Servo Motion

Application Note

Using CANopen for Motion Control

AN518 Rev A (EN)

Introduction

This application note, details how to use the AC500eco V3 to control none-ABB Servo Drives over the CANopen protocol using the tools that are integrated into the Automation Builder. Currently the process requires several manual steps which may change later so always check for the latest version of this application note. Please contact your local ABB support team if you need any additional information.

Compatibility

The PLC program and libraries were written using **Automation Builder v2.8.0** or later which can be downloaded from here: https://new.abb.com/plc/automationbuilder

The PLC Hardware required to run this example is the **ABB AC500eco V3 PLC range**. These come in two main types the AC500 and AC500eco. As all CPUs from this range can support CANopen then any will do. In this instance we will focus on the AC500eco V3 PM**5092**-T-2ETH using the TA5146-CN Option Module.

The AC500 V3 CPU's have a type of designation with four numbers after the PM part of the code e.g. PM**5072**-T-2ETH, unlike the previous generation CPU's which have only three e.g. PM564-ETH.

Contents

1.Introduction

1.1. CANopen with AC500 V3

When a user has a requirement to connect to an axis or device over the CANopen protocol its possible using AC500 V3 PLC's using one of several approaches.

1. AC500eco V3 with a TA5146-CN Option Module fitted.
2. AC500 V3 onboard CANopen
3. AC500 V3 + CM598_CAN Option Module.

In this document we will focus on method one although any of them could be used to achieve the same result.

2. Approach to coding and control

Of course, once the device is connected then we must consider the requirement to control it. The AC500's motion libraries using PLCopen function blocks give a convenient and standard way to control these devices. To achieve the core control of these functions a connection between the hardware and the control protocol is needed. There are two methods to achieve this – write the code manually or use the wizard. The AB Motion Wizard provides a quick and easy mechanism to do this connection automatically for the user, but it currently only supports ECAT and PTO hardware. None the less partially using this method for the implementation is the quickest way to do get to a running program, so this document will outline how to use Wizard method, and the necessary edits and work arounds needed to make it operate with the CANopen devices.

Please note some of the function blocks mentioned in this example are included in the Motion Control Extension library which can be downloaded here:
https://search.abb.com/library/Download.aspx?DocumentID=9AKK108468A9993&LanguageCode=en&LanguageCode=zh&DocumentPartId=1&Action=Launch

2.1. Hardware Configuration

As mentioned earlier the PLC needs an option module 'TA5146-CN' to be fitted to act as a CANopen master for the network, this has dip switches that need to be configured and it needs to be situated in one of the slots in the CPU which make have 2 or 3 slots dependent on the hardware type

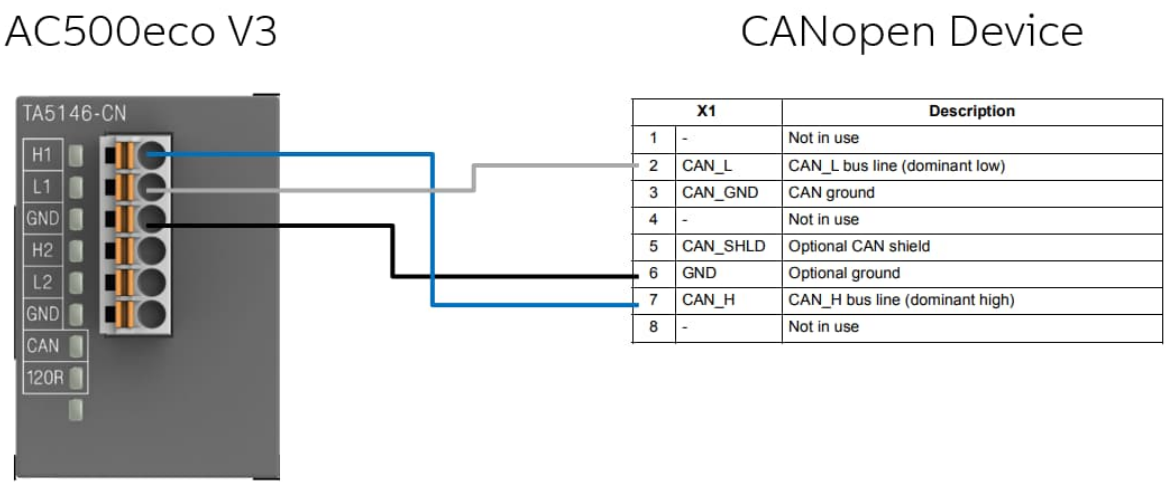
2.2. Fitting the option module

Fitting the TA5146-CN could be simpler. Push in the tab on the left and remove the blank module which occupies the slot and place the option module in its place. You are now ready to power up and use it.

2.3. Cabling and Termination Resistors

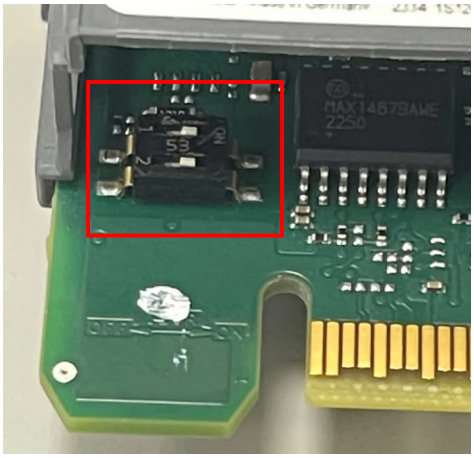
When cabling the devices, shielded twisted pair cable should be used:

The connections can be connected as is shown below:



X1		Description
1	-	Not in use
2	CAN_L	CAN_L bus line (dominant low)
3	CAN_GND	CAN ground
4	-	Not in use
5	CAN_SHLD	Optional CAN shield
6	GND	Optional ground
7	CAN_H	CAN_H bus line (dominant high)
8	-	Not in use

In addition to the cables termination end of line resistors should be used at both ends of the network, these should be 120 ohms connected between the transmission lines and ground. In the AC500eco TA5146-CN these resistors are integrated and can be turned on or off according to the user requirements. In this case as we are one end of the network, we must select switch 1 and 2 to 'On'.



In addition, the connected drive / device should also have these connections, but they depend on the device so the devices user manual should be consulted on how to do this.

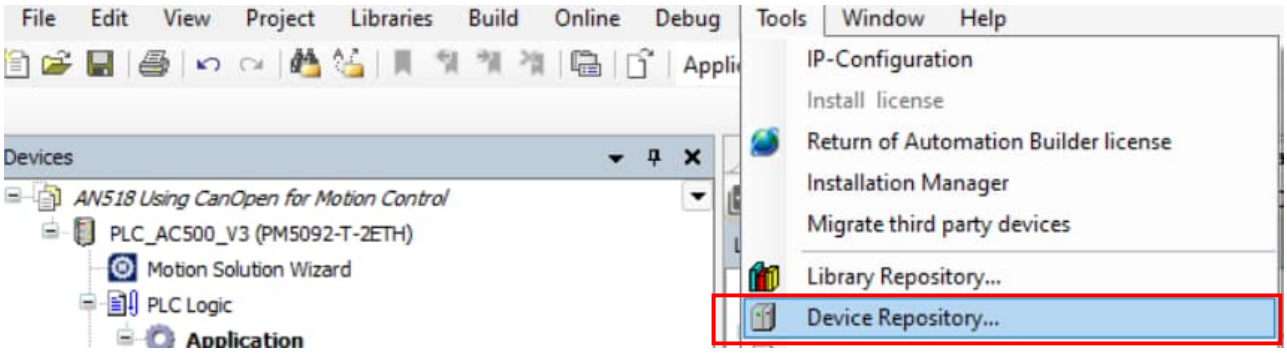
Note: These resistances are needed only on the first and last node in the network

3. Project Configuration

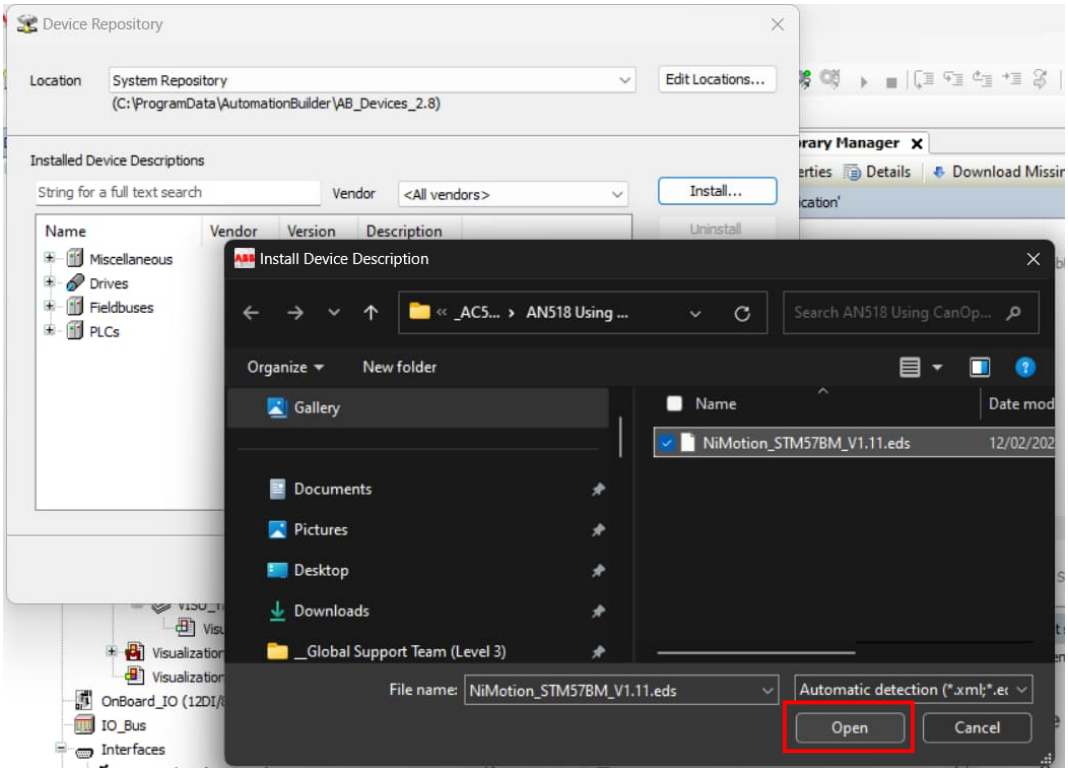
This section describes how to configure your Automation Builder project for communication and control of a connected device. In the below example we will use a third party 'STM57BM' Device which is an integrated drive and motor device.

