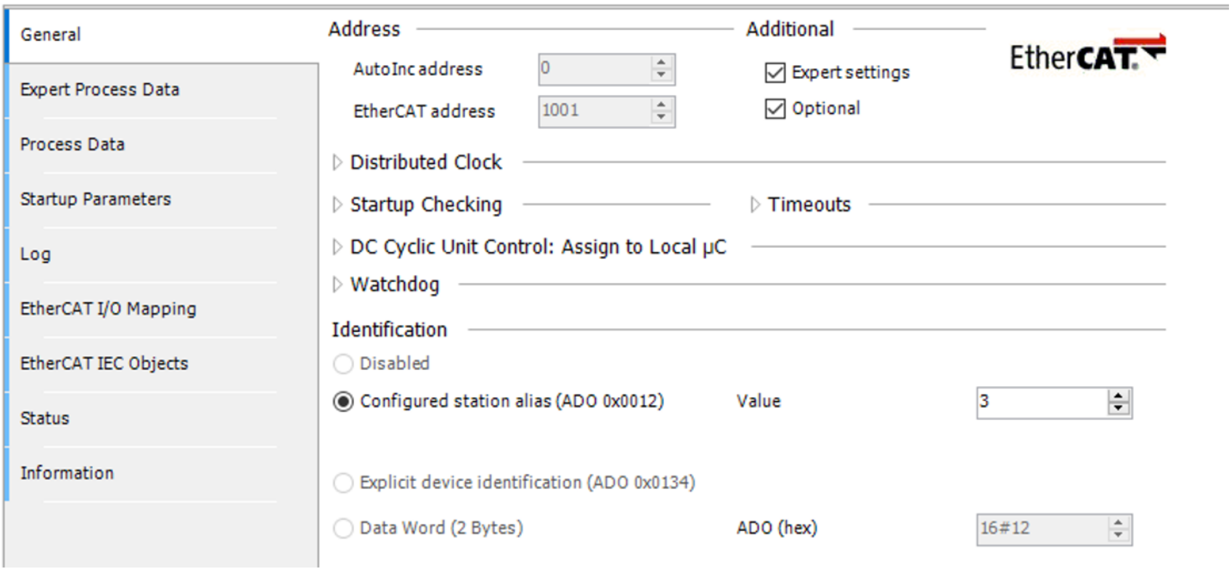


Figure 4.5: *Enable the Optional field.*

- Download the program to the **PLC500MC** and do the monitoring in the **Online** mode.

Note that now the current address will be the address written to the EEPROM of the device, as shown in Figure 4.6.

Figure 4.6: *New EEPROM address.*

NOTE!
To use the address written to the EEPROM memory in a network with more than one SCA06, it is necessary that the devices be marked with the **Optional** option; otherwise, the network will be configured automatically by the network master without using the EEPROM memory address.

4.2 READING AND EDITING PARAMETERS IN THE SCA06 THROUGH THE ETHERCAT NETWORK

Using this method, it is possible to modify configuration parameters of the **SCA06** remotely through the EtherCAT network, without having to use your HMI.

- Use the settings described in Section 2.4.
- In the **Online** mode, open the EtherCAT slave settings (**SCA06_SoftMotion**), and on the **General** tab, enable **Expert setting**.
- Open the **CoE Online** tab and select **Auto Update**, as shown in Figure 4.7.

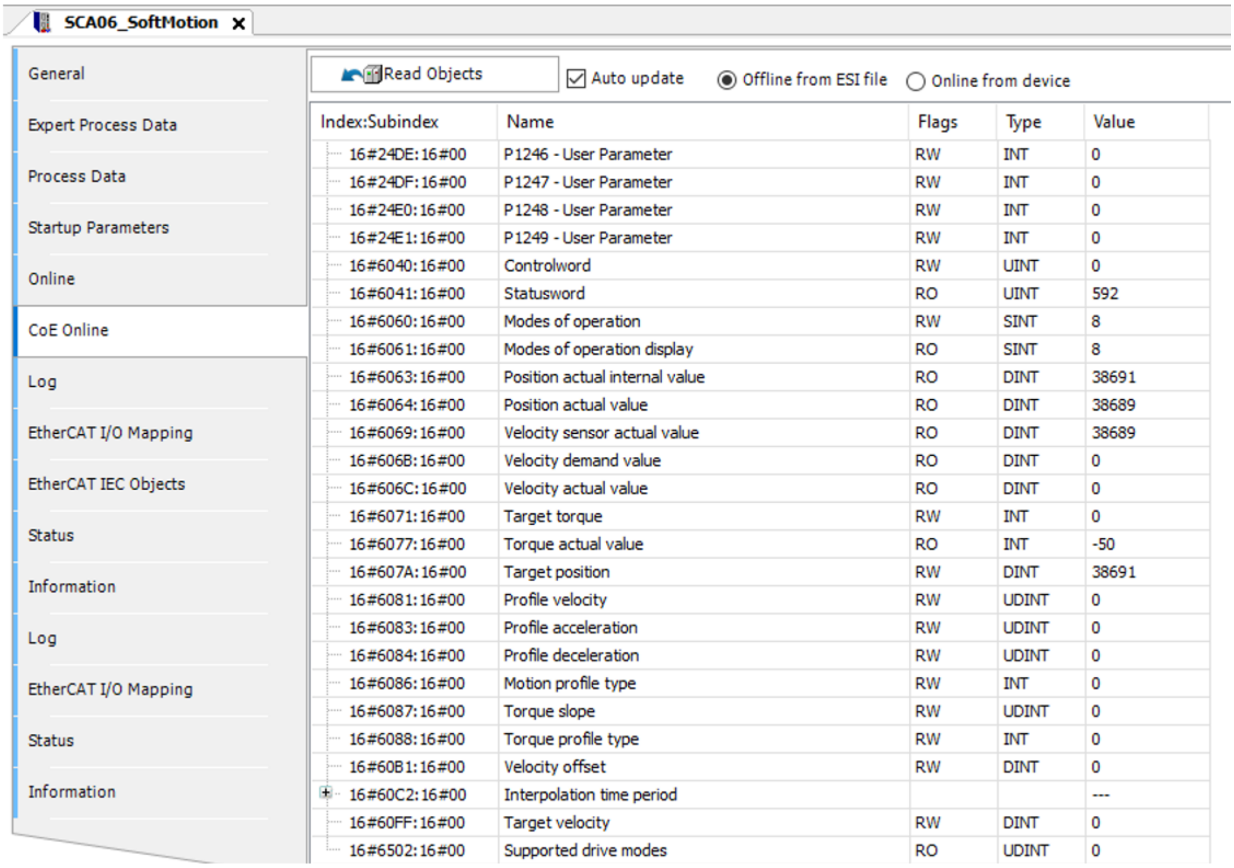


Figure 4.7: *Online CoE*

Operating elements	Description
Read Objects	The object directory is read once.
Auto update	Objects are read in cycles.
Offline from ESI file	The tab shows the contents of the device description object directory.
Online from Device	The tab shows the contents of the device object directory (not available for SCA06).
Flags	**RO:** The value is write-protected. **RW:** The value can be modified.
Value	You can double-click the text field to edit this value. The new value will be written directly to the SCA06.

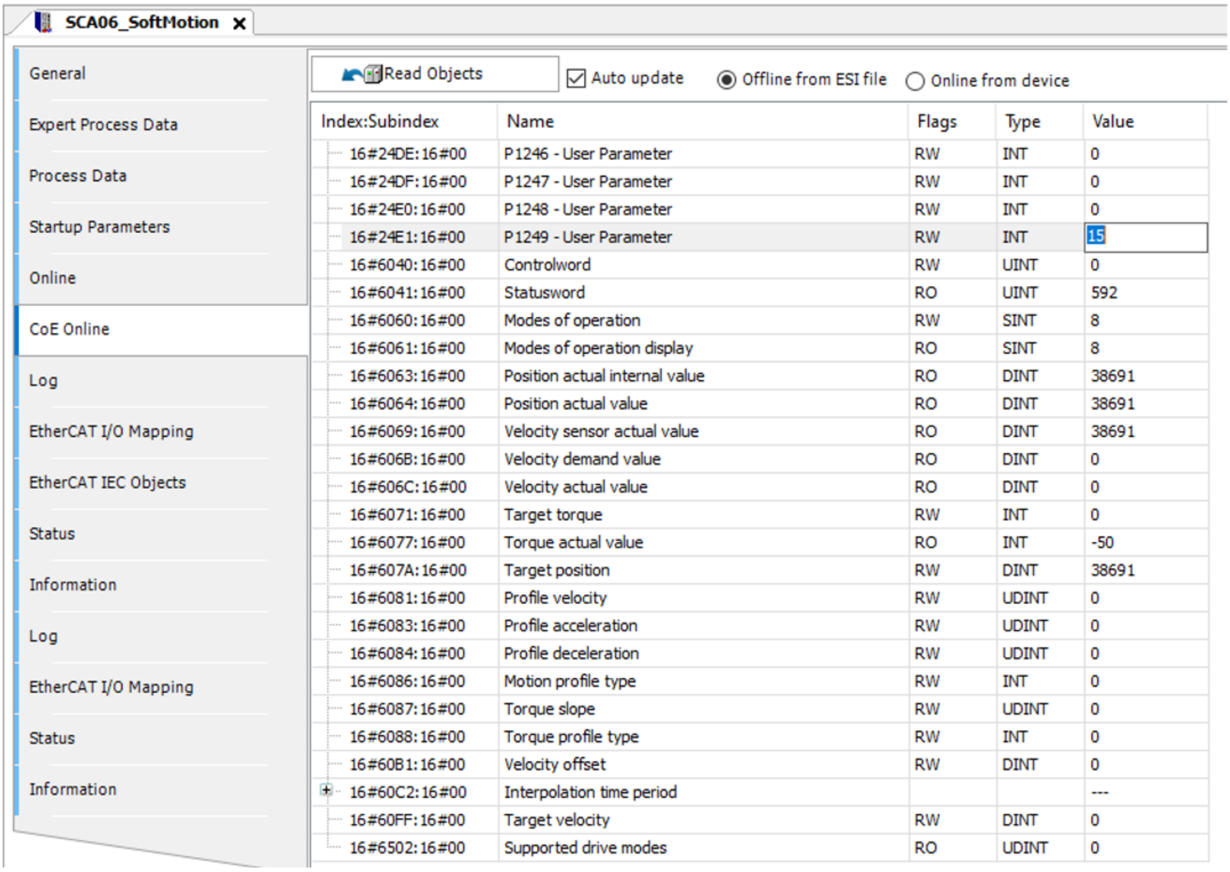
Table 4.1: *Online CoE Elements.*

On this tab, you can read and modify some internal parameters of the SCA06 servo drive.

Carry out a test by changing parameter P1249 to value 15.

- Find the **P1249 - User Parameter** in the list, double-click the **Value** field, enter **15** and press **Enter**, as shown in Figure 4.8.

NOTE!
The variables will be updated in cycles, wait for the parameters to be read (this may take some time).



Index:Subindex	Name	Flags	Type	Value
16#24DE:16#00	P1246 - User Parameter	RW	INT	0
16#24DF:16#00	P1247 - User Parameter	RW	INT	0
16#24E0:16#00	P1248 - User Parameter	RW	INT	0
16#24E1:16#00	P1249 - User Parameter	RW	INT	15
16#6040:16#00	Controlword	RW	UINT	0
16#6041:16#00	Statusword	RO	UINT	592
16#6060:16#00	Modes of operation	RW	SINT	8
16#6061:16#00	Modes of operation display	RO	SINT	8
16#6063:16#00	Position actual internal value	RO	DINT	38691
16#6064:16#00	Position actual value	RO	DINT	38691
16#6069:16#00	Velocity sensor actual value	RO	DINT	38691
16#606B:16#00	Velocity demand value	RO	DINT	0
16#606C:16#00	Velocity actual value	RO	DINT	0
16#6071:16#00	Target torque	RW	INT	0
16#6077:16#00	Torque actual value	RO	INT	-50
16#607A:16#00	Target position	RW	DINT	38691
16#6081:16#00	Profile velocity	RW	UDINT	0
16#6083:16#00	Profile acceleration	RW	UDINT	0
16#6084:16#00	Profile deceleration	RW	UDINT	0
16#6086:16#00	Motion profile type	RW	INT	0
16#6087:16#00	Torque slope	RW	UDINT	0
16#6088:16#00	Torque profile type	RW	INT	0
16#60B1:16#00	Velocity offset	RW	DINT	0
16#60C2:16#00	Interpolation time period			---
16#60FF:16#00	Target velocity	RW	DINT	0
16#6502:16#00	Supported drive modes	RO	UDINT	0

Figure 4.8: *Editing parameters Online*

By doing so, the value will be modified.

- Check the writing of this value directly on the HMI of the **SCA06**, in parameter **P1249**.

4.3 EDITING PDOS ON THE ETHERCAT NETWORK

It is possible to edit the **PDOs** defined as the default of the **SCA06** in the EtherCAT communication.

- Use the settings described in Section 2.4.
- Open the **SCA06_Motion** settings, and on the **General** tab, enable **Expert settings**. After that several advanced settings will be available, as shown in Figure 4.9.

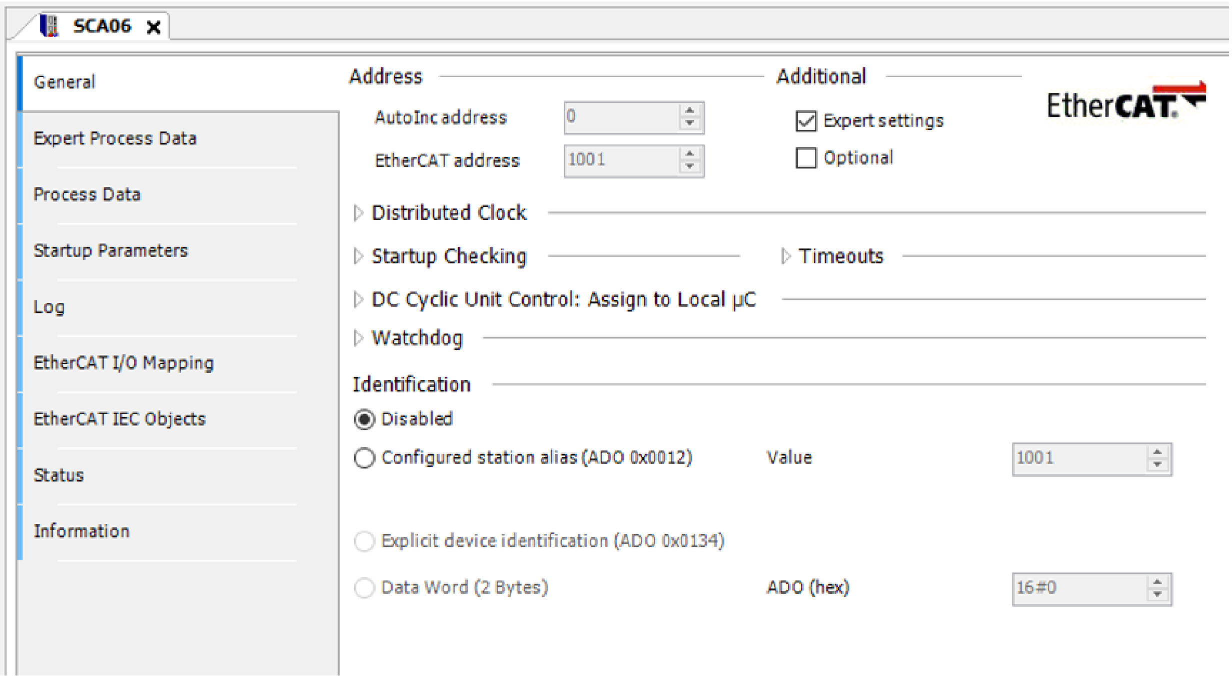


Figure 4.9: *Enabling SCA06_SoftMotion advanced settings.*

- Access the **Expert Process Data** tab, where you can modify the communication **PDOs**.

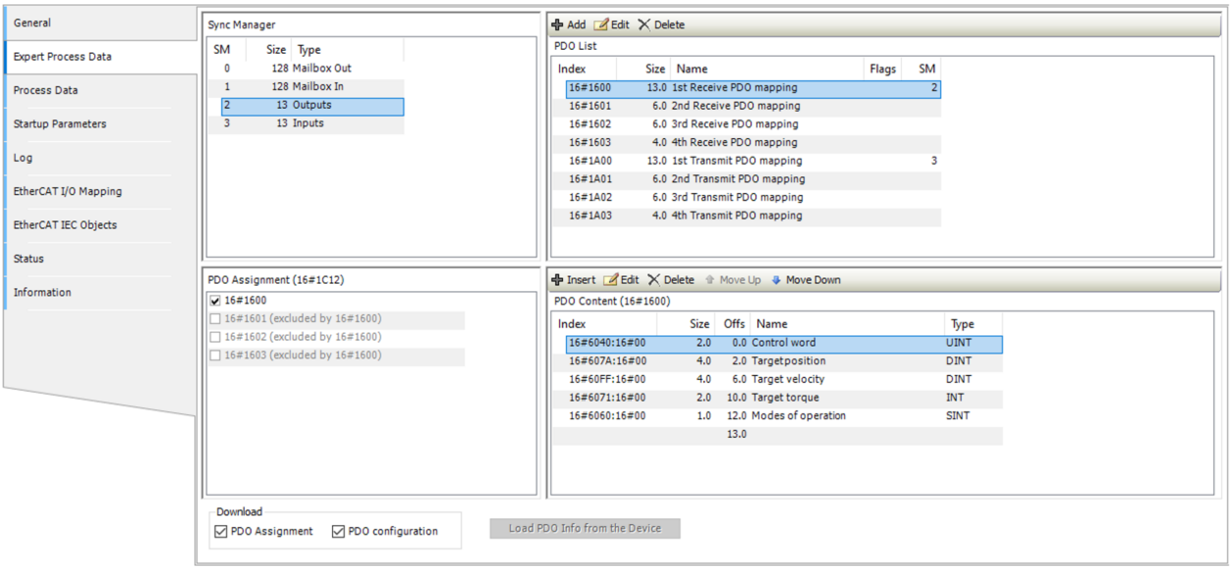


Figure 4.10: *Edit EtherCAT network PDOs.*

- Select the **PDO** that you want to modify in the **PDO List** field and then edit it in the **PDO Content** field.
- To apply the new **PDOs** settings, make sure that, in the **Download** field, **PDO Assignment** and **PDO configuration** are selected.

By using this procedure when you **Download** the program to the **PLC500MC** and start the EtherCAT communication with the SCA06 servo drive, the list will be automatically modified for the new PDOs settings.

4.4 CONFIGURING ETHERCAT REDUNDANCY

It is possible to configure an EtherCAT network with redundancy using the PLC500MC.

