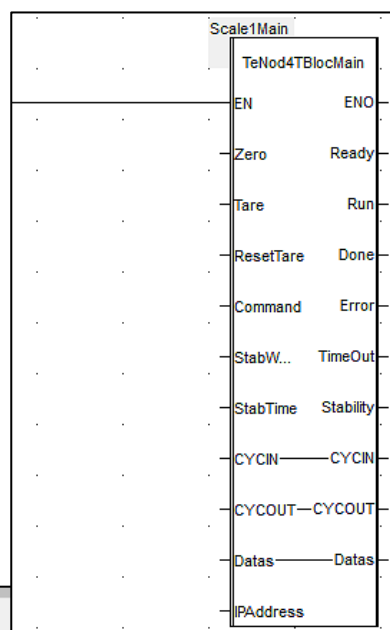
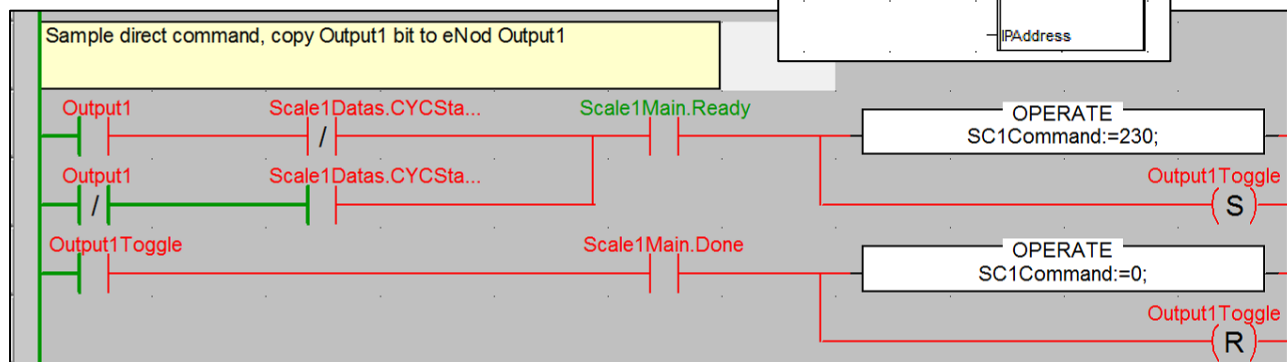




Function Block for eNod4-T EIP

For Unity Pro, Schneider Electric



1 GENERAL PRESENTATION	3
2 INSTALLING EDS FILE IN UNITY PRO.....	4
2.1 Configuring a NOC card with a M340 PLC.....	4
2.2 Installing the EDS file in Unity Pro.....	6
3 ADDING ENOD4T IN THE PROJECT.....	9
3.1 Adding eNod4T on the Ethernet configuration.....	9
3.2 Adding “DFB instructions” in the project.....	11
3.2.1 Selection instructions	11
3.2.2 Installing Add-on instructions.....	12
4 USING ADD-ON INSTRUCTIONS IN THE PROJECT.....	14
4.1 Main bloc “TeNod4TBlocMain”	14
4.1.1 Creating the instance	14
4.1.2 Select global data for the eNod.....	15
4.1.3 Selecting implicit data from hardware configuration	15
4.1.4 Enter the Ethernet/IP link to the eNod.	16
4.1.5 Using the instruction.....	16
4.1.6 Compatibility with different Unity versions.....	19
4.2 “TeNod4TBlocBackupRestore” to restore and backup configuration	20
4.2.1 Creating the instance	20
4.2.2 Select global data for the eNod.....	21
4.2.3 Using the instruction	21
4.3 “TeNod4TBlocCalibration” to calibrate a scale	22
4.3.1 Creating the instance	22
4.3.2 Select global data for the eNod.....	23
4.3.3 Using the instruction	24
4.4 “TeNod4TBlocFilters” to read and write filter configuration.....	31
4.4.1 Creating the instance	31
4.4.2 Select global data for the eNod.....	31
4.4.3 Using the instruction	32
4.5 “TeNod4TBlocLegal” to manage DSD legal values.....	33
4.5.1 Creating the instance	33
4.5.2 Select global data for the eNod.....	33
4.5.3 Using the instruction	34
5 COMMUNICATION ERROR CODES	36
6 BASIC COMMUNICATION.....	38