3.1. Initial Configuration and EDS Import

Before the hardware configuration can be built, we must use import the EDS file to the device repository. To do so we must open Automation Builder and Select



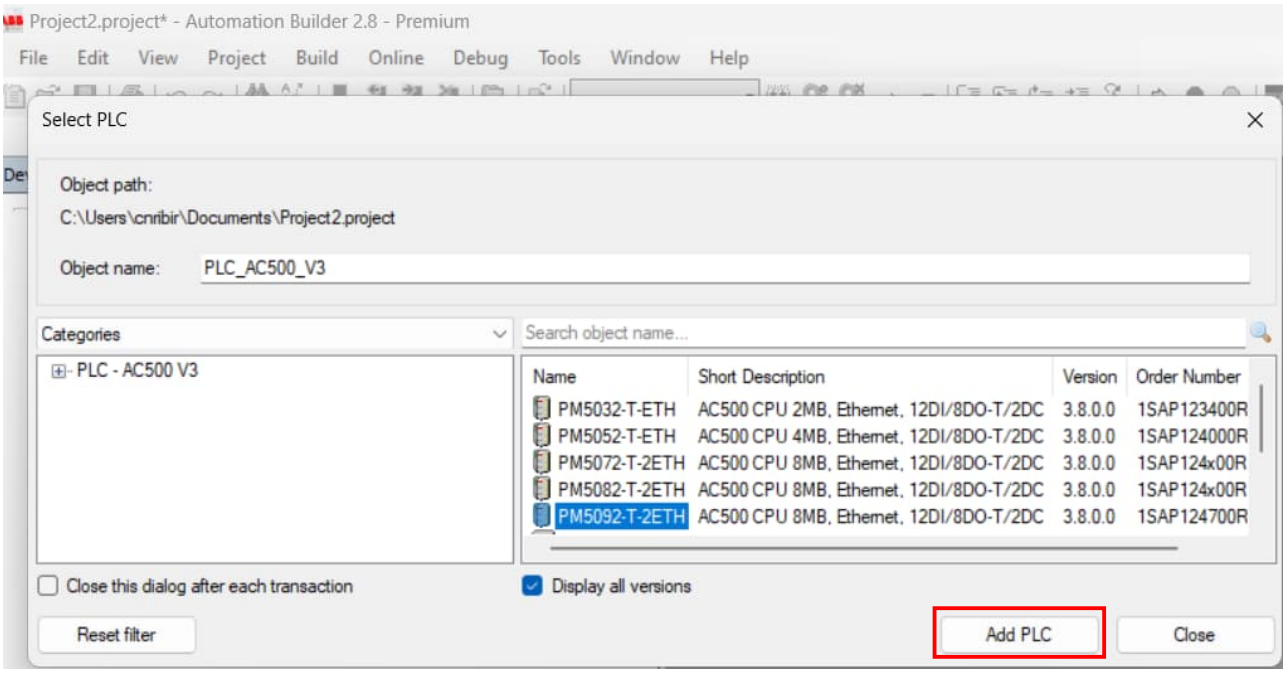
Next Select 'Install···' and navigate to where the file is stored (which should have the extension 'eds') and click Open.



Now you will see that the device 'STM57XXB_CANopen_M' is available to select in your application. Now the code can be written.

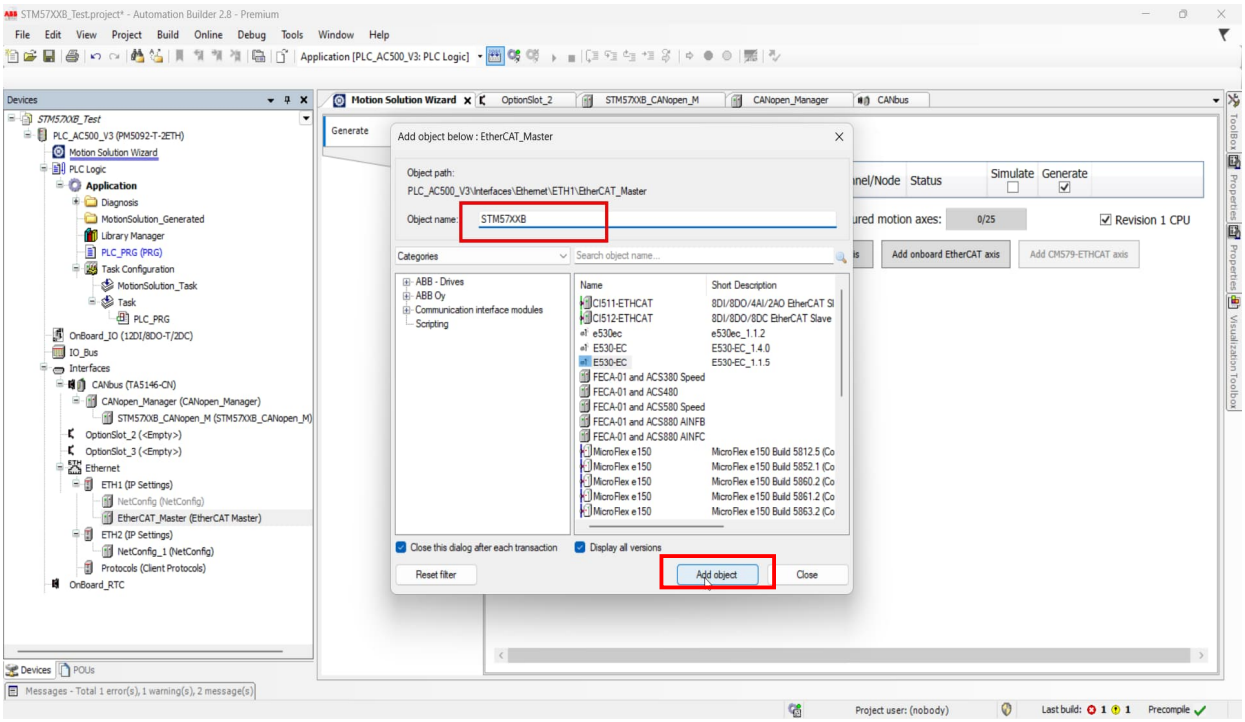
3.2. Create real axis code

To begin open a new project in Automation Builder of the type: 'Motion Solution Project' and then we must select the PLC type which we will use, in this case this will be a PM5092-2ETH



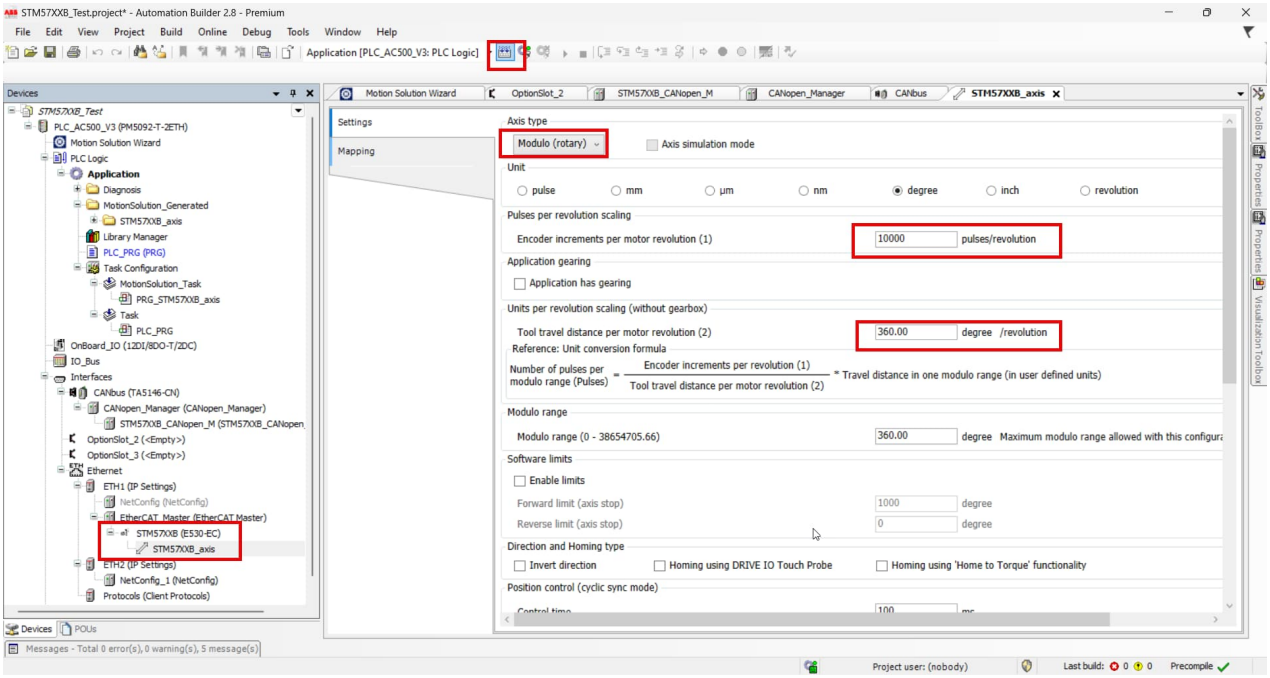
Note: As mentioned earlier we will use the Wizard to generate a template file for us for axis integration. After we do this step, and the template is generated it can be reused for additional CANopen axes and there is no need to repeat the wizard step.

Now the hardware configuration is complete open the Motion Solution Wizard and click 'Add Onboard EtherCAT axis' and select the hardware as e530-EC – its suggested that the object name same as the real axis name you will use later. This will allow for easier tracking.

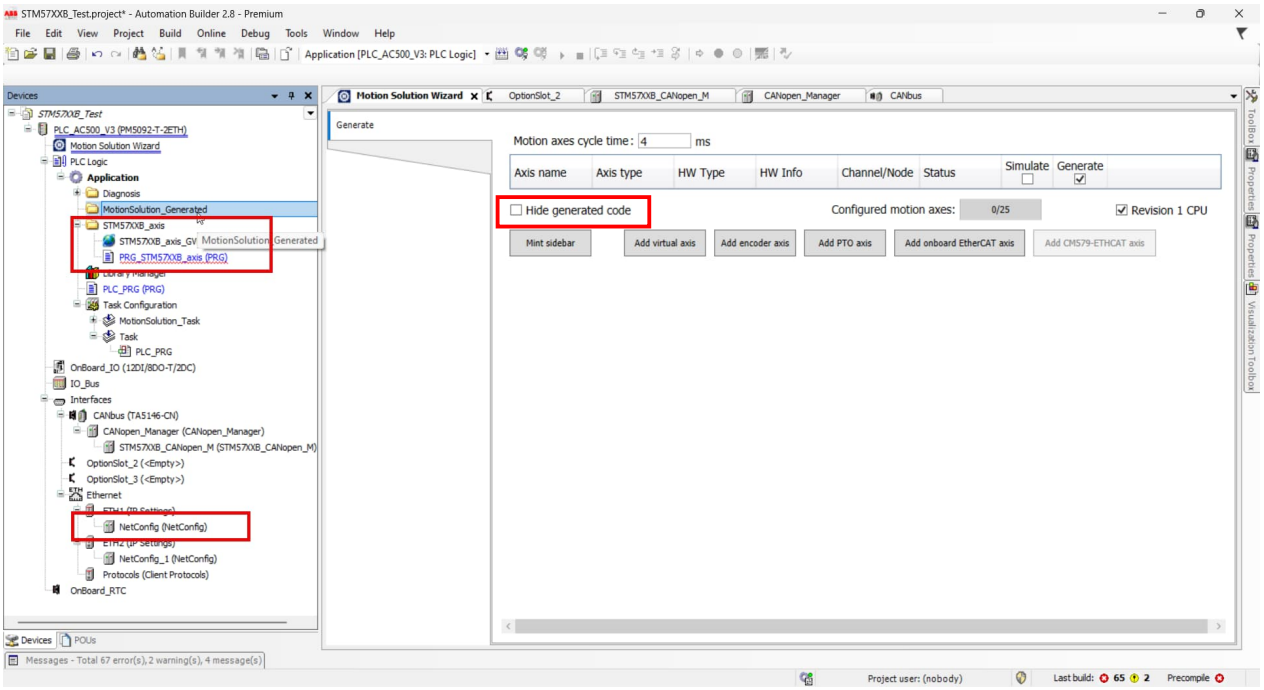


Note: the hardware type isn't important here, but we must pick something.