NOTE!
ETH1 and **ETH2** are independent ports; therefore, it is not possible to carry out EtherCAT ring communication. However, it is possible to carry out EtherCAT ring communication with redundancy.

- Use the settings described in Section 2.4.
- Open the **EtherCAT Master SoftMotion** device settings.
- Check **Redundancy**.
- In the **Redundancy EtherCAT NIC Settings** field, check **Select network by name**.
- In the **Network name** field, enter **ETH1**.

Figure 4.11 shows the previous settings already made.

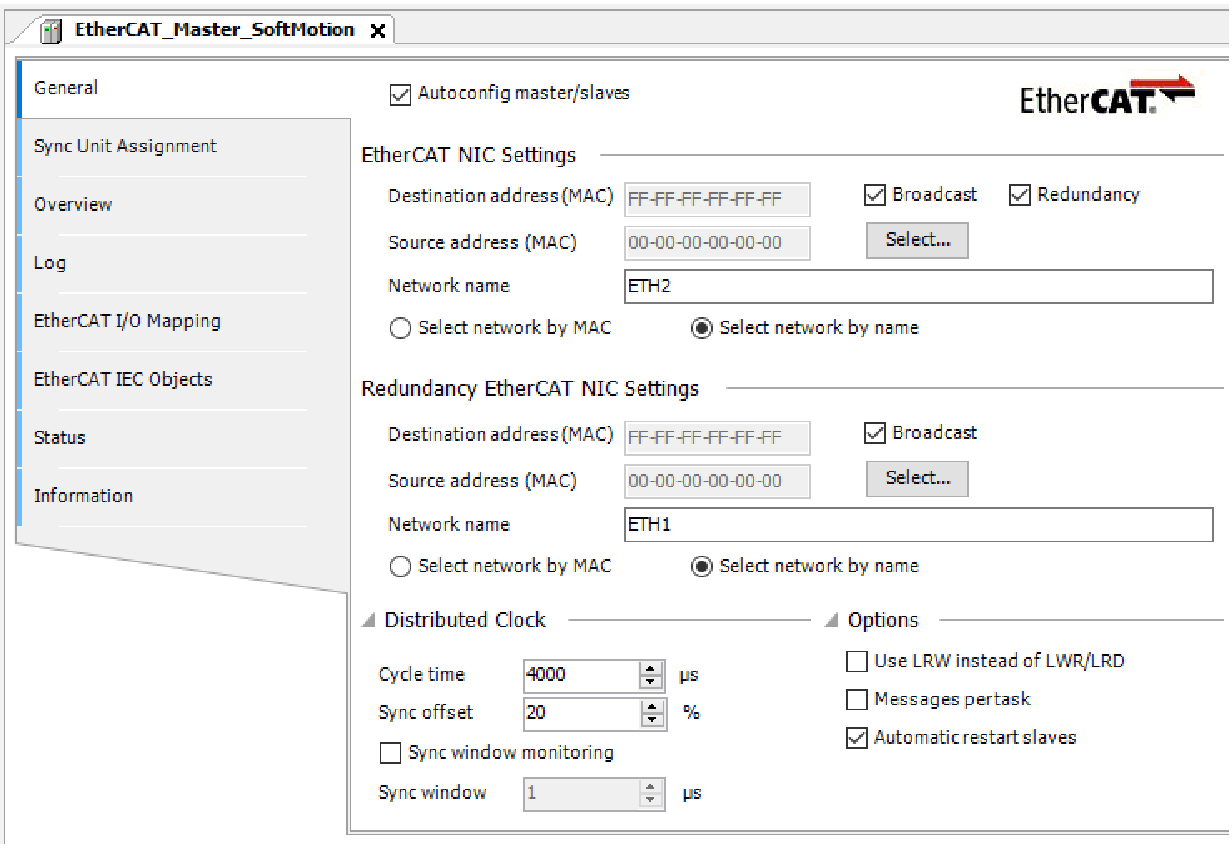


Figure 4.11: *Configuring EtherCAT redundancy.*

By doing so, the redundancy of the network is already configured and ready to be used.

4.5 XML FILE

Each device on an EtherCAT network has an XML configuration file that contains information about the operation of the device on the EtherCAT network, as well as a description of all existing objects for communication. In general, this file is used by a master or configuration software for programming the devices present on the network.

ATTENTION!
It is possible to add EtherCAT slave devices to the Codesys software using XML files. However, for motion control it is recommended that devices already installed and specific for this purpose be used. You can add a generic axis following CiA402, but some SoftMotion functionality may not be available.

5 SOFTMOTION ADDITIONAL INFORMATION

This chapter contains some additional information and advanced settings used for motion control.

5.1 TASK PRIORITY

Motion control needs a high priority for correct operation, thus correct task priority setting is essential.

1 must be set for the task responsible for motion control. When an EtherCAT master device is added, it will create a task (with priority 1) automatically. The application responsible for motion control must run under this task.

Other applications, in addition to motion control or that use great processing power must be executed in a different task with lower priority. Priority 10 or lower (10 - 31) is recommended for such tasks.

NOTE!
The lower the number, the higher its priority, with 0 being the highest priority and 31 the lowest priority.

Figure 5.1 shows an example of task configuration, where motion control is performed in the **MyMotion** program, and the other functionalities are executed in the **PLC_PRG** program.

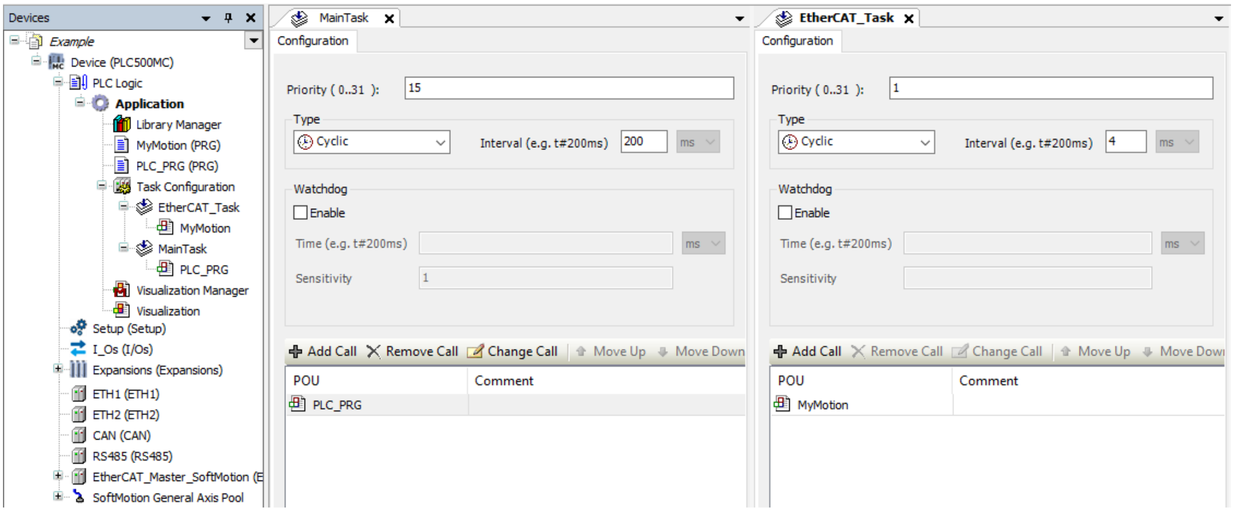


Figure 5.1: *Editing task priority.*

5.2 SCALE SETTINGS FOR SM_DRIVE_ETC_WEG_SCA

This subsection describes the possible scale settings applied to the **SM_Drive_ETC_WEG_SCA** axis.

- Use the settings presented in Section 2.4 as a basis.
- Open the **SM_Drive_ETC_WEG_SCA** settings on the **Scaling/Mapping** tab.

You can choose two types of motors for the scale settings in the **Motor Type** field. Depending on the type of motor selected in the Scaling field, the available settings will be different.

5.2.1 Motor Type: Rotary

Rotary: Generally used for rotary axis settings, as it has a more complete configuration, being able to add gear or pulleys ratios. Figure 5.2 shows a configuration example using the **Motor Type: Rotary**.

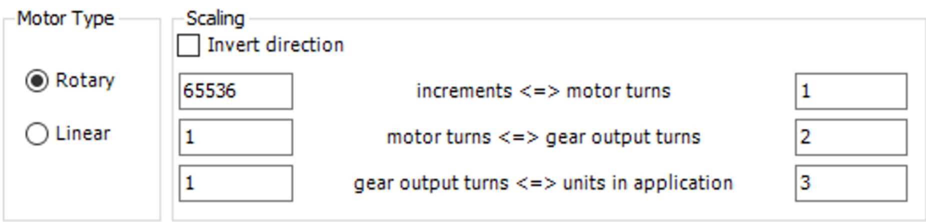


Figure 5.2: *Configuration example using Motor Type: Rotary.*

Each value of the **Scaling** field can be changed according to the mechanics involved in the application.

Operating elements	Description
increments <=> motor turns	Number of increments that correspond to a given number of motor turns.
motor turns <=> gear output turns	Number of motor turns that correspond to a given number of turns at the gear output.
gear output turns <=> units in application	Number of turns at the gear output that correspond to application units.

Table 5.1: *Scaling Elements.*

For this configuration, each application unit will be equivalent to 1/6 of a turn of the servomotor.

5.2.2 Motor Type: Linear

Linear: Generally used for configurations of linear axes, because it has more simplified and straightforward settings. Figure 5.3 shows a configuration example using the **Motor Type: Linear**.

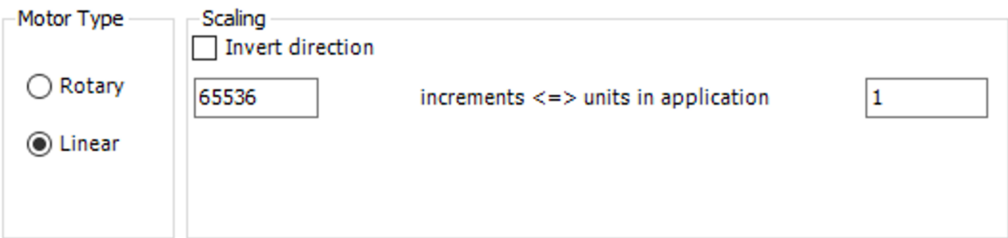


Figure 5.3: *Configuration example using Motor Type: Linear.*

Operating elements	Description
increments <=> units in application	Number of increments that correspond to application units

Table 5.2: *Scaling Elements.*

For this configuration, each application unit will be equivalent to 1 turn of the servomotor.

> **NOTE!**
> When selecting **Invert direction**, the direction of rotation will be reversed. The servo drive will receive the reference values with opposite signs.

More information about the settings can be found directly on the Codesys website, available at: https://help.codesys.com (Add-ons > CODESYS SoftMotion > Reference > User Interface > Objects > SoftMotion Drives).

5.3 ADDING VIRTUAL AXES

Virtual Drives are software emulated drives. With that, you can test your programs without connecting a piece of hardware or implement extended functionality using virtual axes.

To add a virtual axis to an application, follow the steps below.

- Right-click **SoftMotionGeneral axis pool** on the device tree and select **Add device**.
- Select the device **SoftMotionDrives > virtual drives > SM_Drive_Virtual** in the **Add Device** dialog.
- Click **Add Device**.

Figure 5.4 shows the previous steps directly from Codesys.

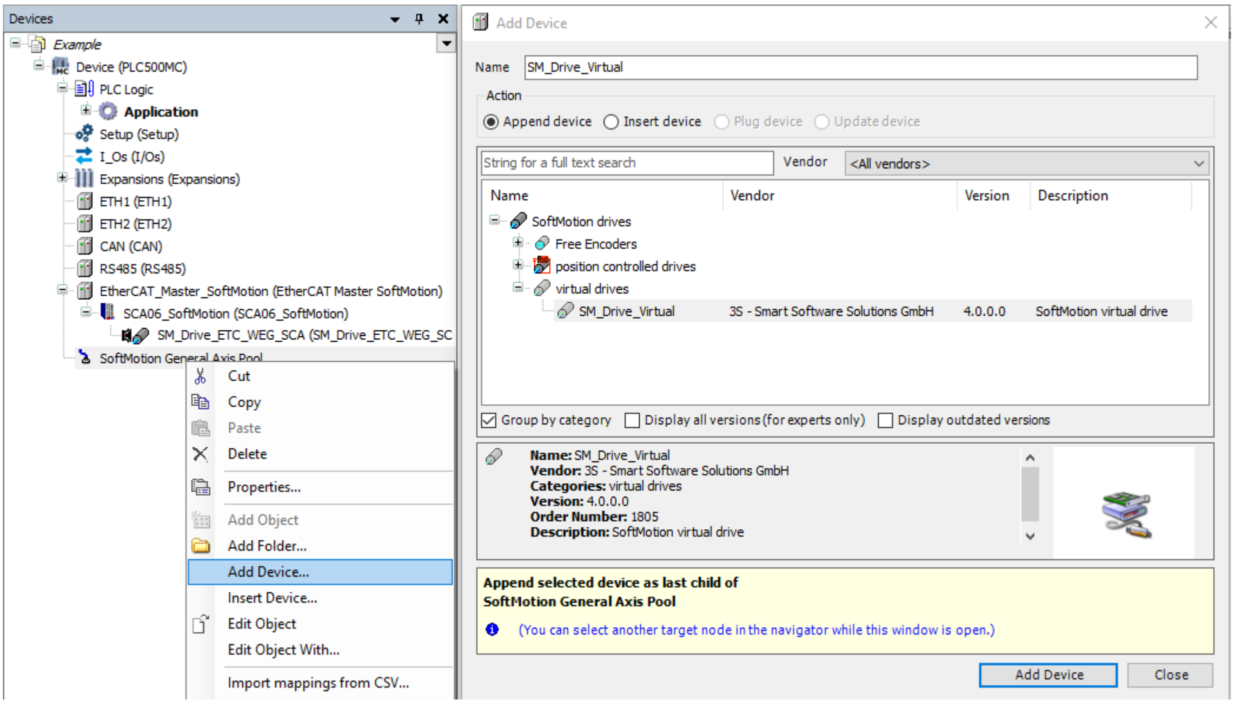


Figure 5.4: *Adding virtual axis.*

By doing so, a virtual axis will be added below the **SoftMotionGeneral axis pool** object. Figure 5.5 shows the device tree with a virtual axis added.

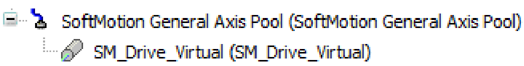


Figure 5.5: *Device tree with a virtual axis added.*

- Open the **SM_Drive_Virtual** settings.
- On the **General** tab, you can set the axis type, limits, acceleration ramp and limit dynamics.
- Set the **General** tab, according to Fig 5.6.

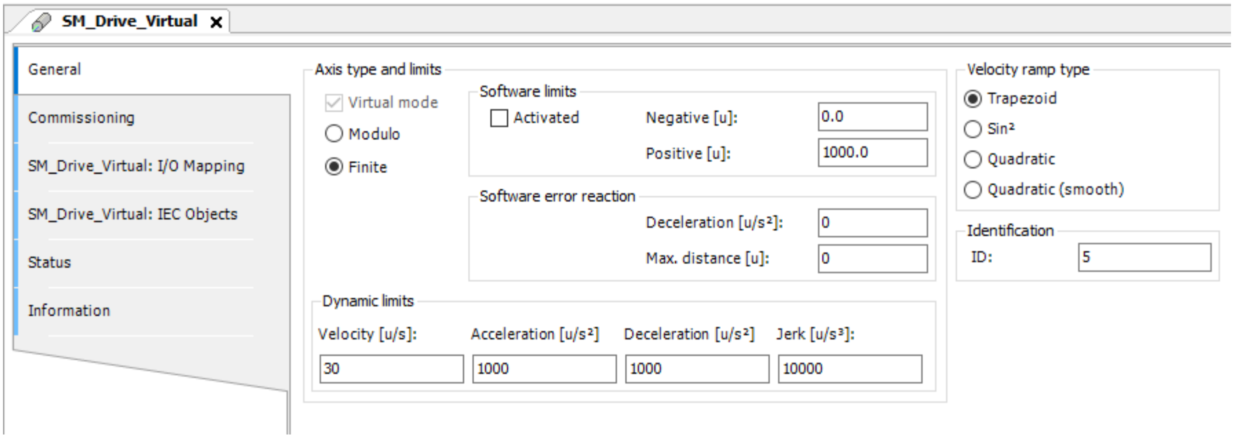


Figure 5.6: *Example of virtual axis settings.*

After these settings, the virtual axis can be used in your applications.

More information about virtual axes can be found directly on the Codesys website, available at: https://help.codesys.com (Add-ons > CODESYS SoftMotion > Reference > User Interface > Objects > SoftMotion Drives > Tab 'Logical Axes').

5.4 ADDING ENCODER AXIS

It is possible to use the two encoder inputs of the **PLC500MC** as **SoftMotion Drives**. To that end, set the **DI1** input of the **PLC500MC** as encoder (**I_Os > DI1 / Encoder1 > Pin type > Pulse/Direction** or **Quadrature**).

Figure 5.7 shows the previous steps directly through Codesys.

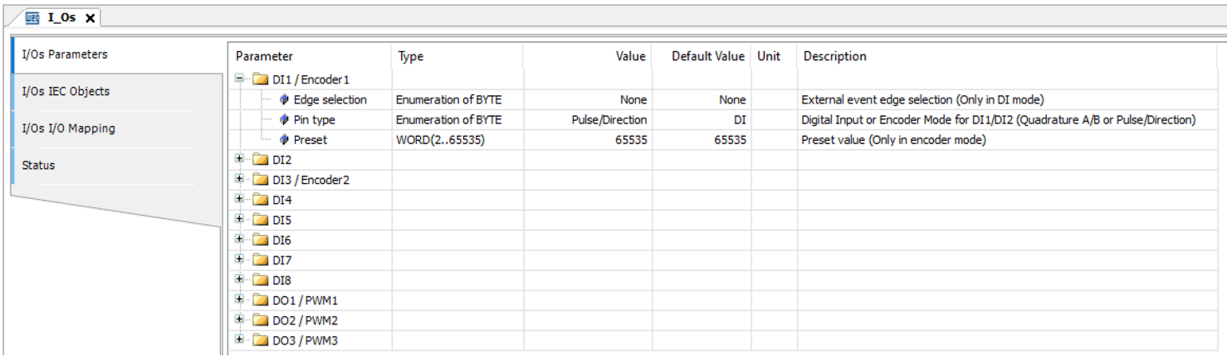


Figure 5.7: *Setting DI as encoder.*

In this way, the **DI1** and **DI2** inputs of the **PLC500MC** cease to be digital inputs and become encoder inputs.