1 GENERAL PRESENTATION

eNod4 is a high-speed digital process transmitter with programmable functions and powerful signal processing capabilities. **eNod4** offers operating modes for advanced static and dynamic process control.

The eNod4-T Block function for Unity Pro is an open source ladder code that eases the integration of an eNod4 with Schneider Electric PLC. This tool integrates pre-programmed ladder functions for Unity Pro. You can quickly get your measurement data, calibrate or restore your eNod4 using this code.

This user manual describes how to integrate the block function in Unity Pro and communicate with an eNod4-T Ethernet/IP through a Schneider Electric PLC.

2 INSTALLING EDS FILE IN UNITY PRO

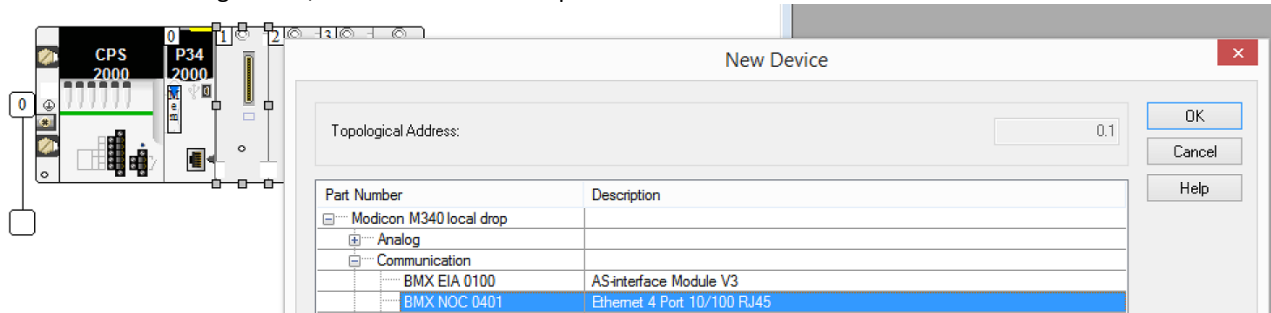
Before using eNod4 in application, EDS file must be imported in Unity Pro to add the eNod4 module in hardware catalog.

EDS cannot be imported if no project is open and if no NOC card is inserted on the hardware configuration. So before importing the EDS file, open the project where you plan to use the eNod4T scale. The project must be with a M580 CPU or a M340 CPU with a BMXNOC0401 card. This is to have an Ethernet link with Ethernet/IP capabilities.

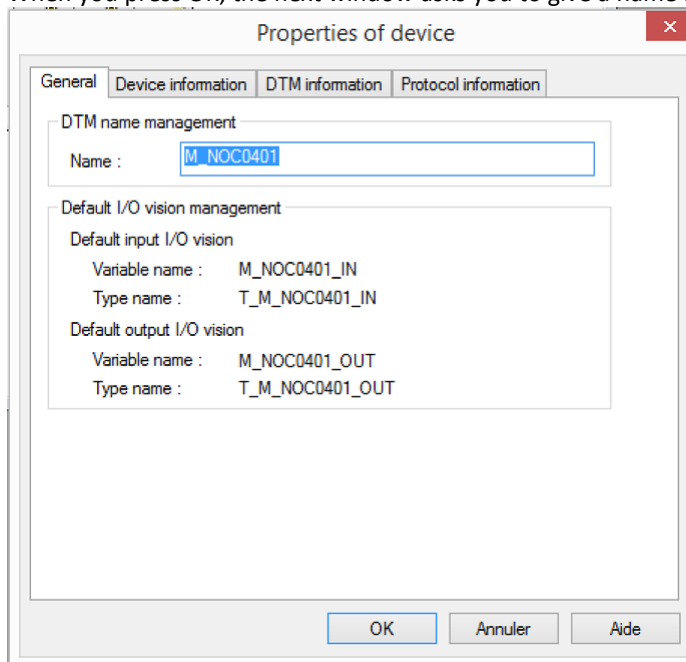
2.1 Configuring a NOC card with a M340 PLC