Next configure the reference axis to follow the real application requirements to set these parameters, in this case 10000 pulses and 360 user units per revolution and then generate the code



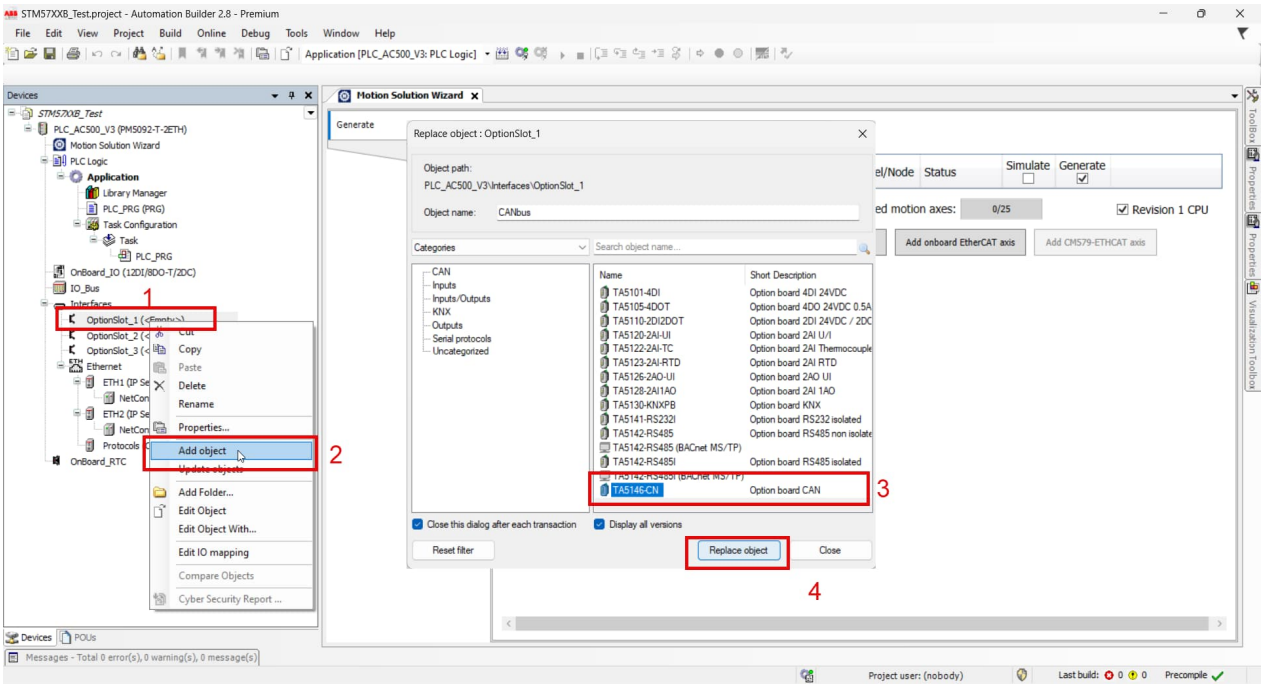
Now we have the basic template for code & GVL List made for a standard **EtherCAT axis** and we must move it from the 'MotionSolution_Generated' folder (which is controlled by the wizard) to the main 'Application' root directory or a new folder in the root. To do so cut the folder called 'STM57XXB_Axis' and move it as instructed, then delete reference axis and the EtherCAT master (unless they are needed by another part of the application)



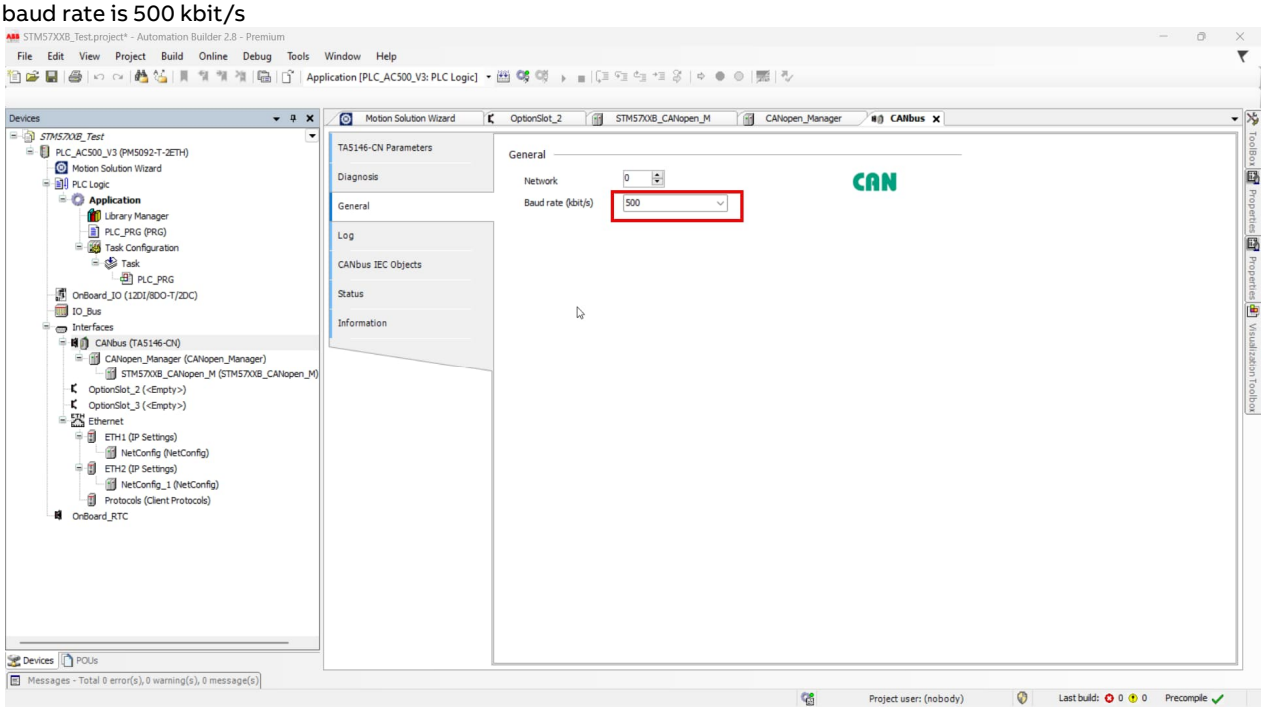
Note: To see the 'MotionSolution_Generated' folder please ensure that the 'Hide Generated Code' tick box is deselected.

3.3. Configuration of hardware and CANopen Manager

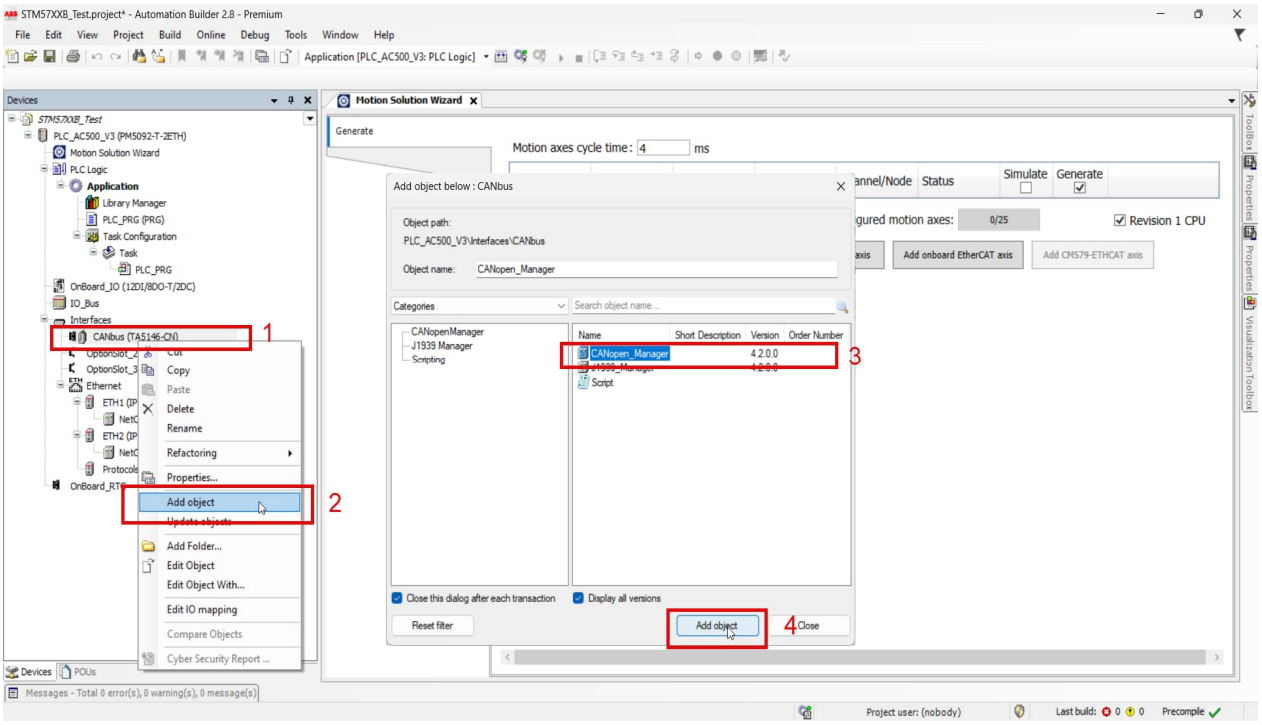
Now add the 'TA5146-CN' to the configuration, to do so we must select the slot we plan to use, in this case 'Slot 1', next right click and 'Add Object', select the TA5146-CN and press 'Replace Object'



Next we must set correct baud rate for the connected devices and network capabilities, for the STM57B5B, the correct baud rate is 500 kbit/s

