

Getting Started Guide

MCz201, MCz601& MCe200

Machine Controllers featuring CODESYS® Control RTE real-time software PLC Runtime

Part Number: 0478-0683-01
Issue: 1

Original Instructions

For the purposes of compliance with the EU Machinery Directive 2006/42/EC, the English version of this manual is the Original Instructions. Manuals in other languages are Translations of the Original Instructions.

Documentation

Manuals are available to download from the following locations: http://www.drive-setup.com/ctdownloads. The information contained in this manual is believed to be correct at the time of printing and does not form part of any contract. The manufacturer reserves the right to change the specification of the product and its performance, and the contents of the manual, without notice.

Warranty and Liability

In no event and under no circumstances shall the manufacturer be liable for damages and failures due to misuse, abuse, improper installation, or abnormal conditions of temperature, dust, or corrosion, or failures due to operation outside the published ratings. The manufacturer is not liable for consequential and incidental damages. Contact the supplier of the drive for full details of the warranty terms.

Environmental policy

Control Techniques Ltd operates an Environmental Management System (EMS) that conforms to the International Standard ISO 14001. Further information on our Environmental Policy can be found at: http://www.drive-setup.com/environment

Restriction of Hazardous Substances (RoHS)

The products covered by this manual comply with European and Intern ational regulations on the Restriction of Haz-hazardous Substances including EU directive 2011/65/EU and the Chinese Administrative Measures for Restriction of Hazardous Substances in Electrical and Electronic Products

Disposal and Recycling (WEEE)



When electronic products reach the end of their useful life, they must not be disposed of along with domestic waste but should be recycled by a specialist recycler of electronic equipment. Control Techniques products are designed to be easily dismantled into their major component parts for efficient recycling. The majority of materials used in the product are suitable for recycling.

Product packaging is of good quality and can be re-used. Large products are packed in wooden crates. Smaller products are packaged in strong cardboard cartons which have a high recycled fibre content. Cartons can be re-used and recycled. Polythene, used in protective film and bags for wrapping the product, can be recycled. When preparing to recycle or dispose of any product or packaging, please observe local legislation and best practice.

REACH legislation

EC Regulation 1907/2006 on the Registration, Evaluation, Authorisation and restriction of Chemicals (REACH) requires the supplier of an article to inform the recipient if it contains more than a specified proportion of any substance which is considered by the European Chemicals Agency (ECHA) to be a Substance of Very High Concern (SVHC) and is therefore listed by them as a candidate for compulsory authorisation.

Further information on our compliance with REACH can be found at: http://www.drive-setup.com/reach

Registered Office

Nidec Control Techniques Ltd, The Gro, Newtown, Powys, SY16 3BE, UK. Registered in England and Wales. Company Reg. No. 01236886

Copyright

The contents of this publication are believed to be correct at the time of printing. In the interests of a commitment to a policy of continuous development and improvement, the manufacturer reserves the right to change the specification of the product or its performance, or the contents of the guide, without notice.
All rights reserved. No parts of this guide may be reproduced or transmitted in any form or by any means, electrical or mechanical including photocopying, recording or by an information storage or retrieval system, without permission in writing from the publisher.
Copyright © November 2021 Nidec Control Techniques Ltd

Contents

1 Safety Information 7
1.1 Warnings, Cautions and Notes 7
1.2 Important safety information. Hazards. Competence of designers and installers 7
1.3 Responsibility 7
1.4 Compliance with regulations 8
1.5 Electrical hazards 8
1.6 Stored electrical charge 8
1.7 Mechanical hazards 8
1.8 Access to equipment 9
1.9 Environmental limits 9
1.10 Hazardous environments 9
1.11 Motor 9
1.12 Mechanical brake control 9
1.13 Adjusting parameters 9
1.14 Electromagnetic compatibility (EMC) 9
2 Introduction 11
3 Before you begin 11
3.1 Prerequisite 11
3.2 Online help and training 11
4 Device targets 11
4.1 Select the desired target when creating a project 11
4.2 Converting an existing example to a different target type 12
5 EtherCAT Master 13
5.1 New project 13
5.2 Adding an ESI file 13
5.3 Adding the EtherCAT master and EtherCAT slave axis 15
5.4 Add the program code 16
5.5 Configure the PDO mappings 21
5.6 Configure mappings to the program variables 24
5.7 Test your application 24
5.8 Supplementary - Access to slot parameters 26
6 EtherNet/IP Scanner 27
6.1 New project 27

6.2 EDS file 27
6.3 Adding the EtherNet/IP Scanner and Device 27
6.4 Add the program code 30
6.5 Configure mappings to the program variables 31
6.6 Test your application 32
7 EtherNet/IP Device 33
7.1 New project 33
7.2 Adding the EtherNet/IP Device 33
7.3 Add the program code 35
7.4 Configure the mappings 36
7.5 Generate EDS file for Studio 5000 36
7.6 Create a Studio 5000 application 37
7.7 Test your application 37
8 Profinet IO Device 38
8.1 New project 38
8.2 Generate GSDML file for TIA Portal 38
8.3 Create a TIA portal application 38
8.4 Adding the Profinet Device 39
8.5 Add the program code 40
8.6 Configure the mappings 40
8.7 Test your application 41
9 Modbus TCP/IP Client 41
9.1 New project 41
9.2 Adding the Modbus TCP Master 42
9.3 Add the program code 47
9.4 Configure the mappings 47
9.5 Test your application 48
9.6 Supplementary - 32-bit drive parameters 49
9.7 Supplementary - Access to slot parameters 49
10 OPC UA server 51
10.1 New project 51
10.2 New project 51
10.3 Adding the program 51
10.4 Adding the Symbol Configuration 51
10.5 Configuring UaExpert 53
11 Retained variables 57

11.1 VAR RETAIN 57
11.2 VAR PERSISTENT RETAIN 57
11.3 Critical user data 58
12 Glossary of terms 59

1 Safety Information

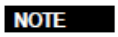
1.1 Warnings, Cautions and Notes



A **Warning** contains information which is essential for avoiding a safety hazard.



A **Caution** contains information which is necessary for avoiding risk of damage to the equipment and motor.



A **Note** contains information which helps to ensure correct operation of the product.



The user must refer to the "MCe200 User Guide" and the "MCz201 and MCz601 User Guide" for safety related information.



The user must refer to the relevant drive manual for operating and safety instructions before proceeding.

1.2 Important safety information. Hazards. Competence of designers and installers

This guide applies to products which control electric motors either directly (drives) or indirectly (controllers, option modules and other auxiliary equipment and accessories). In all cases the hazards associated with powerful electrical drives are present, and all safety information relating to drives and associated equipment must be observed.

Specific warnings are given at the relevant places in this guide.

Drives and controllers are intended as components for professional incorporation into complete systems. If installed incorrectly they may present a safety hazard. The drive uses high voltages and currents, carries a high level of stored electrical energy, and is used to control equipment which can cause injury. Close attention is required to the electrical installation and the system design to avoid hazards either in normal operation or in the event of equipment malfunction. System design, installation, commissioning/start-up and maintenance must be carried out by personnel who have the necessary training and competence. They must read this safety information and this guide carefully.

1.3 Responsibility

It is the responsibility of the installer to ensure that the equipment is installed correctly with regard to all instructions given in this guide. They must give due consideration to the safety of the complete system, so as to avoid the risk of injury both in normal operation and in the event of a fault or of reasonably foreseeable misuse.

The manufacturer accepts no liability for any consequences resulting from inappropriate, negligent or incorrect installation of the equipment.

1.4 Compliance with regulations

The installer is responsible for complying with all relevant regulations, such as national wiring regulations, accident prevention regulations and electromagnetic compatibility (EMC) regulations. Particular attention must be given to the cross-sectional areas of conductors, the selection of fuses or other protection, and protective ground (earth) connections.

This guide contains instructions for achieving compliance with specific EMC standards.

All machinery to be supplied within the European Union in which this product is used must comply with the following directives:

2006/42/EC Safety of machinery.

2014/30/EU: Electromagnetic Compatibility.

1.5 Electrical hazards

The voltages used in the drive can cause severe electrical shock and/or burns, and could be lethal. Extreme care is necessary at all times when working with or adjacent to the drive. Hazardous voltage may be present in any of the following locations:

- AC and DC supply cables and connections
- Output cables and connections
- Many internal parts of the drive, and external option units

Unless otherwise indicated, control terminals are single insulated and must not be touched.

The supply must be disconnected by an approved electrical isolation device before gaining access to the electrical connections.

The STOP and Safe Torque Off functions of the drive do not isolate dangerous voltages from the output of the drive or from any external option unit.

The drive must be installed in accordance with the instructions given in this guide. Failure to observe the instructions could result in a fire hazard.

1.6 Stored electrical charge

The drive contains capacitors that remain charged to a potentially lethal voltage after the AC supply has been disconnected. If the drive has been energized, the AC supply must be isolated at least ten minutes before work may continue.

1.7 Mechanical hazards

Careful consideration must be given to the functions of the drive or controller which might result in a hazard, either through their intended behaviour or through incorrect operation due to a fault. In any application where a malfunction of the drive or its control system could lead to or allow damage, loss or injury, a risk analysis must be carried out, and where necessary, further measures taken to reduce the risk - for example, an over-speed protection device in case of failure of the speed control, or a fail-safe mechanical brake in case of loss of motor braking.

With the sole exception of the Safe Torque Off function, none of the drive functions must be used to ensure safety of personnel, i.e. they must not be used for safety-related functions.

The Safe Torque Off function may be used in a safety-related application. The system designer is responsible for ensuring that the complete system is safe and designed correctly according to the relevant safety standards.

The design of safety-related control systems must only be done by personnel with the required training and experience. The Safe Torque Off function will only ensure the safety of a machine if it is correctly incorporated into a complete safety system. The system must be subject to a risk assessment to confirm that the residual risk of an unsafe event is at an acceptable level for the application.

1.8 Access to equipment

Access must be restricted to authorized personnel only. Safety regulations which apply at the place of use must be complied with.

1.9 Environmental limits

Instructions in this guide regarding transport, storage, installation and use of the equipment must be complied with, including the specified environmental limits. This includes temperature, humidity, contamination, shock and vibration. Drives must not be subjected to excessive physical force.

1.10 Hazardous environments

The equipment must not be installed in a hazardous environment (i.e. a potentially explosive environment).

1.11 Motor

The safety of the motor under variable speed conditions must be ensured.

To avoid the risk of physical injury, do not exceed the maximum specified speed of the motor.

Low speeds may cause the motor to overheat because the cooling fan becomes less effective, causing a fire hazard. The motor should be installed with a protection thermistor. If necessary, an electric forced vent fan should be used.

The values of the motor parameters set in the drive affect the protection of the motor. The default values in the drive must not be relied upon. It is essential that the correct value is entered in the Motor Rated Current parameter.

1.12 Mechanical brake control

Any brake control functions are provided to allow well co-ordinated operation of an external brake with the drive. While both hardware and software are designed to high standards of quality and robustness, they are not intended for use as safety functions, i.e. where a fault or failure would result in a risk of injury. In any application where the incorrect operation of the brake release mechanism could result in injury, independent protection devices of proven integrity must also be incorporated.

1.13 Adjusting parameters

Some parameters have a profound effect on the operation of the drive. They must not be altered without careful consideration of the impact on the controlled system. Measures must be taken to prevent unwanted changes due to error or tampering.

1.14 Electromagnetic compatibility (EMC)

Installation instructions for a range of EMC environments are provided in the relevant Power Installation Guide. If the installation is poorly designed or other equipment does not comply with suitable standards for EMC, the product might cause or suffer from disturbance due to electromagnetic interaction with other equipment. It is the responsibility of the installer to

ensure that the equipment or system into which the product is incorporated complies with the relevant EMC legislation in the place of use.

2 Introduction

This document details simple examples utilising the MCe200 machine controller with differing fieldbus technologies. All examples may be simply adapted for use with the MCz201 or MCz601 Industrial PCs

3 Before you begin

3.1 Prerequisite

Before beginning to explore the examples in this document please refer to section 7 of the *MCe200 User Guide* or section 8 of the *MCz201 and MCz601 User Guide*. Both documents may be downloaded from Support Suite or www.controltechniques.com.

Upon completion of the sections detailed above, you should have downloaded the required target (DDF) as required, set up your controller, gained an understanding of how to create projects using the CODESYS® Development System IDE, set the communications path, written variables, and successfully created and tested your first application.

3.2 Online help and training

Click Help>Contents from the toolbar within the CODESYS Development System to launch the online help. You may search from the search box to open documentation on all the features, settings, and IEC 61131-3 language information.

There are rich online forums dedicated to the CODESYS Development System, and many websites offering examples and tips.

Control Techniques periodically offers training through the Control Techniques Academy and online, please contact your supplier for further information.

4 Device targets

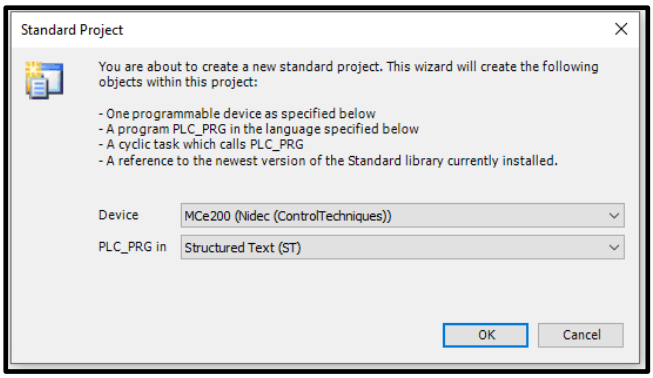
The examples may be used with either the MCz201, MCz601 or MCe200 device targets.

Depending on the version of the CODESYS Development System you are using the MCz201, MCz601 and MCe200 device targets (DDF files) may not be included, if they are not included then they can be downloaded from Support Suite or are available from your local Drive Centre or supplier.

Once downloaded the required device can be added to the CODESYS Development System's *Device Repository*. To do so, click on "*Tools>Device Repository…"* on the toolbar. Click "*Install..."* and navigate to the location of the downloaded DDF file, click on the file and then click *Open.* (see section 3).

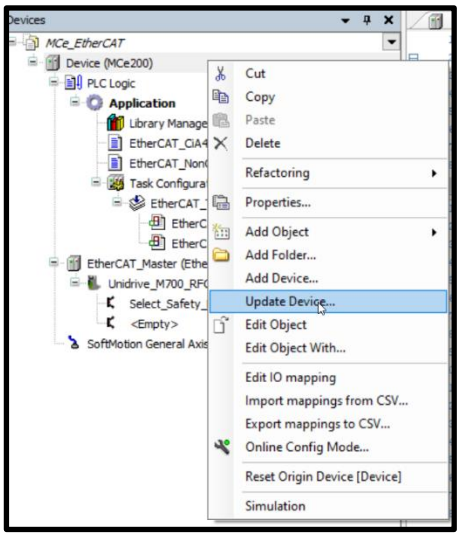
4.1 Select the desired target when creating a project

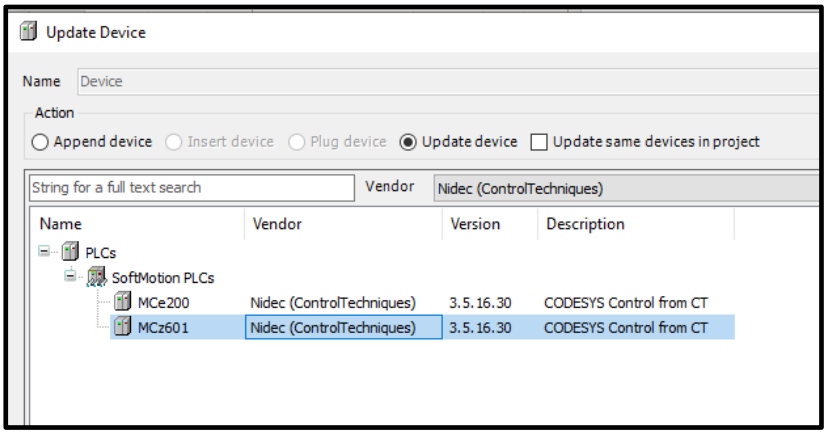
When creating a new project (see section 3) select the desired target (MCe200, MCz201 or MCz601) from the *Standard Project* dialogue.



4.2 Converting an existing example to a different target type

When an existing project is open and you wish to change the device target, right-click on the controller in the *Devices tree* and select *Update Device*. Select the required device in the dialogue.





5 EtherCAT Master

Typically EtherCAT may be used in three ways to control a drive.

- CiA402 drive profile, where standardized profile objects control the drive.
- By the user directly writing to the drive's parameters represented as objects, such as the drive's control word, speed references or AMC parameters etc.
- Using the SoftMotion PLCopen function blocks provided in the CODESYS Development System.

This example was created with the M700 drive in RFC-S mode and SI-EtherCAT option module. It demonstrates non-cyclic data transfer, cyclic data transfer and the CiA402 drive profile. A simple example of using the SoftMotion PLCopen function blocks is included in section 7 of the *MCe200 User Guide*, or section 8 of the *MCz201 and MCz601 User Guide*.

NOTE The companion project archive *MCeEtherCAT.projectarchive* is available to download from Support Suite or is available from your local Drive Centre or supplier. An archive imports to your system along with the application code and fieldbus configuration, all the required devices (DDFs and ESI files etc).

It is assumed that the default IP address and subnet mask (192.168.1.1, 255.255.255.0) for port 1 on the MCe200, or X102 on the MCzx01 are configured within the MCxxxx, and an IP address in the same subnet has been configured on the user's PC or laptop. The user's PC should be connected to port 1 (LAN A) of the MCz200 or X102 of the MCzx01. Port 2 (LAN B) of the MCe200 or X108 of the MCzx01 should be connected to the left-hand port (A) of the SI-EtherCAT module.

If the example archive (*MCeEtherCAT.projectarchive*) is converted to an MCzx01 target then the network must be scanned in the *Communications Settings* tab of the PLC, and subsequently section 5.3 will need to be followed to set the EtherCAT port to X108.

5.1 New project

If you are not installing the project archive (see section 5) then create a new project for your desired device target type (add the device target DDF as required) and configure the gateway by scanning the network (set the network path in the *Communications Settings* tab of the PLC) to ensure your device is visible in the CODESYS Development System. (See section 3.1).

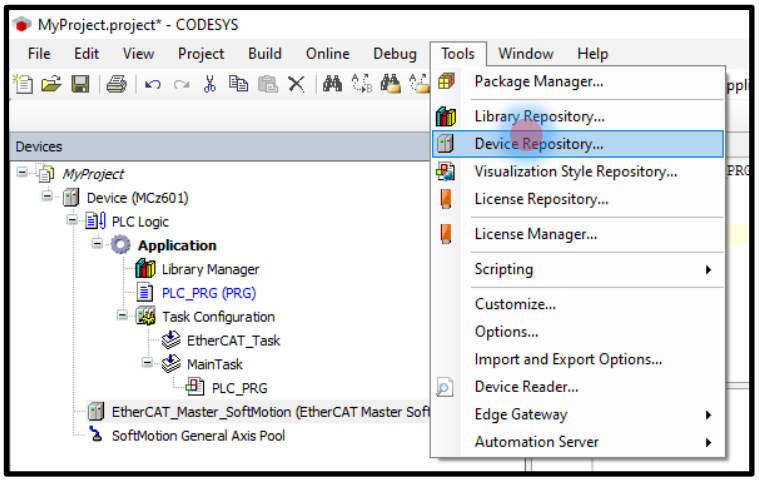
The example and its associated images are based upon the MCe200, but users may choose to use the MCzx01.

5.2 Adding an ESI file

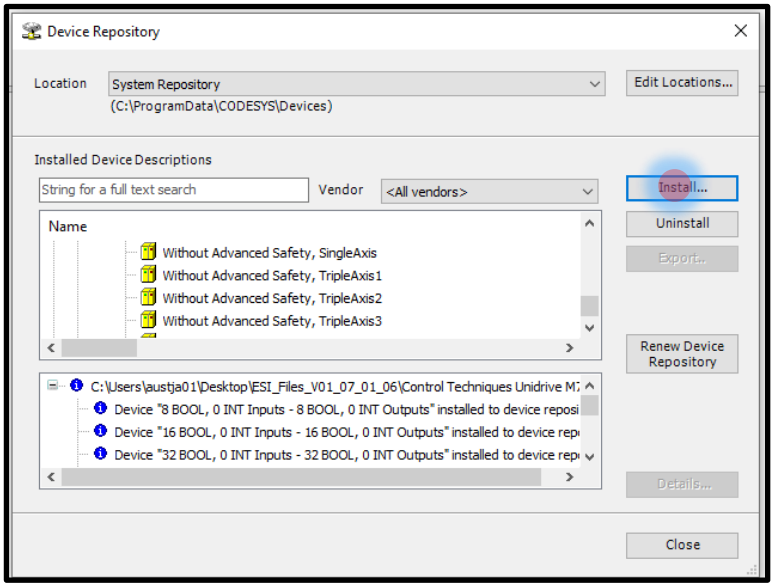
ESI files for Control Techniques drives are available from your supplier or www.controltechniques.com. Select the version that is recommended for the SI-EtherCAT firmware you intend to use.