This procedure is to explain quickly how to configure the NOC card. If you already know how to do it, you can skip this part and go to “B” section.

On the material configuration, double click on a free place and select the NOC0400 or NOC0400.2 card.

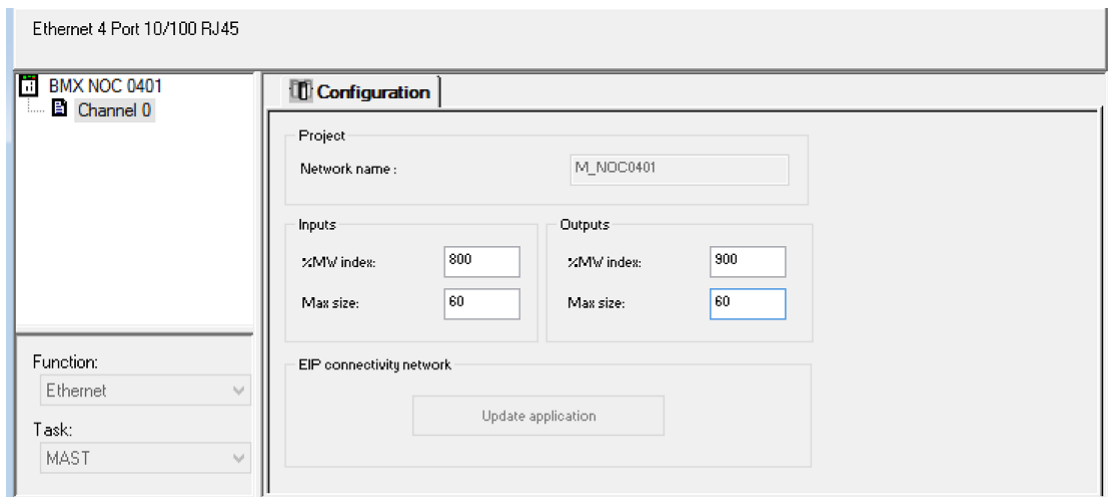


When you press OK, the next window asks you to give a name for the card:



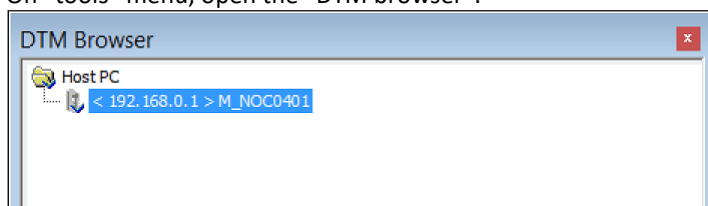
You can change the name or just click OK to continue.

Then double click on the NOC0401 card to open card properties. On the properties you need to define where on the %MW memory the input data and output data will be copied. You need to set a data length equal or bigger of the data received or sent. If the sizes are too small, you will have an error message during the program compilation.