- Use the settings presented in Section 3 as a basis.
- Right-click **SoftMotion General Axis Pool** on the device tree.
- Click **Add Device...**
- On the **Add Device** tab, in the **Action** field, select **Append device**.
- Select the device **SoftMotion drives > Free Encoder > SMC_FreeEncoder** in the dialog.
- Click **Add Device**.

Figure 5.8 shows the previous steps directly through Codesys.

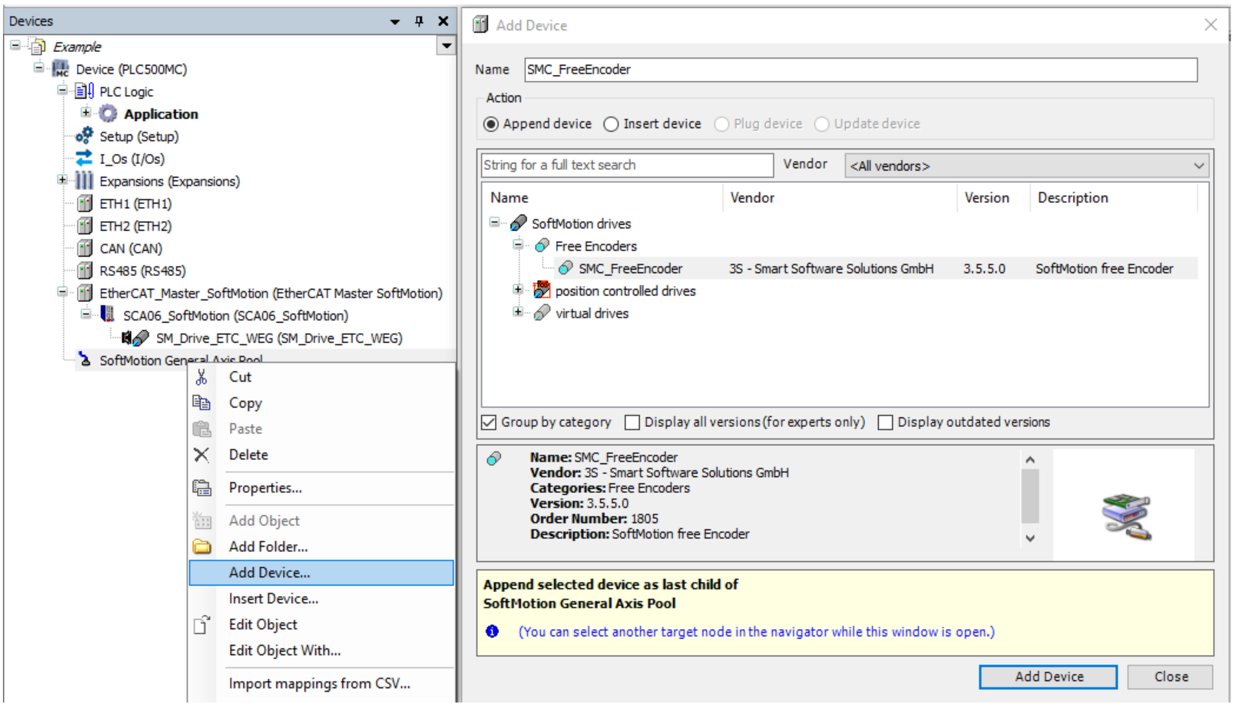


Figure 5.8: Adding encoder axis.

By doing so, a **SMC_FreeEncoder**, as in Figure 5.9 axis will be added to the device tree.

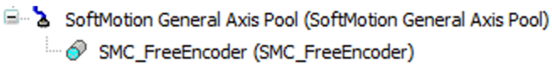


Figure 5.9: Encoder Drive added to the device tree.

- Open the **SMC_FreeEncoder** settings on the **Scaling** tab, and make the proper settings for the encoder used in your application.

Figure 5.10 shows a configuration example where every thousand pulses on the encoder will correspond to an application unit.

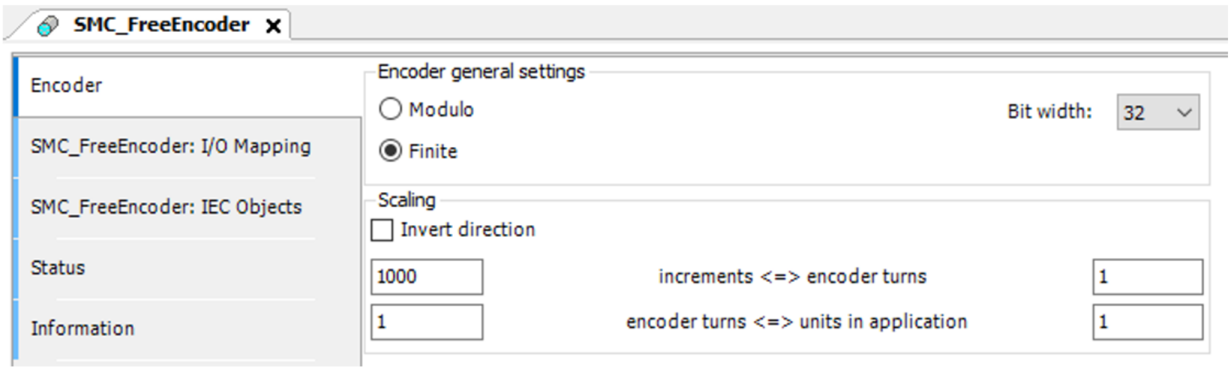


Figure 5.10: Encoder Drive added to the device tree.

For the current value of the added Drive to be updated with the value of the PLC500MC encoder input, it is necessary to assign its value to the **<FREE_ENCODER_AXIS>.diEncoderPosition** variable—this must occur in the task responsible for the motion (**EtherCAT_Task**). It is also necessary to convert the type of the variable from **LINT** to **DINT** (use the **LINT_TO_DINT() function** to do so).

The field below presents an example of the command that must be used to assign the value of **contain_Encoder1** to the **SMC_FreeEncoder** drive variable.

```
SMC_FreeEncoder.diEncoderPosition := LINT_TO_DINT(counter_Encoder1);
```

NOTE!
It is also possible to use the function blocks available in the **IoDrvGPIO (WEG)** library to update the encoder position values.

- Open the **MyMotion** POU and add the command to assign the encoder value to the **SMC_FreeEncoder**, as shown in Figure 5.11 drive.

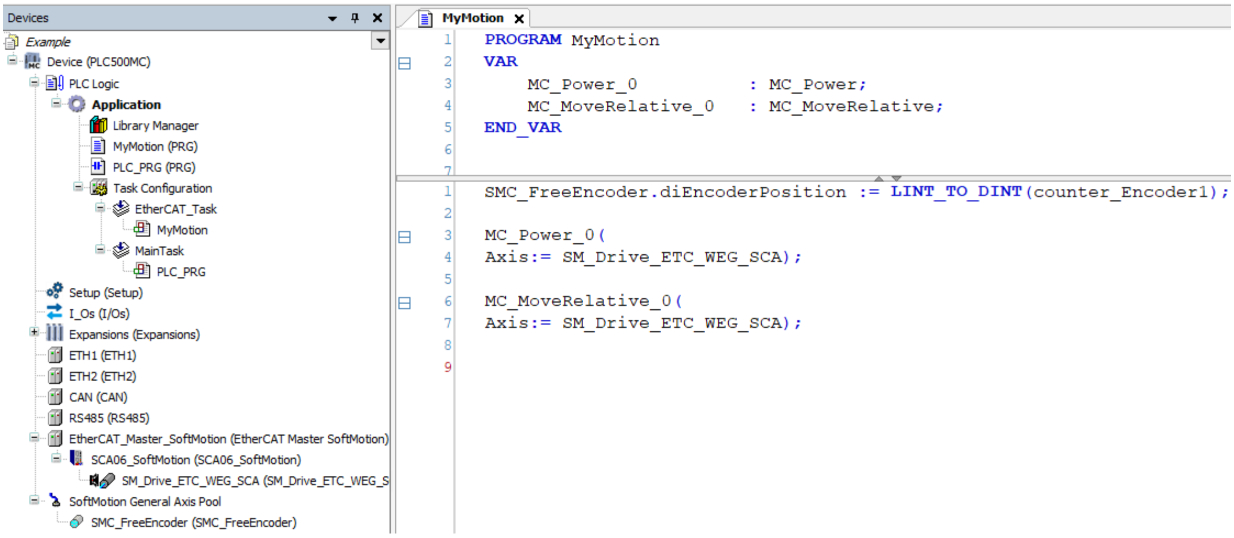


***Figure 5.11:** Adding the command to the POU associated with the motion.*

MyMotion Application:

```
PROGRAM MyMotion
VAR
   MC_Power_0            : MC_Power;
   MC_MoveRelative_0     : MC_MoveRelative;
END_VAR

SMC_FreeEncoder.diEncoderPosition := LINT_TO_DINT(counter_Encoder1);

MC_Power_0(
   Axis:= SM_Drive_ETC_WEG_SCA);

MC_MoveRelative_0(
   Axis:= SM_Drive_ETC_WEG_SCA);
```

- Connect an encoder to DI1 and DI2 inputs.
- Download the program to the PLC500MC and do the monitoring in the **Online** mode.
- Open the **SMC_FreeEncoder** settings on the **Encoder** tab, as shown in Figure 5.12.

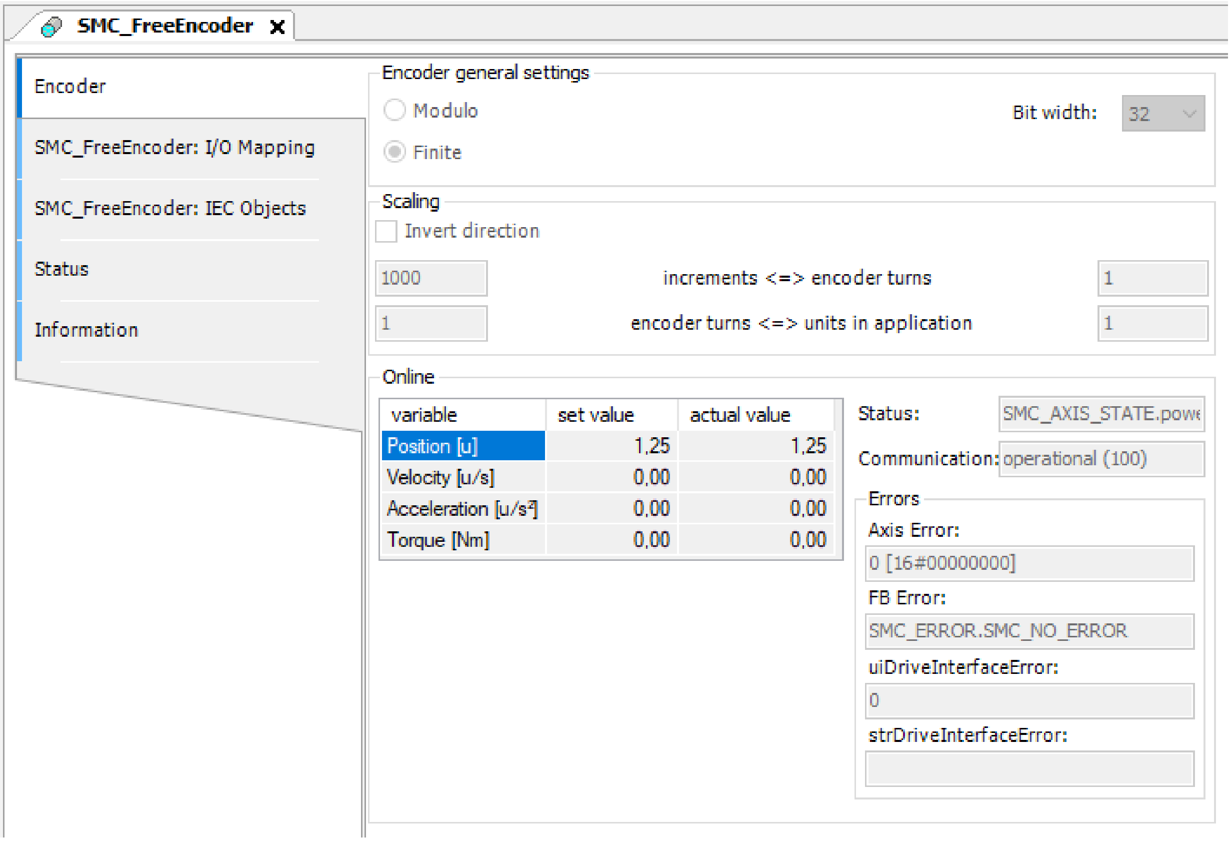


Figure 5.12: *Monitoring encoder.*

- Move the encoder axis and observe the position value changing in **Position [u] - actual value**.

More information about encoder axes can be found directly on the Codesys website, available at: https://help.codesys.com (Add-ons > CODESYS SoftMotion > Reference > User Interface > Objects > SoftMotion Drives > Tab ‘Encoder’).

5.5 CAM SYNCHRONIZATION

A cam describes the functional dependence of motion of one unit (slave) on another unit (master). The relationship is described by a continuous function (or curve) that maps a defined range of master values to slave values.

5.5.1 Creating cam application

This subsection describes the necessary settings and function blocks used to execute a cam motion using virtual axes.

- Create a new project in **File > New Project**. Select **Standard Project**, define a directory and application name (Example_Cam). Select the **PLC500MC** device and the **Continuous Function Chart (CFC)** programming language.
- On the device tree, right-click **Application > Add Object > Cam table...**
- In the opened dialog, define the name, as shown in Figure 5.13.
- Click **Add**.

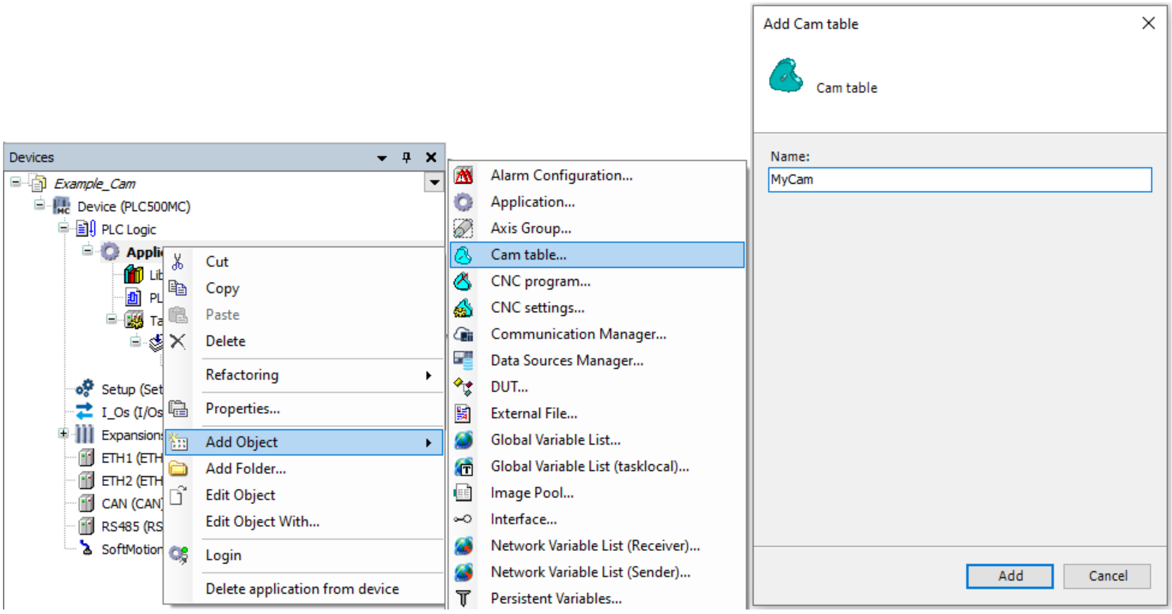


Figure 5.13: *Creating cam table.*

- Open the **MyCam** object previously created.

The Codesys software has an integrated graphical cam editor that allows the quick creation and editing of cam tables.

In this object, cam tables are defined. You can toggle between the graphical editor (**cam** tab) and the alternative table editor (**cam table** tab) any time.

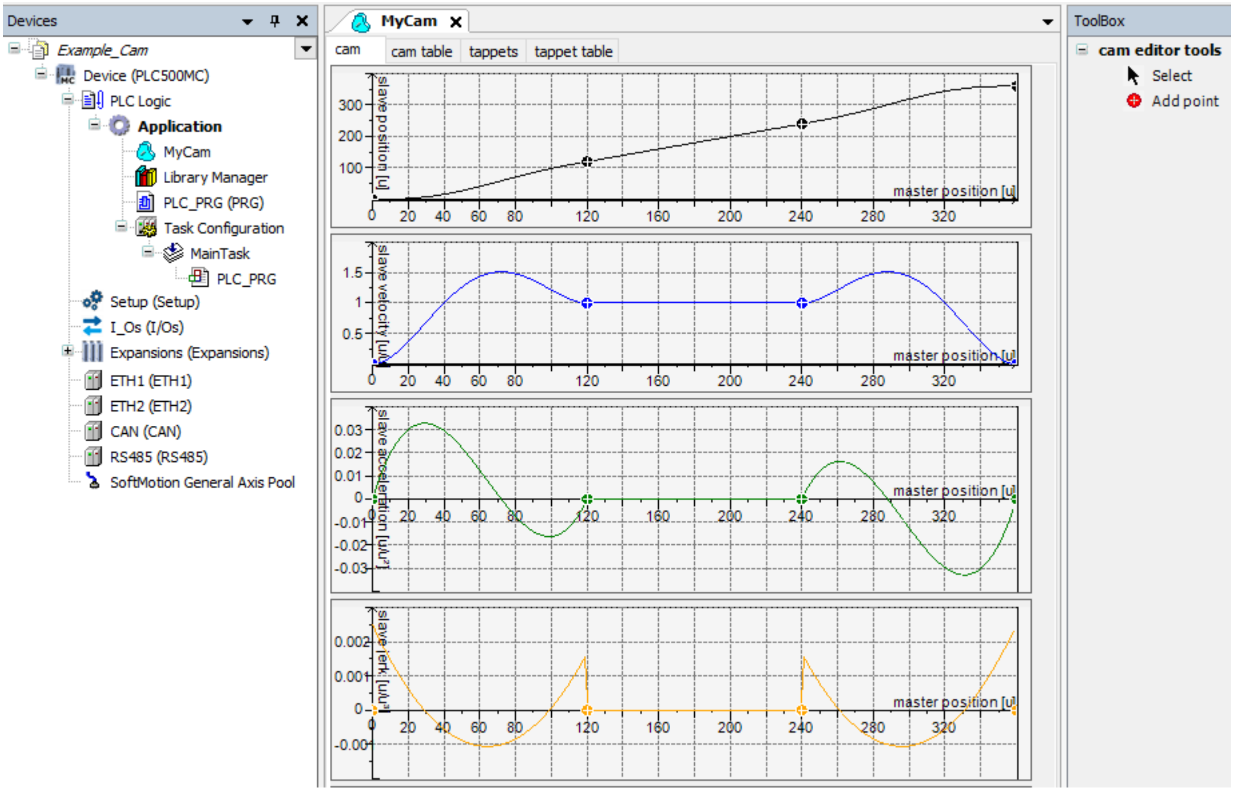


Figure 5.14: *Cam editor.*

5.5.2 Importing a cam table

In addition to creating a cam table through the editor, it is also possible to import and export these tables.