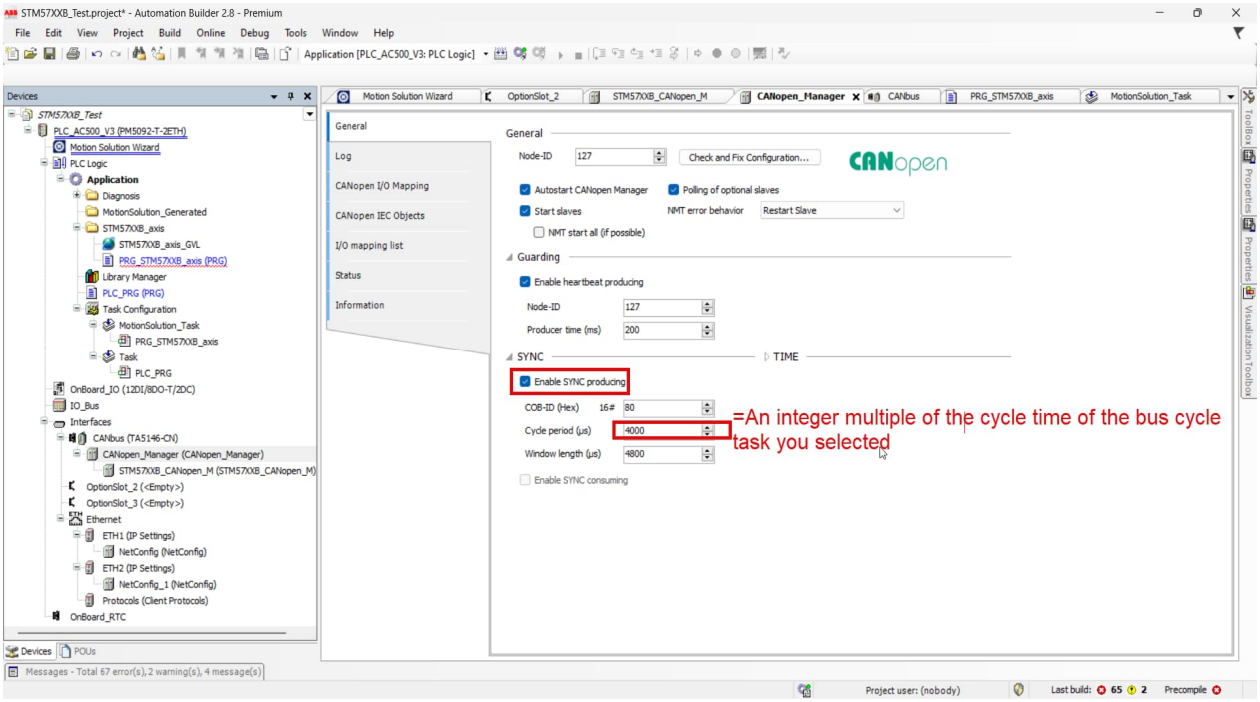


Now under the 'TA5146-CN' add the CANopen Manager, to do so right click on CANbus (TA5146-CN), 'Add Object', select the CANopen Manager and press 'Add Object'.

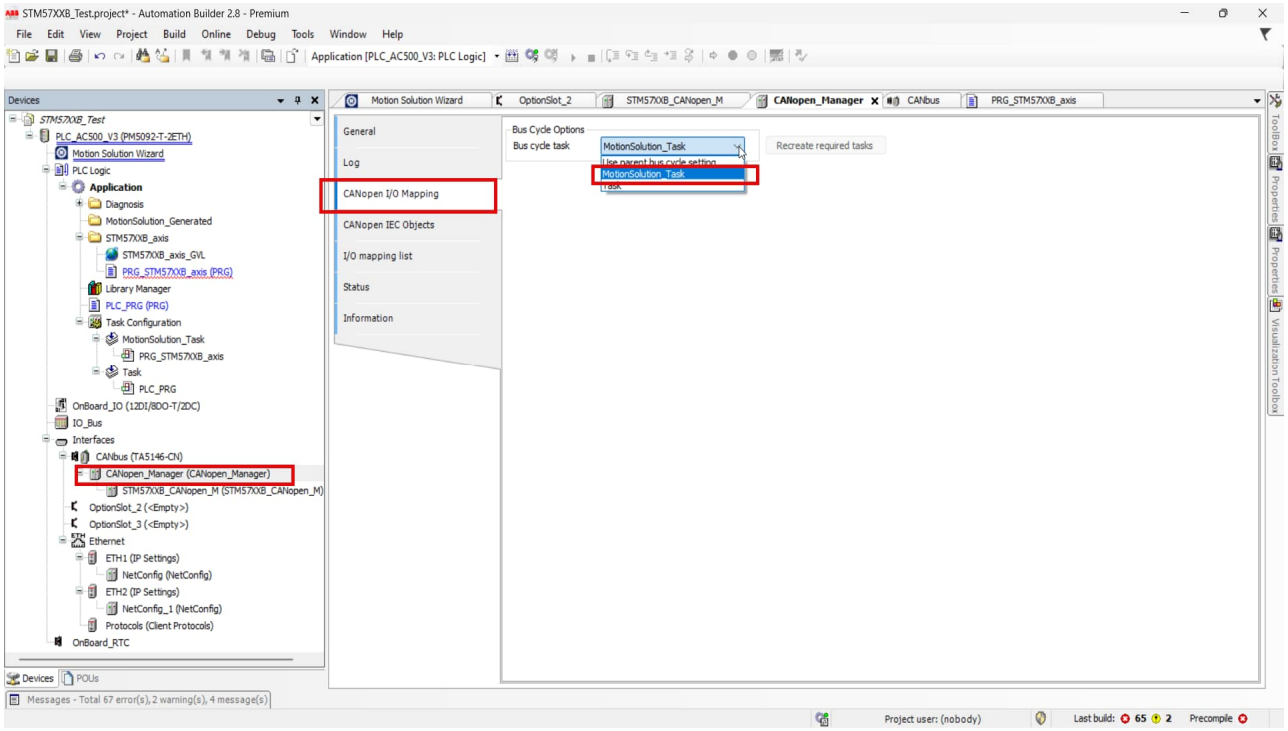


Now the manager is added configure it as per the network and device requirements.

First configure the 'General' settings. Here 'Enable SYNC producing' is selected along with a cycle period that matches the task cycle time it will be called by; in this instance we have selected 4000µs/4ms.

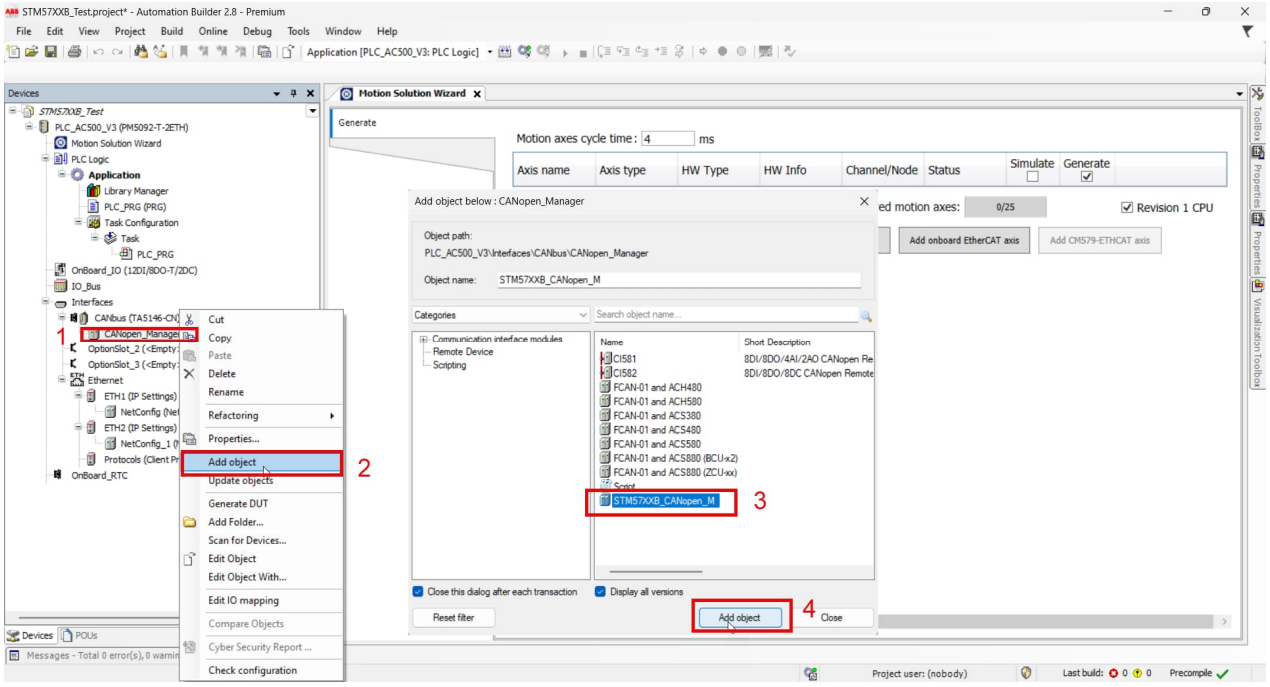


In line with the above setting, we can now navigate to the 'CANopen I/O Mapping page and select the task we want to use to drive these communications. As we use the 'Motion Solution Wizard' already it created the MotionSolution_Task which we are free to select here



3.4. Adding the Drive

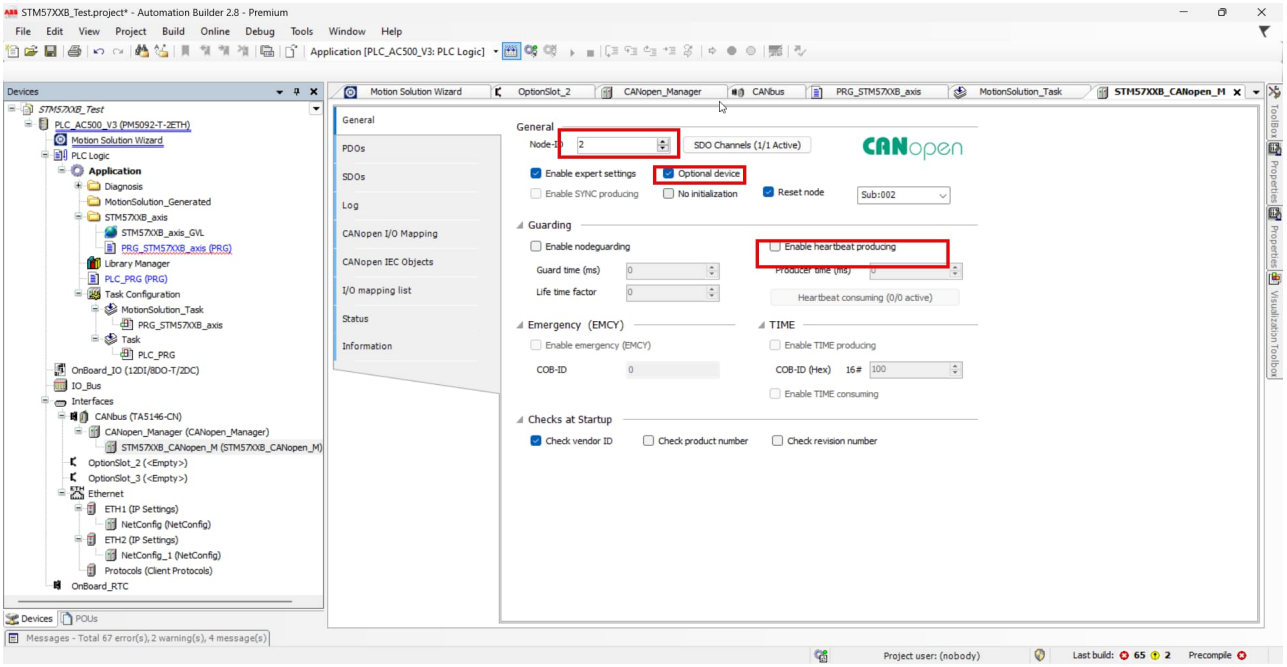
Now the CANbus hardware and manager are added we can add the Drive(s) to the configuration. To do so right click on CANopen Manager, select 'Add Object', select the 'STM57XXB_CANopen_M' (we previously imported) and press 'Add Object'