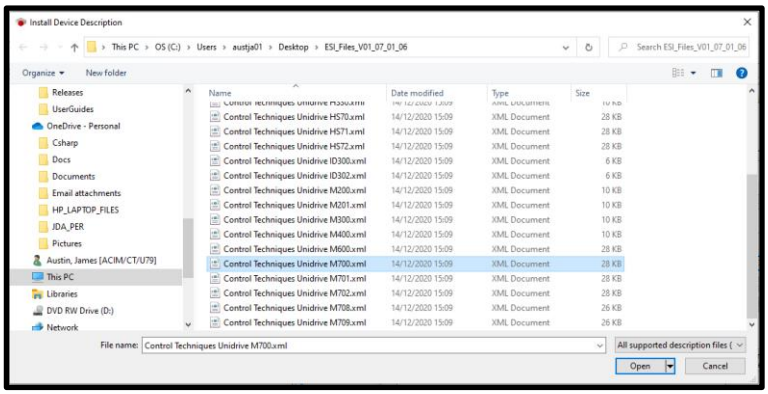
It is assumed the user has downloaded the ESI files and unzipped them to a suitable location on their PC.

From the toolbar click on "*Tools>Device Repository…*"



Click the *Install* button and navigate to the location of the ESI files previously downloaded and unzipped. Select the required file, e.g. *Control Techniques Unidrive M700.xml*. Once installed click the *Close* button.



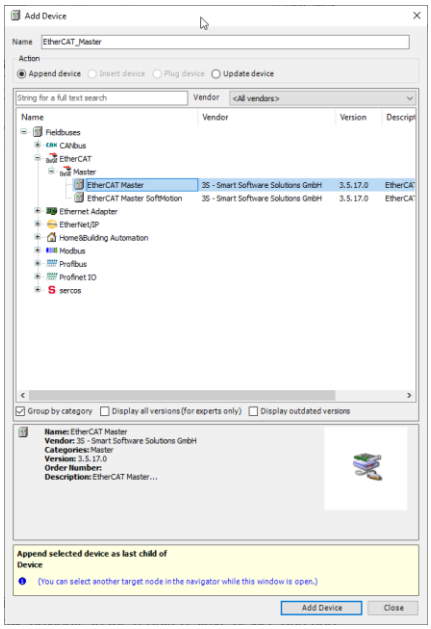


5.3 Adding the EtherCAT master and EtherCAT slave axis

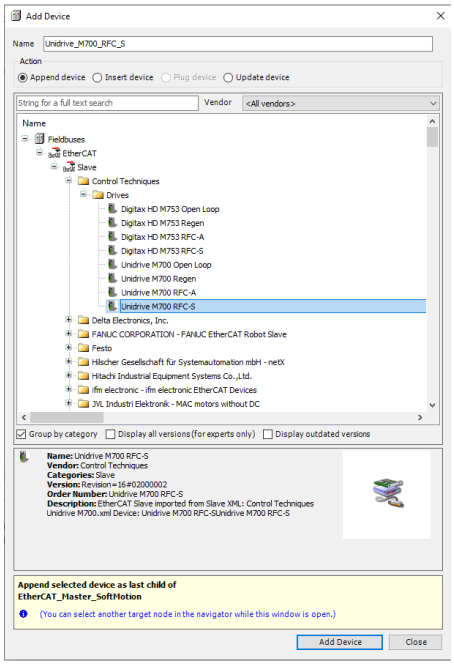
Both an EtherCAT master and EtherCAT slave device that represents the drive must be added. We will now add the EtherCAT master followed by the slave.

Right-click on *Device(MCe200)* in the *Devices* tree, then click "*Add Device…".* From the *Add Device* window that appears, expand the *EtherCAT* folder and select *EtherCAT Master.* Now click the *Add Device* button.

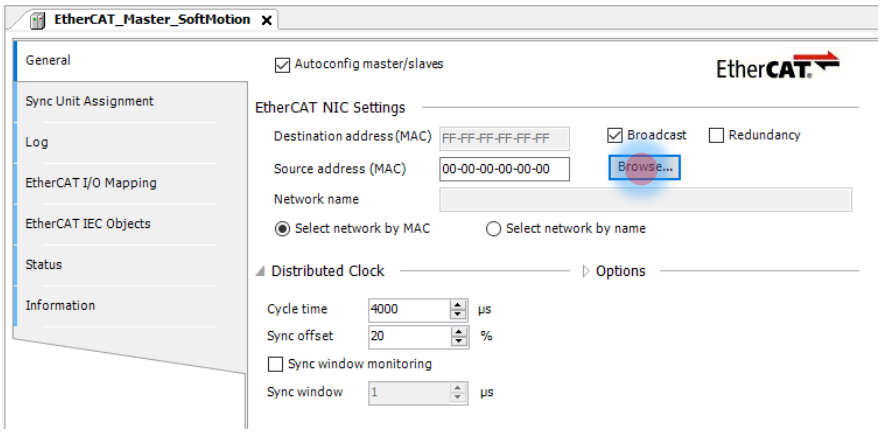
NOTE Do not close the *Add Device* window until both devices have been added.



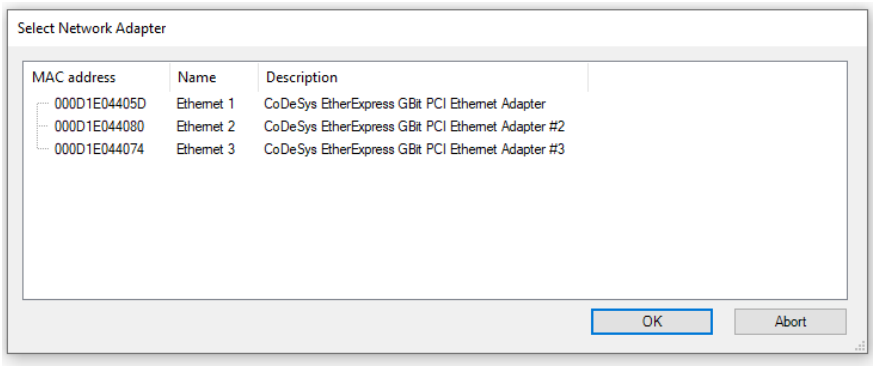
With the *Add Device* window still open click on the *EtherCAT_Master* that has appeared in the *Device* tree. In the *Add Device* window, navigate to the required slave device and add the device. Close the *Add Device* window.



Double click on the EtherCAT Master in the *Devices* window, and in the *General* tab click *Browse.* (If a message box displaying *Gateway not configured properly* is displayed, then the network has not been previously successfully scanned. See section 5.1).

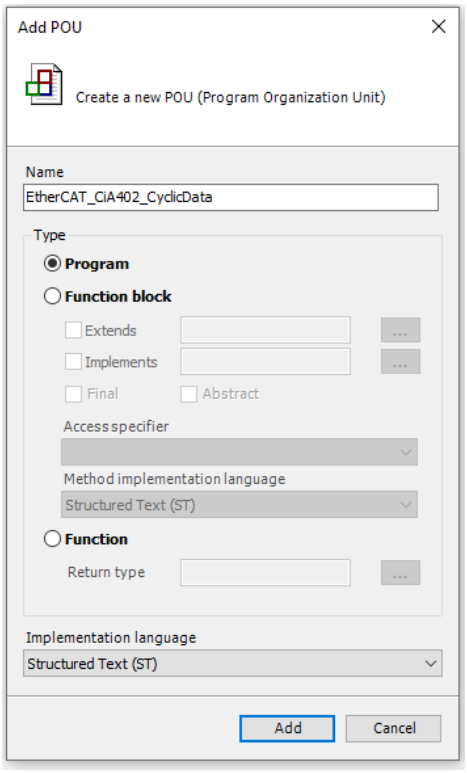


Select the port *Ethernet 2*, and click *OK.* (For MCzx01 select X108).

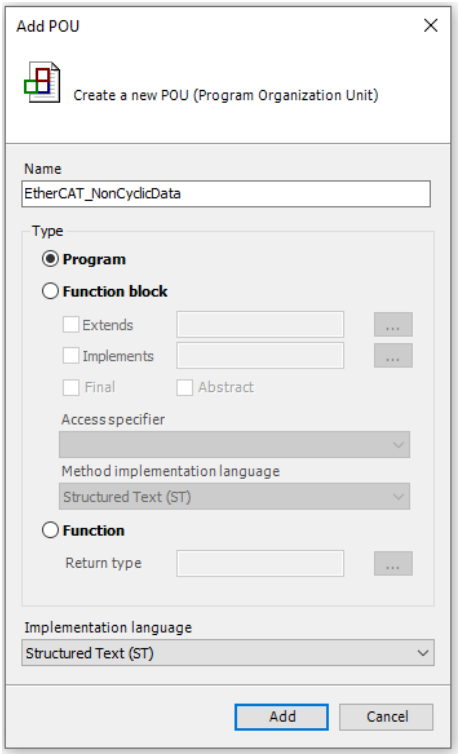


5.4 Add the program code

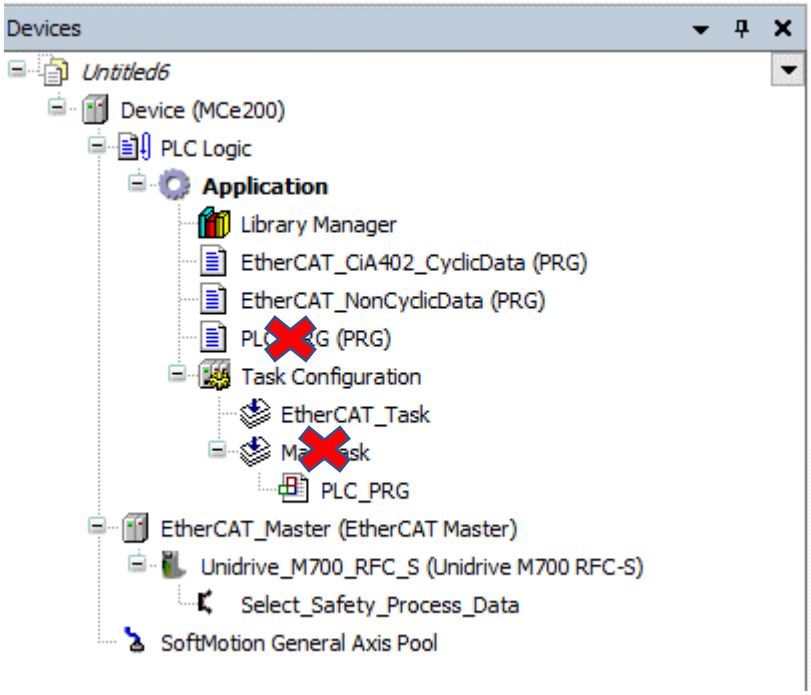
Right-click on *Application* in the *Devices* tree and select *Add Object>POU.* In the dialogue that appears enter the name, type as program and language as Structured Text (ST).



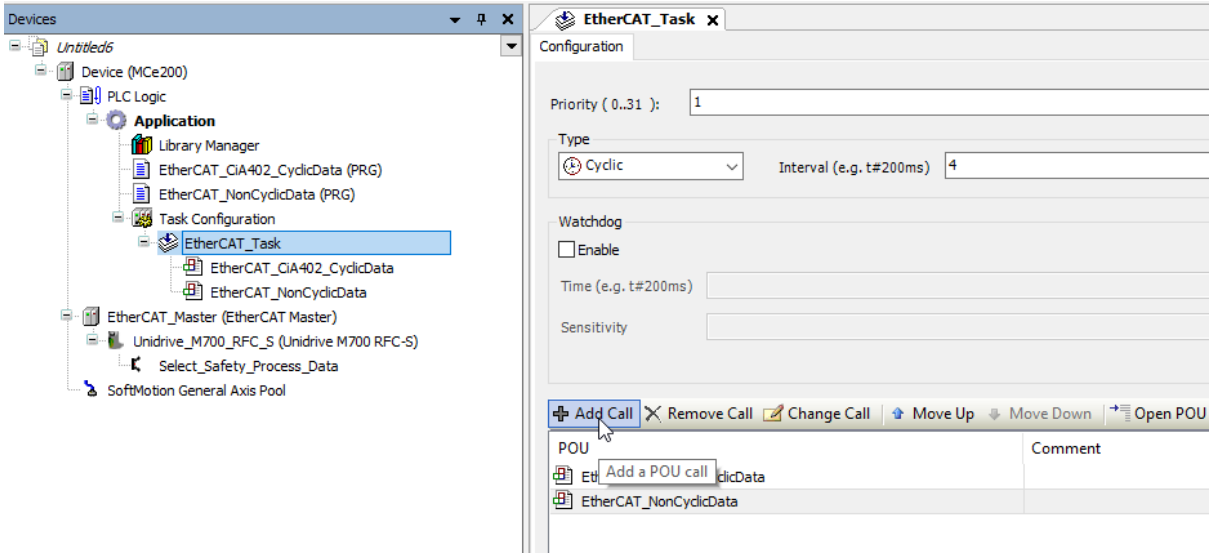
Repeat for a second program.



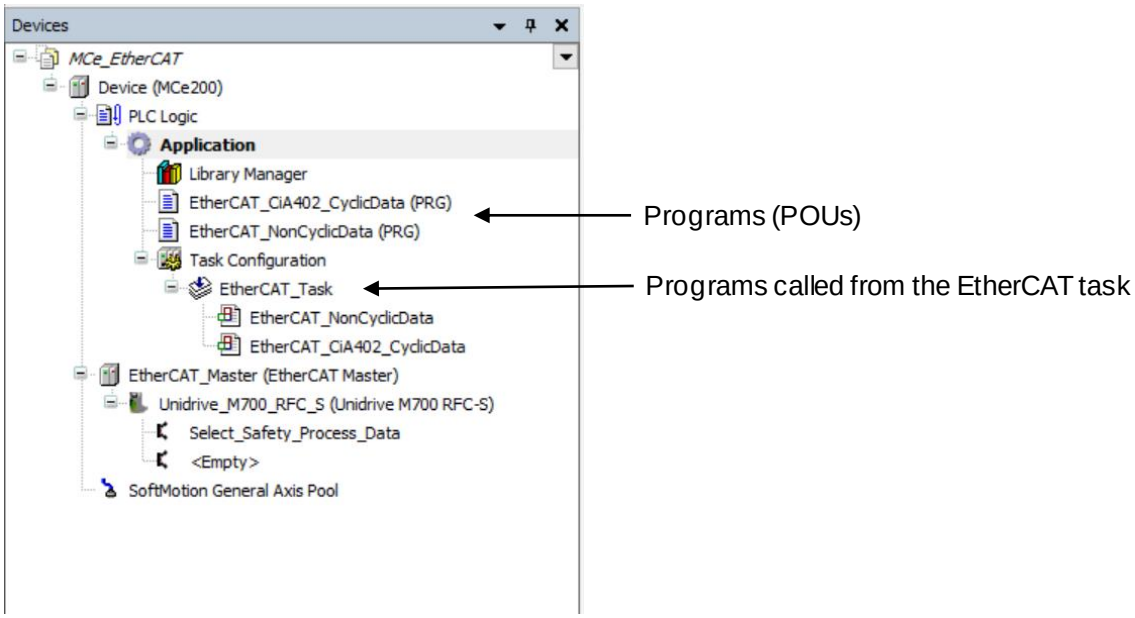
The *PLC_PRG(PRG)* and *MainTask* in the *Devices* tree may be deleted. (Right click>Delete)



Add the two new programs into the *EtherCAT_Task* in the *Devices* tree.



The *Devices* tree should look similar to the following.



Double click on the *EtherCAT_CiA402_CyclicData(PRG)* program in the *Devices* tree and in the declaration window and program window enter the following:

```
//The following is entered in the declaration window or copy and paste

PROGRAM EtherCAT_CiA402_CyclicData
VAR
(*These variables are mapped to the CiA402 objects process data in the
Unidrive_M700_RFC_S>EtherCAT I/O Mapping tab (see the Devices tree on the left)*)
   StatusWord:  UINT;
   ControlWord: UINT;
   TargetPos:   DINT;
   ActPos:      DINT;

   Count:       INT;
   StateIssue:  BOOL;
   ShaftPos:    DINT;

(*These variables are mapped to the objects process data in the Unidrive_M700_RFC_S>EtherCAT
I/O Mapping tab.  They have been added in the Expert Process Data tab (see the Devices tree on
the left)*)
   Write:       DINT;
   Read:        DINT;
END_VAR



//The following is entered in the code window or copy and paste

//Wait for the EtherCAT network to be ready
IF Ethercat_Master.xConfigFinished THEN

  //Cyclic data, use the CiA402 drive profile to control the position of the motor shaft
  //Cycle through the CiA402 drive profile state machine
  IF StatusWord.3 THEN                //Fault
     ControlWord := 128;              //Fault reset
  ELSIF StatusWord.6  THEN            //Switch on disabled
     ControlWord := 6;                //Shutdown
  ELSIF StatusWord.0 AND StatusWord.5 AND NOT StatusWord.1 THEN     //Ready to switch on
     ControlWord := 7;                 //Switch on
  ELSIF StatusWord.0 AND StatusWord.1 AND StatusWord.5 AND NOT StatusWord.2 THEN //Switched on
     ControlWord := 15;                //Enable operation
  ELSIF StatusWord.0 AND StatusWord.1 AND StatusWord.2 AND StatusWord.5 THEN//Operationenabled
```

```
      //Move the shaft ~ every second
      StateIssue := FALSE;
      IF Count > 249 THEN
         TargetPos :=  TargetPos + 1092;
       Count := 0;
      ELSE
        Count := Count + 1;
      END_IF

 ELSE
    StateIssue := TRUE;         //There is a problem with the state machine/logic/STO off etc
 END_IF

 ShaftPos := ActPos;

 //Cyclic data, read and write M20.P22.
 Write := Read + 1;


END_IF
```

Double click on the *EtherCAT_NonCyclicData (PRG)* program in the *Devices* tree and in the declaration window and program window enter the following:

```
//The following is entered in the declaration window or copy and paste

PROGRAM EtherCAT_NonCyclicData
VAR
        SdoRead:                ETC_CO_SdoRead;
        DataRead:               DINT;
        Read:                   BOOL;
        SdoWrite:               ETC_CO_SdoWrite;
        DataToWrite:            DINT;
        Write:                  BOOL;
END_VAR


//The following is entered in the code window or copy and paste

//Wait for the EtherCAT network to be ready
IF Ethercat_Master.xConfigFinished THEN


//SDO communications take time to be performed, and this time can be longer than one task
//cycle. The xExecute input of the block should remain high until the SDO has completed.
//In this example this is of course trivial.  However, the user may examine the xDone and
//xError outputs of the function blocks to control the xExecute input if desired.

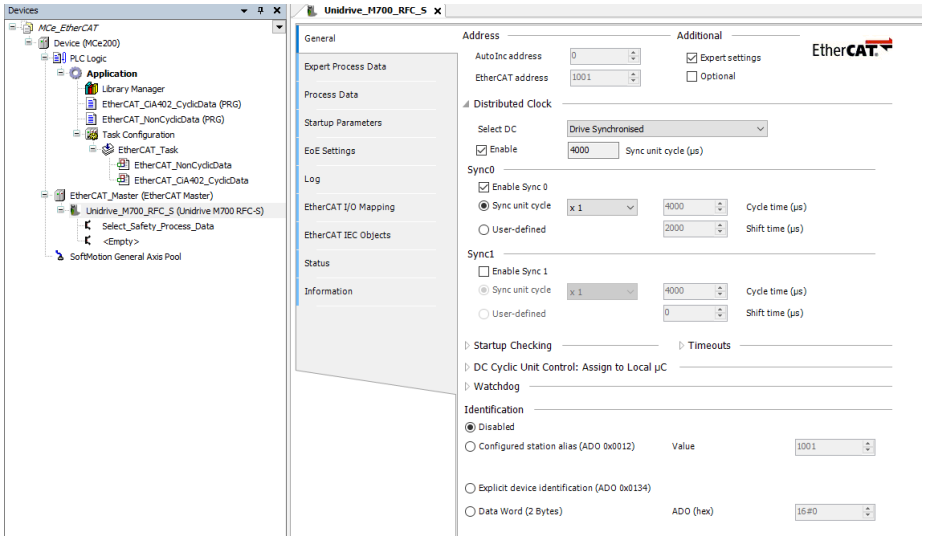

//Read parameter 20.21
SdoRead(
        xExecute:= Read,                //Begin reading the data on transition to true
        usiCom:= 1 ,                    //Number of the master
        uiDevice:= 1001 ,               //Number of the slave
        wIndex:= 16#2014 ,              //Menu 20
        bySubindex:= 21 ,               //Parameter 21
        udiTimeOut:= 1000 ,             //Timeout in ms
        pBuffer:= ADR(DataRead) ,       //Pointer to returned data, 4 bytes will be returned
        szSize:= SIZEOF(DataRead) );    //Size of data

//Write to parameter 20.21
SdoWrite(
        xExecute:= Write,
        usiCom:= 1 ,                    //Number of the master
        uiDevice:= 1001 ,               //Number of the slave
        wIndex:= 16#2014 ,              //Menu 20
        bySubindex:= 21 ,               //Parameter 21
        udiTimeOut:= 1000,              //Timeout in ms
        pBuffer:= ADR(DataToWrite) ,    //Pointer to data to be sent, 4 bytes will be sent
        szSize:= SIZEOF(DataToWrite) ,//Size of data
        eMode:= ETC_CO_AUTO );          //Mode for transmission is automatically chosen

END_IF
```