- To import or export a cam table, open the **MyCam** object.

With the object open, a new option called **Cam** is enabled in the top menu of the Codesys software; this tab contains the options for importing and exporting cam tables.

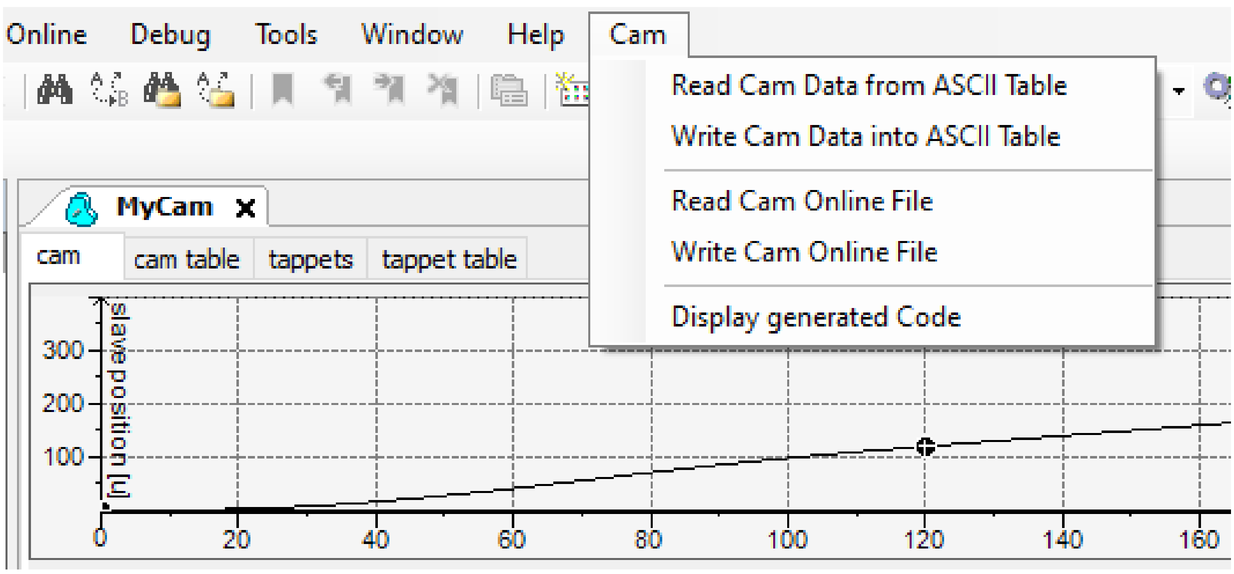


Figure 5.15: *Importing/exporting cam tables.*

More information about cam tables can be found directly on the Codesys website, available at: https://help.codesys.com (Add-ons > CODESYS SoftMotion > Reference > User Interface > Objects > Object 'Cam Table').

5.5.3 Running a cam table

To execute a cam table, it is necessary to configure the axes that will be part of the motion.

- Add two virtual axes to this application (**Axis_A** and **Axis_B**), as described in Subsection 5.3.
- Make the settings on the **General** tab of both axes created, as shown in Figure 5.16.

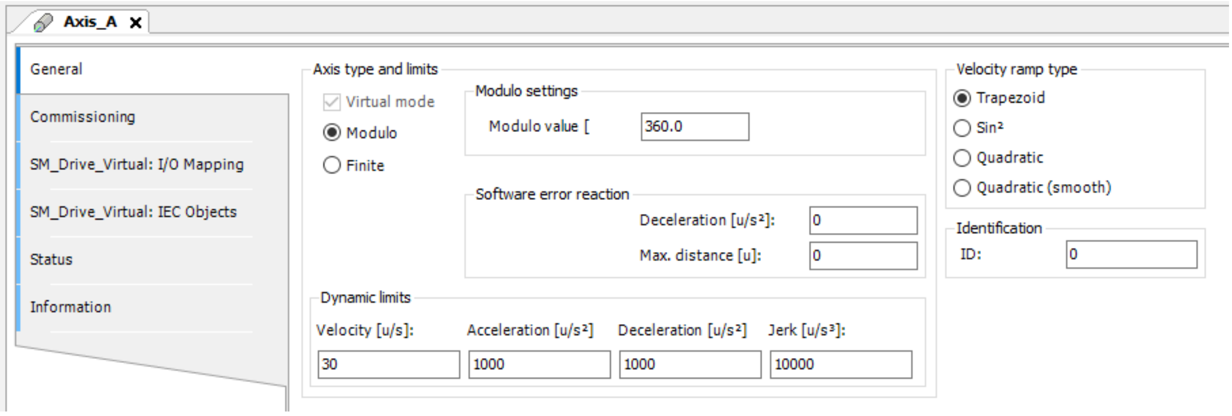


Figure 5.16: *Cam task settings.*

- Modify the priority of the **MainTask** to 1 and set it to cyclic interval of 4 ms.

Figure 5.17 shows the task settings and the objects already added.

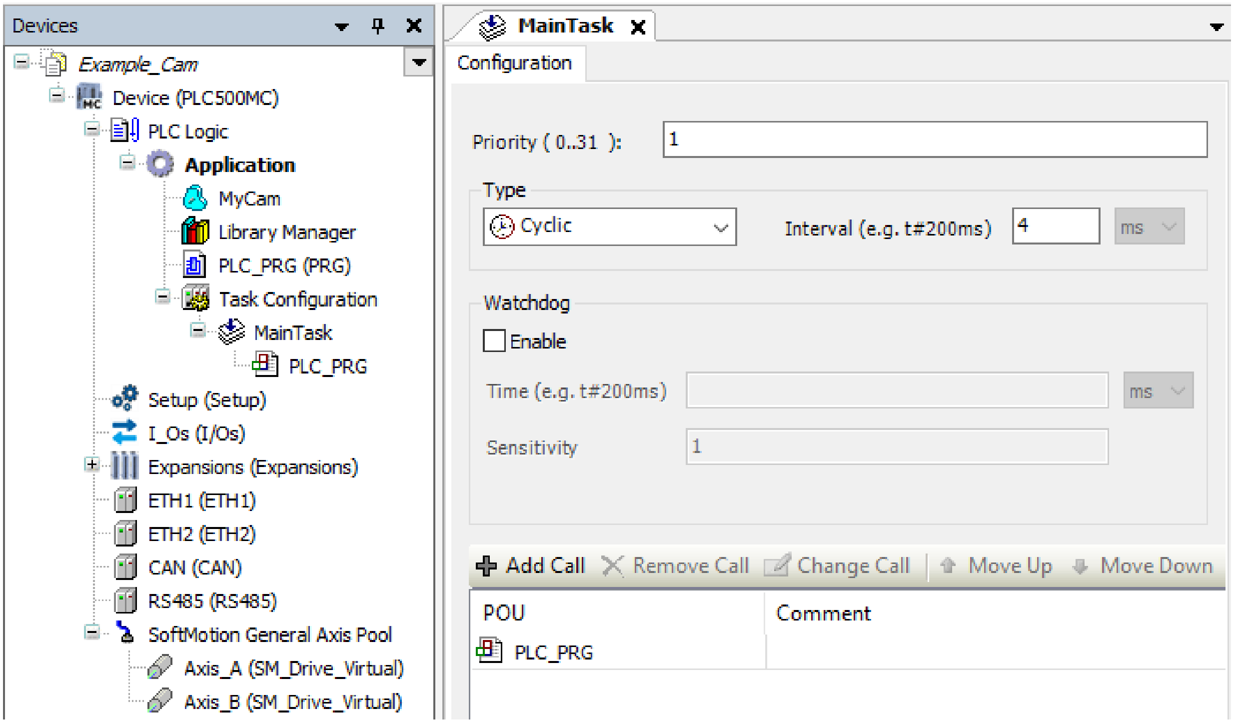


Figure 5.17: *Cam task settings.*

The default program for executing a cam table is shown in Figure 5.18.

- Open the settings of the **PLC_PRG(PRG)** program.
- Declare the function block instances and link the blocks as shown in Figure 5.18

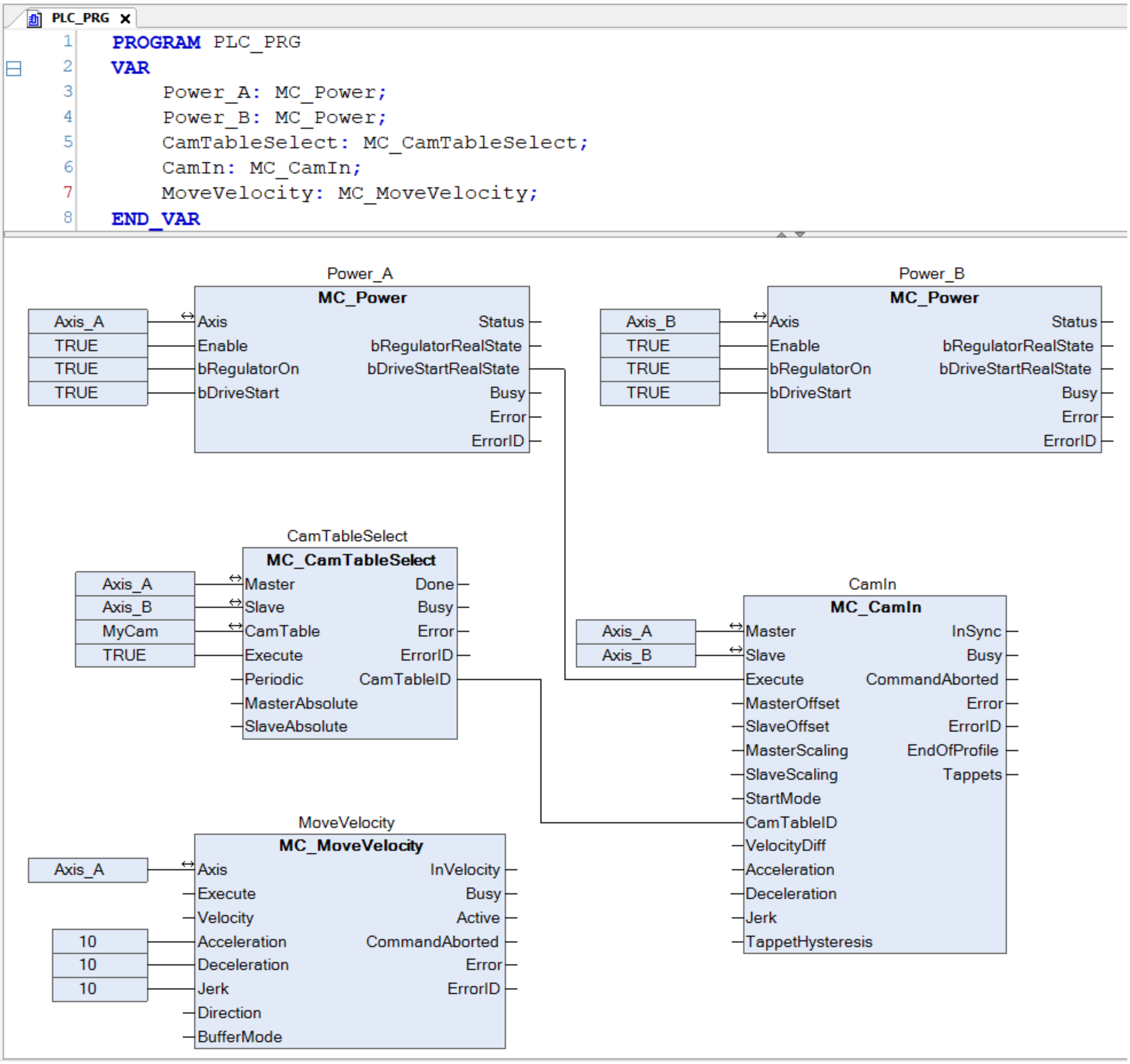


Figure 5.18: *Program to execute cam tables.*

> **NOTE!**
> Appendix A shows this same program using the ST language.

Next, you will find some information regarding each block of the program and their connections.

Function blocks of the **MC_Power** type are responsible for enabling the axes.

The **MC_CamTableSelect** function block selects the cam table to be executed. The **CamTable** input must reference the cam table of the device tree, and the **CamTableID** output must be connected to the **CamTableID** input of the **MC_CamIn** function block

The **MC_CamIn** function block implements the selected cam table.

The **MC_MoveVelocity** function block controls the speed of the master axis.

- Create a **Visualization** object.
- Add and reference the **VISU_NEW_MC_MoveVelocity** visualization template to the **MC_MoveVelocity** function block.
- Add and reference a **RotDrive** visualization template for each **Axis_A** and **Axis_B**.

Figure 5.19 shows the **Visualization** object with the templates added.

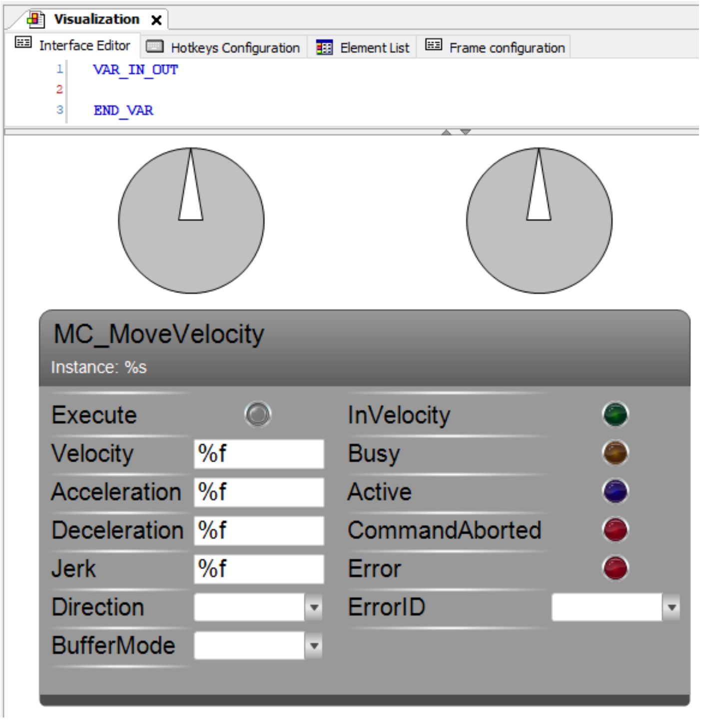


Figure 5.19: *Cam visualization.*

- Download the program for the **PLC500MC**.
- In the **Online** monitoring mode, open the **Visualization** object.
- In the **VISU_NEW_MC_MoveVelocity** visualizaton template, select the rotational speed for the master axis and click **Execute**.
- The motion of the axes can be observed through the **RotDrive** visualization templates.

Modify the cam table through the editor and run some more tests.

Other application examples using cam tables can be found directly on the Codesys website, available at: https://help.codesys.com (Add-ons > CODESYS SoftMotion > Application Examples).

5.6 INTERPRETING AND EXECUTING CNC FILES

The PLC500MC can interpret G-code (according to DIN 60025) using the 3D CNC editor present in the Codesys software.

5.6.1 Range of commands (G-Code) supported

- Quick positioning (G0).
- Linear interpolation (G1), circular interpolation (G2/G3).
- Timing (G4).
- Helical interpolation (G5, G10).
- Parabolic interpolation (G6), elliptical interpolation (G8, G9).
- Selections of interpolation plane for circular arcs (G16 - G19).
- Conditional leaps (G20).
- Write/increment IEC variable (G36, G37).
- Tool radius compensation (G40 - G42).
- Rounding and smoothing of angles (G50, G51, G52).
- Shift of the coordinate system (G53 to G56).
- Loop suppression (G60, G61).
- Time synchronization with interpolator (G75).
- Absolute and relative coordinates (G90, G91).
- Position setting (G92).
- Absolute and relative coordinates (G98, G99).
- M functions (M), Path tappets (H).
- Definition of speed and acceleration (F, E).
- Supported dimensions: X, Y, Z (primary interpolation axes).
- A, B, C (orientation axes).
- P, Q, U, V, W (additional axes).

5.6.2 Creating a CNC application

This subsection describes the necessary settings and the function blocks used to execute a CNC path for a 2D gantry type plant.

- Create a new project in **File > New Project**. Select **Standard Project**, define a directory and application name (Example_CNC). Select the **PLC500MC** device and the **Continuous Function Chart (CFC)** programming language.
- On the device tree, right-click the object **Application > Add Object > CNC program...**
- In the opened dialog, set it as shown in Figure 5.20.
- Click **Add**.

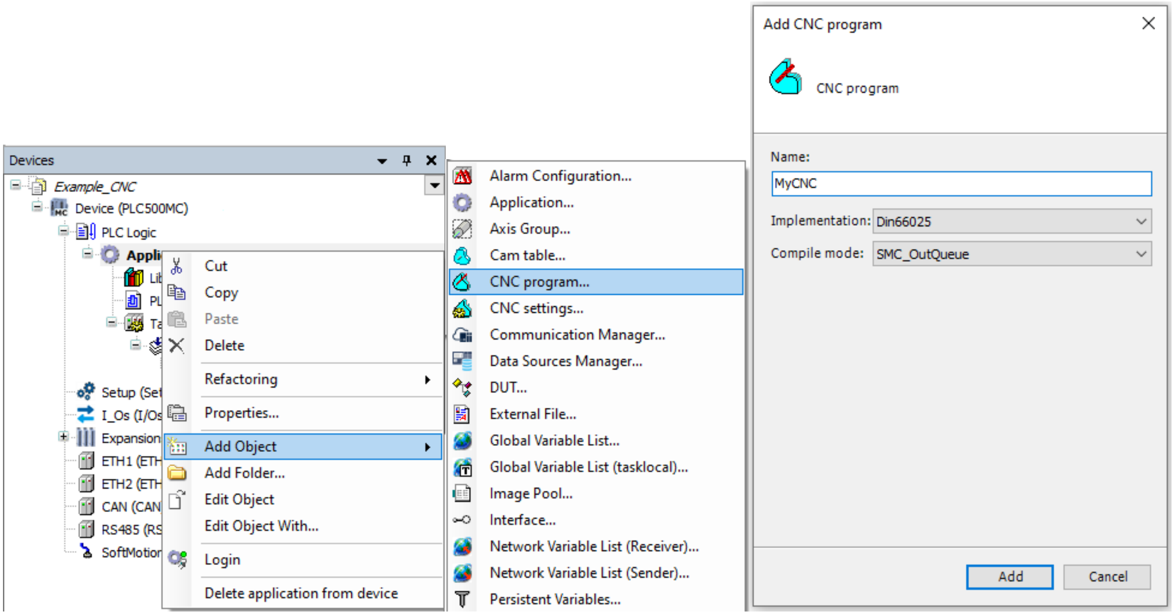


Figure 5.20: *Creating a CNC program.*

When adding the object, observe on the device tree that, in addition to the CNC program (MyCNC), an object called **CNC Settings** is added. The settings of this object are valid for all CNC objects in the application. In the **CNC Settings**, you can specify settings for the modules of path preprocessing, preinterpolation and CNC table editor.