Now the drive object is added we can configure it, to do so complete the below steps

Firstly, set the basic communications settings, firstly open the drive object and select ‘General’

1. Set real Node-ID of drive
2. Select Optional device option
3. Deselect ‘Enable heartbeat producing.’



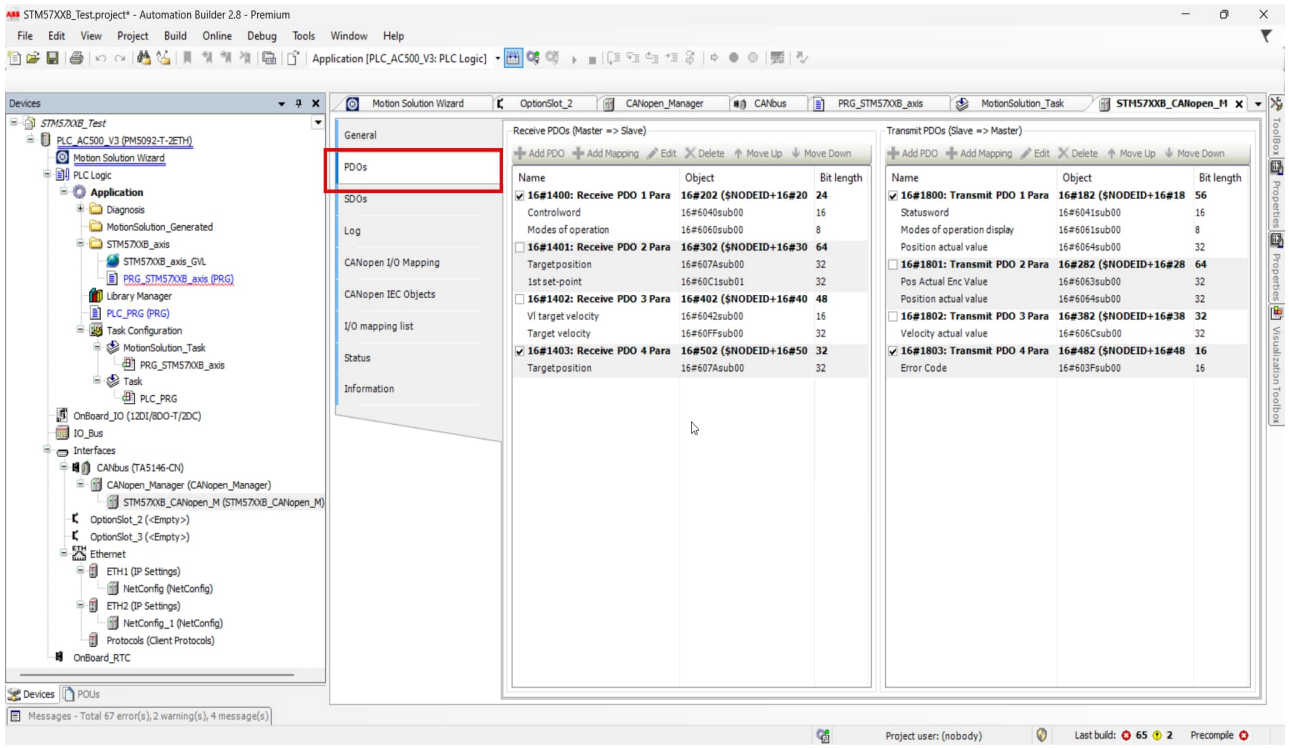
Now set the Drives PDO settings. To do so firstly open the drive object and select ‘PDOs’

For Receive PDOs

1. Select 16#1401 if you plan to use IP or CSP mode
2. Select 16#1403 if you plan to use CSP mode

For Transmit PDOs

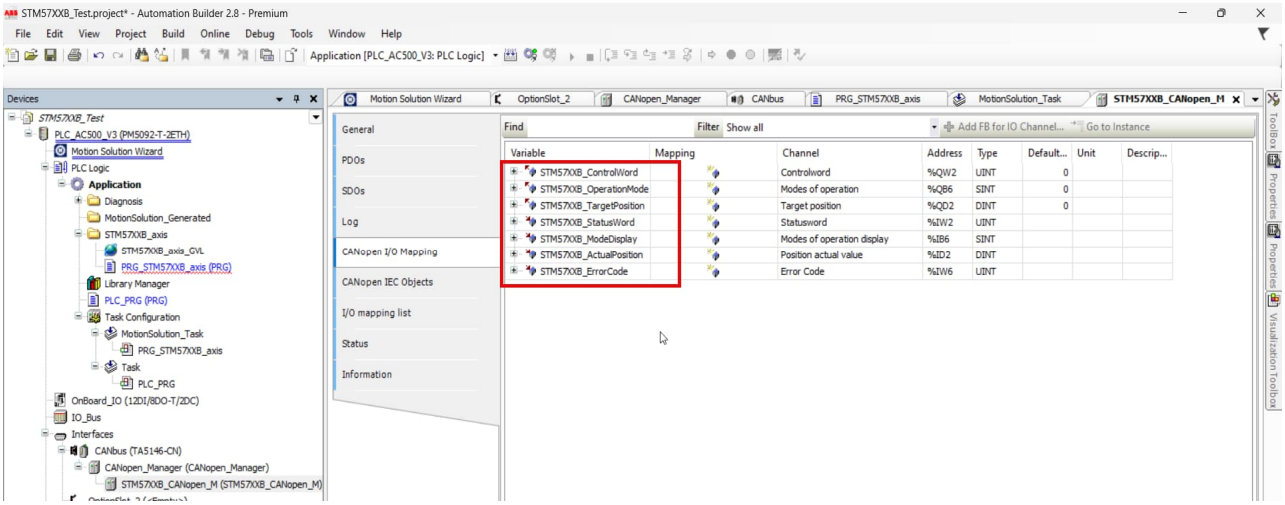
3. Select 16#1800 and 16#1803 in all cases.



Now we've selected the PDO Drive I/O define the variables that are associated with them. To do so go to the CANopen I/O Mapping tab.

To most closely match with the generated code, we should use the prefix the same as we selected as the '*axis name*' before in Section 4.2, followed by '_' followed by the *function*.

In this case the axis was called STM57XXB so for example the first Variable name becomes 'STM57XXB_ControlWord' and so on.

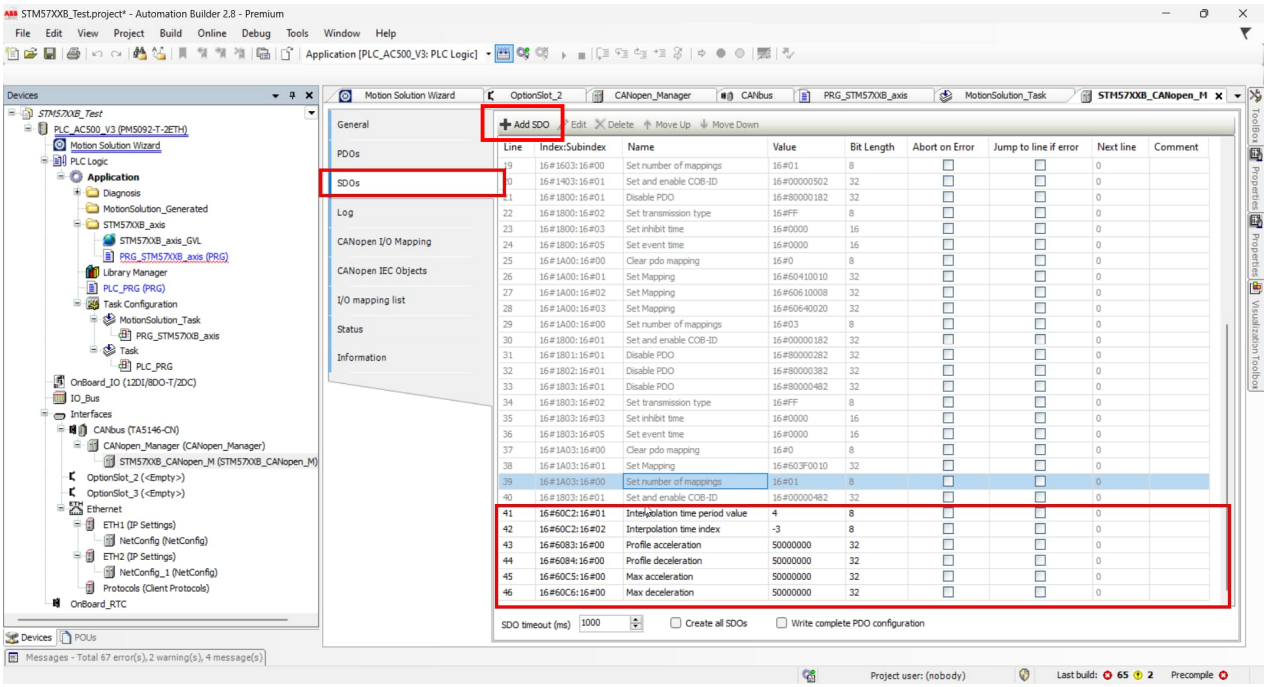


Note: you can also check in 'PRG_STM57XXB_axis' to look at the variable names used and copy them across directly.

As this is a simple drive which ideally needs little user configuration, some parameters are set via SDO. Set the Drives SDO settings to do so firstly open the drive object and select 'SDOs'.