5.5 Configure the PDO mappings

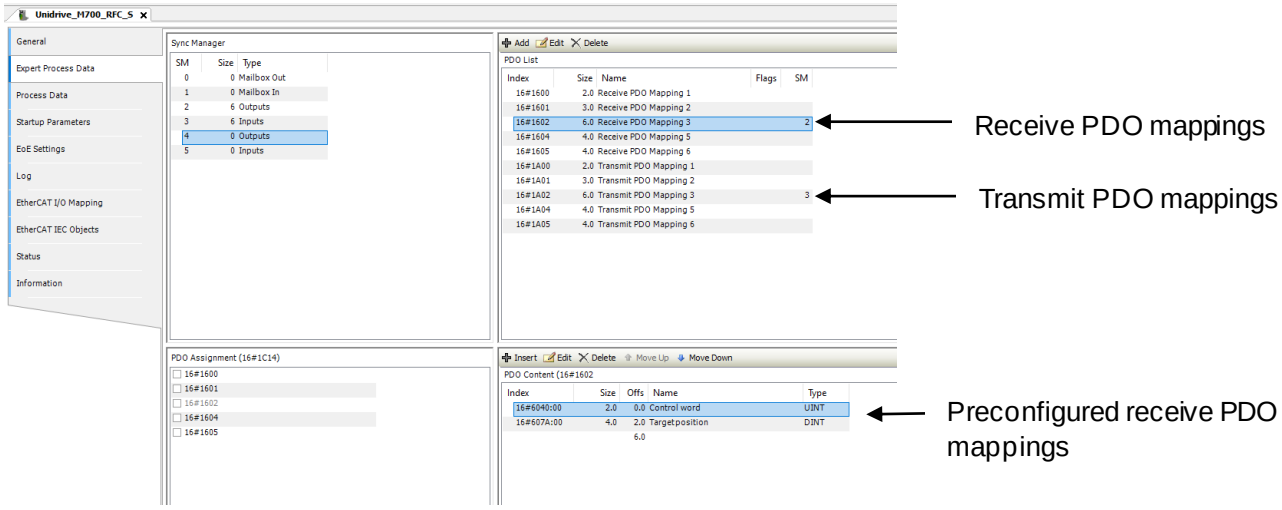
Select the *Unidrive_M700_RFC_S* slave from the *Devices* tree and choose the *Expert settings* check box.



Select the *Expert Process Data* tab. Here you will see the preconfigured PDO mappings.

As the drive has been selected to be RFC-S, the *Modes of Operation* object in the *Startup Parameters* tab is automatically set to *Cyclic Synchronous Position* mode, the preconfigured PDO mappings reflect position control. See the SI-EtherCAT User Guide for further details.

The CAN over EtherCAT concept of *objects* is used, these are numerical identifiers in the format of *index:sub-index*. It is usual for the index to be expressed in hexadecimal format and the sub-index(s) to be expressed as decimal. For example the object number for the CiA402 control word is 16#6040:00



We will shortly map our program's variables to these PDO mappings. However, firstly we shall manually add a further receive and transmit object to the PDO mappings. In this case it will be the drive Application menu parameter, and for this example we will use the same parameter in the transmit and receive PDO. (It should be noted that if you wish to control the drive's speed, torque or position using drive parameters directly such as the drive's

control word etc. Then to prevent the CiA402 drive profile within the EtherCAT module from writing to such drive parameters, parameter 17.033 of the EtherCAT module should be set to a value of 1).

To calculate the EtherCAT object that represents a particular drive parameter use the following:

Index = 16#2000 + 16#Drive menu

Sub-index = Drive parameter

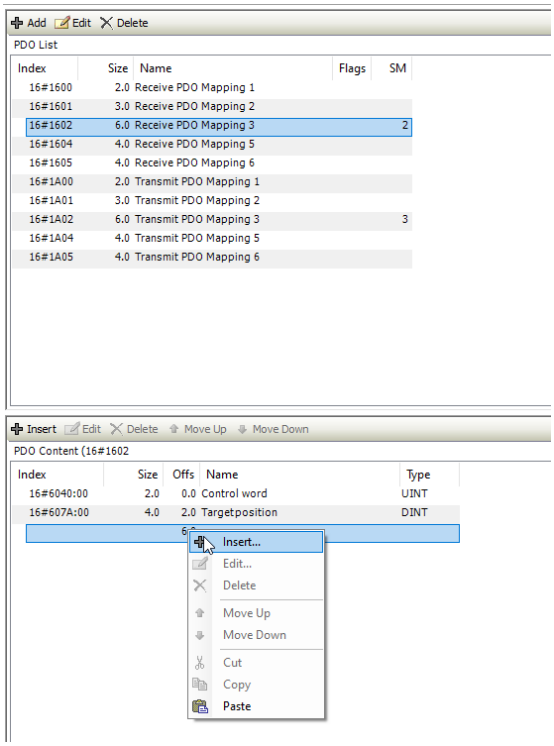
We will select drive parameter 20.22. Therefore, to calculate the object:

Index = 16#2000 + 16#14 = 16#2014

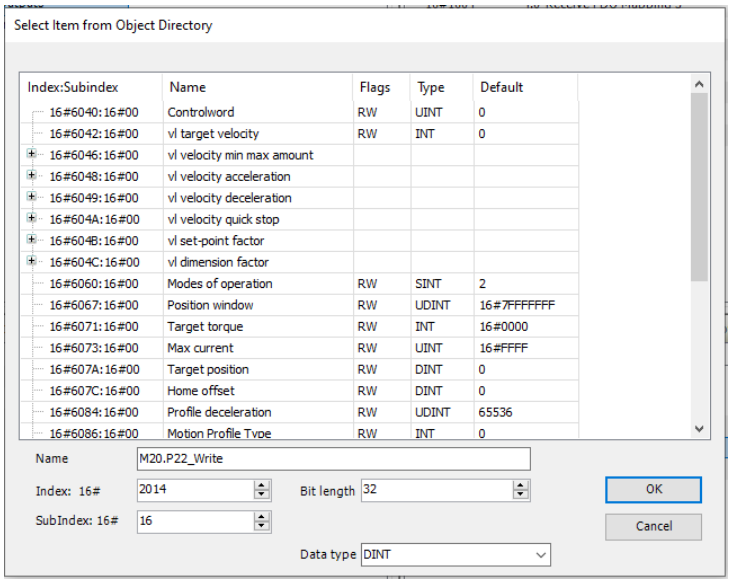
Sub-index = 22

Even though the convention is to show the sub-index as decimal, in the following steps you will see that the CODESYS Development System expects the sub-index to be entered in hexadecimal format, so the sub-index would be 16#16.

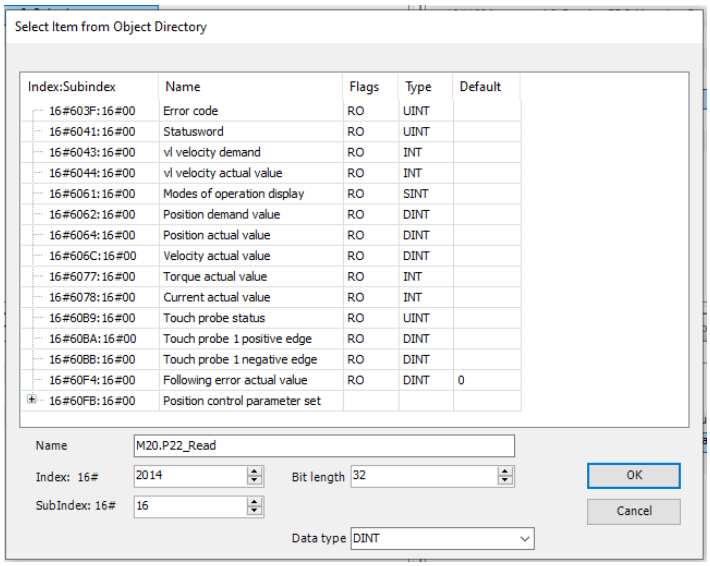
To add a new mapping select the receive PDO in the upper *PDO* List window and then in the lower *PDO Content* window right click below the last mapping and select *Insert.*



In the dialogue that appears enter a *Name* for the object, the *Index,* the *SubIndex* (noting it is entered in hexadecimal format) and the *Data type,* then click *OK.*

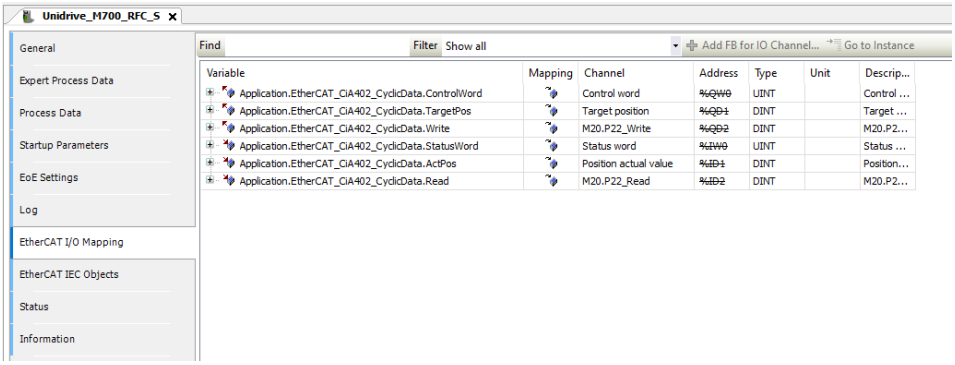


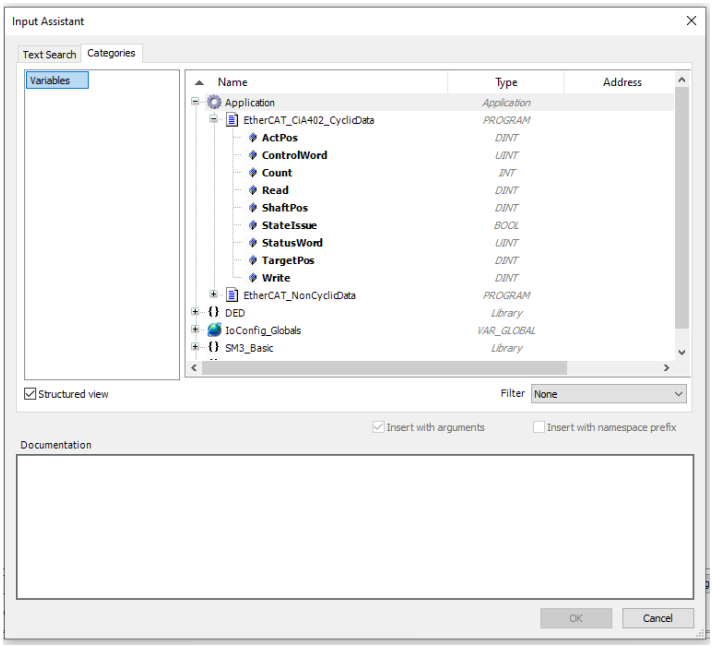
Repeat for the Transmit PDO using the same object configuration, but a different *Name*.



5.6 Configure mappings to the program variables

Select the *EtherCAT I/O Mapping* tab and click in the variable section to add the program variables for each mapping.





5.7 Test your application

The motor's shaft may rotate.

If safe to do so download and login to your application. (See section 3). Click *Debug>Start.*

Select the program *EtherCAT_CiA402_CyclicData(PRG)*. It is assumed that the drive is correctly configured for its motor and the drive's STO is turned off.

Turn on the drive's STO. By means of cyclic communications (implicit messaging), if the state machine has successfully transitioned to *Operation Enabled,* then the motor's shaft will index and the variable *ShaftPos* update.

Additionally parameter 20.022 should increment as the value is read cyclically from the drive and a value of one added before the value is written cyclically to the drive.

```
//Wait for the EtherCAT network to be ready
IF Ethercat_Master.xConfigFinished TRUE THEN

    //Cyclic data, use the CiA402 drive profile to control the position of the motor shaft
    //Cycle through the CiA402 drive profile state machine
    IF StatusWord 5687 .3 FALSE THEN                //Fault
       ControlWord 15 := 128;                 //Fault reset
    ELSIF StatusWord 5687 .6 FALSE  THEN              //Switch on disabled
       ControlWord 15 := 6;                  //Shutdown
    ELSIF StatusWord 5687 .0 TRUE AND StatusWord 5687 .5 TRUE AND NOT StatusWord 5687 .1 TRUE THEN      //Ready to switch on
       ControlWord 15 := 7;                   //Switch on
    ELSIF StatusWord 5687 .0 TRUE AND StatusWord 5687 .1 TRUE AND StatusWord 5687 .5 TRUE AND NOT StatusWord 5687 .2 TRUE THEN      //Switched on
       ControlWord 15 := 15;                  //Enable operation
    ELSIF StatusWord 5687 .0 TRUE AND StatusWord 5687 .1 TRUE AND StatusWord 5687 .2 TRUE AND StatusWord 5687 .5 TRUE THEN //Operation enabled

        //Move the shaft 360' (6deg) ~ every second
        StateIssue FALSE := FALSE;
        IF Count 48 > 249 THEN                      //1000 / 4 (the task calling this program is 4ms)
            TargetPos 7644 :=  TargetPos 7644 + 1092;     //One complete rotation per minute of time (65536 encoder counts per rotation / 60 seconds of time)
            Count 48 := 0;
        ELSE
            Count 48 := Count 48 + 1;
        END_IF

    ELSE
        StateIssue FALSE := TRUE;         //There is a problem with the state machine/logic/STO off etc
    END_IF

    ShaftPos 7642 := ActPos 7642;
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

    //Cyclic data, read and write M20.P22.
    Write 1264606 := Read 1264605 + 1;
```

Select the *EtherCAT_NonCyclicData(PRG)* program. Using the drive's keypad set a value in parameter 20.021. Set the variable *Read* to true, the variable *DataRead* should show the value of parameter 20.021. Now set *Read* to false, set *DataToWrite* to some value within its range and make *Write* true. Parameter 20.021 should update with the new value, set *Write* to false. You have now successfully read and written data non cyclically (explicit messaging).

```
//Wait for the EtherCAT network to be ready
IF Ethercat_Master.xConfigFinished TRUE THEN


//Read parameter 20.21
SdoRead(
    xExecute FALSE := Read FALSE,             //Read the data on transition to true
    usiCom 1 := 1 ,                    //Number of the master
    uiDevice 1001 := 1001 ,            //Number of the slave
    wIndex 8212 := 16#2014 ,               //Menu 20
    bySubindex 21 := 21 ,              //Parameter 21
    udiTimeOut 1000 := 1000 ,              //Timeout in ms
    pBuffer 18446694113735983532 := ADR(DataRead 1 ) ,      //Pointer to returned data, 4 bytes will be returned
    szSize 4 := SIZEOF(DataRead 1 ) );    //Size of data

//Write to parameter 20.21
SdoWrite(
    xExecute TRUE := Write TRUE,
    usiCom 1 := 1 ,                    //Number of the master
    uiDevice 1001 := 1001 ,            //Number of the slave
    wIndex 8212 := 16#2014 ,               //Menu 20
    bySubindex 21 := 21 ,              //Parameter 21
    udiTimeOut 1000 := 1000,                   //Timeout in ms
    pBuffer 18446694113735992664 := ADR(DataToWrite 1234 ) ,   //Pointer to data to be sent, 4 bytes will be sent
    szSize 4 := SIZEOF(DataToWrite 1234 ) ,  //Size of data
    eMode ETC_CO_AUT := ETC_CO_AUTO );          //Mode for transmission is automatically chosen

END_IF RETURN
```

SDO communications take time to be performed, and this time can be longer than one task cycle. The xExecute input of the block should remain high until the SDO has completed. In this example this is of course trivial. However, the user may examine the xDone and xError outputs of the function blocks to control the xExecute input if desired as shown below.

```
SdoRead(
        xExecute:= Read,              //Begin Reading the data on transition to true
        usiCom:= 1 ,                  //Number of the master
        uiDevice:= 1001 ,             //Number of the slave
        wIndex:= 16#2014 ,            //Menu 20
        bySubindex:= 21 ,             //Parameter 21
        udiTimeOut:= 1000 ,           //Timeout in ms
        pBuffer:= ADR(DataRead) ,     //Pointer to returned data, 4 bytes will be returned
        szSize:= SIZEOF(DataRead) );  //Size of data

IF SdoRead.xDone OR SdoRead.xError THEN
        Read := FALSE;
ELSE
        Read := TRUE;
END_IF
```

This concludes a brief example to using EtherCAT with the MCxxxx controllers. For further information regarding specific features of the CODESYS Development Environment please see the on-line help available within CODESYS. For further details of SI-EtherCAT please see the SI-EtherCAT User Guide.

5.8 Supplementary - Access to slot parameters

To access drive slot parameters as objects use the following addressing scheme.

Index = Menu Index + Menu Number

Sub Index = Parameter number

Menu Index	Destination
0x2000	Drive
0x2100	Slot 1
0x2200	Slot 2
0x2300	Slot 3 (internal slot Digitax HD M75x
0x2400	Slot 4 (internal slot M70x)

6 EtherNet/IP Scanner

This example was created with the M700 drive, it demonstrates non-cyclic data transfer and cyclic data transfer from the MCxxxx acting as a Scanner (master).

NOTE The companion project archive *MCeEtherNet_IP.projectarchive* is available to download from Support Suite or is available from your local Drive Centre or supplier.

NOTE The drive's communication interface should be configured for Protocol Mode (parameter 04.02.018 = EtherNet/IP).

It is assumed that the default IP address and subnet (192.168.2.1, 255.255.255.0) for port 2 on the MCe200, or X105 on the MCzx01 are configured within the MCxxxx, and IP addresses in the same subnet have been configured on the user's PC or laptop and drive (this example uses the drive IP address 192.168.2.10). For convenience it is suggested that the user's PC is connected to one of the drive's Ethernet ports and the remaining Ethernet port on the drive connects to port 2 (LAN B) of the MCz200 or X105 on the MCzx01.

If the example archive (*MCeEtherNet_IP.projectarchive*) is converted to an MCzx01 target then the network must be scanned in the *Communications Settings* tab of the PLC, and subsequently section 6.3 will need to be followed to set the Ethernet port to X105.

Drive parameter 01.014 should be set to *Preset*, and drive parameter 06.043 should be set to *On.*

6.1 New project

If you are not installing the project archive (see section 6) then create a new project for your desired device target type (add the device target DDF as required) and configure the gateway by scanning the network (set the network path in the *Communications Settings* tab of the PLC) to ensure your device is visible in the CODESYS Development System. (See section 3.1).

The example and its associated images are based upon the MCe200, but users may choose to use the MCzx01.

6.2 EDS file

This example will use the manufacturer's specific CIP objects 100 and 101. Therefore, no EDS file is required for the drive.