The available preprocessing settings are shown in Table 5.3.

Function block	Description
SMC_CheckVelocities	Reduces the speed to zero if there are sharp curves.
SMC_AvoidLoop	Disregards *loop* in the code.
SMC_ExtendedVelocityChecks	Checks the speed of additional axes.
SMC_LimitCircularVelocity	Limits the speed in circular motions.
SMC_ObjectSplitter	Divides a curve into multiple points.
SMC_RotateQueue2D	Rotates the 2D path on the plane.
SMC_RoundPath	Rounds corners using circular arcs.
SMC_ScaleQueue3D	Adjusts the path scale factor.
SMC_SmoothAddAxes	Smooths motions of additional axes.
SMC_SmoothPath	Smooths the edges of a given path.
SMC_SmoothMerge	Approximates a number of points by a polynomial.
SMC_ToolCorr | **SMC_ToolRadiusCorr**	Corrects the tool radius.
SMC_TranslateQueue3D	Shifts the path in X, Y, and Z.
SMC_SmoothBSpline	Smooths consecutive **G1** element segments with a **B-Spline** of fifth degree.
SMC_RecomputeABCSlopes	Recalculates the inclinations of additional axes A,B,C to perform a smooth motion.
SMC_ReduceVelEndAtCorner	Reduces the final speed if there is an edge between two consecutive path elements.

Table 5.3: *Description of preprocessing function blocks.*

More information on the configuration of the **CNC Settings** object path preprocessing can be found directly on the Codesys website, available at: https://help.codesys.com (Libraries > SM3_CNC Library Documentation > SM_CNC_POUs > SoftMotion CNC > SoftMotion Function Blocks).

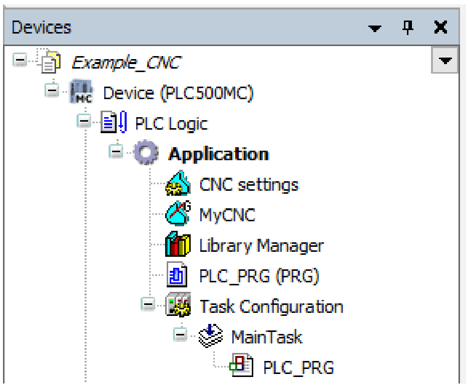


Figure 5.21: *CNC device tree.*

- Open the CNC program settings (MyCNC).
- In the CNC editor, type the commands of Figure 5.22.

Note that when typing commands, the CNC path will be displayed in the graphic editor.

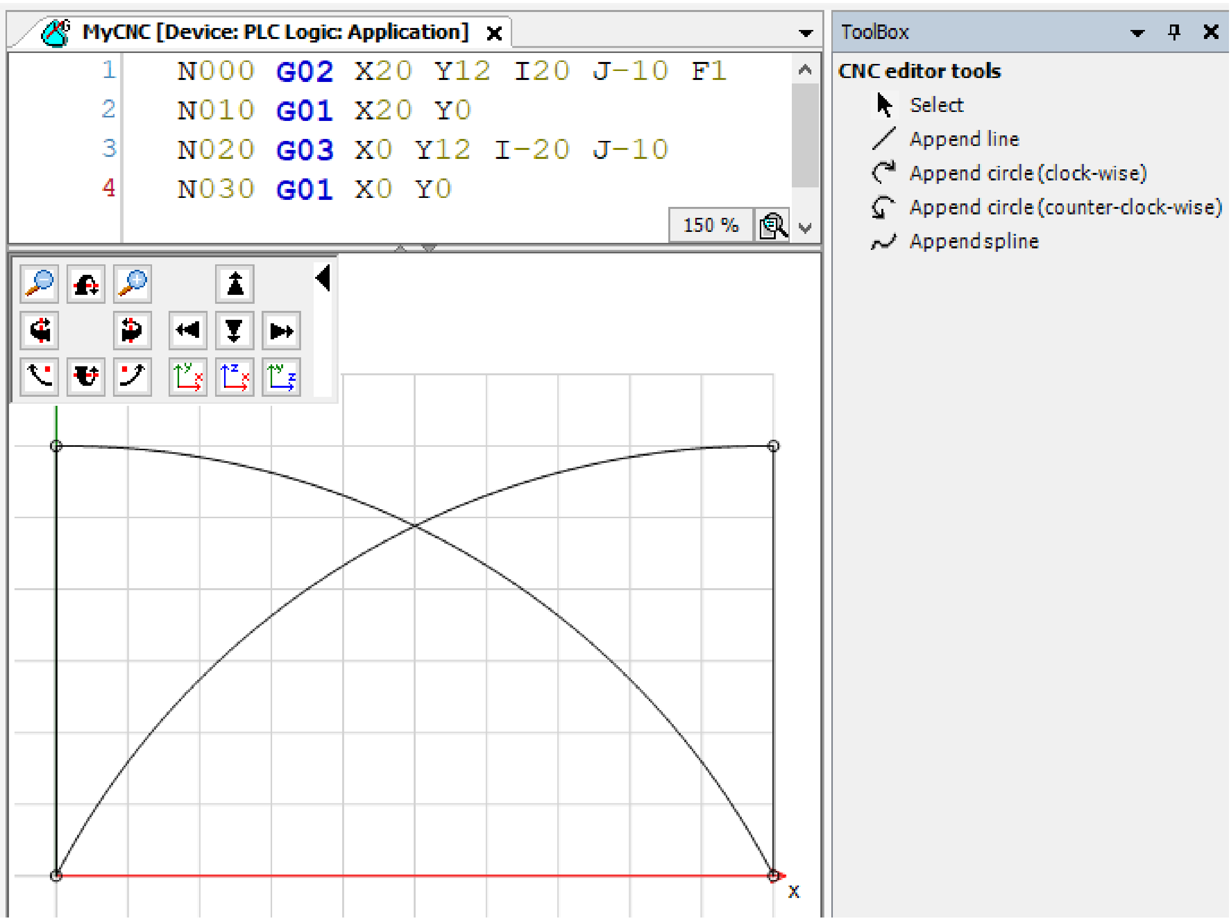


Figure 5.22: *Basic CNC Program.*

G code used:
N000 **G02** X20 Y12 I20 J-10 F1 N010 **G01** X20 Y0 N020 **G03** X0 Y12 I-20 J-10 N030 **G01** X0 Y0

5.6.3 Import CNC files

In addition to creating a CNC path, using the editor you can also import DXF or ASCII files (.cnc, .gcode, .txt).

- To import a file, open the **CNC** (MyCNC) object on the device tree.

With the object open, a new option called **CNC** is enabled in the top menu of the Codesys software; this tab contains the options for importing files.

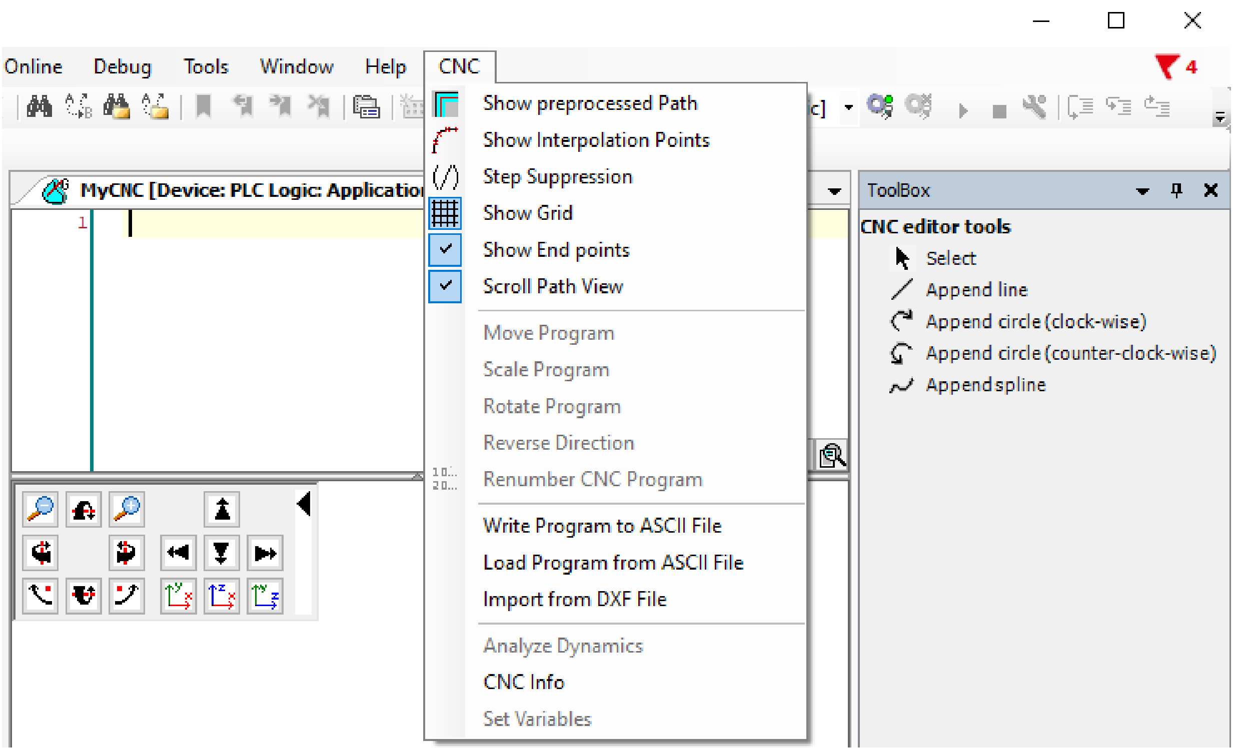


Figure 5.23: *Import/export CNC paths.*

- Click **Import from DXF File** or **Load Program from ASCII File** and select the file.

By doing so, the file will be imported, and it can be viewed in the graphic editor, as in Figure 5.24.

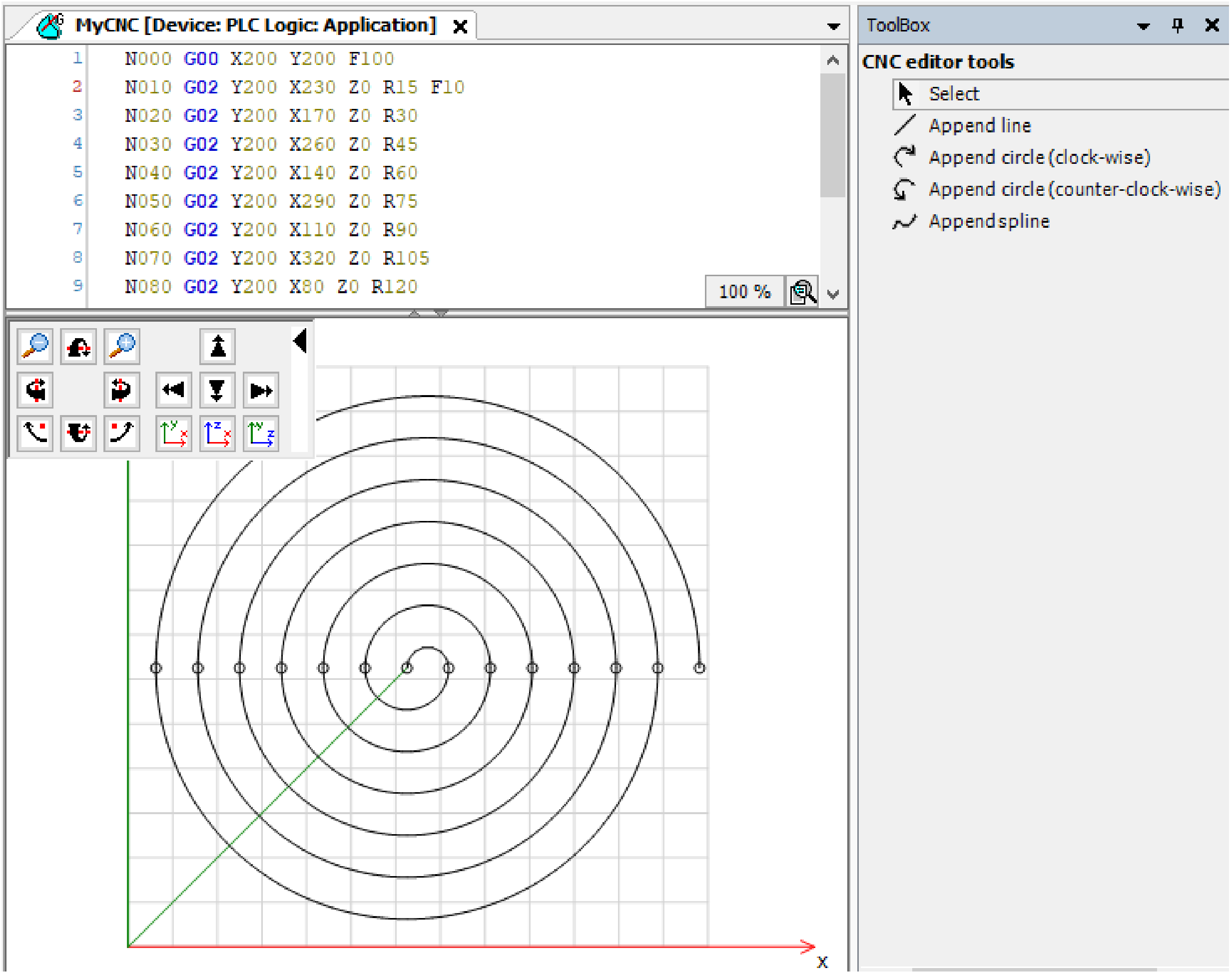


Figure 5.24: *Imported CNC path.*

ATTENTION!
The units used in the CNC path are application units; make the correct settings on the scales for the axes.

More information about CNC files can be found directly on the Codesys website, available at: https://help.codesys.com (Add-ons > CODESYS SoftMotion > Reference > User Interface > Commands > CNCCommand).

5.6.4 Running CNC path

To execute a CNC path, it is necessary to configure the axes that will be part of the motion.

- Add two virtual axes to this application (**Axis_A** and **Axis_B**), as described in Subsection 5.3.
- Modify the priority of the **MainTask** to 1 and set it to cyclic interval of 4 ms.

Figure 5.25 shows the task settings and the objects already added.

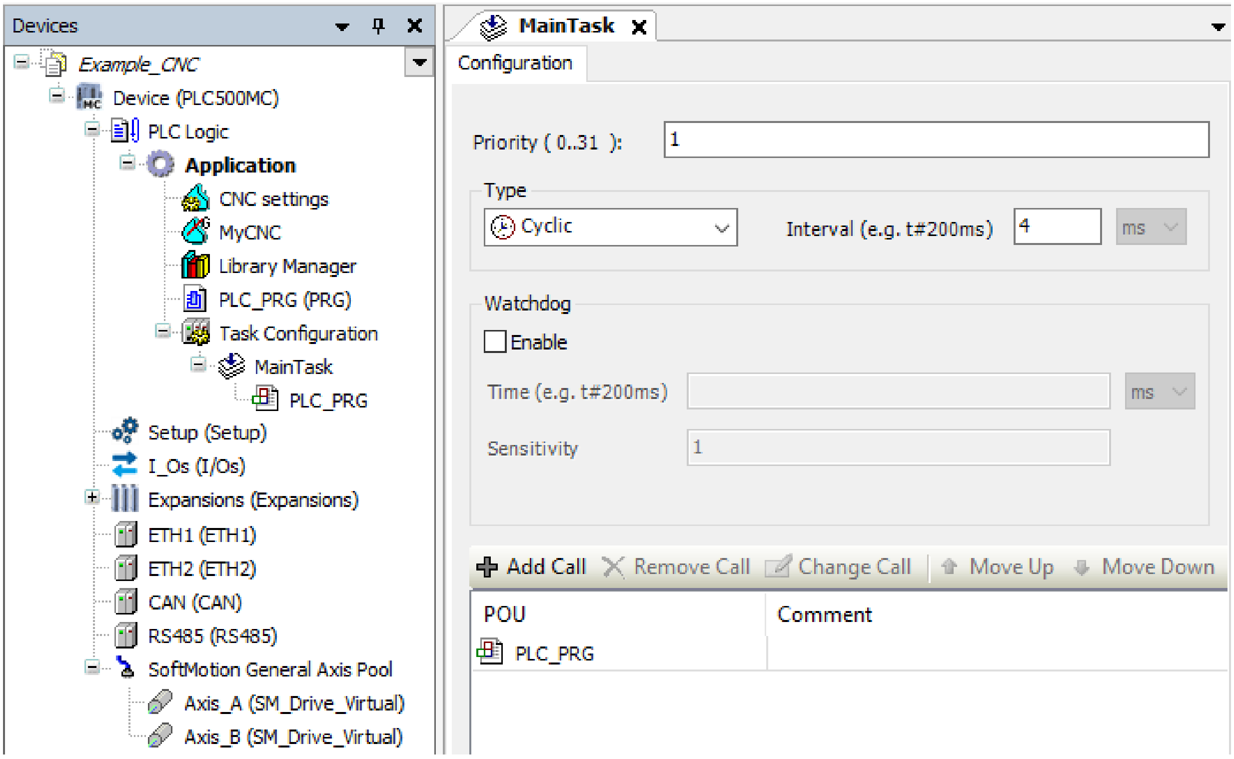


Figure 5.25: CNC task settings.

The default program for executing a CNC path controlling a 2D gantry type system is shown in Figure 5.26.