To add additional SDO's which aren't shown as default use the '+ Add SDO' button as shown below

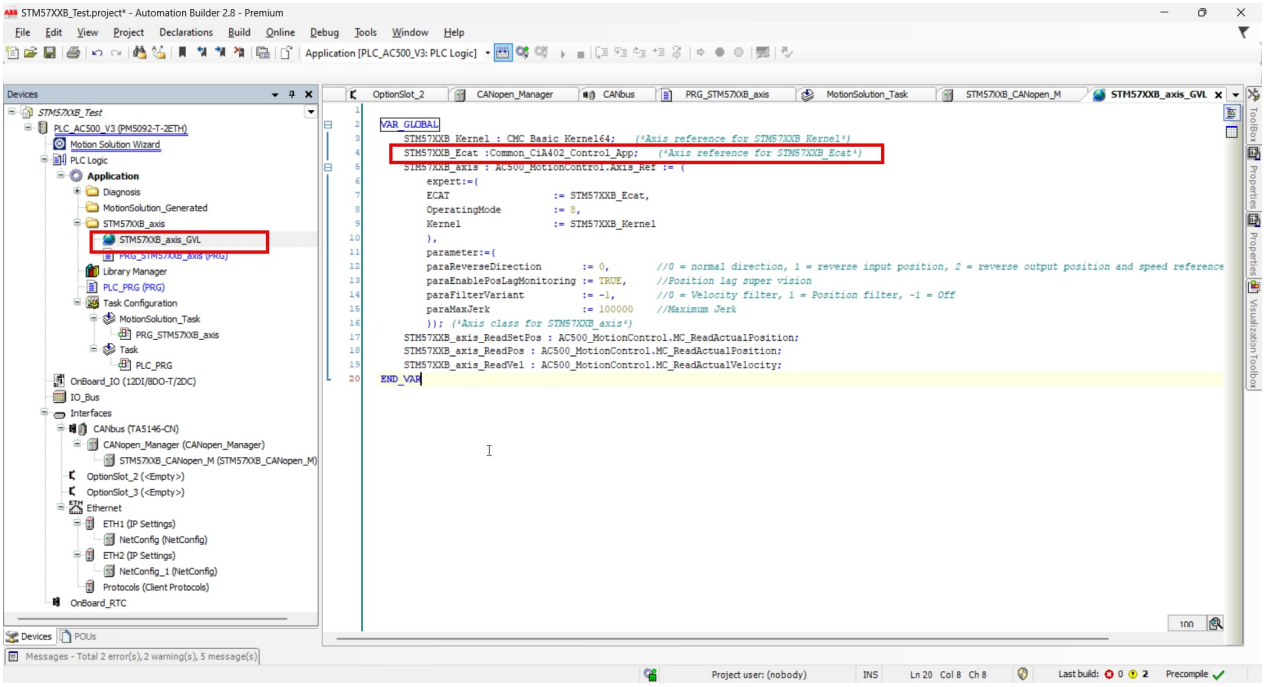
1. Select 16#160C2:16#01 Interpolation time period Value = 4 (4 cycles to use for interpolation)
2. Select 16#160C2:16#02 Interpolation time Index = -3 (as per the user manual)
3. Select 16#16083:16#0 Profile Accel = 50000000 (user units)
4. Select 16#16084:16#0 Profile Decel = 50000000 (user units)
5. Select 16#160C5:16#0 Max Accel = 50000000 (user units)
6. Select 16#160C6:16#0 Max Decel = 50000000 (user units)



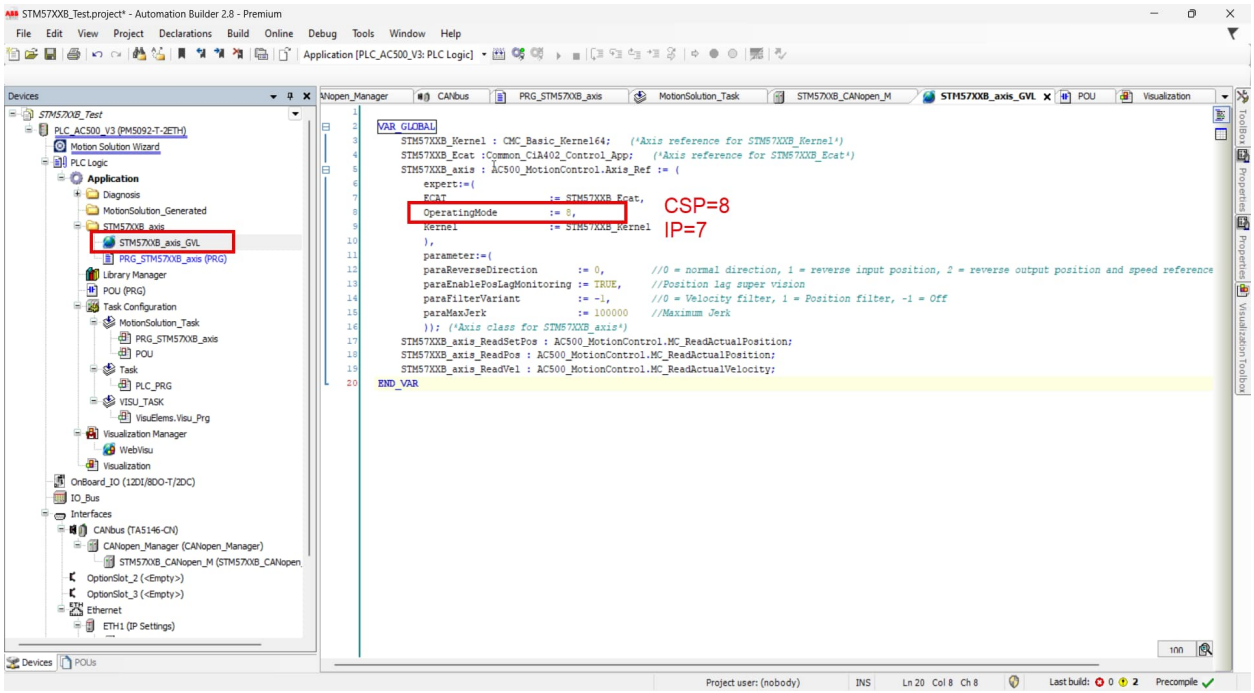
Note: The above SDOs are examples for this specific hardware and might vary in other devices

3.5. Programming

Next the drive interface can be selected (here is the first change the generated code). The key point here is that we will be changing the generated code declaration from an instance of '***ECAT***_CiA402_Control_App' to an instance of '***Common***_CiA402_Control_App'. This new function block keeps the handling of the CiA402 state machine but has no EtherCAT parts which makes it perfect for CANopen applications. To make the correct edits open the Global Variable Declaration for our axis 'STM57XXB_axis_GVL' and locate the line referring to the '**E530_EC**_Ecat : ***ECAT***_CiA402_Control_App; (*Axis reference for **E530_EC_Ecat***)' and then edit it to be '**STM57XXB_Ecat** : **Common**_CiA402_Control_App; (*Axis reference for **STM57XXB_Ecat***)'



Next set the desired drive operation mode. To make the required edit open the Global Variable Declaration for our axis 'STM57XXB_axis_GVL' and find the line 'OperatingMode := **8**,' and change the constant value if another control mode is desired.

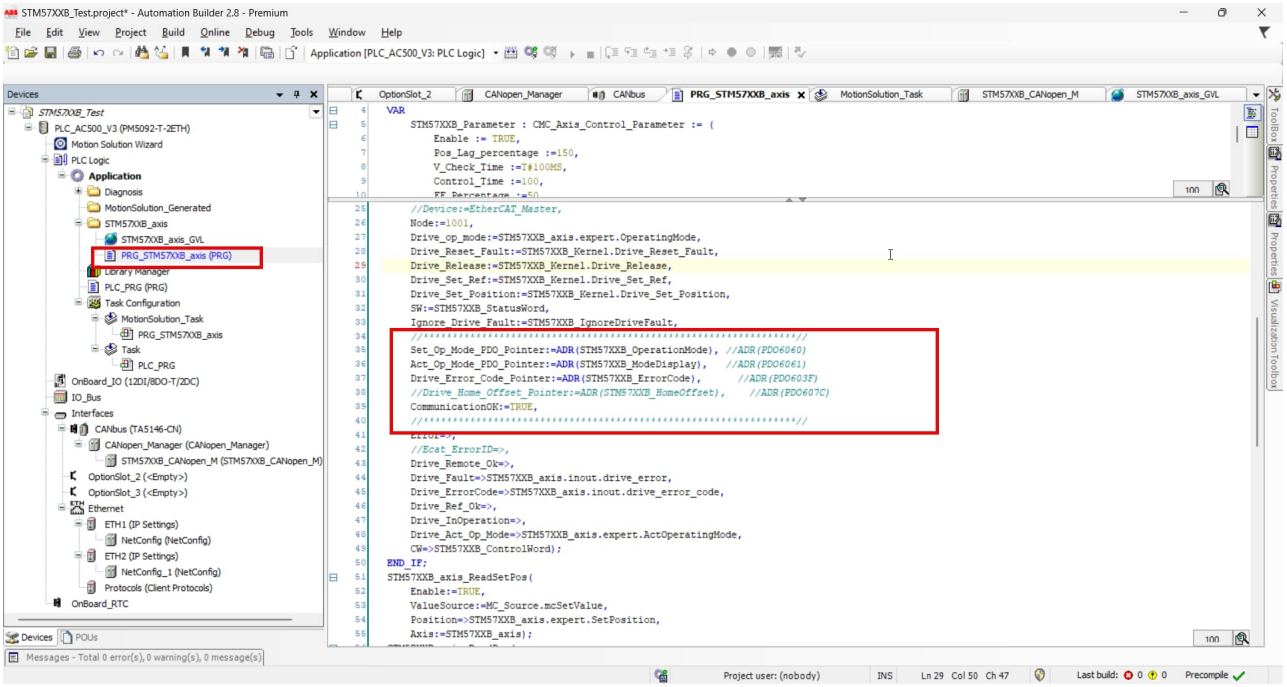


Now the unwanted (EtherCAT) code can be edited out. To make the required edits open the Program for our axis 'PRG_STM57XXB_axis' and find the lines mentioned below and add '//' in front of them to 'comment out' the undesired code as shown below:

```
1   STM57XXB_Parameter();
2   //IF EtherCAT_Master.xConfigFinished THEN
3       xRunCode :=TRUE;
4   //END_IF
5   IF xRunCode THEN
23  STM57XXB_Ecat(
24      En:=TRUE,
25      //Device:=EtherCAT_Master,
26      Node:=1001,
27      Drive_op_mode:=STM57XXB_axis.expert.OperatingMode,
28      Drive_Reset_Fault:=STM57XXB_Kernel.Drive_Reset_Fault,
29      Drive_Release:=STM57XXB_Kernel.Drive_Release,
30      Drive_Set_Ref:=STM57XXB_Kernel.Drive_Set_Ref,
31      Drive_Set_Position:=STM57XXB_Kernel.Drive_Set_Position,
32      SW:=STM57XXB_StatusWord,
33      Ignore_Drive_Fault:=STM57XXB_IgnoreDriveFault,
34      Error=>,
35      //Ecat_ErrorID=>,
36      Drive_Remote_Ok=>,
79  //STM57XXB_axis.expert.ActualTorque :=STM57XXB_ActualTorque;
80
81  IF NOT xFirstScan AND xRunCode THEN
82      STM57XXB_axis.parameter.paraMaxVelocityAppl :=6000;
83      STM57XXB_axis.parameter.paraMaxAccelerationAppl :=10000;
84      STM57XXB_axis.parameter.paraMaxDecelerationAppl :=10000;
85      xFirstScan :=TRUE;
86  END_IF;
87  // STM57XXB_TouchProbeFunction :=STM57XXB_axis.expert.TouchProbeFunction;
88  // STM57XXB_axis.expert.TouchProbeStatus :=STM57XXB_TouchProbeStatus;
89  // STM57XXB_axis.expert.TouchProbePositionPos1 :=STM57XXB_TouchProbePositionPos1;
90  // STM57XXB_axis.expert.TouchProbePositionPos2 :=STM57XXB_TouchProbePositionPos2;
```

We must also add some new code to handle the data exchange between the drives PDO data and the function blocks. To make the required edits open the Program for our axis 'PRG_STM57XXB_axis' and add the lines mentioned below under the declaration for 'STM57XXB_Ecat(' in lines 35 to 39:

- Set_Op_Mode_PDO_Pointer:=ADR(STM57XXB_OperationMode), //ADR(PDO6060)
- Act_Op_Mode_PDO_Pointer:=ADR(STM57XXB_ModeDisplay), //ADR(PDO6061)
- Drive_Error_Code_Pointer:=ADR(STM57XXB_ErrorCode), //ADR(PDO603F)
- CommunicationOK:=TRUE,



These lines connect the new variable we created to the relevant inputs in the instance of 'Common_CiA402_Control_App' we created earlier.