6.3 Adding the EtherNet/IP Scanner and Device

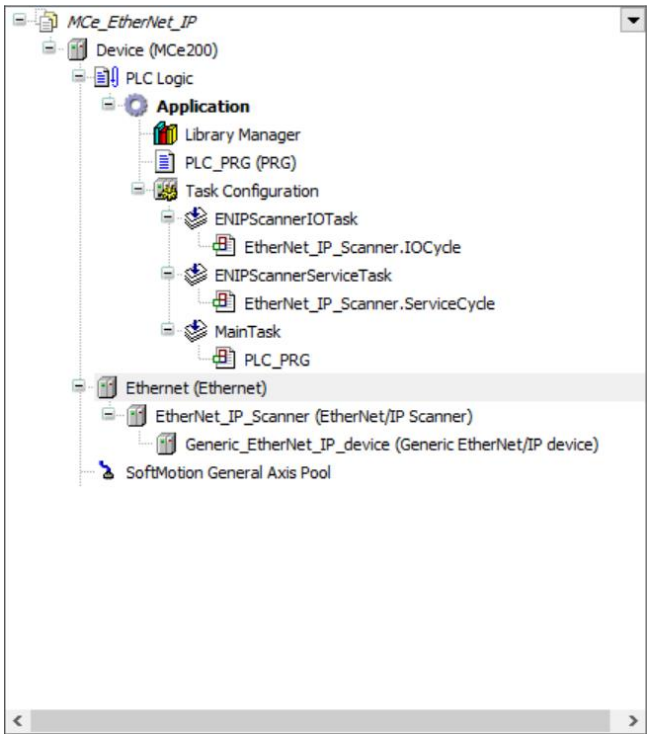
Both an EthetNet/IP Scanner (master) and EtherNet/IP Device (slave) must be added. We will now add the EtherNet/IP Scanner followed by the Device that represents the drive.

Right click on *Device(MCe200)* in the *Devices* tree, then click "*Add Device…".* From the *Add Device* window that appears, expand the *EtherNet/IP* folder, and select *Ethernet Adapter.* Now click the *Add Device* button.

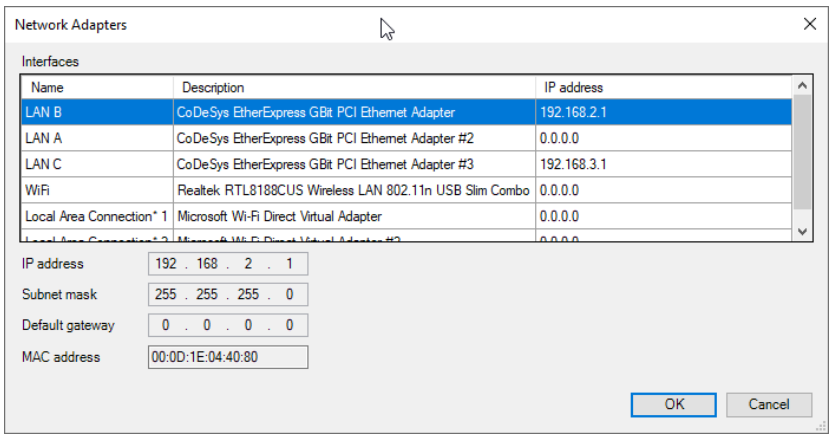
NOTE Do not close the *Add Device* window until all the tree devices have been added.

Click on the *Ethernet* device that has appeared in the *Devices* window, select *EtherNet/IP Scanner*. Click on the *EtherNet_IP_Scanner* in the *Devices* window and select *Generic EtherNet/IP device.* Close the *Add Device* window.

The *Devices* window should look similar to the following.

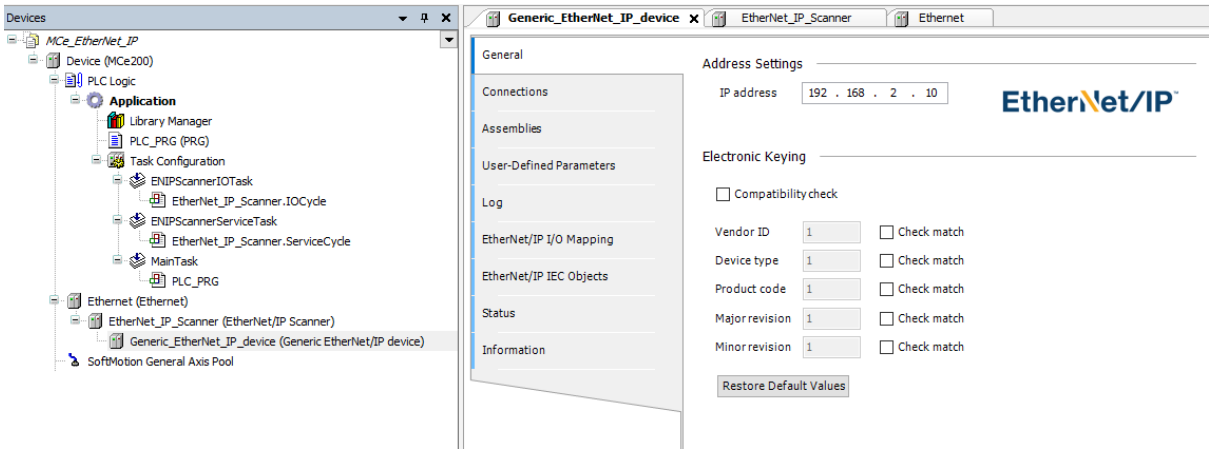


Click on the *Ethernet* device in the *Devices* window, and in the *General tab* click next to the *Network* interface, in the dialogue that appears assign LAN B to the *Network interface*. (Or X105 for the MCzx01).

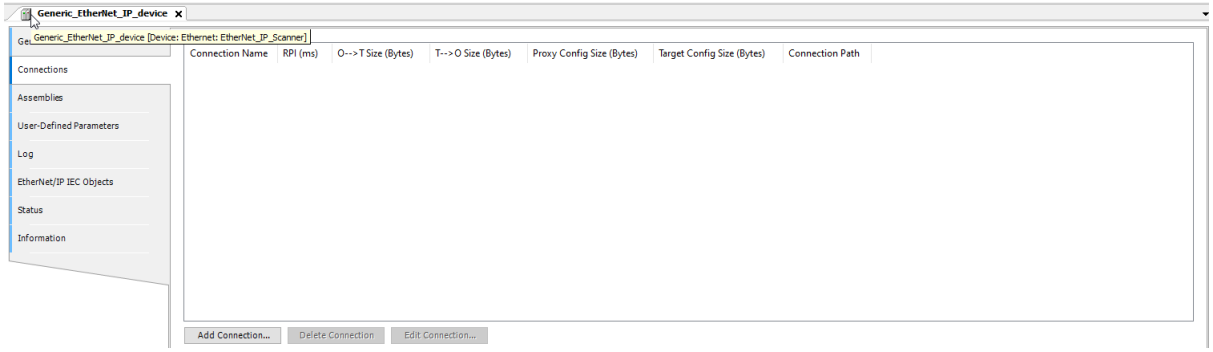


Click on the *Generic_EtherNet_IP_device* in the *Devices* window. In the *General* tab Enter the IP address that is set in the drive (192.168.2.10 in this example). Deselect all the *Check match* check boxes.

(If electronic keying is required, the Vendor ID for Control Techniques is 257, and the device type is 2. Parameter 20.033 in the Ethernet interface will show the product code. The Major revision is the first two digits (left) of the Ethernet interface's firmware version, and the Minor revision is represented by the second two digits i.e. XX.YY.**.** of the Ethernet interface's parameter 00.02).

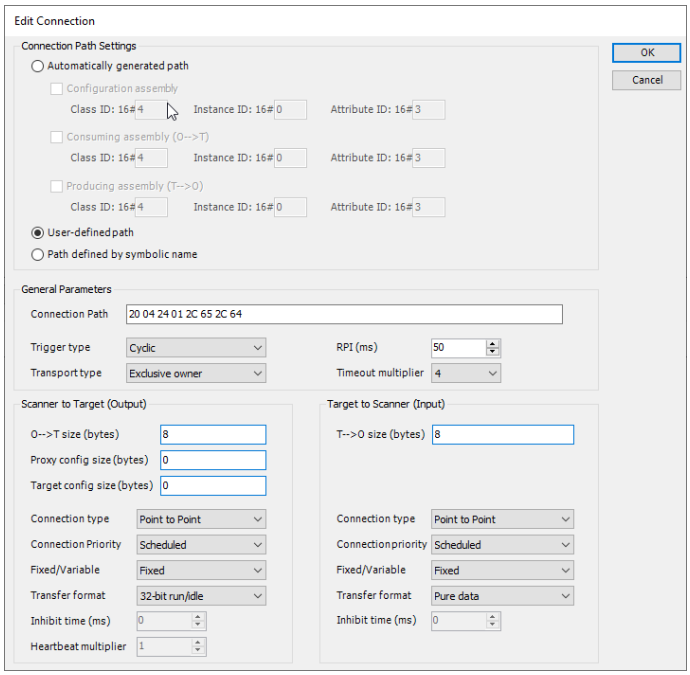


Select the *Connections* tab and click *Add Connection..*



In the dialogue that appears, select *User-defined path, and* enter the following values. The Connection path *20 04 24 01 2C 65 2C 64* is valid for a factory fitted Ethernet V2 (FFE V2) module, (for SI-Ethernet and FFE V1 the *Connection path* should be *20 04 24 01 2C 14 2C 46).*

The drive's default input and output assembly objects (parameters *mm.20.020* and *mm.20.021*) sizes are 8 bytes.



6.4 Add the program code

Double click on the *PLC_PRG(PRG)* program in the *Devices* tree and in the declaration window and program window enter the following:

```
//The following is entered in the declaration window or copy and paste
PROGRAM PLC_PRG
VAR
	Write_M01_P015:		ENIP.Set_Attribute_Single;
	Read_M20_P021:		ENIP.Get_Attribute_Single;
	PreSetSelector:		DINT;
	SetPreSet:		BOOL;
	Read:			BOOL;
	M20_P21:		DINT;

	Speed:			DINT;
	Run:			BOOL;

	Control:		UINT;
	ControlWord:		ARRAY [0..1] OF BYTE;
	StatusWord:		ARRAY [0..1] OF BYTE;
	SpeedRef:		ARRAY [0..3] OF BYTE;
	Status:			UINT;
END_VAR


//The following is entered in the code window or copy and paste
//********Non cyclic communications********//
//Write to select Preset reference = 1 (parameter 01.015)
PreSetSelector := 1;
Write_M01_P015(
	xExecute:= SetPreSet ,				//Write the data on transition to true
	itfEtherNetIPDevice:= Generic_Ethernet_IP_device ,	//The EtherNet/IP device
	eClass:= 100 ,					//CIP class
	dwInstance:= 1,					//Menu 1
	wAttribute:= 15 ,				//Parameter 15
	pData:= ADR(PreSetSelector) ,			//Pointer to data to send
	udiDataSize:= SIZEOF(PreSetSelector ) );	//Size of data

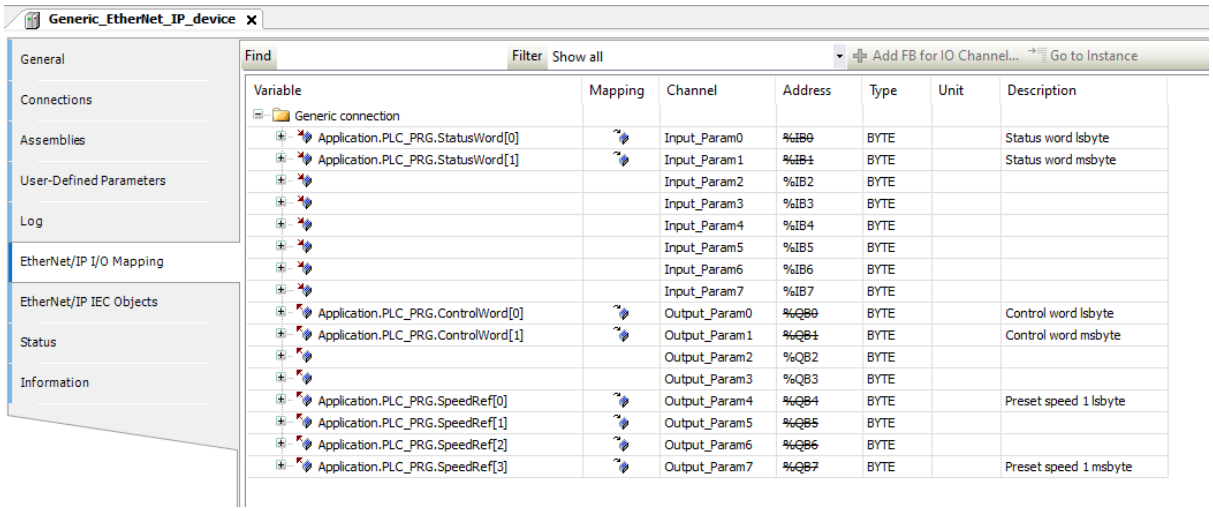
//Read M20.P021
Read_M20_P021(
	xExecute:= Read ,				//Read the data on transition to true
	itfEtherNetIPDevice:= Generic_Ethernet_IP_device ,	//The EtherNet/IP device,
	eClass:= 100 ,					//CIP class
	dwInstance:= 20,				//Menu 20
	wAttribute:= 21 ,				//Parameter 25
	pData:= ADR(M20_P21),				//Pointer to data to read
	udiDataSize:=SIZEOF(M20_P21) );			//Size of data

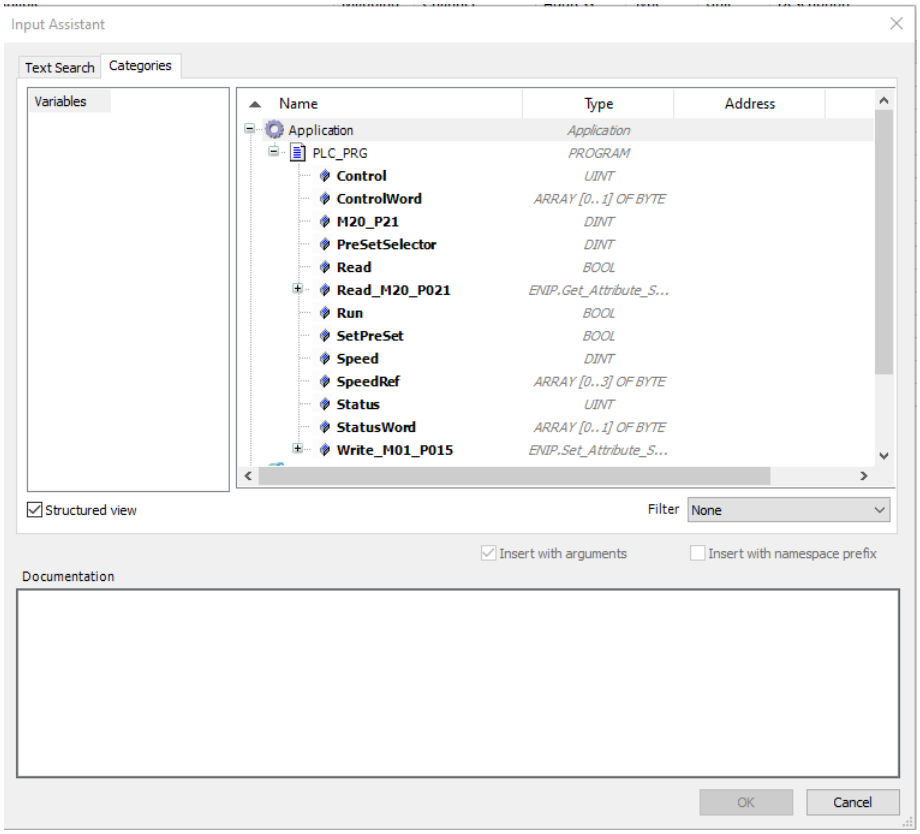

//********Cyclic communications********//
//Run the motor
IF Run THEN
	Control := 131;		//Drive's control word = Auto, Run Forward, Drive Enable.  The
				//drive's control word is mapped to output mapping 1 by default
				//in the drive.
	Speed := Speed + 1;	//Speed of motor.  Preset ref 1.  Preset reference 1 is mapped
				//to output mapping 2 by default in the drive.
ELSE
	Control := 0;
	Speed := 0;
END_IF


//Bytes are mapped in the Generic_EtherNet_IP device.  Therefore, convert to and from bytes//
//Construct bytes from Control
ControlWord[0] := UINT_TO_BYTE(Control);		//least significant byte
ControlWord[1] := UINT_TO_BYTE( SHR(Control,8) );	//most significant byte
//Construct Status from bytes
Status := (StatusWord[1]);
Status := SHL(Status, 8) OR StatusWord[0];
//Construct bytes from Speed
SpeedRef[0] := DINT_TO_BYTE(Speed);
SpeedRef[1] := DINT_TO_BYTE( SHR(Speed,8) );
SpeedRef[2] := DINT_TO_BYTE( SHR(Speed,16) );
SpeedRef[3] := DINT_TO_BYTE( SHR(Speed,24) );
```

6.5 Configure mappings to the program variables

Click on the *Generic_IP_Scanner_device* in the Devices window. Select the *EtherNet/IP I/O Mapping* tab and click in the variable section to add the program variables for each mapping. The arrays' indexes and brackets should be typed manually.





6.6 Test your application



The motor's shaft may rotate.

If safe to do so download and login to your application. (See section 3). Click *Debug>Start.*

Select the program *PLC_PRG(PRG)*. It is assumed that the drive is correctly configured for its motor and the drive's STO is turned off.

By means of non-cyclic communications (explicit messaging) we will set the drive's preset selector to 1 and read parameter 20.021.

Make *SetPreSet* true to write a value of 1 to M01.P015. Set *SetPreSet* back to false.

Using the drive's keypad adjust M20.P021 to some value other than zero, now set *Read* to true. The value set in parameter M20.P021 should be seen in variable *M20_P21.* Set *Read* back to false.

Turn on the drive's STO. Utilising Cyclic communications (implicit messaging) we will command the motor's shaft to rotate by writing to the drive's control word and preset speed reference. Set *Run* to true, the motor shaft should begin to accelerate.

Non-cyclic communications take time to be performed, and this time can be longer than one task cycle. The xExecute input of the block should remain high until the communications has completed. In this example this is of course trivial. However, the user may examine the xDone and xError outputs of the function blocks to control the xExecute input if desired as shown below.

```
Read_M20_P021(
        xExecute:= Read ,                               //Read the data on transition to true
        itfEtherNetIPDevice:= Generic_Ethernet_IP_device ,   //The EtherNet/IP device,
        eClass:= 100 ,                                  //CIP class
        dwInstance:= 20,                                //Menu 20
        wAttribute:= 21 ,                               //Parameter 25
        pData:= ADR(M20_P21),                           //Pointer to data to read
        udiDataSize:=SIZEOF(M20_P21) );                 //Size of data

IF Read_M20_P021.xDone OR Read_M20_P021.xError THEN
        Read := FALSE;
ELSE
        Read := TRUE;
END_IF
```

This concludes a brief introduction to using the EtherNet/IP Scanner with the MCxxxx controllers. For further information regarding specific features of the CODESYS Development Environment please see the on-line help available within CODESYS. For further details of EtherNet/IP and Control Techniques' drives please see the SI-Ethernet and relevant drive's User Guides.

7 EtherNet/IP Device

This example demonstrates data transfer from an Allen-Bradley Rockwell PLC acting as a Scanner (master) and the MCxxxx acting as the device (slave).

It is expected that the user is familiar with the Rockwell's Studio 5000 Logix Designer software. The example was tested with an Allen-Bradley CompactLogix 1769-L24ER-QB1B PLC.

NOTE The companion project archive *MCe_EtherNet_IP_Device.projectarchive* along with MCeEtherNet_IP_Slave.ACD Studio 5000 V32 project are available to download from Support Suite or is available from your local Drive Centre or supplier.

It is assumed that the default IP address and subnet (192.168.1.1, 255.255.255.0) for port 1 on the MCe200, or X102 on the MCzx01 are configured within the MCxxxx, and an IP address in the same subnet has been configured on the user's PC or laptop. The user's PC should be connected to port 1 (LAN A) of the MCe200 or X102 of the MCzx01. The Allen-Bradley PLC in this example is connected to the same network, this is best achieved with a suitable switch. For convenience a second PC was used for Connection to the Allen-Bradley PLC and Studio 5000 Logix Designer.

If the example archive (*MCe_EtherNet_IP_Device.projectarchive*) is converted to an MCzx01 target then the network must be scanned in the *Communications Settings* tab of the PLC, and subsequently section 7.2 will need to be followed to set the Ethernet port to X105.

7.1 New project

If you are not installing the project archive (see section 7) then create a new project for your desired device target type (add the device target DDF as required) and configure the gateway by scanning the network (set the network path in the *Communications Settings* tab of the PLC) to ensure your device is visible in the CODESYS Development System. (See section 3.1).