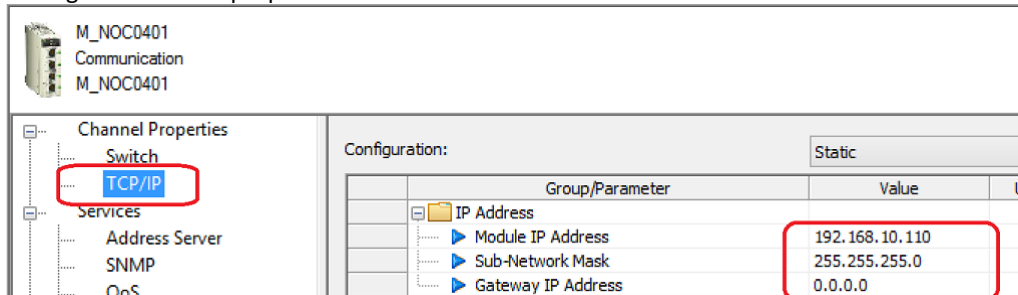


Select the base addresses you want and the data length. Then confirm use the “validate” on the tool bar.

On “tools” menu, open the “DTM browser”.



On the DTM browser, double click on the NOC card to open the NOC DTM properties. Then go to “channel properties->TCP/IP” to set the IP address of the NOC card.

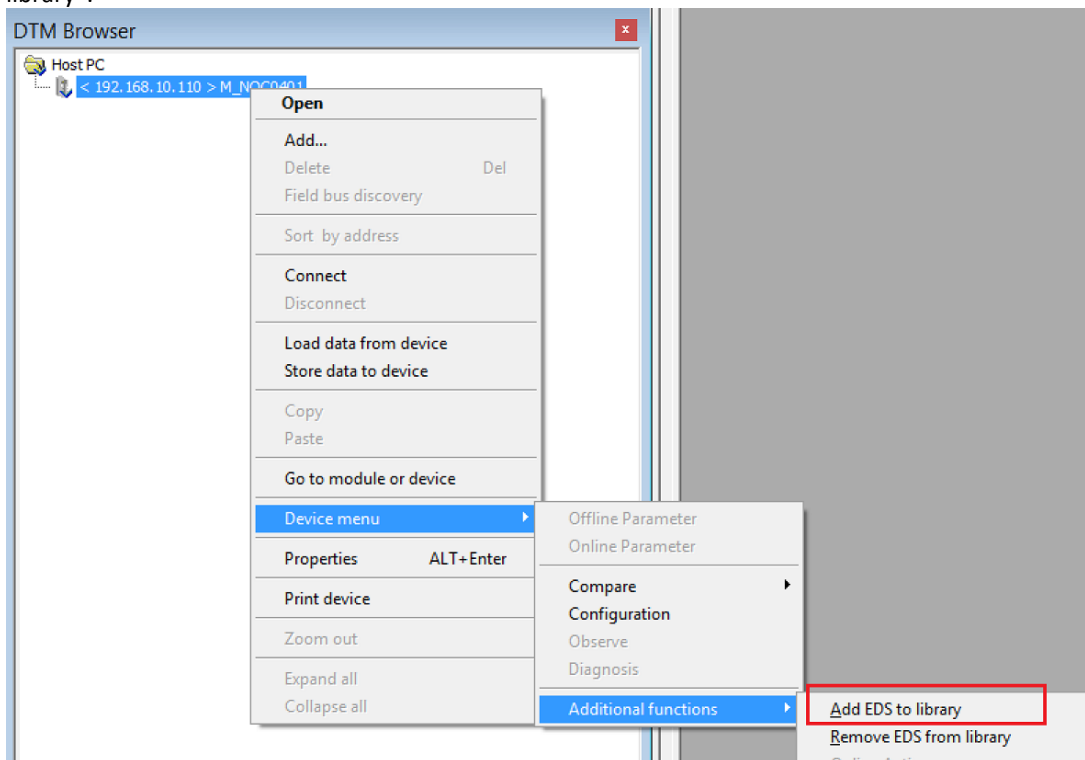


Click “OK” to confirm.