- On the device tree, open the **PLC_PRG(PRG)** program.
- Declare the function block instances and link the blocks as shown in Figure 5.26

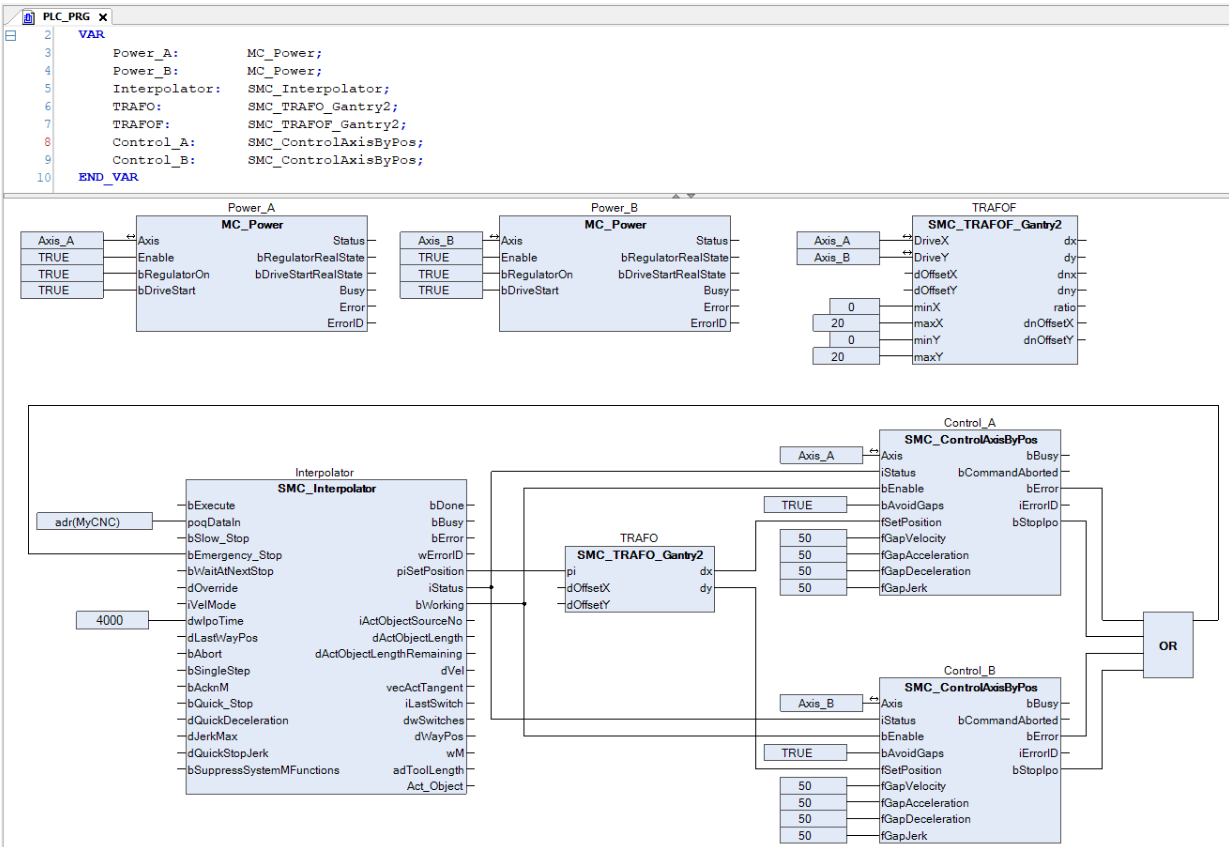


Figure 5.26: Program to execute CNC paths.

NOTE!
Appendix B shows this same program using the ST language.

Next, you will find some information regarding each block of the program and their connections.

Function blocks of the **MC_Power** type are responsible for enabling the axes.

The **SMC_Interpolator** function block converts a path defined by GEOINFO objects into discrete path points. The function block receives the address from the CNC program at the **poqDataIn** input and the IEC task cycle time it will run at the **dwIpoTime** input.

The **SMC_TRAFOF_Gantry2** function block corresponds to the direct transform of the **Gantry2** system, and it is only needed for visualization.

The **SMC_TRAFO_Gantry2** function block corresponds to the inverse transform of the **Gantry2** system, and it is responsible for generating the reference for each axis at its output.

The **SMC_ControlAxisByPosition** function block controls the position of the axis connected to the **Axis** input. As the application does not guarantee that the interpolator outputs are constant (for example, the path ends at a different point from where it started), it is necessary to activate gap prevention (bAvoidGaps, fGapVelocity, fGapAcceleration, fGapDeceleration).

- Create an object of the **Visualization** type.
- Add and reference visualization templates of the **VISU_NEW_SMC_Interpolator** and **SMC_VISU_Gantry2** type to the **SMC_Interpolator** and **SMC_TRAFOF_Gantry2** function blocks, respectively.

Figure 5.27 shows the **Visualization** object with the templates added.

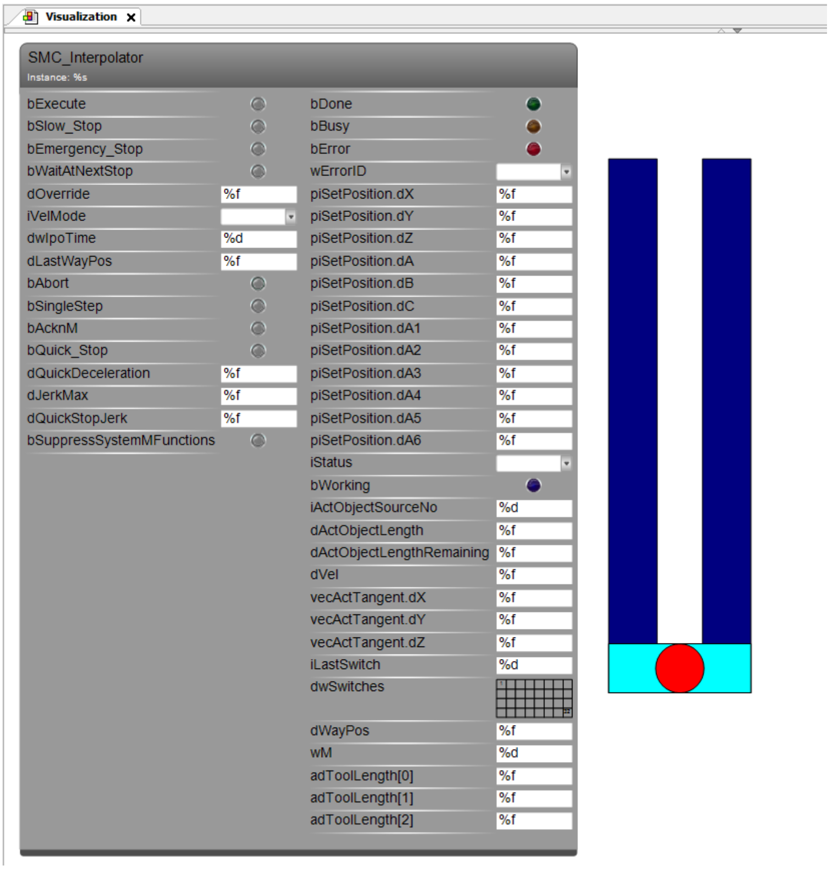


Figure 5.27: *CNC visualization.*

- Download the program for the **PLC500MC**.

- In the **Online** monitoring mode, open the **Visualization** object.
- The program executes the CNC motion as soon as the **Execute** input of the interpolator is activated.
- After the complete execution of the program, you can restart it by means of a new rising edge at the interpolator **Execute** input.
- The motion can be observed through the **SMC_VISU_Gantry2** visualization template.

If you wish, perform some more tests.

Application examples using CNC paths can be found directly on the Codesys website, available at: https://help.codesys.com (Add-ons > CODESYS SoftMotion > Application Examples).

5.6.5 Tangential axis on CNC paths

The PLC500MC enables applications that require an axis that is tangent to the CNC path during motion. This type of application is generally used in a cutting machine.

- Create a new application, as shown in Subsection 5.6.2.

To execute a CNC path with a tangential axis, it is necessary to configure the axes that will be part of the motion.

- Add two virtual axes to this application (**Axis_A** and **Axis_B**), as described in Subsection 5.3.
- Add a third virtual axis (**Axis_R**), as described in Subsection 5.3; however, on the **General** tab, change **Axis type** to **Module**. This will be the tangential axis.
- Modify the priority of the **MainTask** to 1 and set it to cyclic interval of 4 ms.

The default program for executing a CNC path controlling a 2D gantry system with a tangential axis is shown in Figure 5.28.

- On the device tree, open the **PLC_PRG(PRG)** program.
- Declare the function block instances and link the blocks as shown in Figure 5.28.

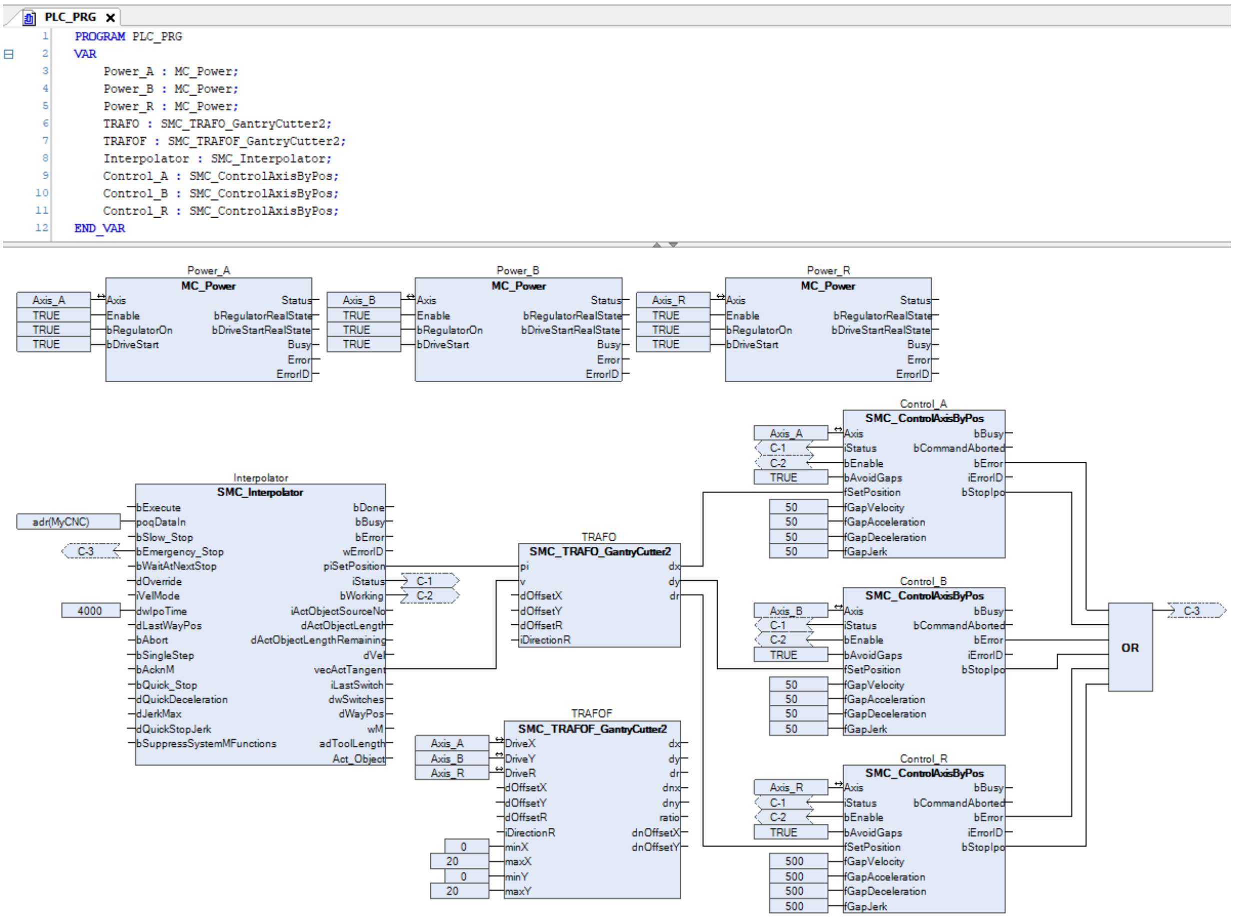


***Figure 5.28:** Program to execute CNC paths.*

> **NOTE!**
> Appendix C apresenta este mesmo programa utilizando a linguagem ST.

The **SMC_TRAFOF_GantryCutter2** function block corresponds to the direct transform of the **GantryCutter2** system, and it is only needed for visualization.

The **SMC_TRAFO_GantryCutter2** function block corresponds to the inverse transform of the **GantryCutter2** system, and it is responsible for generating the reference for each axis at its output.

> **NOTE!**
> The tangential axis reference is calculated directly by the **SMC_Interpolator** function block, and it is interpreted by the **GantryCutter2** function block, thus eliminating the need for its reference in the G-Code.

- Create an object of the **Visualization** type.
- Add and reference visualization templates of the **VISU_NEW_SMC_Interpolator** and **SMC_VISU_GantryCuuter2** type to the **SMC_Interpolator** and **SMC_TRAFOF_GantryCutter2** function blocks, respectively.

Figure 5.29 shows the **Visualization** object with the templates added.

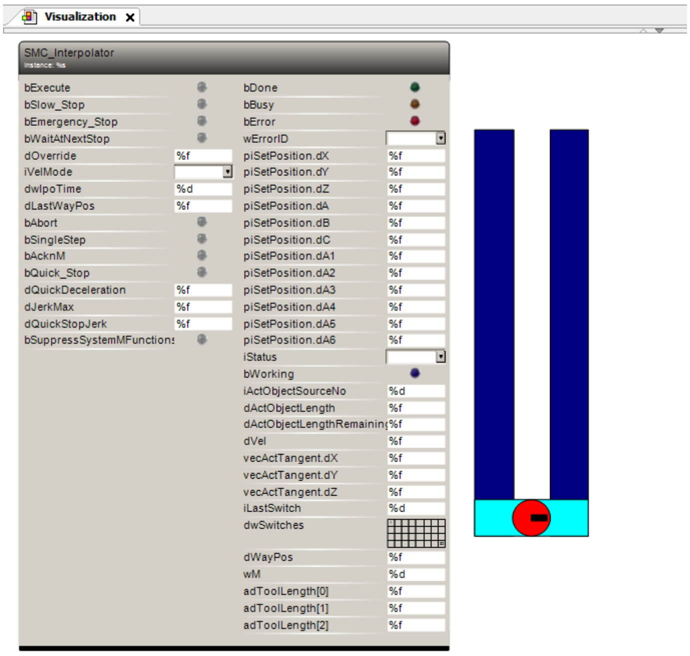


Figure 5.29: CNC visualization.

- Download the program for the **PLC500MC**.
- In the **Online** monitoring mode, open the **Visualization** object.
- The program executes the CNC motion as soon as the **Execute** input of the interpolator is activated.
- After the complete execution of the program, you can restart it by means of a new rising edge at the interpolator **Execute** input.
- The motion can be observed through the **SMC_VISU_GantryCutter2** visualization template.

If you wish, perform some more tests.

Application examples using CNC paths can be found directly on the Codesys website, available at: https://help.codesys.com (Add-ons > CODESYS SoftMotion > Application Examples).

5.7 CHANGING CONTROL MODE

Currently, the SCA06 servo drive supports two types of operating mode: cyclic synchronization position mode (cusp) and cyclic synchronization velocity mode (csv).

The **SMC_SetControllerMode** function block can be used to switch the control mode of the **SCA06_Motion**.

Preconditions:

1. The servo drive must support the desired control mode.
2. The necessary transmission and reception **PDOs** must be mapped.
3. The axis must not be in the **errorstop**, **stop** or **homing** state when the **SMC_SetControllerMode** function block is executed.
4. The **SCA06_Motion** will only accept the new control mode when disabled.

Figure 5.30 shows the **SMC_SetControllerMode** block visualization template.

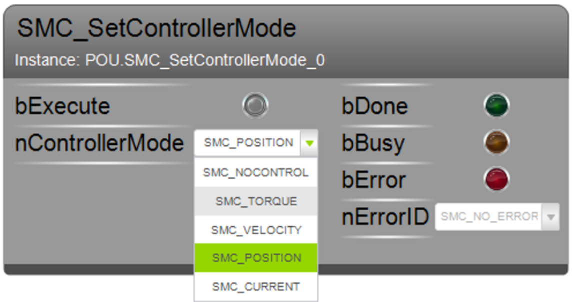


Figure 5.30: *Visualization template of the **SMC_SetControllerMode** function block.*

To change the control mode:

- Add the **SMC_SetControllerMode** function block to your application.
- With the application in the **Online** mode, make sure the axis is disabled (**MC_Power** function block).
- In the **SMC_SetControllerMode** function block, select the desired control mode.
- Activate **bExecute** function block input. The **bBusy** output of the function block will be active for 1000 cycles.
- During this period, enable the Axis.

By doing so, the **bDone** output of the **SMC_SetControllerMode** function block will become active, indicating that the control mode has been changed.

6 CREATING AND CONFIGURING CAN NETWORK + SOFTMOTION

This section describes the necessary steps to perform motion control using CAN communication between the PLC500MC and the SCA06 servo drive through the Codesys software.

> **ATTENTION!**
> For motion control using the CANopen network, use a generic CiA402 axis.

6.1 SCA06 CAN SERVO DRIVE SETTINGS

Correctly connect the CAN communication cable and the servomotor to the SCA06 servo drive.

Starting from the factoring settings of the SCA06:

- Change parameter **P0202** to **5** (control via CAN/EtherCAT network).
- Change parameter **P0385** to the value corresponding to the relevant servomotor model.
- Change parameter **P0700** to **1** (sets the CAN communication protocol to CANopen.)
- Change parameter **P0701** to **3** (sets the servo address on the CAN network to 3).
- Change parameter **P0702** to **0** (sets the baud rate of the CAN interface to 1 Mbit/s).

Follow the recommendations described in the SCA06 servo drive manual to set the device parameters related to the motor settings, desired functions for the I/O signals, etc.

For further explanations, refer to the SCA06 servo drive Programming Manual.

- Restart the servo drive.

By doing so, the SCA06 servo drive will be ready to be accessed through the CAN network.

6.2 CREATING A PROJECT IN CODESYS

- Create a new project in **File > New Project**. Select **Standard Project**, define a directory and the name of the application. Select the **PLC500MC** Device and the desired programming language.

- Add a new task responsible for the motion control (**Motion_Task**) in this application. Apply the settings as show in Figure 6.1.