The example and its associated images are based upon the MCe200, but users may choose to use the MCzx01.

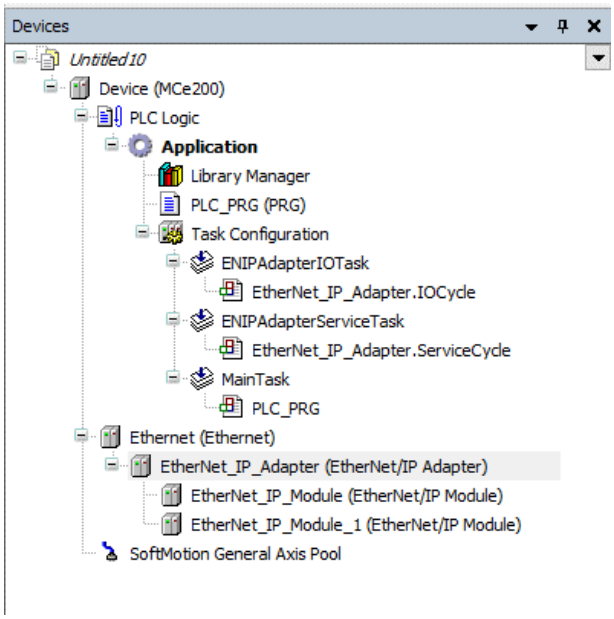
7.2 Adding the EtherNet/IP Device

Right click on *Device(MCe200)* in the *Devices* tree of the CODESYS Development Environment, then click "*Add Device…".* From the *Add Device* window that appears, expand the *EtherNet/IP* folder, and select *Ethernet Adapter.* Now click the *Add Device* button.

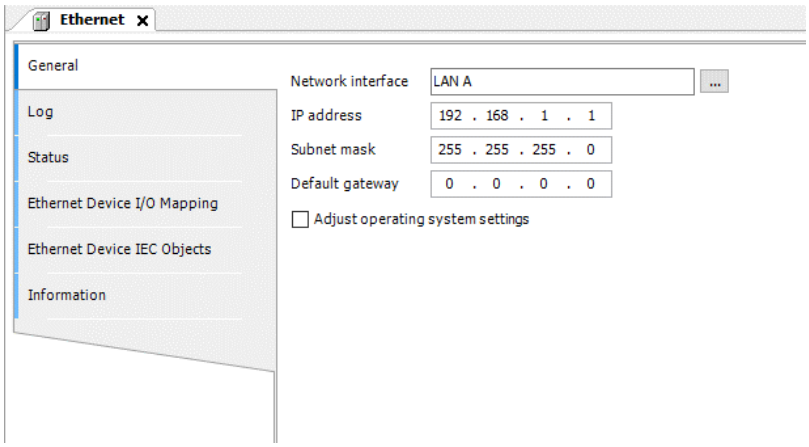
NOTE Do not close the *Add Device* window until all the tree devices have been added.

Click on the *Ethernet* device that has appeared in the *Devices* window, select *EtherNet/IP Adapter* Click on the *EtherNet/IP* Adaptor that has appeared in the *Devices* window, select *EtherNet/IP Module* in the *Devices* window, add two instances of the *EtherNet/IP Module.* Close the *Add Device* window.

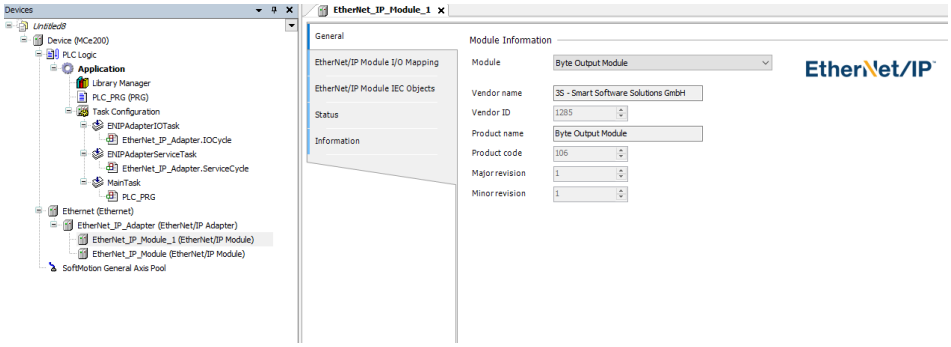
The *Devices* window should look similar to the following.

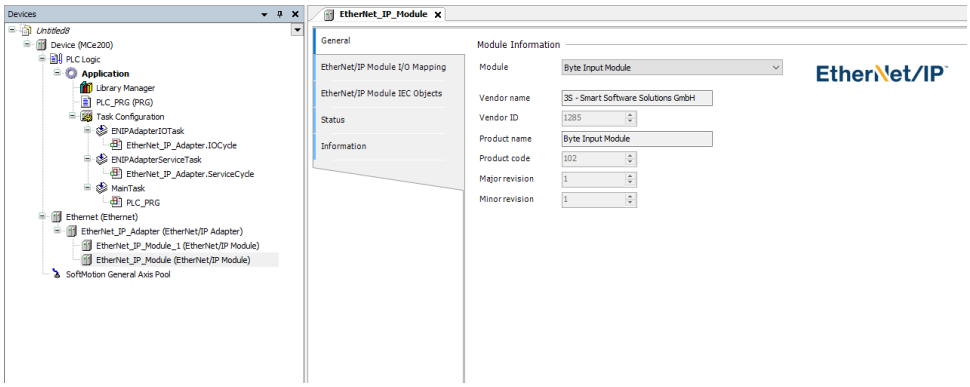


Click on the *Ethernet* device in the *Devices* window, and in the *General tab* click next to the *Network* interface, in the dialogue that appears assign LAN A to the *Network interface* (X102 for the MCzx01), set the IP address to an address in the same subnet as the Allen-Bradley PLC, in this example the default address of 192.168.1.1 has been used. This address will be required to configure your Studio 5000 project.



Click on the *EtherNet_IP Module* device in the *Devices* window and configure the first *EtherNet/IP Adapter* as a *Byte Output Module*, repeat for the second module as *Byte Input Module.*





7.3 Add the program code

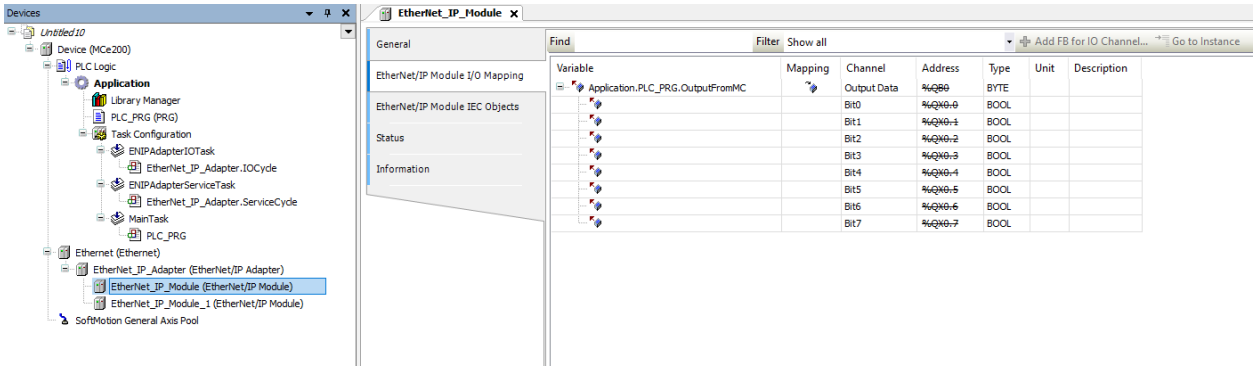
Double click on the *PLC_PRG(PRG)* program in the *Devices* tree and in the declaration window and program window enter the following:

```
//The following is entered in the declaration window or copy and paste
PROGRAM PLC_PRG
VAR
        OutputFromMC:           BYTE;
        Output:                 BYTE;
        Input:                  BYTE;
        InputToMC:              BYTE;
END_VAR

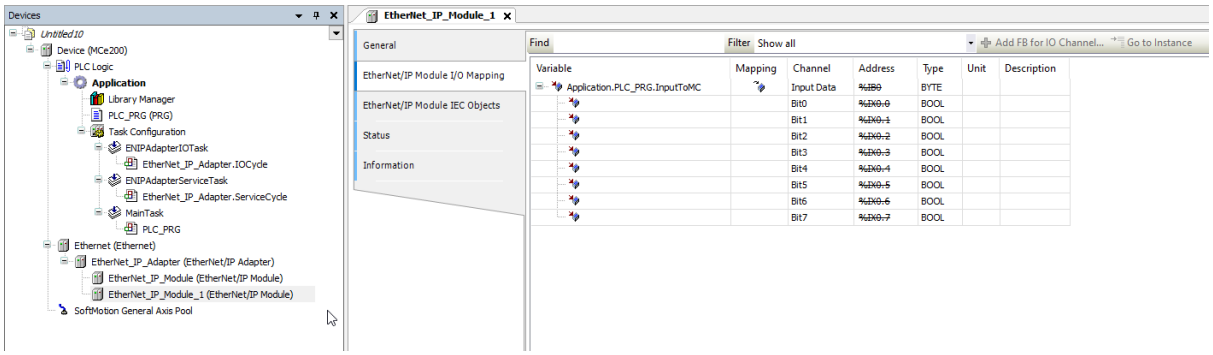
//The following is entered in the code window or copy and paste
OutputFromMC := Output;         //Output to Allen-Bradley PLC
Input := InputToMC;             //Input from Allen-Bradley PLC
```

7.4 Configure the mappings

Click on the *EtherNet_IP_Module*(*Byte Output Module)* in the *Devices* window and select the *EtherNet/IP Module I/O Mapping* tab. Add the variable *OutputFromMC* to the *Output Data*.

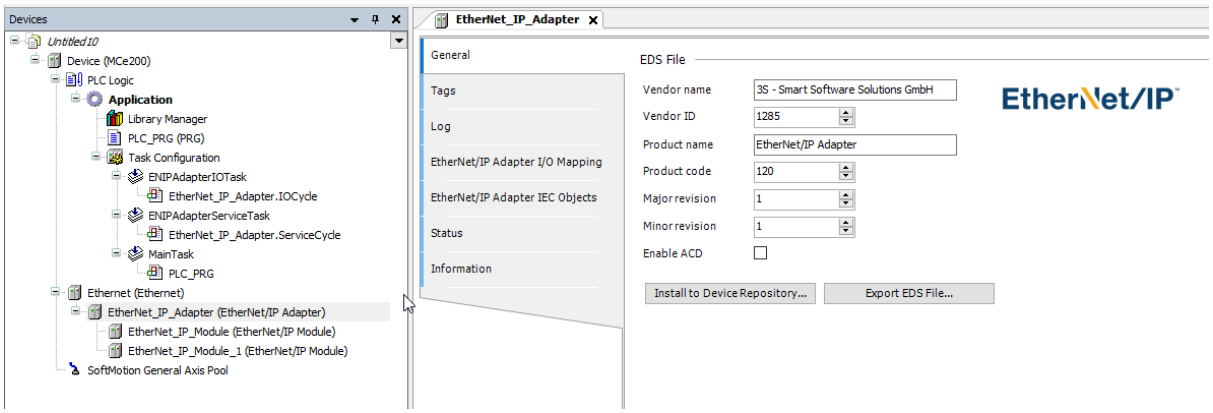


Click on the *EtherNet_IP_Adapter_1* (*Byte Input Module*) in the *Devices* window and select the *EtherNet/IP Module I/O Mapping* tab. Add the variable *InputToMC* to the *Input Data*.



7.5 Generate EDS file for Studio 5000

Click on the *EtherNet_IP_Adaptor* in the *Devices* window, and in the *General* tab select *Export EDS File…* You will be prompted for a location where you wish to save the file. Install the EDS file into Studio 5000 in the normal way.



7.6 Create a Studio 5000 application

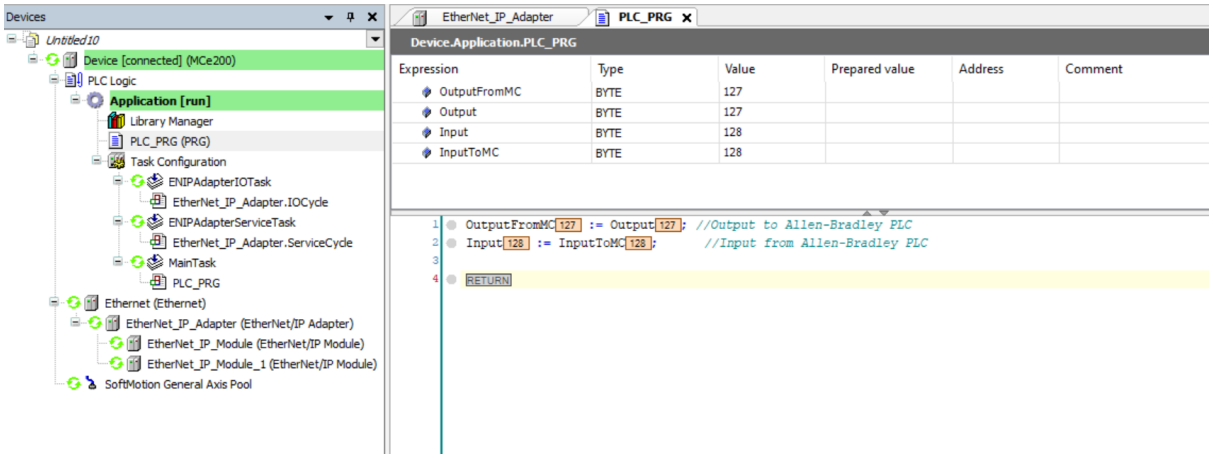
From within Studio 5000 create a new project, in this example the 176-L24ER-QB1B PLC has an integrated Ethernet port, other PLCs may require that the Ethernet hardware is added. Configure an IP address for the Ethernet adapter in the same subnet as the MCxxxx. Add a new module to the Ethernet adapter and search for *0505_000C_0078 CODESYS EtherNet/IP Adapter*. Configure the device's IP address as 192.168.1.1. (See the Studio 5000 project *MCeEtherNet_IP_Slave.ACD* for an example).

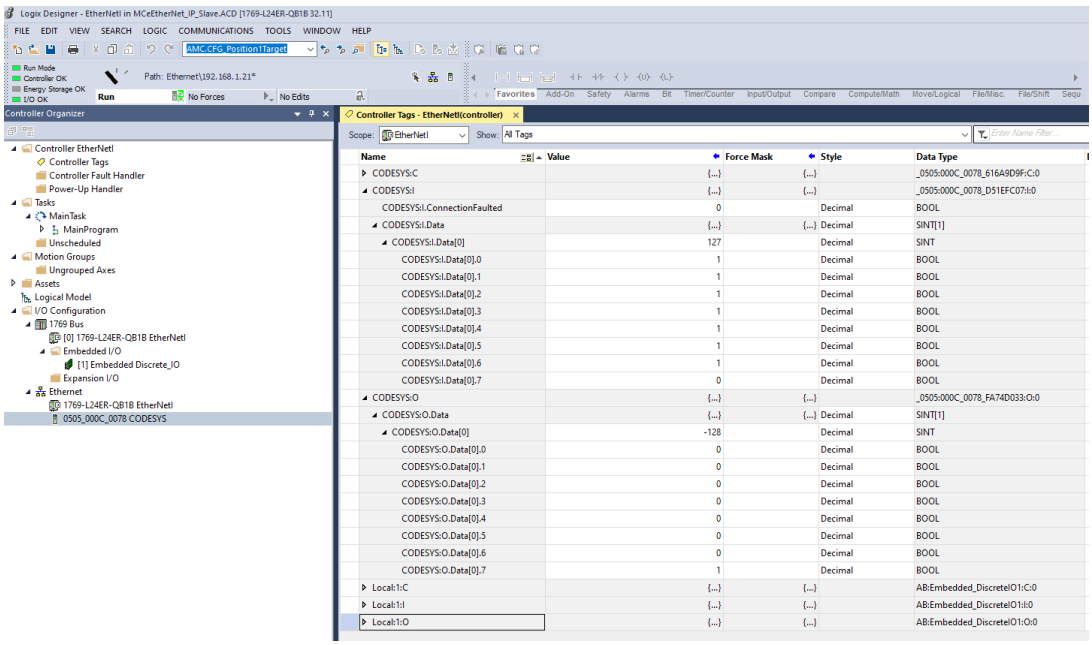
Download the application to the Allen-Bradley PLC and ensure the PLC is set in *Run* mode.

7.7 Test your application

If safe to do so download and login to your application. (See section 3). Click *Debug>Start.*

Select the program *PLC_PRG(PRG)*. Setting the variable *Output* to a value other than zero should result in the corresponding controller tag within Studio 5000 assuming the same value. Setting the tag within TIA Portal associated with the output should result in the variable *Input* within the CODESYS Development Environment reflecting the value in TIA Portal. (By default TIA Portal will show the data type as SINT).





8 Profinet IO Device

This example demonstrates data transfer from a Siemens PLC acting as the master and the MCxxxx acting as the slave.

It is expected that the user is familiar with the Siemens TIA portal software. The example was tested with a Siemens S7-1200 PLC and TIA Portal V14.

NOTE The companion project archive *MCe_ProfinetDevice.projectarchive* along with MCeProfinetSlave Siemens TIA Portal V14 project are available to download from Support Suite or is available from your local Drive Centre or supplier.

In this example we will change the IP address of the MCe's port 2 or the MCzx01's port X105 to 192.168.0.2 (see section 8.4). This port will connect to the Siemens PLC.

It is assumed that the default IP address and subnet (192.168.1.1, 255.255.255.0) for port 1 on the MCe200, or X102 on the MCzx01 are configured within the MCxxxx, and an IP address in the same subnet has been configured on the user's PC or laptop. The user's PC should be connected to port 1 (LAN A) of the MCe200 or X102 of the MCzx01. For convenience a second PC or second Ethernet port on the user's PC is used for Connection to the Siemens PLC as required.

If the example archive (*MCe_ProfinetDevice.projectarchive*) is converted to an MCzx01 target then the network must be scanned in the *Communications Settings* tab of the PLC, and subsequently section 8.4 will need to be followed to set the Ethernet port to X105 (including modifying the IP address).

8.1 New project

If you are not installing the project archive (see section 7) then create a new project for your desired device target type (add the device target DDF as required) and configure the gateway by scanning the network (set the network path in the *Communications Settings* tab of the PLC) to ensure your device is visible in the CODESYS Development System. (See section 3.1).

The example and its associated images are based upon the MCe200, but users may choose to use the MCzx01.

8.2 Generate GSDML file for TIA Portal

Within the CODESYS Development Environment click "*Tools>Device Repository…"* on the toolbar. Click "*Install..."* and navigate to the Profinet IO Device folder, select the *CODESYS Profinet Device*, click on the file and then click *Export.* You will be prompted for a location to store the device archive. Once generated, unzip the archive, and locate the GSDML .xml file. Install the GSDML file into TIA portal in the usual way.

8.3 Create a TIA portal application

Within TIA Portal, from the *Hardware Catalog* add *Other Field Devices>PROFINET IO>I/O>3S – Smart Software Solutions GmbH>CODESYS PLC>Head module>CODESYS Profinet Device.*

Select *Other Field Devices>PROFINET IO>I/O>3S – Smart Software Solutions GmbH>CODESYS PLC>Input>.* Then add an *Input 8Bit* and *Output 8Bit* module.

Create a hardware configuration and program in the normal way (including assigning a device name) that will write to Bit0 of the output (%Q2.0) and to read Bit0 of the input (%I2.0). (See the *MCeProfinetSlave* Siemens TIA Portal V14 project for an example). The IP address of the Profinet IO Device should be assigned 192.168.0.2.

Download the application to the Siemens PLC and ensure the PLC is set in *Run* mode.