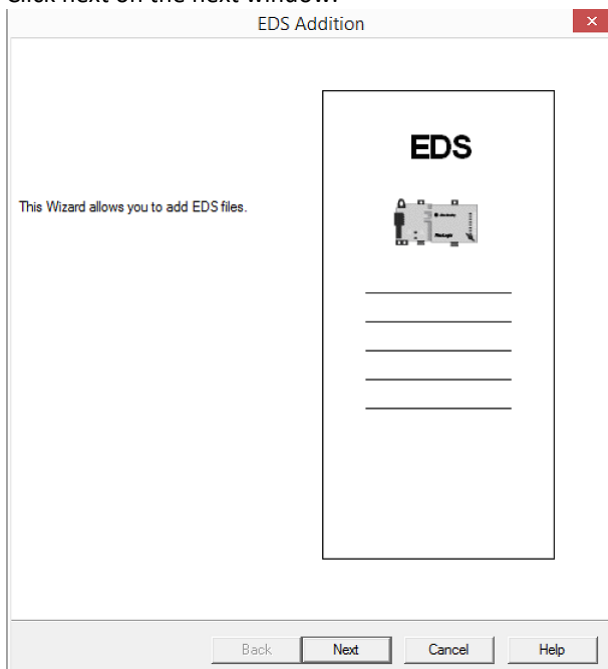
2.2 Installing the EDS file in Unity Pro

This needs to be done only once for the eNo4T, the EDS will stay available for all projects.

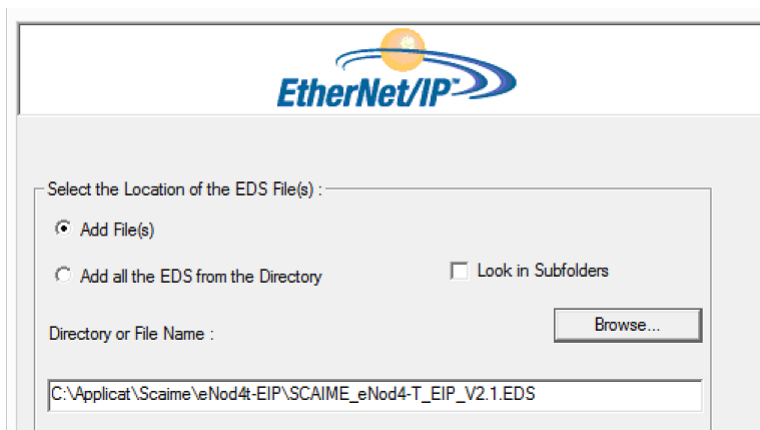
On the DTM navigator, right click on the NOC card and select “Device menu->Additional functions>Add EDS file to library”.



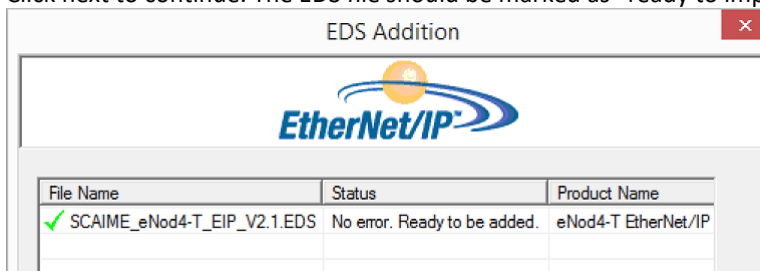
Click next on the next window.