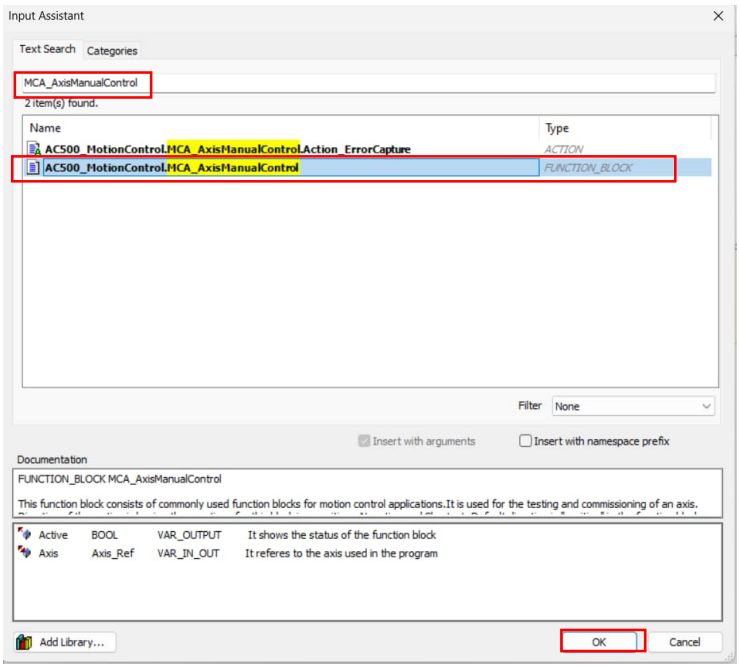
If any additional axes are required the steps mentioned can be repeated using new drive instance names each time. Now all the edits are done the code is ready to use for a single axis.

4. Commissioning

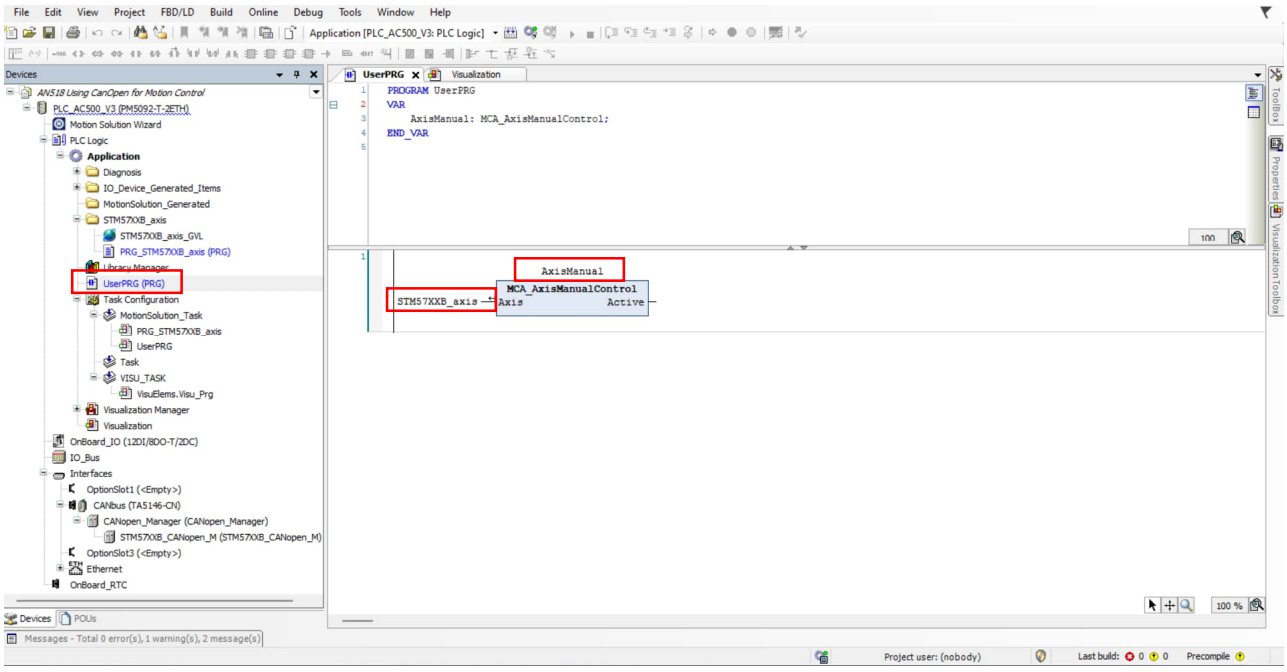
Now we can create a test program to check the application works correctly, as the previous steps have allowed us to connect the Kernel and the hardware, we can use most of the MC or MCA function blocks to control the axes we have set up (other than those dealing with EtherCAT). In this case for a quick and simple test we will use the function block 'MCA_AxisManualControl' which contains a method to enable and jog the axis combined with a simple visual interface, all in once simple implementation.

4.1. Creating a Test program

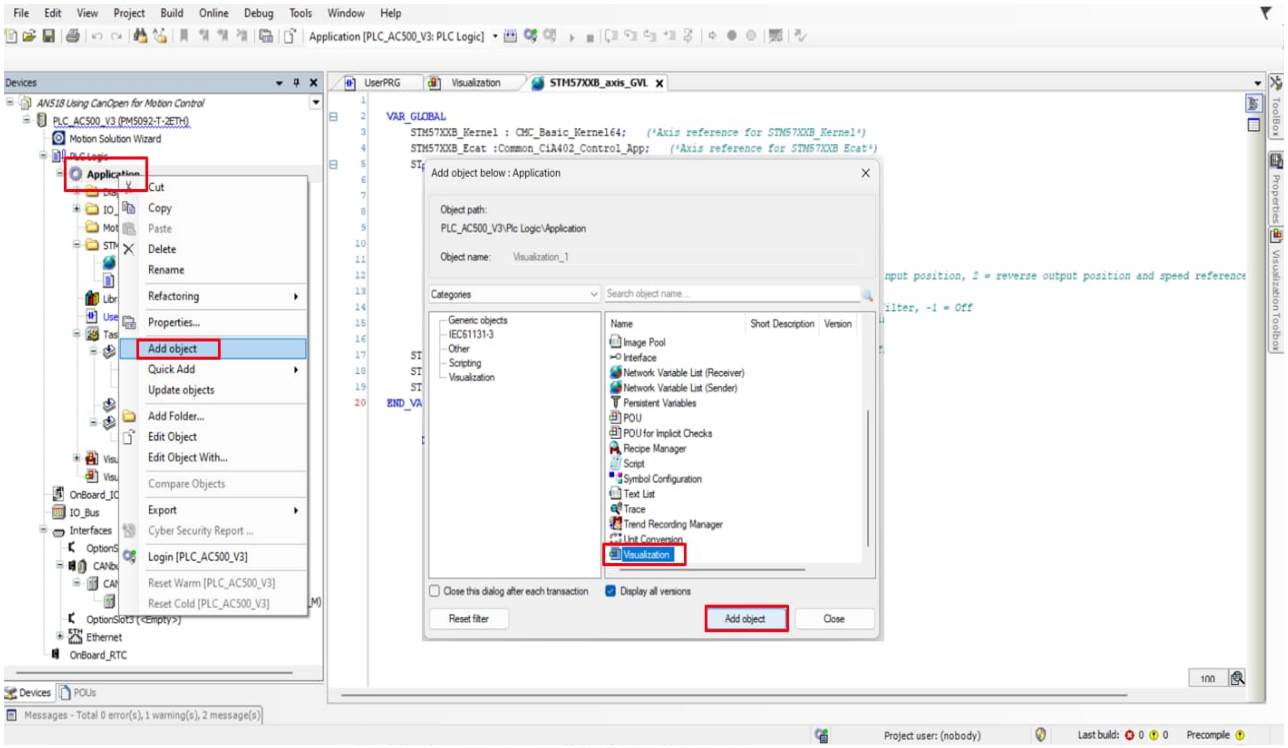
Add or create a PRG which we will use to run the axis program. In this instance the PRG is called 'UserPRG'. Next insert a 'New Box' into the network, select 'Test Search' and type in 'MCA_AxisManualControl' find in the list then press 'Ok'



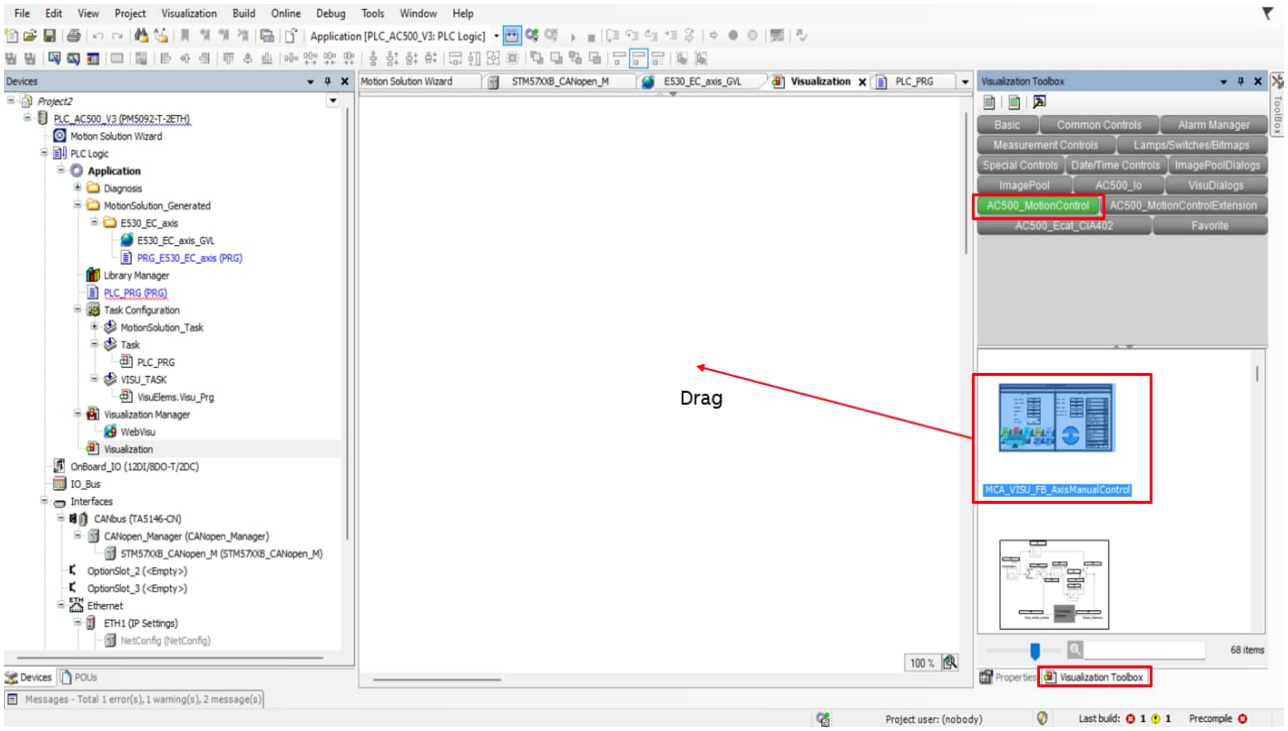
Now give it an instance name of 'AxisManual'. Finally connect the input 'Axis' to the axis reference for the CANopen axis. This can be found in the Global variable declaration for our axis 'STM57XXB_axis_GVL'



Now we must add a visualization to run the test. Right click on Application, Add Object, Select 'Visualization' from the list and select 'Add Object' and 'Add' from any pop ups that appear afterwards.