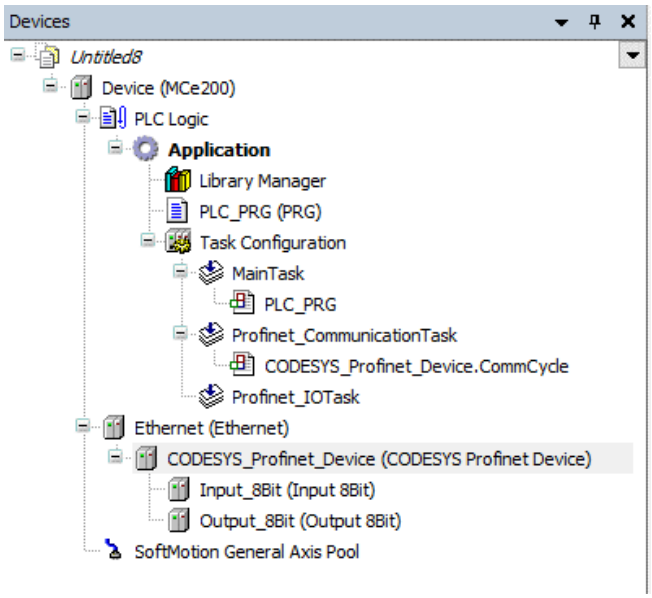
8.4 Adding the Profinet Device

Right click on *Device(MCe200)* in the *Devices* tree of the CODESYS Development Environment, then click "*Add Device…".* From the *Add Device* window that appears, expand the *Profinet IO* folder and select *Ethernet Adapter.* Now click the *Add Device* button.

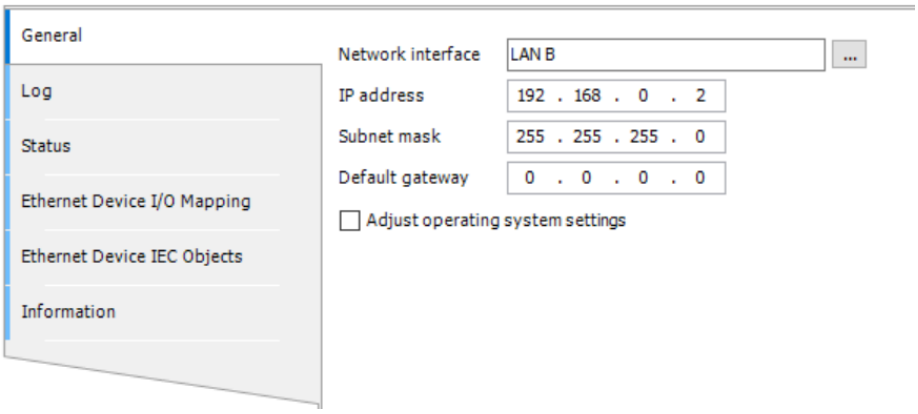
NOTE Do not close the *Add Device* window until all the tree devices have been added.

Click on the *Ethernet* device that has appeared in the *Devices* window, select *Profinet IO Device* Click on the *Ethernet* device that has appeared in the *Devices* window, select *Profinet IO Device* in the *Devices* window, and select *Input 8Bit and Output 8Bit*.

The *Devices* window should look similar to the following.



Click on the *Ethernet* device in the *Devices* window, and in the *General tab* click next to the *Network* interface, in the dialogue that appears assign LAN B to the *Network interface* (X105 for MCzx01), set the IP address to an address in the same subnet as the Siemens PLC, in this example 192.168.0.2 has been used. This address will be required to configure your TIA Portal project.



8.5 Add the program code

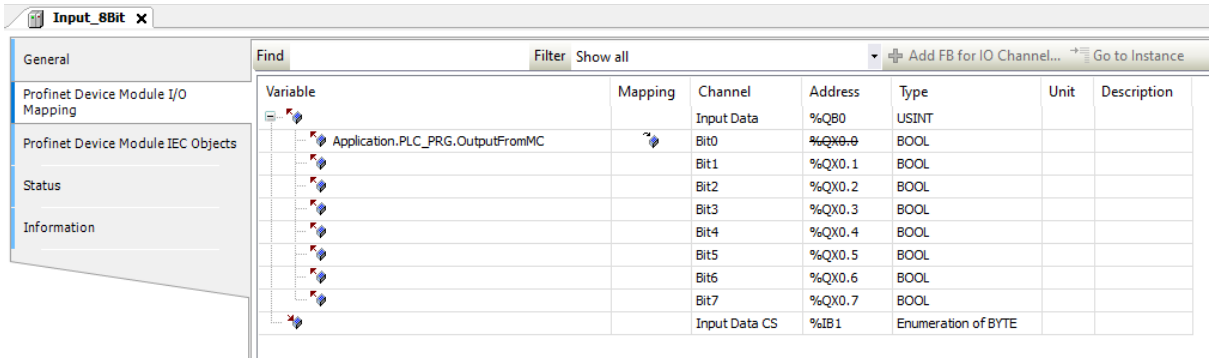
Double click on the *PLC_PRG(PRG)* program in the *Devices* tree and in the declaration window and program window enter the following:

```
//The following is entered in the declaration window or copy and paste
PROGRAM PLC_PRG
VAR
        OutputFromMC:   BOOL;
        InputToMC:      BOOL;
        Bit0_Input:     BOOL;
        Bit0_Output:    BOOL;
END_VAR

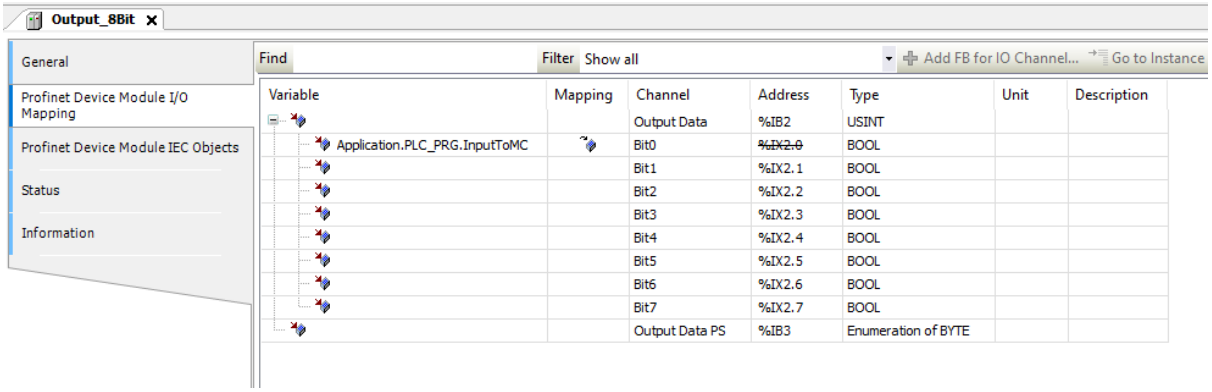
//The following is entered in the code window or copy and paste
OutputFromMC := Bit0_Output;  //Turn on bit 0 in Input_8Bit to Siemens PLC
Bit0_Input := InputToMC;      //Read bit 0 in Output_8Bit from Siemens PLC
```

8.6 Configure the mappings

Click on the *Input_8Bit* device in the *Devices window,* select the *Profinet Device Module I/O Mapping* tab and click in the variable section for Bit0 to add the program variable *OutputFromMC*.



Click on the *Output_8Bit* device in the *Devices window,* and as above add the mapping *InputToMC*.



8.7 Test your application

If safe to do so download and login to your application. (See section 3). Click *Debug>Start.*

Select the program *PLC_PRG(PRG)*. Setting the variable *Bit0_Output* to true should result in the corresponding tag within TIA Portal becoming true. Setting the tag within TIA Portal associated with the output should result in *Bit0_Input* within the CODESYS Development Environment becoming true. (See *Main [OB1]* program and *Watch table_1* in the TIA Portal example project).

9 Modbus TCP/IP Client

This example Modbus TCP/IP communications between the MCe200 acting as a client (master) and a Unidrive M700 acting as a server (slave).

NOTE The companion project archive *MCe_Modbus_client.projectarchive* is available to download from Support Suite or is available from your local Drive Centre or supplier.

It is assumed that the default IP address and subnet (192.168.1.1, 255.255.255.0) for port 1 on the MCe200, or X102 on the MCzx01 are configured within the MCxxxx, and an IP address in the same subnet has been configured on the user's PC and in the drive. The user's PC should be connected to one port of the drive's Ethernet interface, the second port is then connected to port 1 (LAN A) of the MCe200 or X102 of the MCzx01.

If the example archive (*MCe_Modbus_client.projectarchive*) is converted to an MCzx01 target then the network must be scanned in the *Communications Settings* tab of the PLC, and subsequently section 0 will need to be followed to set the Ethernet port to X102.

Drive parameter 01.014 should be set to *Preset*, and drive parameter 06.043 should be set to *On.*

9.1 New project

If you are not installing the project archive (see section 7) then create a new project for your desired device target type (add the device target DDF as required) and configure the gateway by scanning the network (set the network path in the *Communications Settings* tab

of the PLC) to ensure your device is visible in the CODESYS Development System. (See section 3.1).

The example and its associated images are based upon the MCe200, but users may choose to use the MCzx01.

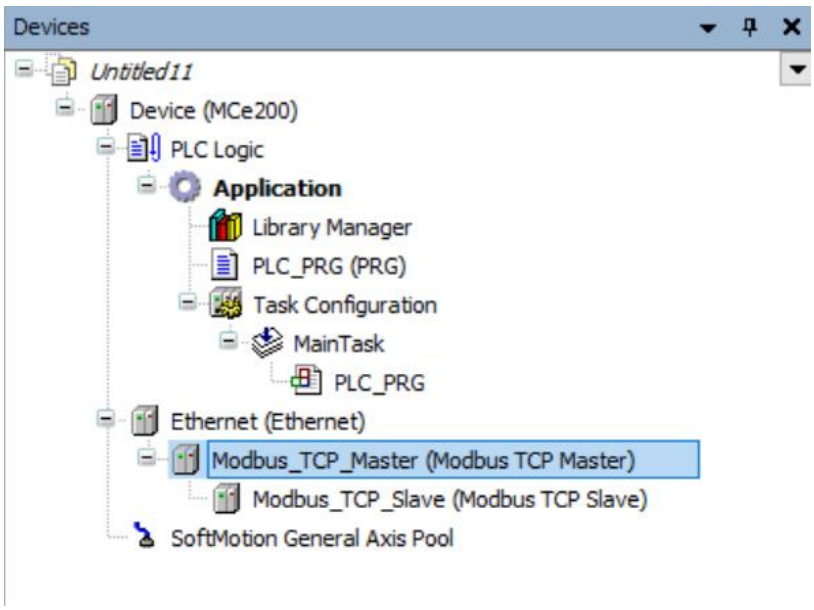
9.2 Adding the Modbus TCP Master

Right click on *Device(MCe200)* in the *Devices* tree of the CODESYS Development Environment, then click "*Add Device…".* From the *Add Device* window that appears select *Ethernet Adapter.* Now click the *Add Device* button.

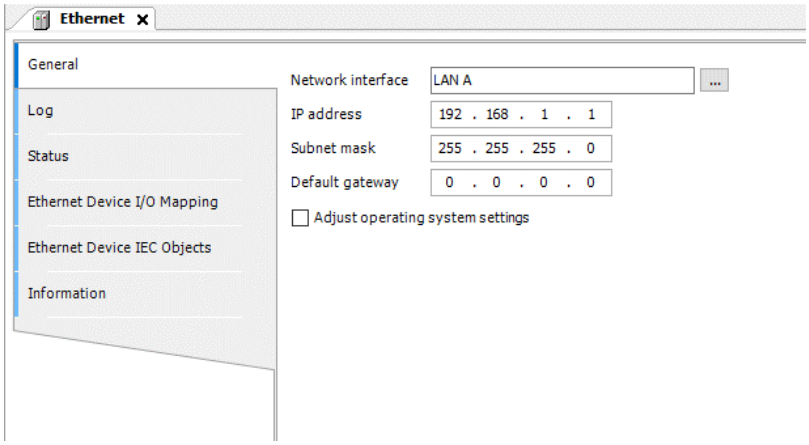
NOTE Do not close the *Add Device* window until all the tree devices have been added.

Click on the *Ethernet* device that has appeared in the *Devices* window, select *Ethernet Adapter* Click on the *Ethernet* Adaptor that has appeared in the *Devices* window, add *Modbus>Modbus TCP Master,* then click *Modbus TCP Master* in the *Devices* window, add the *Modbus TCP Slave.*

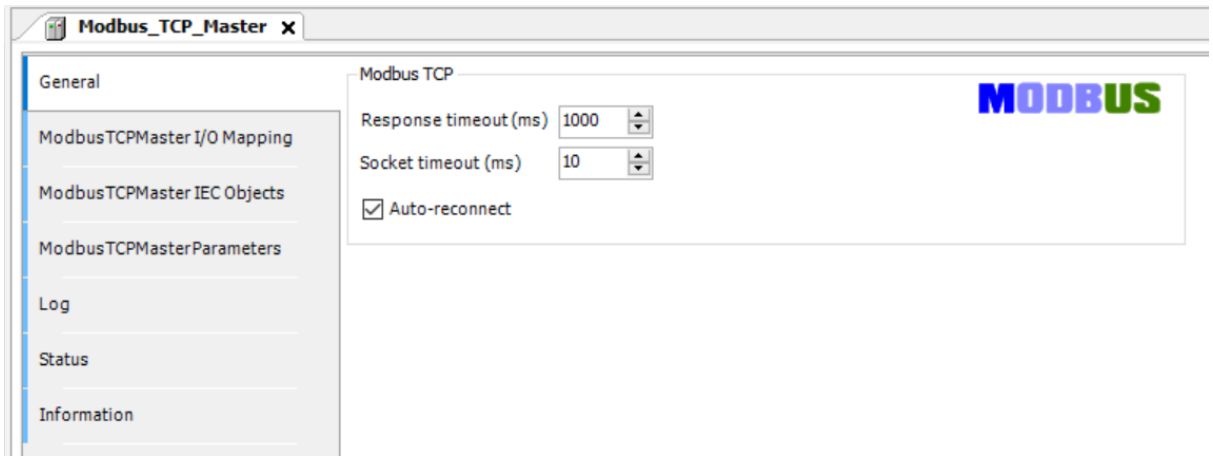
The *Devices* window should look similar to the following.



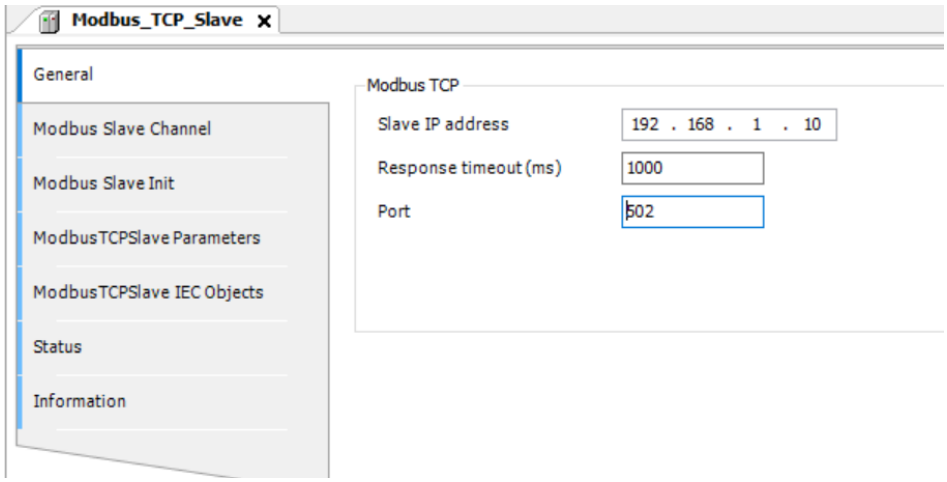
Click on the *Ethernet* device in the *Devices* window, and in the *General tab* click next to the *Network* interface, in the dialogue that appears assign LAN A to the *Network interface* (for the MCzx01 select port X102), set the IP address to an address in the same subnet as the drive and PC, in this example the default address for LAN A of 192.168.1.1 has been used.



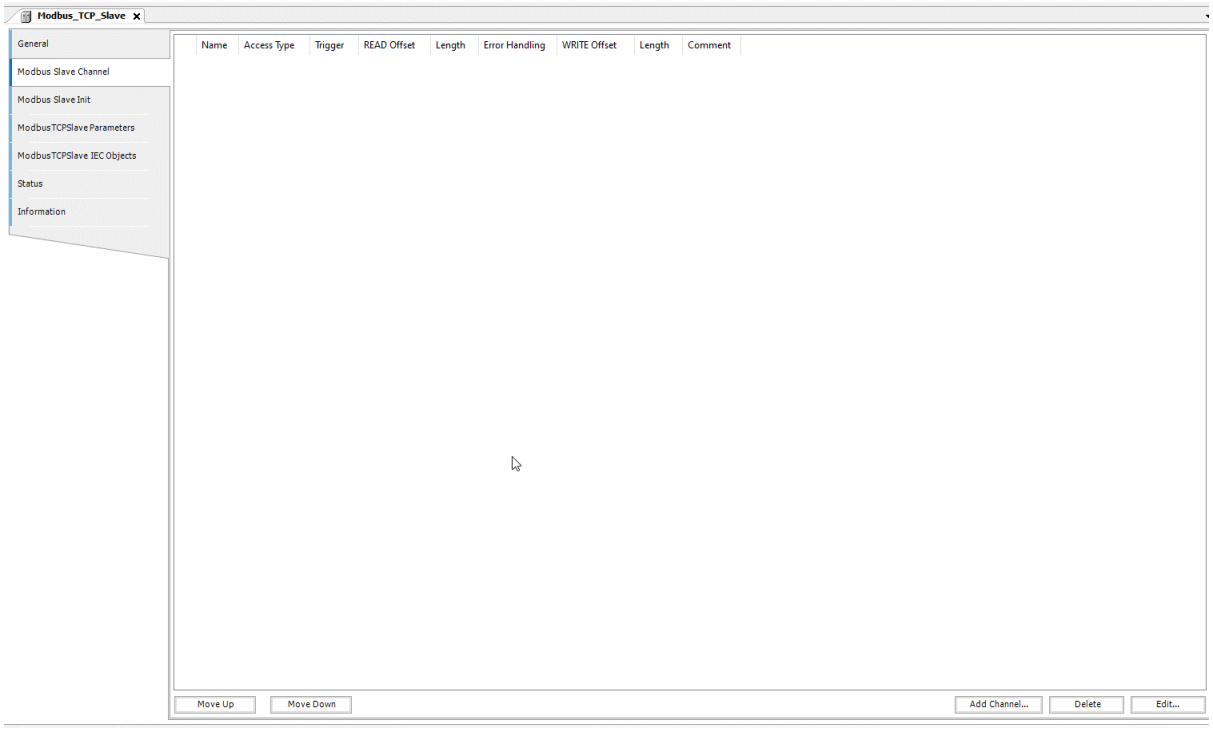
Click on the *Modbus_TCP_Master* in the *Devices* window, and in the *General* tab check the *Auto-reconnect* box.



Click on the *Modbus_TCP_Slave* in the *Devices* window. In the *General* tab enter the drive's IP address, in this example 192.168.1.10.



Select the *Modbus Slave Channel* tab and click the *Add Channel…* button.



In this example we wish to access a 16-bit drive parameter's value. The *Offset* for the register that represents the individual drive parameter is calculated using the following relationship.