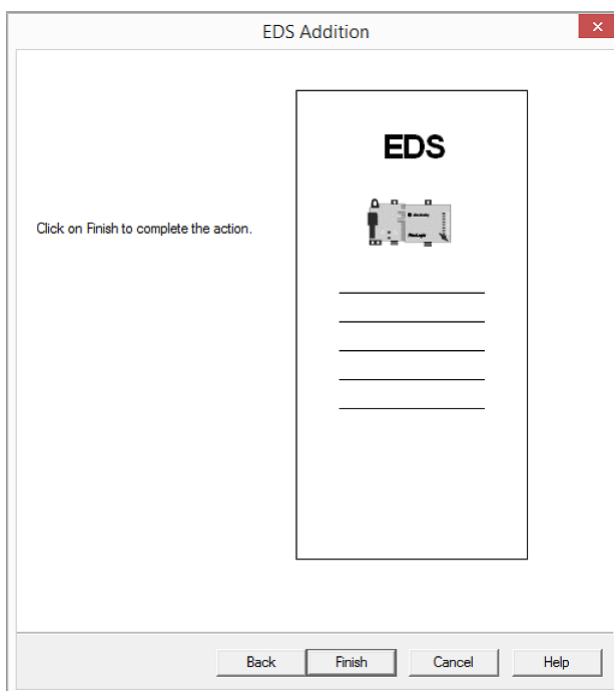
On the next window select the “Add files” option and select the EDS.



Click next to continue. The EDS file should be marked as “ready to import no error”.

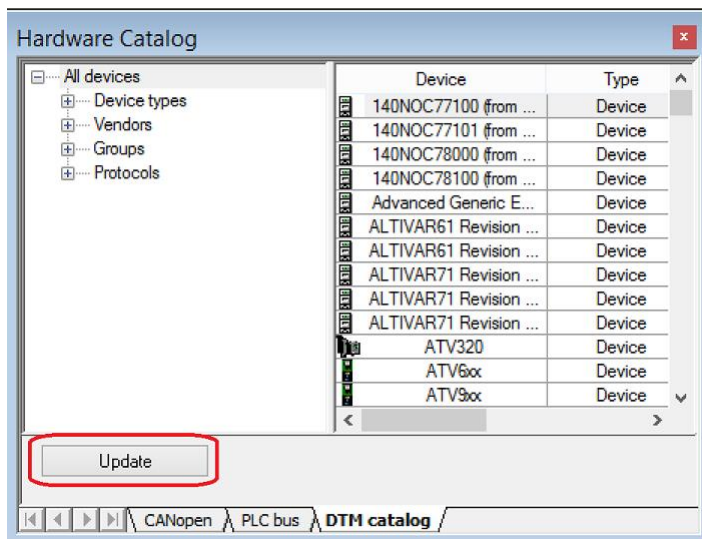


Click next to import the file.



Click on “finished” to close the importation tool.

In Unity program, go to “tools menu->Hardware catalog”. And then select the “DTM catalog” tab. On the DTM click on the “update” button to compile the catalog with the new EDS file.



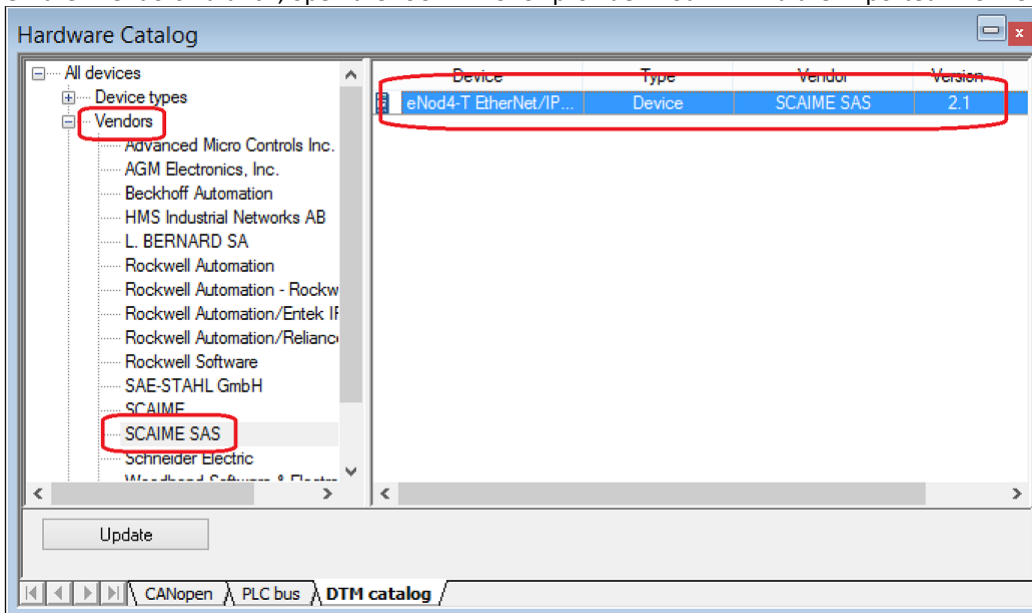
Now the eNod4T is ready to be used on the PLC projects.