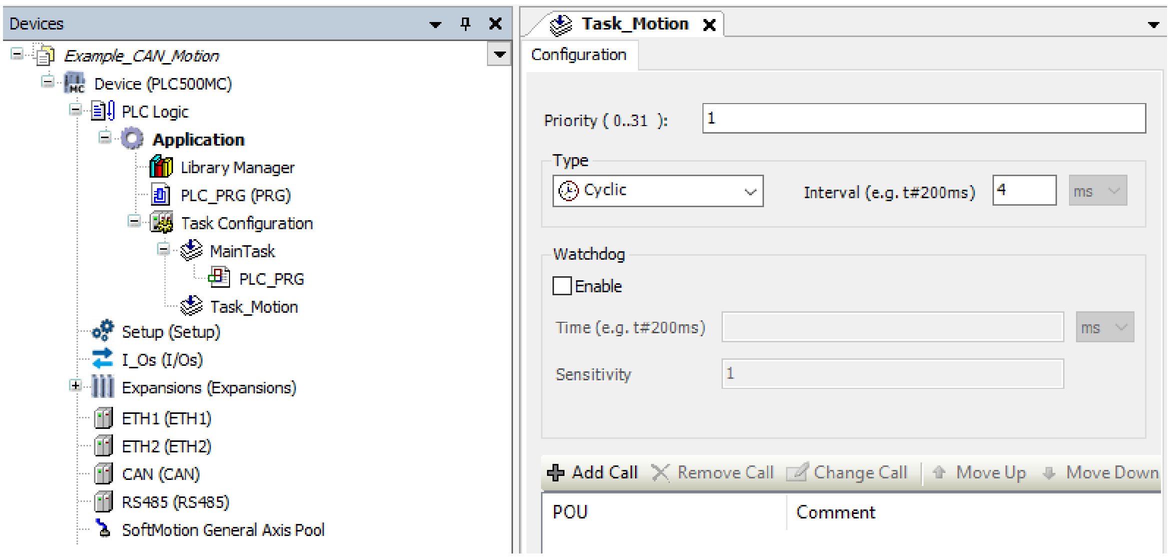


***Figure 6.1:** Priority settings.*

6.2.1 Adding CANopen Manager SoftMotion

- To add a new **CANopen Manager SoftMotion** communication interface, right-click the **CAN** object on the device tree, click **Add Device**, in the dialog select **Add Device**, and then **Fieldbuses > CANopen > CANopen_Manager_SoftMotion**, and click **Add Device** to add it to the device tree, as shown in Figure 6.2.

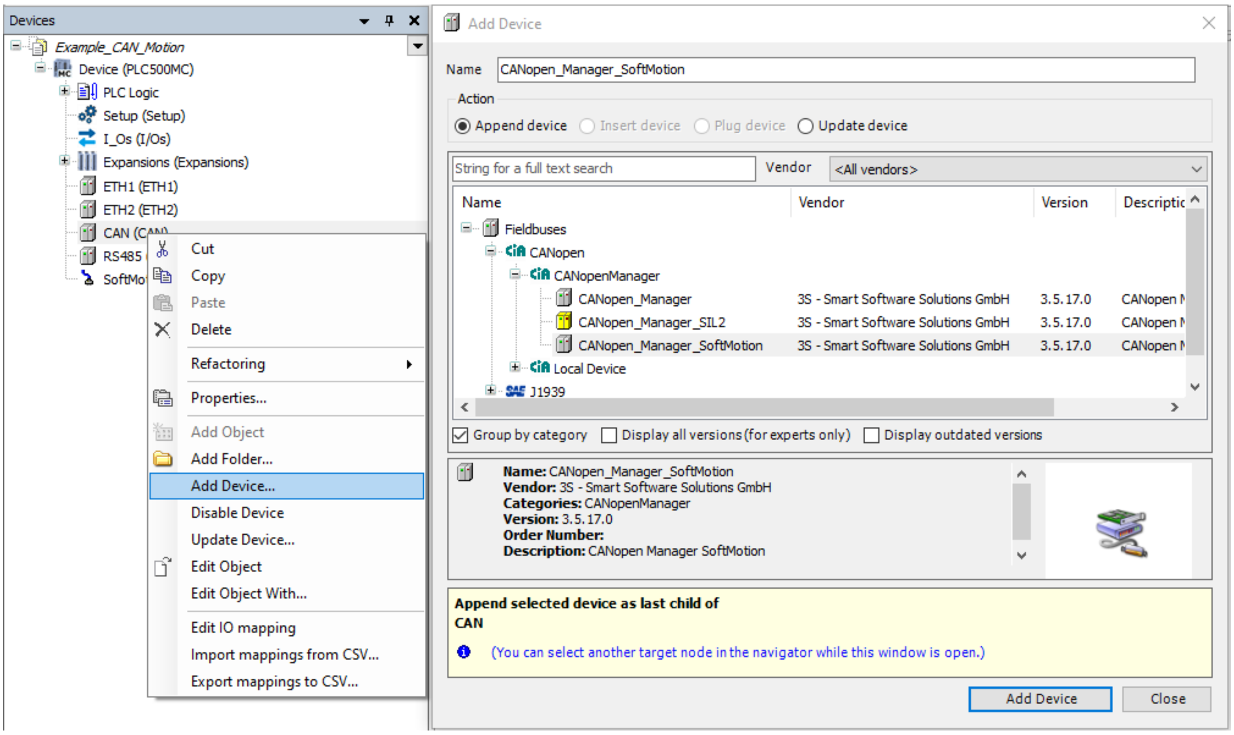


*Figure 6.2: Adding **CANopen Manager SoftMotion** to the device tree.*

6.2.2 Add SCA06 as a slave to the CANopen network

- To add the **SCA06** device as a slave to the CANopen network, right-click the **CANopen Manager SoftMotion** device previously created and select **Add Device**.)
- In the **Action** section, from the opened dialog, make sure that **Append device** is selected. Search for **SCA06 Fieldbuses > CANopen > CANopen Remote Devices > SCA06**.
- Click on **Add Device**.

Figure 6.3 shows the previous steps directly in the Codesys software.

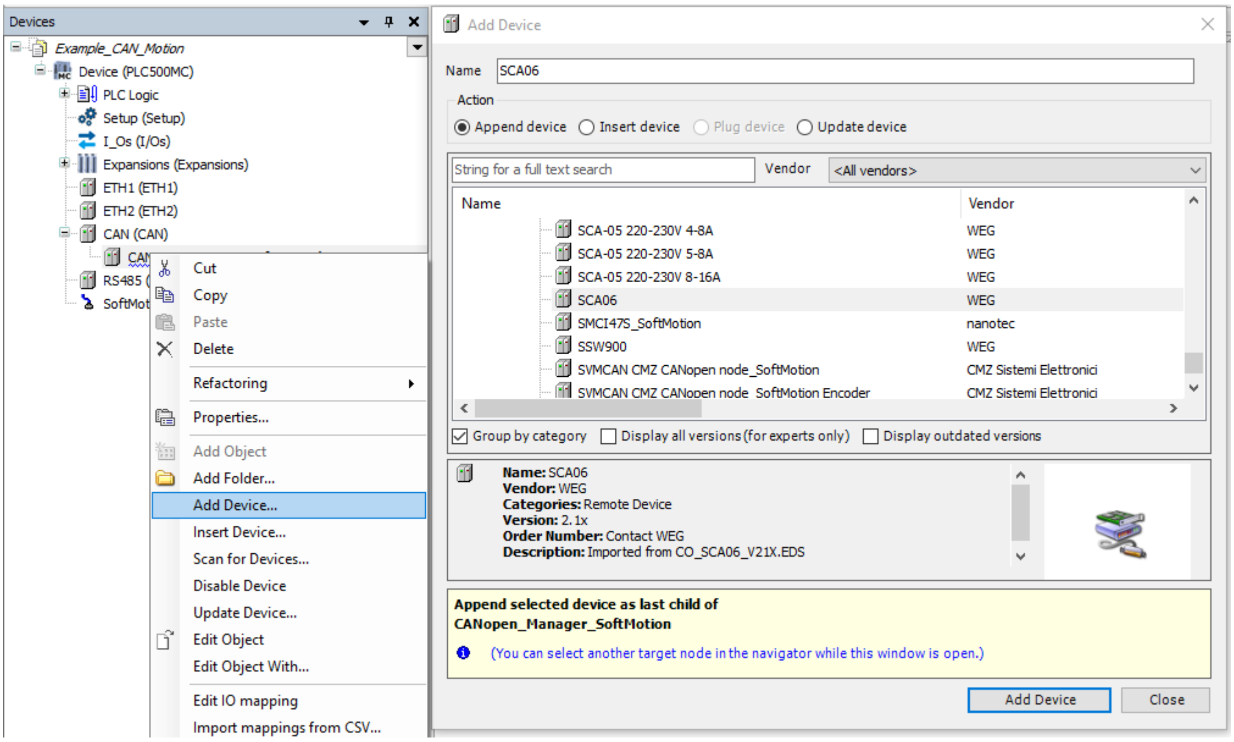


Figure 6.3: *Adding SCA06 as a slave to the CANopen network.*

> **NOTE!**
> If the SCA06 servo drive is not available, download the .EDS file directly from the WEG website at: https://www.weg.net/ and add it to the Codesys device repository (**Tools > Device Repository... > Install...**).

- To add a SoftMotion axis to the **SCA06**, right-click the **SCA06** device previously added and select **Add SoftMotion CiA402 Axis**.

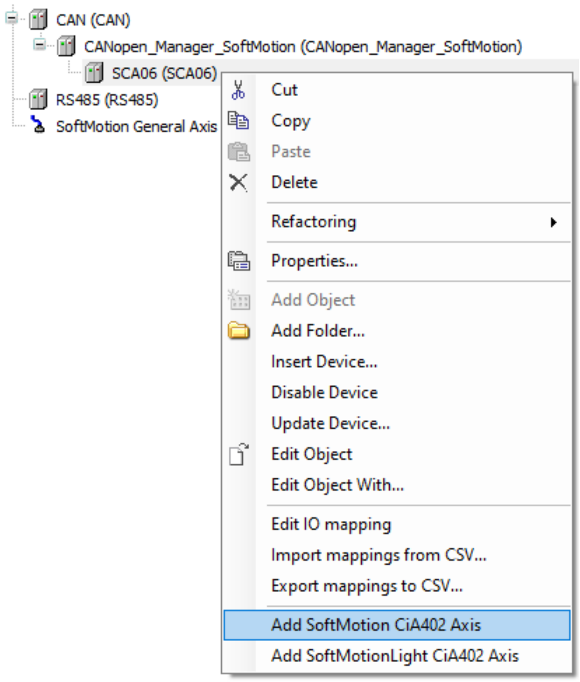


Figure 6.4: *Adding SoftMotion axis to SCA06.*

When a SoftMotion axis is manually added, the dialog of Figure 6.5 will be displayed.

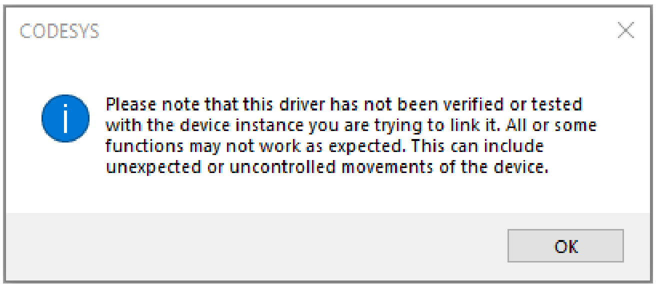


Figure 6.5: *Alert message when manually adding a SoftMotion axis.*

- Read the message and click **OK**.

After these settings, the device tree should contain the icons shown in Figure 6.6.

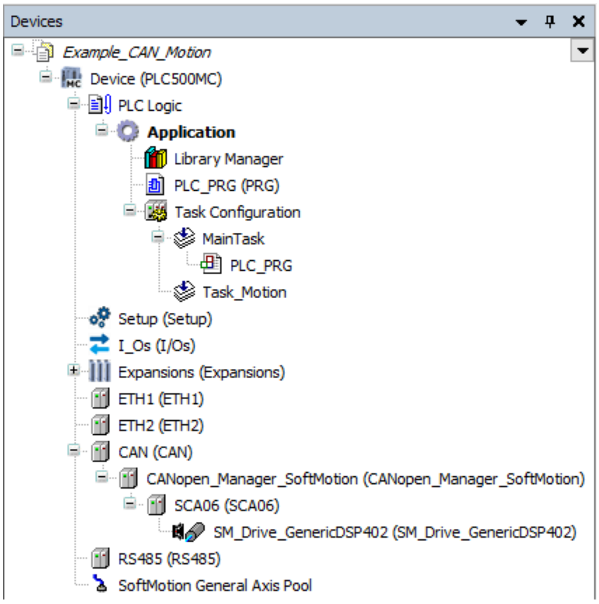


Figure 6.6: *Device tree for using SoftMotion.*

6.2.3 Setting a CAN object

- Open the **CAN** device settings, on the **General** tab, set the page options as shown in Figure 6.7.

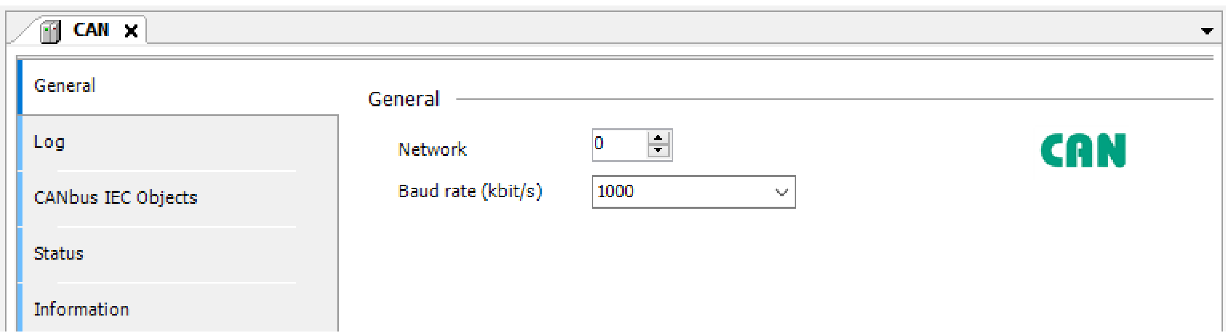


Figure 6.7: *CAN default settings.*

6.2.4 Setting a CANopen Manager SoftMotion object

- Open the **CANopen Manager SoftMotion** object settings, and on the **General** tab, set the page options as shown in Figure 6.8.

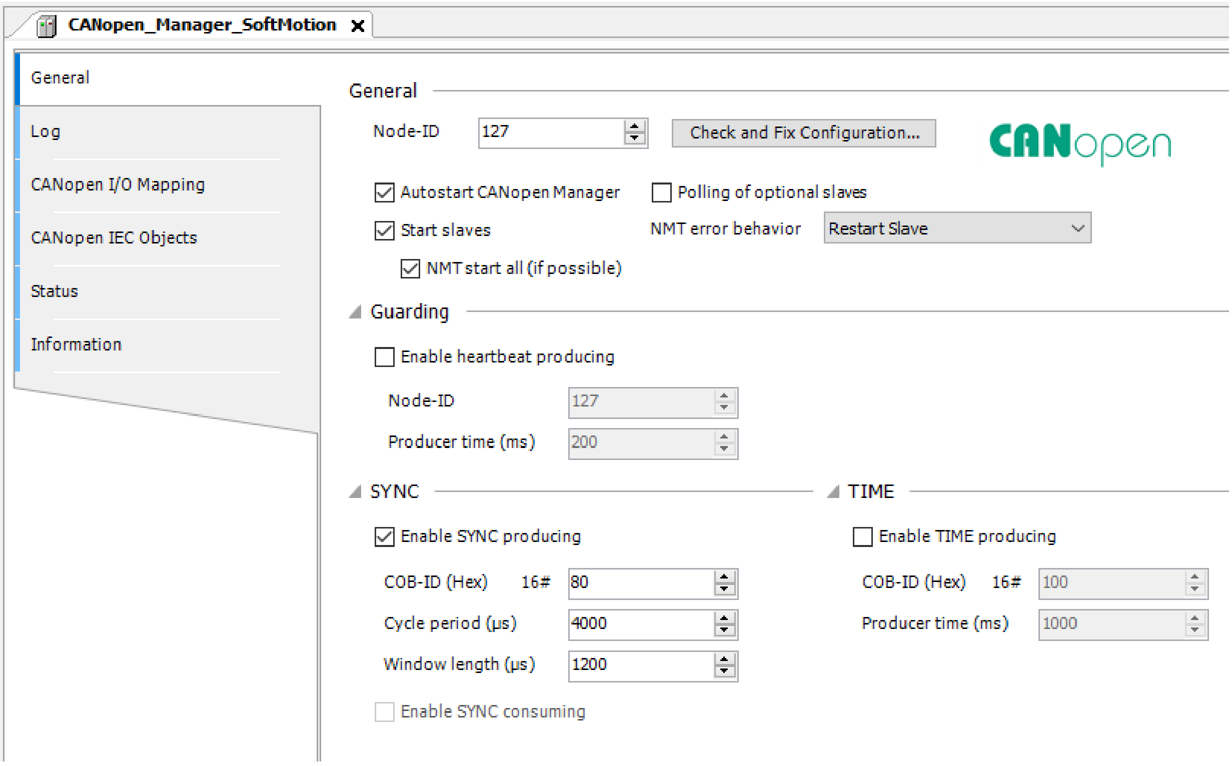


Figure 6.8: *Default CANopen Manager SoftMotion settings.*

- Still in the settings of the **CANopen Manager SoftMotion** object, on the **CANopen I/O Mapping** tab, select the task responsible for the motion (**task_Motion**), as shown in Figure 6.9.

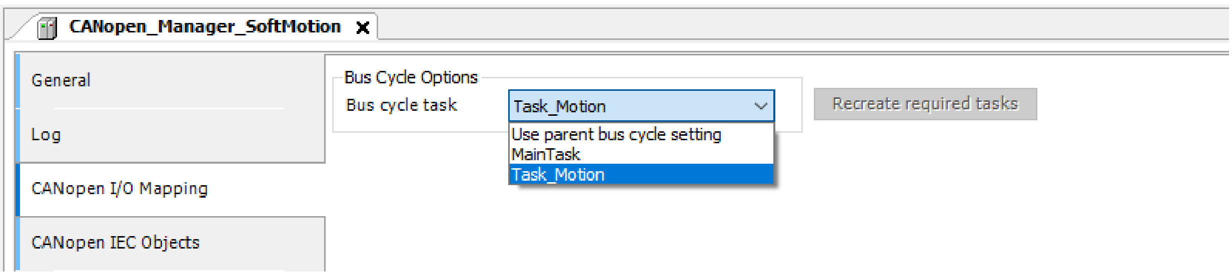


Figure 6.9: *Default CANopen Manager SoftMotion settings.*

6.2.5 Setting SCA06 as a SoftMotion CAN slave

- Open the **SCA06** object settings, and on the **General** tab, set the page options as shown in Figure 6.10.