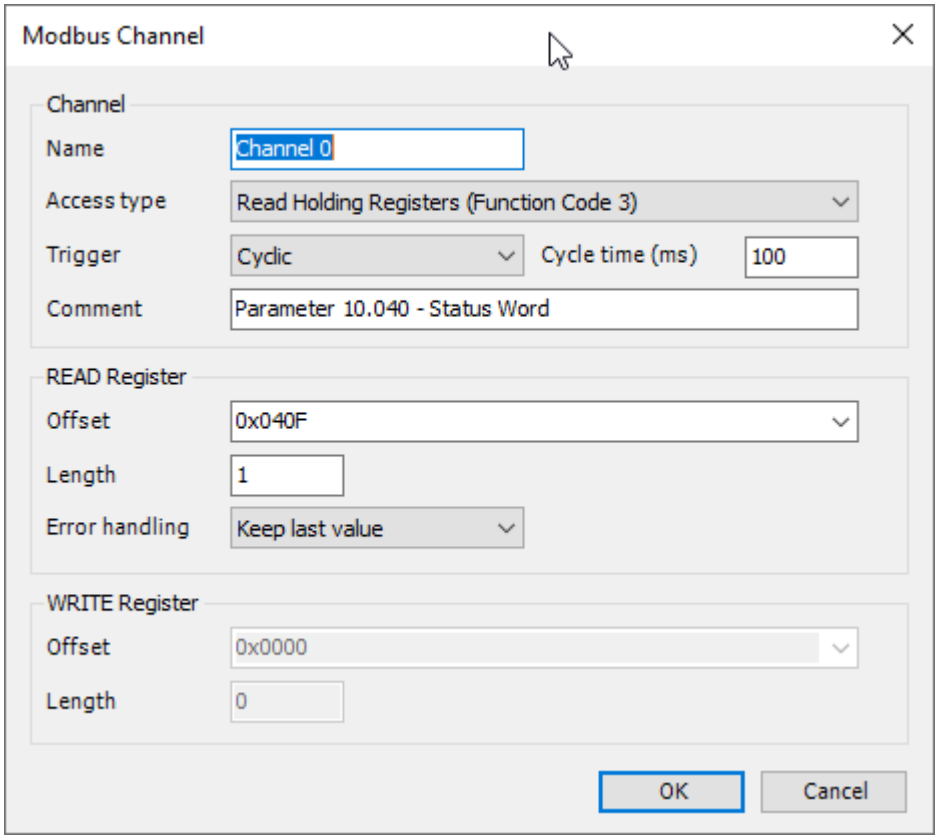
*Offset = drive menu * 100 + drive parameter -1*

For example parameter 10.040

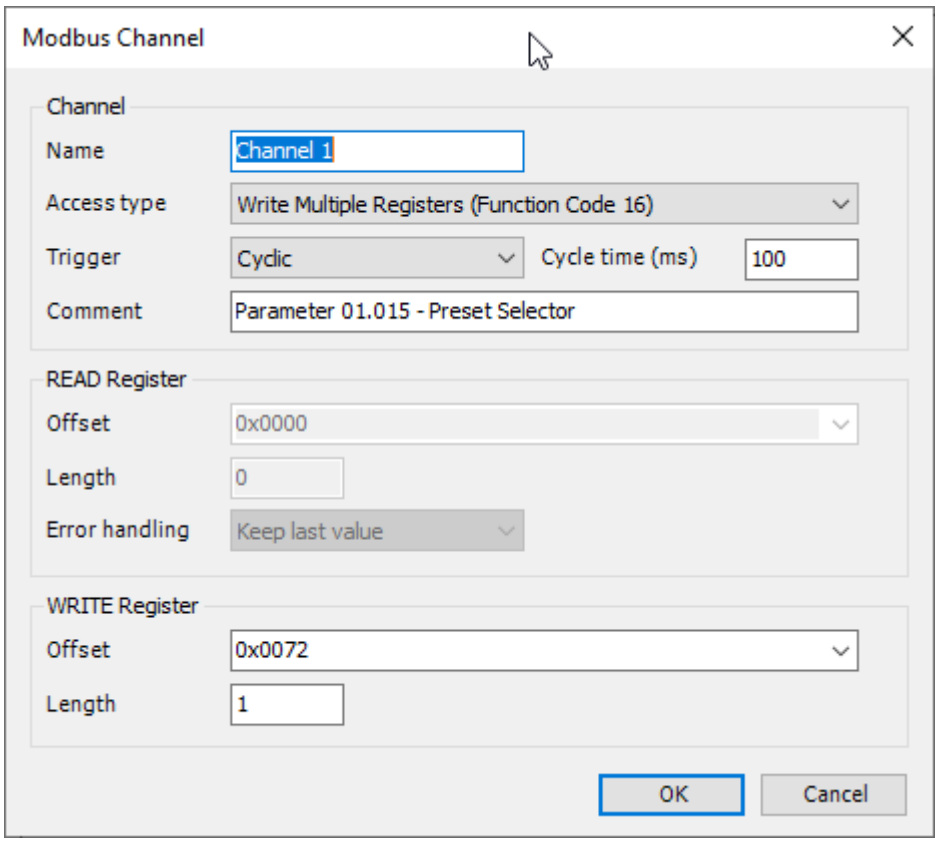
*1039 = 10 * 100 + 40 -1*

The offset is expressed in hexadecimal base.

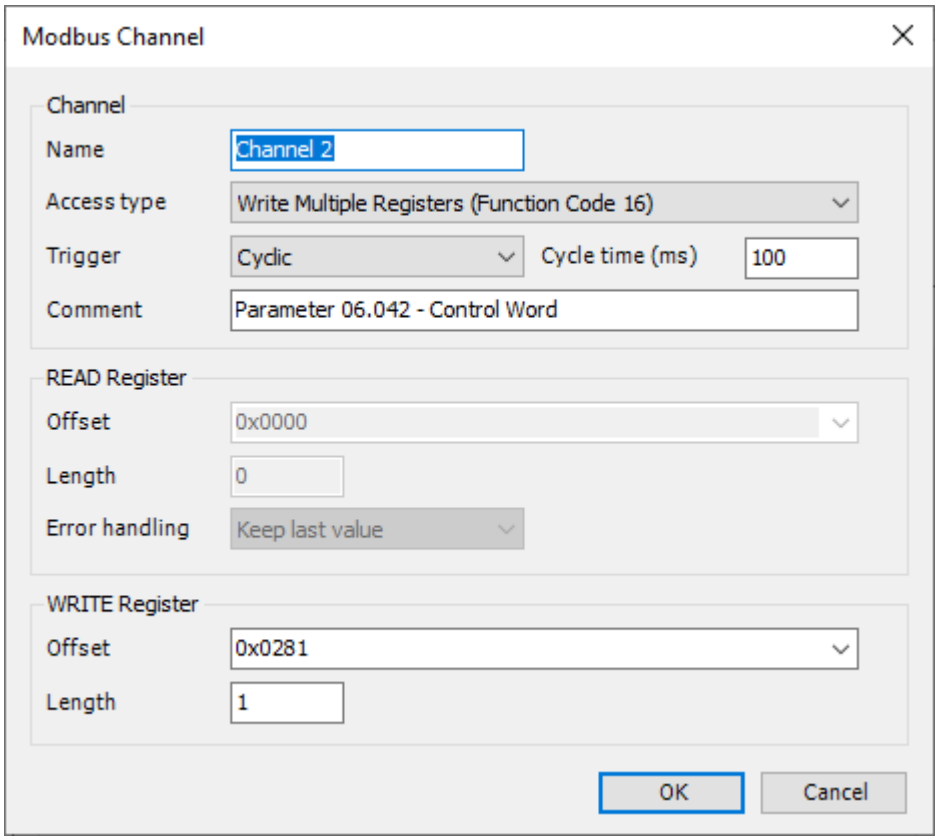
Therefore, to read the drive's status word (parameter 10.040) we will use Modbus *Function Code 3* and an offset of 0x040F. Enter the details in the *Modbus Channel* box and then click *OK.*



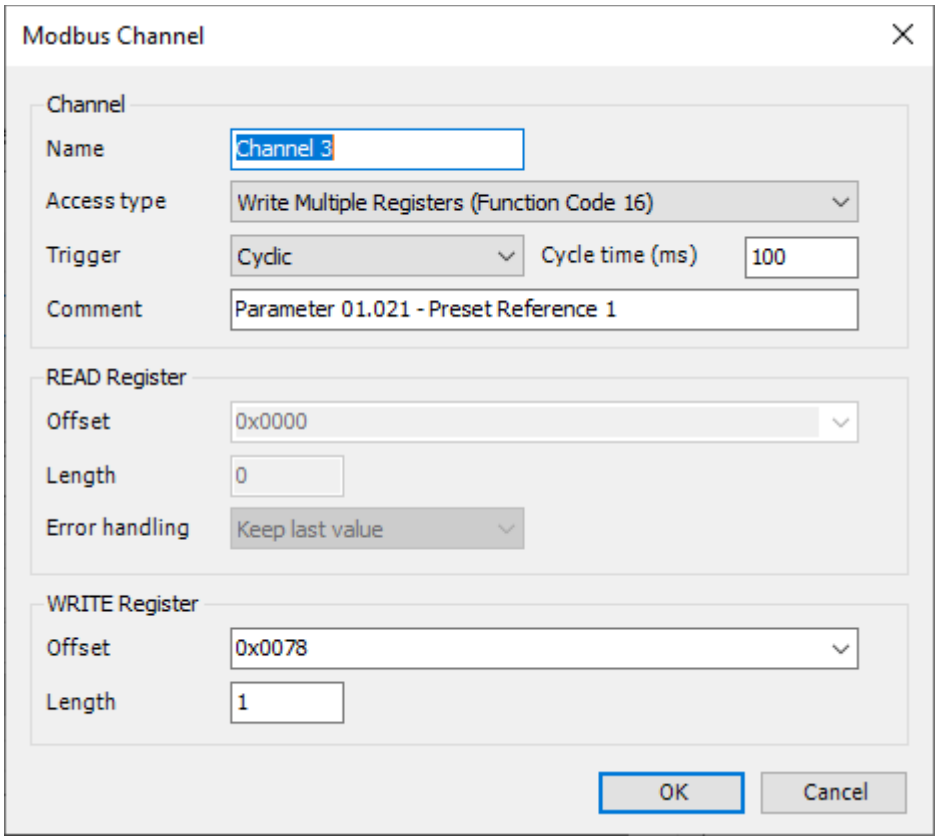
Repeat for the following channel, here we are writing to the drive using *Function Code 16* to write to the Preset Selector (parameter 01.015).



And again to write to the Control Word.



Then for the final channel we will write to Preset Reference 1.



9.3 Add the program code

Double click on the *PLC_PRG(PRG)* program in the *Devices* tree and in the declaration window and program window enter the following:

```
//The following is entered in the declaration window or copy and paste

PROGRAM PLC_PRG
VAR
	Run:			BOOL;
	ControlWord:		WORD;
	Speed:			WORD;
	StatusWord:		WORD;
	Status:			WORD;
	PresetRefSelector:	WORD;
END_VAR



//The following is entered in the code window or copy and paste

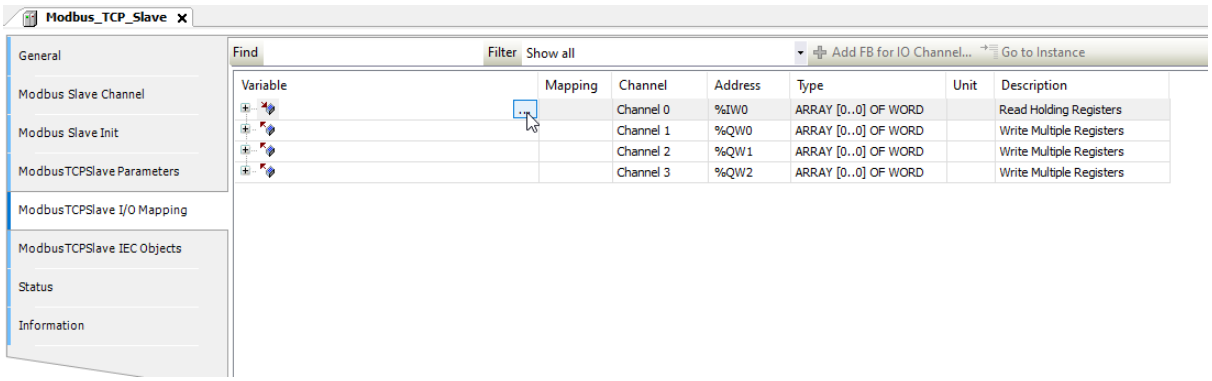
//Set Preset selector to 1
IF PresetRefSelector <> 1 THEN
	PresetRefSelector := 1;
END_IF

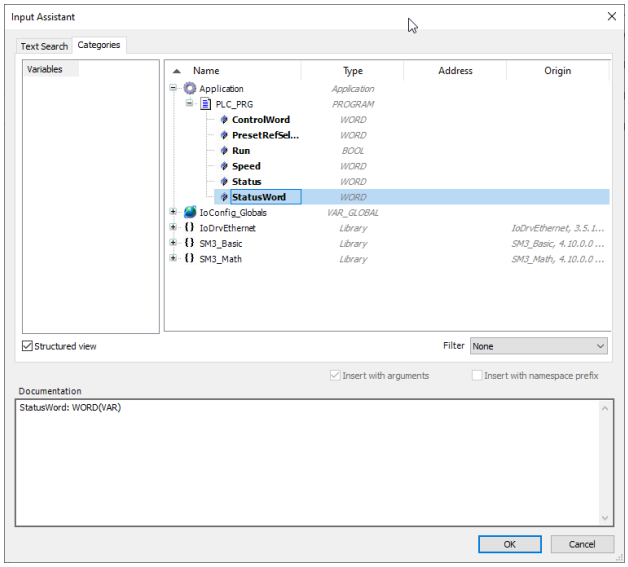

//Run the motor
IF Run THEN
ControlWord := 131;		//Drive's control word = Auto, Run Forward, Drive Enable.
Speed := Speed + 1;		//Speed of motor.  Preset ref 1.
ELSE
	ControlWord := 0;
	Speed := 0;
END_IF

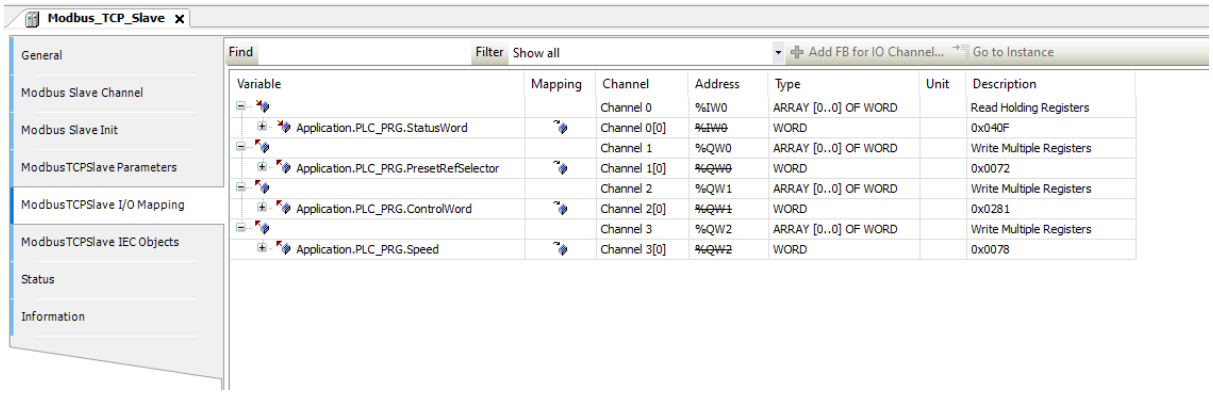
Status := StatusWord;
```

9.4 Configure the mappings

Click on the *Modbus_TCP_slave* in the *Devices* window and select the *Modbus TCPSlave I/O Mapping* tab and click in the variable section to add the program variables for each mapping.







9.5 Test your application



The motor's shaft may rotate.

If safe to do so download and login to your application. (See section 3). Click *Debug>Start.*

Select the program *PLC_PRG(PRG)*. It is assumed that the drive is correctly configured for its motor and the drive's STO is turned off.

Turn on the drive's STO We will command the motor's shaft to rotate by writing to the drive's control word and preset speed reference. Set *Run* to true, the motor shaft should begin to accelerate.

This concludes a brief introduction to using the Modbus TCP Client with the MCxxxx controllers. For further information regarding specific features of the CODESYS Development Environment please see the on-line help available within CODESYS. For further details of Modbus TCP/IP and Control Techniques' drives please see the SI-Ethernet and relevant drive's User Guides.

9.6 Supplementary - 32-bit drive parameters

The Modbus protocol does not implicitly know which drive parameters are 16-bit and 32-bit.

Standard Modbus registers are 16 bits in size and reference a single drive/option module parameter. To access a 32-bit data value the multiple read/write services must be used to transfer a contiguous array of 16-bit registers. To instruct the client to select either 16-bit or 32-bit access, bit 14 of the register address is used.

The Control techniques specific scheme for mapping 32-bit drive parameters via Modbus TCP/IP is as follows:

Set bit 14 of the register offset.

Set the number of registers to two. (If considering a single 32-bit parameter).

The drive then uses two consecutive 16-bit Modbus registers.

Hardware version 1 Unidrive M and Digitax HD drives use 'big endian' format; i.e. most significant word transmitted first. Hardware version 2 Unidrive M and Digitax HD drives use 'small endian' format; i.e. least significant word transmitted first.

For example:

Drive Menu * 100 + Drive Parameter -1

Pr20.021 = 20*100 + 21 -1 = 2000 or 0x07E4

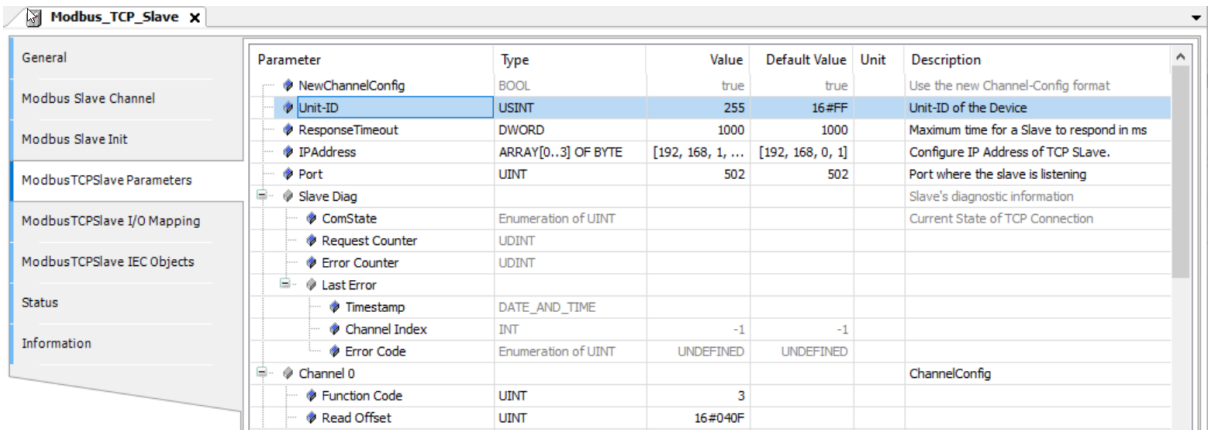
To specify the drive replies with 32-bits, add 0x4000 to set bit 14.

0x07E4 + 0X4000 = 0x47E4

9.7 Supplementary - Access to slot parameters

To access drive slot parameters with Modbus TCP/IP the Unit ID (Unit Identifier section of the MBAP header) is assigned a value.

Set the *Unit-ID* in the *ModbusTCPSlave I/O Mapping* tab to one of the following values to access the required slot



.

Unit Identifier	Destination
0 or 255	Drive
1	Slot 1
2	Slot 2
3	Slot 3 (internal slot Digitax HD M75x
4	Slot 4 (internal slot M70x)

10 OPC UA server

This example uses the OPC UA server running on an MCe200 and the software OPC UA client from Unified Automation.

NOTE The companion project archive *MCe_OPCUA.projectarchive* is available to download from Support Suite or is available from your local Drive Centre or supplier.

It is assumed that the default IP address and subnet (192.168.1.1, 255.255.255.0) for port 1 on the MCe200, or X102 on the MCzx01 are configured within the MCxxxx, and an IP address in the same subnet has been configured on the user's PC. . The user's PC should be connected to port 1 (LAN A) of the MCe200 or X102 of the MCzx01.

10.1 New project

If you are not installing the project archive (see section) then create a new project for your desired device target type (add the device target DDF as required) and configure the gateway by scanning the network (set the network path in the *Communications Settings* tab of the PLC) to ensure your device is visible in the CODESYS Development System. (See section 3.1).

The example and its associated images are based upon the MCe200, but users may choose to use the MCzx01.

10.2 New project

If you are not installing the project archive (see section 10) then create a new project for your desired device target type, add the device target DDF as required and scan the network to ensure your device is visible in the CODESYS Development System. (See section 3). Set the network path in the *Communications Settings* tab of the PLC.