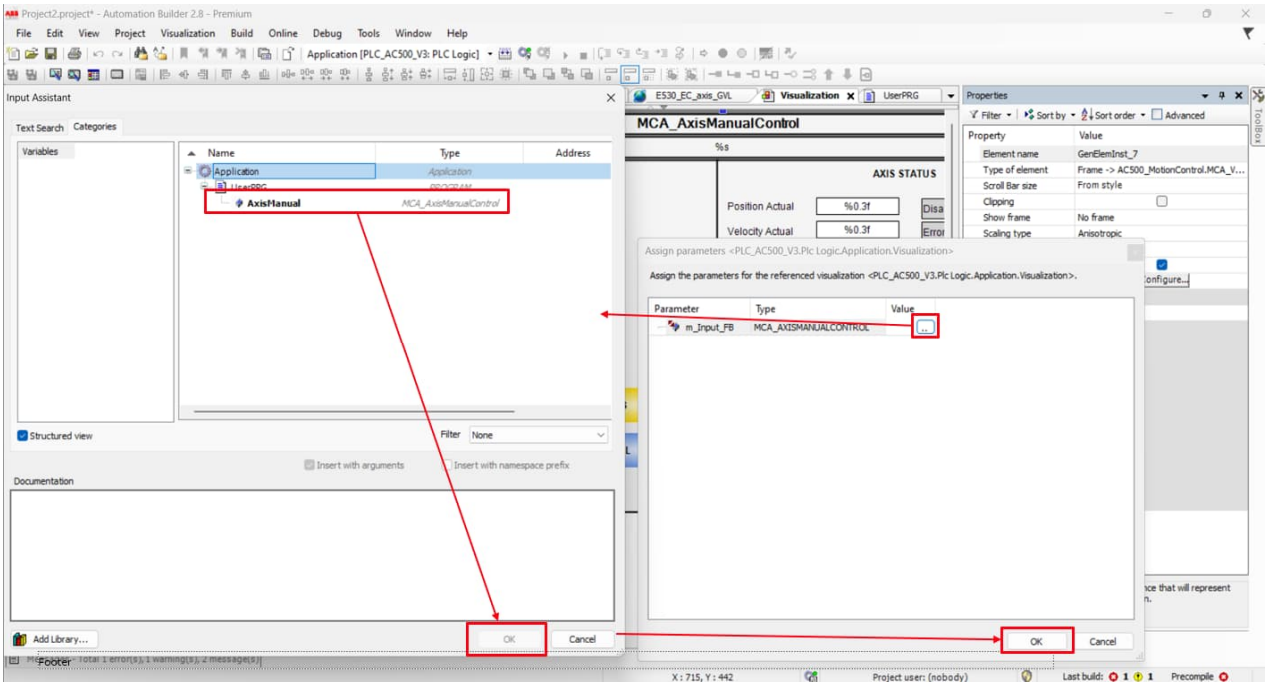


Next we must add an instance of the Visualization we require. Open up the Visualization, select Visualization Toolbox, 'AC500_MotionControl' tab and find the Visualization we require which in this case is 'MCA_VISU_FB_AxisManualControl' then we can drag it into the work area.

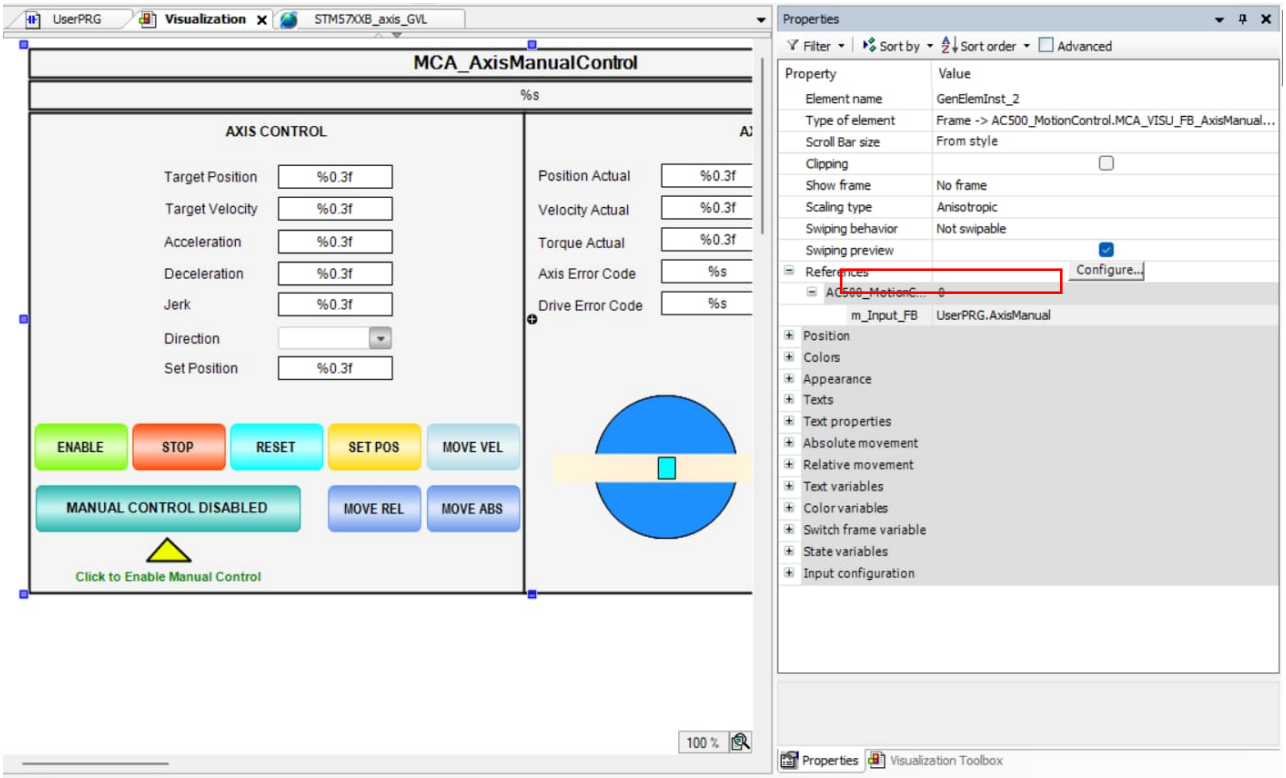


This will automatically generate a pop up which will prompt the user to connect the instance of the visualization to an instance of the Function Block.

To add the connection first click on the '…', then as we added this instance in 'UserPRG (PRG)' the user should look here for the instance, then click 'OK' and click 'OK' to close this part of the process.



You can see that these steps have configured the references within the Visualization.



Now the test program is completed and is ready to be used

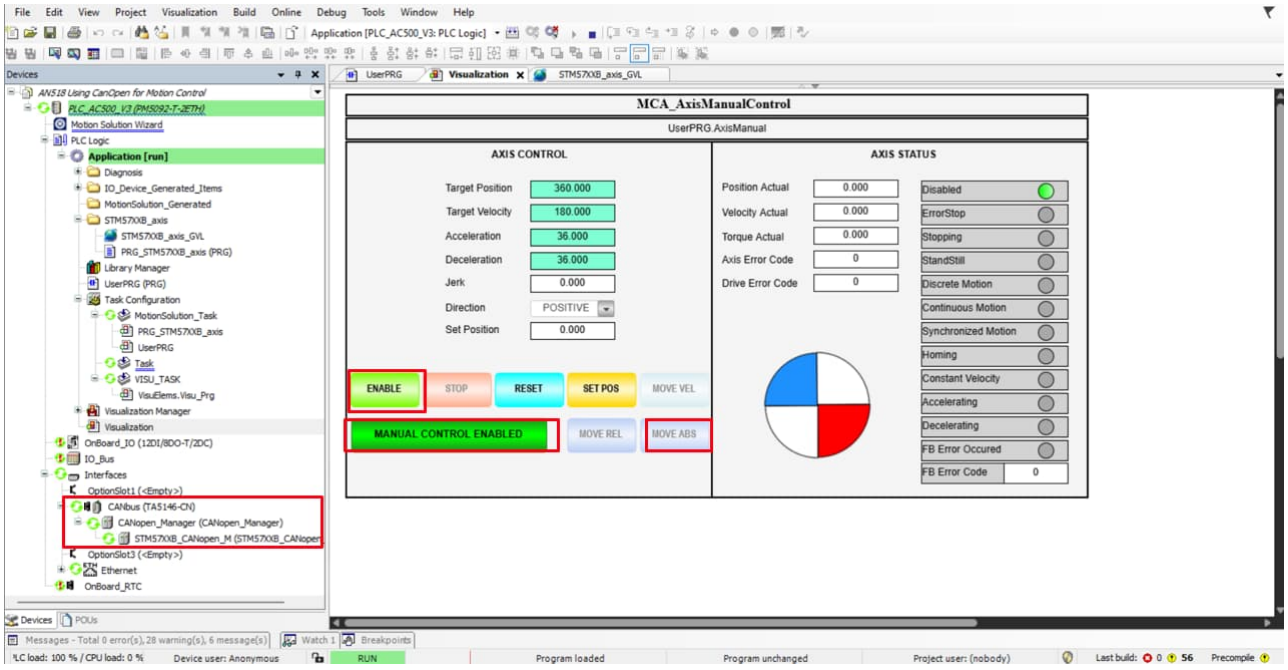
4.2. Testing the solution

Now the code can be compiled via Build > Generate Code, and as long as no errors downloaded. To down Online > Login (using the users preferred method), then selecting 'Create Boot Project' and then putting the program into run.

Once the program is running successfully then the RUN LED should be illuminated Green, and the 'CAN' LED on the 'TA5146-CN' should be illuminated 'Orange' this shows the hardware and program are running successfully.

In addition, there are hardware diagnostics built into the Device tree where the user can see there are green circles ( ) to show the health status of a particular branch as in the below picture.

If no errors exist, then the user is free 'Enable manual control,' enable the drive and to enter the required move profile parameters and start issuing moves.



The program to match this example is included along with this document.

Contact us.

For more information, please contact your
local ABB representative or one of the following:

new.abb.com/drives/low-voltage-ac/servo-products
new.abb.com/drives
new.abb.com/drivespartners
new.abb.com/PLC