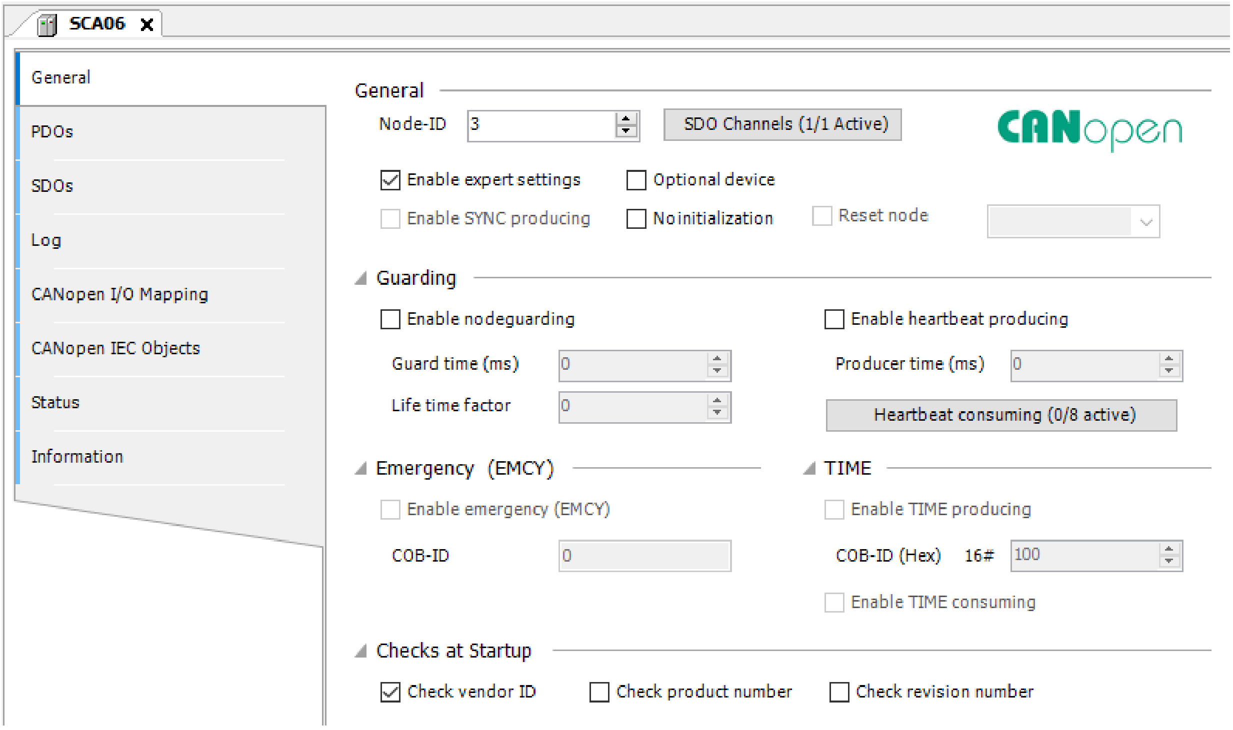


***Figure 6.10:** Default SCA06 settings on the CANopen SoftMotion network.*

- Still in the **SCA06** object settings, on the **PDOs** tab, select only the PDOs from Figure 6.11.

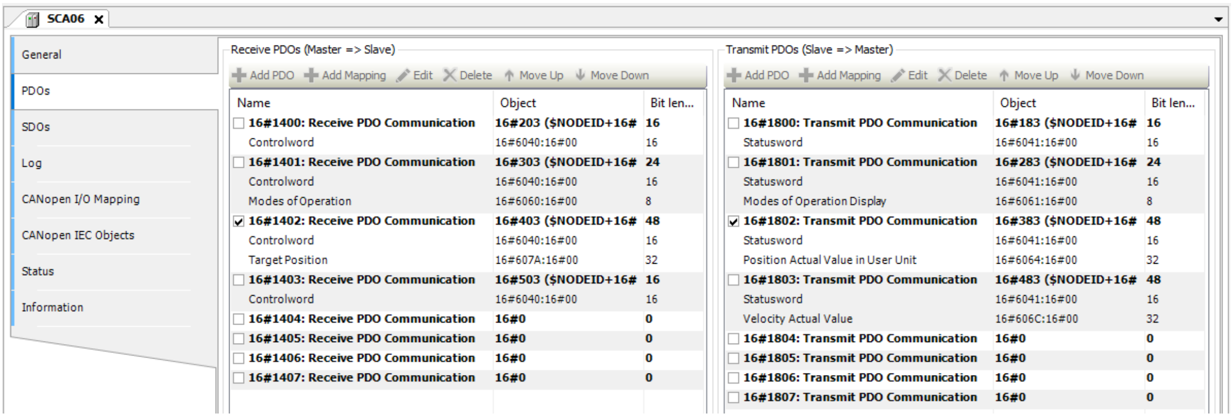


***Figure 6.11:** Default SCA06 settings on the CANopen SoftMotion network.*

- Change the **Transmissiontype** of the transmission and reception PDOs to **Cyclic - synchronous (Type 1-240)** (to open the properties, double-click **PDO Communication**).

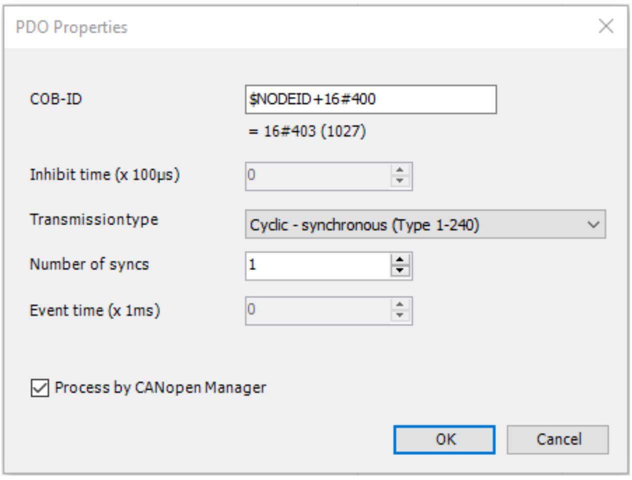


Figure 6.12: *Transmissiontype settings of the PDOs.*

6.2.6 Setting SM_Drive_GenericDSP402

- Apply the same settings as in Subsection 2.4.5.

6.3 MONITORING

6.3.1 CAN communication status

The status of the CAN network can be monitored in the **Online** mode of Codesys. When encountering connection problems, as shown in Figure 6.13, check again if the cables are properly connected and review the settings in Section 6.

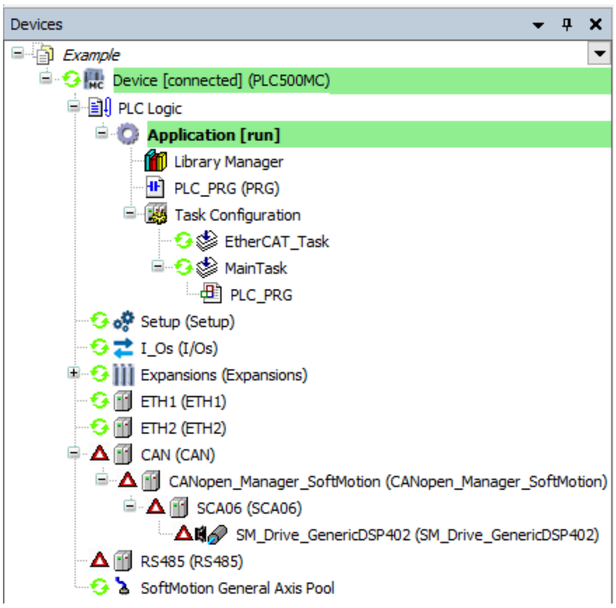


Figure 6.13: *Indication of EtherCAT communication error.*

When the settings are correct and the devices are communicating properly, all CAN communication items will be green, as shown in Figure 6.14.

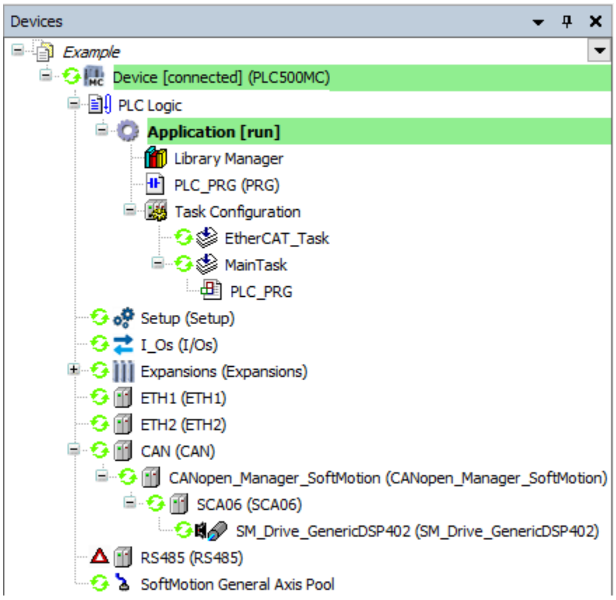


Figure 6.14: *Communication properly configured and devices communicating.*

6.3.2 Check variation in the current position of the servomotor

- After correctly setting the CAN network and still in **Online** mode, open the **SM_Drive_GenericDSP402** settings.

When the PLC is in **Online** mode, on the **General** tab, a field will be enabled for viewing the axis, as shown in Figure 6.15.

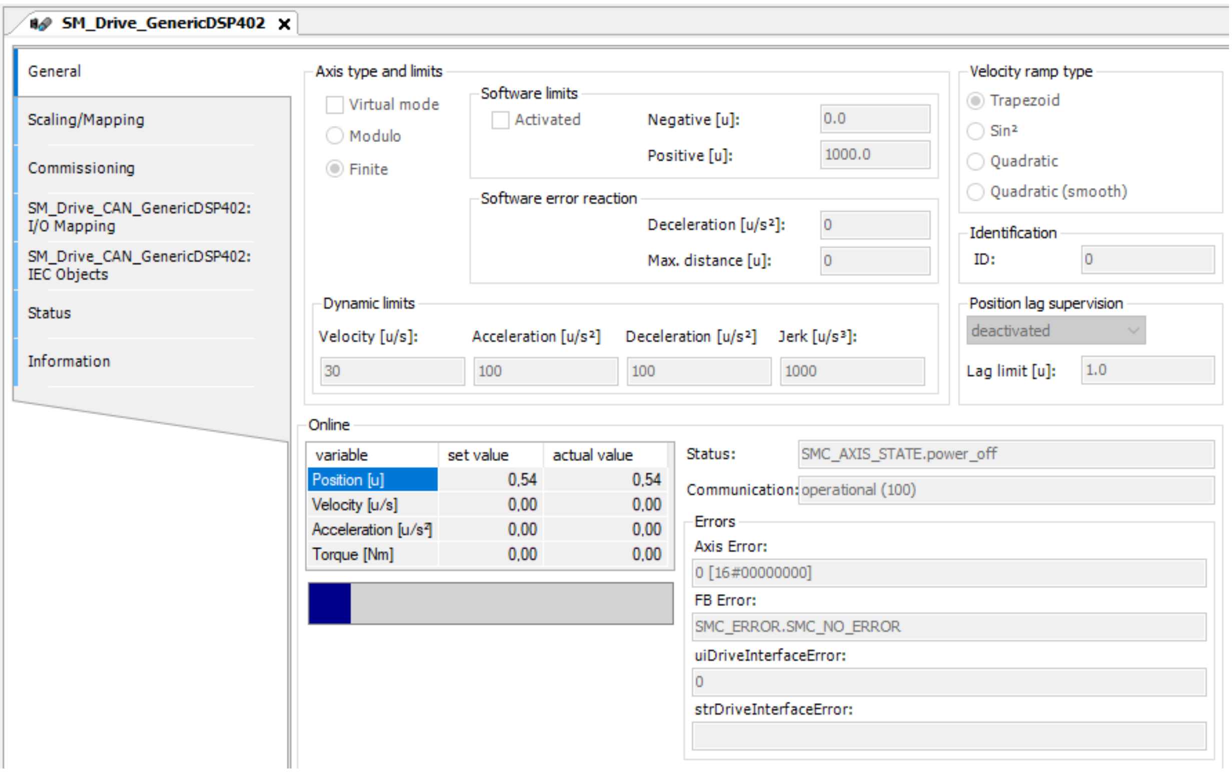


Figure 6.15: *Servomotor online monitoring.*

In this field it is possible to observe the communication and axis status, position variables, speed, acceleration and torque, with their references and current values.

- Move the servomotor shaft manually and observe the position value changing in **Position [u] - actual value**.

6.4 COMMISSIONING

To test the settings, follow the same instructions presented in Subsection 2.6.

With the settings applied in this section, the axis can now be used in the applications.

A CAM APPLICATION

This appendix contains the PLC_PRG application, from Subsection 5.5, in ST.

PLC_PRG application:

```
PROGRAM PLC_PRG
VAR
   Power_A               : MC_Power;
   Power_B               : MC_Power;
   CamTableSelect        : MC_CamTableSelect;
   CamIn                 : MC_CamIn;
   MoveVelocity          : MC_MoveVelocity;
END_VAR
```

```
Power_A(
   Axis:= Axis_A,
   Enable:= TRUE,
   bRegulatorOn:= TRUE,
   bDriveStart:= TRUE);

Power_B(
   Axis:= Axis_B,
   Enable:= TRUE,
   bRegulatorOn:= TRUE,
   bDriveStart:= TRUE);

CamTableSelect(
   Master:= Axis_A,
   Slave:= Axis_B,
   CamTable:= MyCam,
   Execute:= TRUE);

CamIn(
   Master:= Axis_A,
   Slave:= Axis_B,
   Execute:= Power_A.bDriveStartRealState,
   CamtableID:= CamTableSelect.CamTableID);

MoveVelocity(
   Axis:= Axis_A,
   Acceleration:= 10,
   Deceleration:= 10,
   Jerk:= 10);
```

B CNC APPLICATION

This appendix contains the MyMotion application from Section 5.6, in ST.

MyMotion Application:

```
PROGRAM MyMotion
VAR
   Power_A            : MC_Power;
   Power_B            : MC_Power;
   Interpolator       : SMC_Interpolator;
   Control_A          : SMC_ControlAxisByPos;
   Control_B          : SMC_ControlAxisByPos;
   TRAFO              : SMC_TRAFO_Gantry2;
   TRAFOF             : SMC_TRAFOF_Gantry2;
END_VAR
```

```
Power_A(
   Axis:= Axis_A,
   Enable:= TRUE,
   bRegulatorOn:= TRUE,
   bDriveStart:= TRUE);

Power_B(
   Axis:= Axis_B,
   Enable:= TRUE,
   bRegulatorOn:= TRUE,
   bDriveStart:= TRUE);

TRAFOF(
   DriveX:= Axis_A,
   DriveY:= Axis_B,
   minX:= 0,
   maxX:= 20,
   minY:= 0,
   maxY:= 20);

Interpolator(
   poqDataIn:= ADR(MyCNC),
   bEmergency_Stop:= Control_B.bError OR Control_B.bStopIpo OR Control_A.bError OR Control_A.bStopIpo,
   dwIpoTime:= 4000);

TRAFO(
   pi:= Interpolator.piSetPosition);

Control_A(
   Axis:= Axis_A,
   iStatus:= Interpolator.iStatus,
   bEnable:= Interpolator.bWorking,
   bAvoidGaps:= TRUE,
   fSetPosition:= TRAFO.dx,
   fGapVelocity:= 50,
   fGapAcceleration:= 50,
   fGapDeceleration:= 50,
   fGapJerk:= 50);

Control_B(
   Axis:= Axis_B,
   iStatus:= Interpolator.iStatus,
   bEnable:= Interpolator.bWorking,
   bAvoidGaps:= TRUE,
   fSetPosition:= TRAFO.dy,
   fGapVelocity:= 50,
   fGapAcceleration:= 50,
   fGapDeceleration:= 50,
   fGapJerk:= 50);
```

C TANGENTIAL CNC APPLICATION

This appendix contains the MyMotion application from Section 5.6.5, in ST.

MyMotion Application:

```
PROGRAM MyMotion
VAR
   Power_A            : MC_Power;
   Power_B            : MC_Power;
   Power_R            : MC_Power;
   Interpolator       : SMC_Interpolator;
   Control_A          : SMC_ControlAxisByPos;
   Control_B          : SMC_ControlAxisByPos;
   Control_R          : SMC_ControlAxisByPos;
   TRAFO              : SMC_TRAFO_GantryCutter2;
   TRAFOF             : SMC_TRAFOF_GantryCutter2;
END_VAR
```

```
Power_A(
   Axis:= Axis_A,
   Enable:= TRUE,
   bRegulatorOn:= TRUE,
   bDriveStart:= TRUE);

Power_B(
   Axis:= Axis_B,
   Enable:= TRUE,
   bRegulatorOn:= TRUE,
   bDriveStart:= TRUE);

Power_R(
   Axis:= Axis_R,
   Enable:= TRUE,
   bRegulatorOn:= TRUE,
   bDriveStart:= TRUE);

TRAFOF(
   DriveX:= Axis_A,
   DriveY:= Axis_B,
   DriveR:= Axis_R,
   minX:= 0,
   maxX:= 20,
   minY:= 0,
   maxY:= 20);

Interpolator(
   poqDataIn:= ADR(MyCNC),
   bEmergency_Stop:=  Control_B.bError  OR  Control_B.bStopIpo  OR  Control_A.bError  OR  Control_A.bStopIpo  OR
Control_R.bError OR Control_R.bStopIpo,
   dwIpoTime:= 4000);

TRAFO(
   pi:= Interpolator.piSetPosition,
   v:= Interpolator.vecActTangent );
```

```
Control_A(
   Axis:= Axis_A,
   iStatus:= Interpolator.iStatus,
   bEnable:= Interpolator.bWorking,
   bAvoidGaps:= TRUE,
   fSetPosition:= TRAFO.dx,
   fGapVelocity:= 50,
   fGapAcceleration:= 50,
   fGapDeceleration:= 50,
   fGapJerk:= 50);

Control_B(
   Axis:= Axis_B,
   iStatus:= Interpolator.iStatus,
   bEnable:= Interpolator.bWorking,
   bAvoidGaps:= TRUE,
   fSetPosition:= TRAFO.dy,
   fGapVelocity:= 50,
   fGapAcceleration:= 50,
   fGapDeceleration:= 50,
   fGapJerk:= 50);

Control_R(
   Axis:= Axis_R,
   iStatus:= Interpolator.iStatus,
   bEnable:= Interpolator.bWorking,
   bAvoidGaps:= TRUE,
   fSetPosition:= TRAFO.dr,
   fGapVelocity:= 500,
   fGapAcceleration:= 500,
   fGapDeceleration:= 500,
   fGapJerk:= 500);
```

WEG Drives & Controls - Automation LTDA.
Jaraguá do Sul - SC - Brazil
Phone 55 (47) 3276-4000 - Fax 55 (47) 3276-4020
São Paulo - SP - Brazil
Phone 55 (11) 5053-2300 - Fax 55 (11) 5052-4212
automacao@weg.net
www.weg.net