10.3 Adding the program

Double click on the *PLC_PRG(PRG)* program in the *Devices* tree and in the declaration window and program window enter the following:

```
//The following is entered in the declaration window or copy and paste

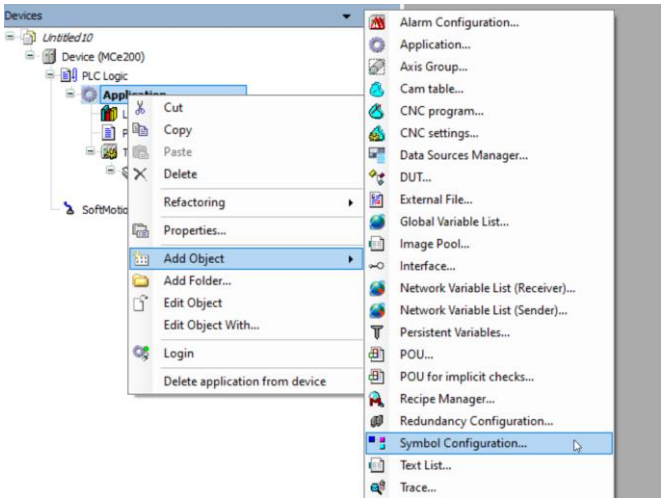
PROGRAM PLC_PRG
VAR
        Out:    INT;
        In:     INT;
        Input: INT;
END_VAR


//The following is entered in the code window or copy and paste

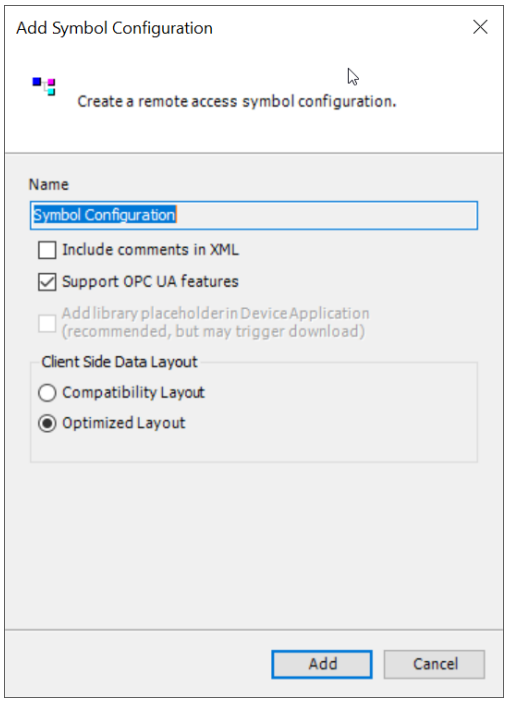
Out := Out + 1;
Input := In;
```

10.4 Adding the Symbol Configuration

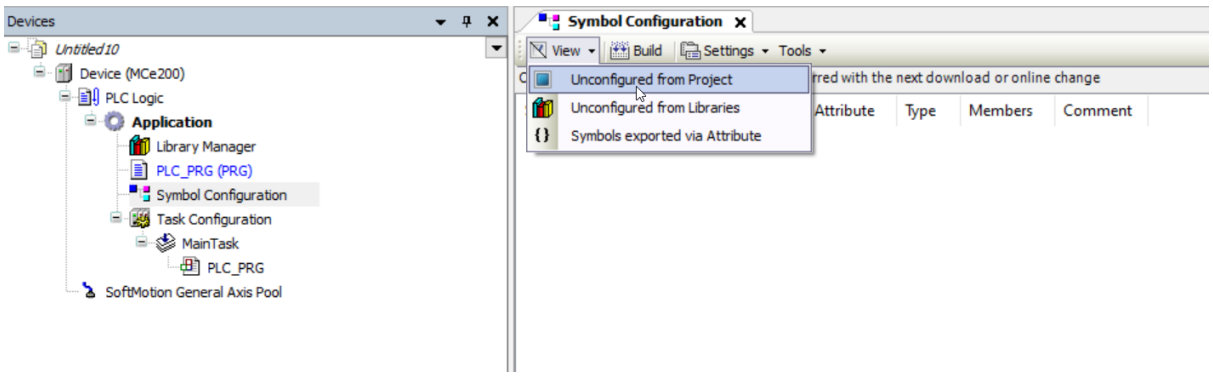
Right click on *Application* in the *Devices* tree and select *Add Object*. Select *Symbol Configuration*.



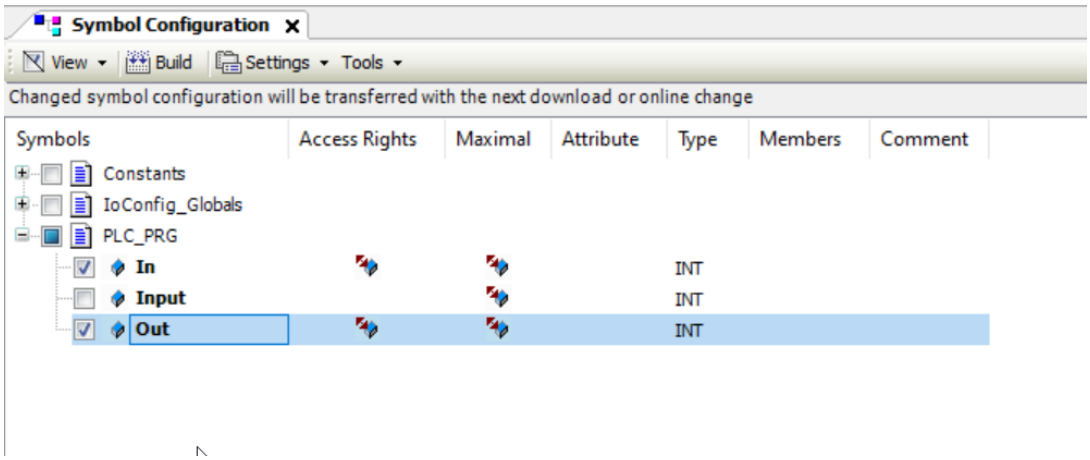
Click *Add.*



Build the project by clicking the *Build* button. Select the *Symbol Configuration* and enable the *Unconfigured from Project* button.



Select the variables from the program as shown.



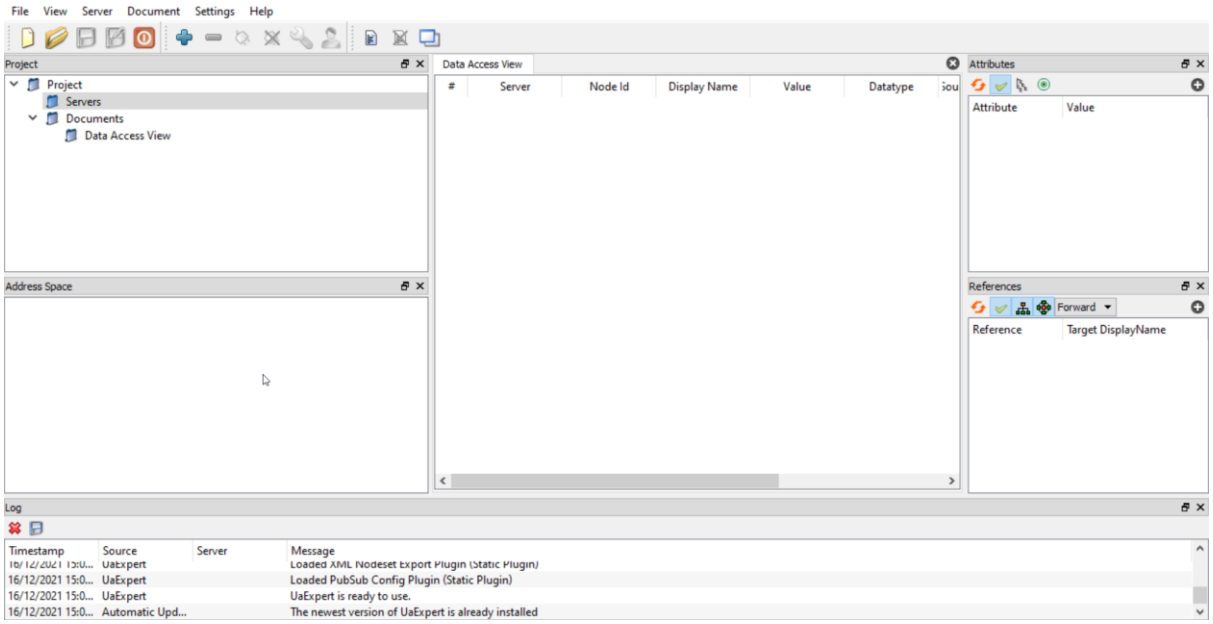
If safe to do so download and login to your application. (See section 3). Click *Debug>Start.* The OPC UA server should now be running and ready for the client to access it.

10.5 Configuring UaExpert

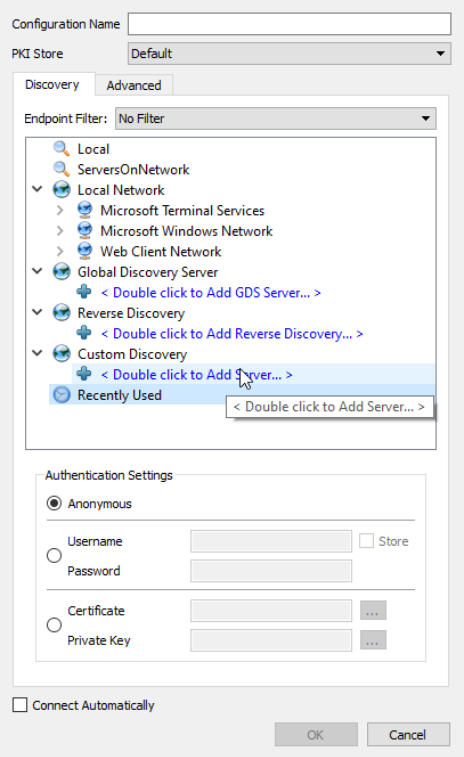
UaExpert Copyright © Unified Automation GmbH can be obtained from the Automation Expert's web site at https://www.unified-automation.com.

Download and install UaExpert following the manufacturer's instructions. Open the software, a new project is created for you.

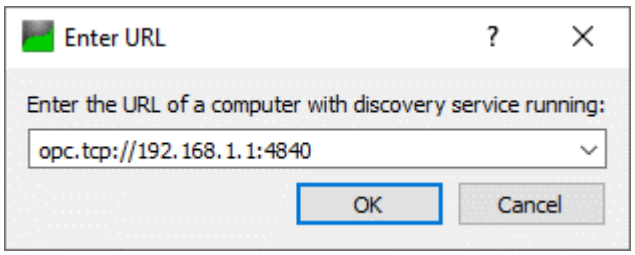
Click the plus button in the toolbar.



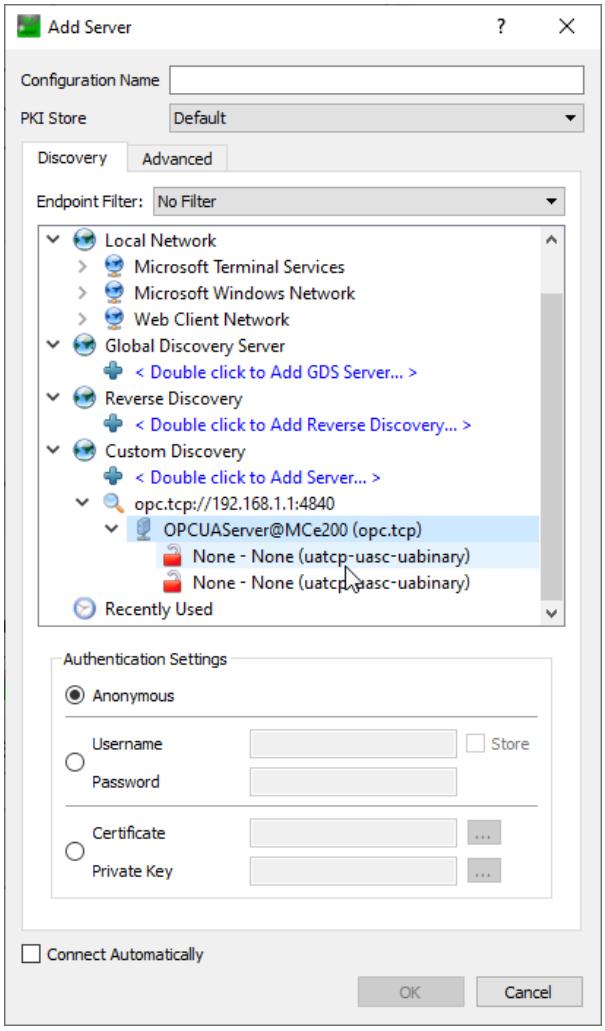
Select *Custom Discovery*.



Enter the following, noting the IP address is that of the MCe

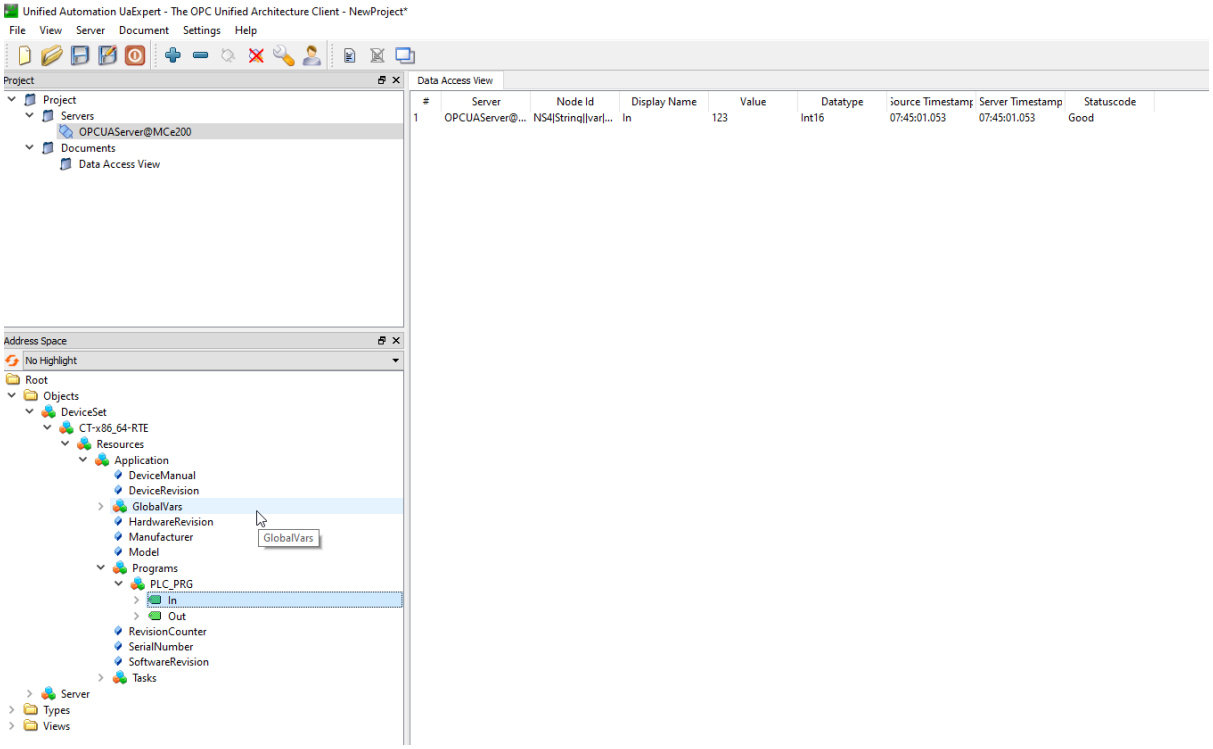


Expand the folder and for this demonstration select *None* for no security.

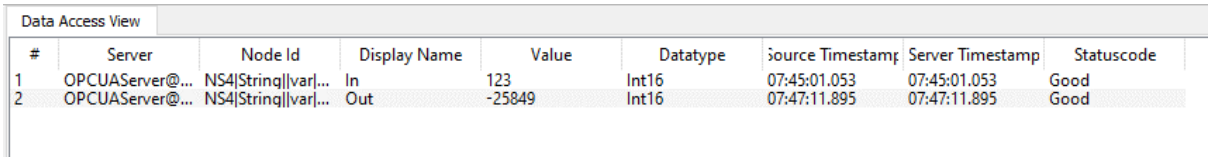


Click *Connect Server* on the toolbar 

Several seconds later the *Address Space* should populate. Locate the PLC variables from your program and drag them both from the *Address Space* window into the *Data Access View* window.



From the *Data Access View* you can view and edit the variables used in the MCe program.



11 Retained variables

1MB of physical non-volatile random-access memory (NVRAM) is a standard feature of the MCz201and MCz601, the MCe200 provides 128KB Utilizing NVRAM provides the user with a simple method of implementing non-volatile data storage.

The following table documents the behaviour of each type of variable.

Type of Variable	Reset Warm	Reset Cold	Reset Origin	Uncontrolled Shutdown
VAR	Reset	Reset	Reset	Reset
VAR RETAIN	Saved	Reset	Reset	Saved
VAR PERSISTENT RETAIN	Saved	Saved	Reset	Saved

11.1 VAR RETAIN

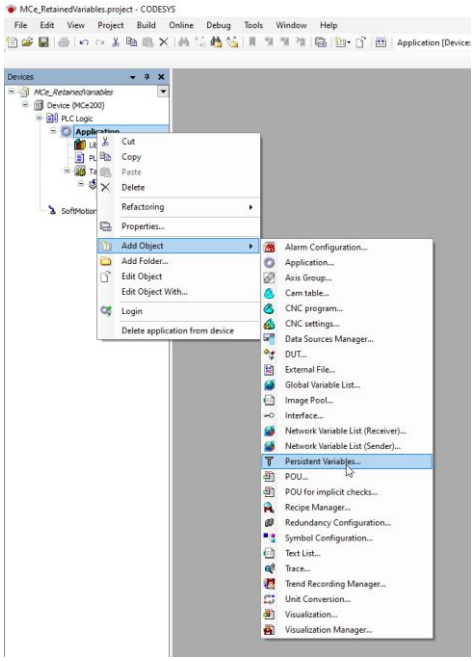
To define a VAR RETAIN, simply insert the variable's definition between VAR RETAIN and END_VAR as shown in the following example.

```
PROGRAM PLC_PRG
VAR
        x:              DINT;
        y:              INT;
        z:              BYTE;
END_VAR

VAR RETAIN
        MyRetain:       DINT;
END_VAR
```

11.2 VAR PERSISTENT RETAIN

If the user wants to use the VAR Persistent retain storage method, then a *Persistence Variables…* object needs to be added to the project.

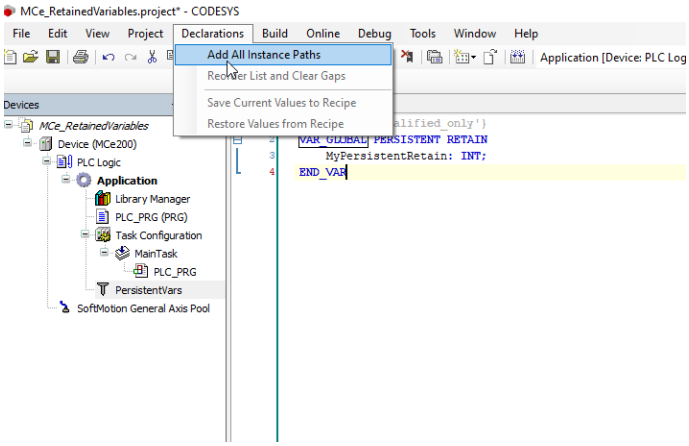


And the variable declared in the global variable list. (The default is for the pragma *attribute 'qualified_only'* to be included. This pragma means that variables from a global variable list can only be addressed by specifying the global variable list name and variable separated by a full stop).

```
{attribute 'qualified_only'}
VAR_GLOBAL PERSISTENT RETAIN
        MyPersistentRetain:    INT;
END_VAR
```

Build the project (F11).

Add All Instance Paths should be selected from the Declarations menu.



11.3 Critical user data

If the user's data integrity is deemed critical, it is strongly suggested that a UPS is incorporated to protect against loss of power while an NVRAM write may be occurring.

12 Glossary of terms

AB: A digital incremental encoder with two output channels.

Address: This is the unique network identification given to a networked device to allow communication on a network. When a device sends or receives data the address is used to determine the source and the destination of the message.

Byte: A collection of eight binary digits that collectively store a value.

Control word: A collection of binary digits that are used to control the drive.

CPR: Counts Per Revolution. The number of quadrature decoded states that exist between an encoder's two incremental outputs (PPR * 4). Alternatively referred to as lines per revolution. For communication encoders such as EnDat CPR equals the number (n) of bits per revolution 2^n the encoder supports.

EoE: Ethernet over EtherCAT. An EtherCAT master tunnels standard Ethernet communication via EtherCAT. Control Techniques' PC tools including Connect receive and send standard Ethernet frames and are unaware of the EoE protocol.

MCzx01: In this document MCzxo1 refers to the MCz201 and MCz601.

MCxxx: In this document MCxxx refers to the MCe200, MCz201 or MCz601.

PC: Personal computer.

PLC: Programmable logic controller.

PPR: Pulses Per Revolution. Represents the number of pulses an incremental rotary encoder will have on either of its outputs over a single revolution.

rpm: Revolutions per minute. Measure of rotational speed (min^{-1}).

RTU: Remote Terminal Unit. In the case of the PC tool Connect's terminology RTU refers to Modbus RTU. This is one of the serial protocols used by Connect (and other Control Techniques' PC tools) and is transmitted over EIA-485 hardware. Sometimes in the case of Control Technique's products referred to as "serial".

Status word: A value that denotes the status of the drive. Each bit within the word will have a specific meaning.

STO: Safe Torque Off.

Word: A collection of sixteen binary digits.



0478-0683-01