3 ADDING ENOD4T IN THE PROJECT

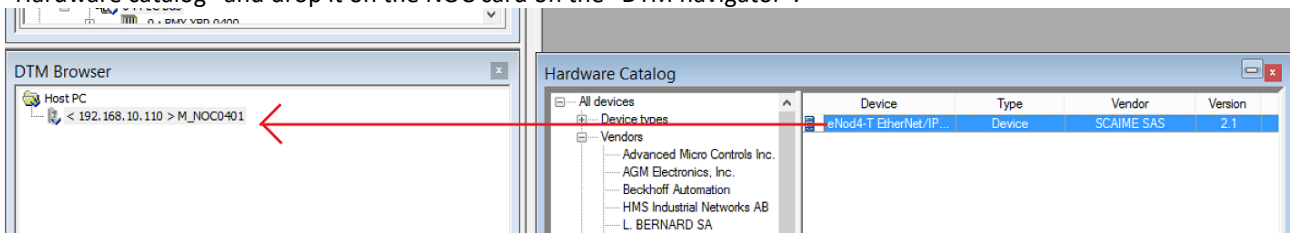
3.1 Adding eNod4T on the Ethernet configuration

The easiest way is to add it from the “Hardware catalog”. On the “Tools” menu select “Hardware catalog”. Then select the “DTM catalog” tab.

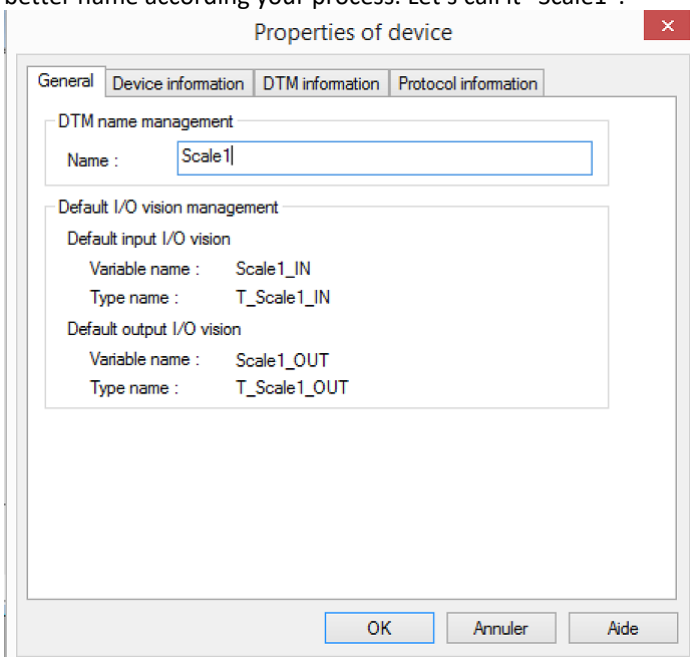
On the “Vendors” branch, open the “SCAIME SAS” provider. You will find the imported EDS file.



To add the eNod to the project, keep the catalog open and open the “DTM navigator”. Then drag the eNod from the “Hardware catalog” and drop it on the NOC card on the “DTM navigator”.



After you drop the eNod, unity ask you for the name of the equipment. You can keep the default name or enter the better name according your process. Let's call it “Scale1”.



Click OK to add the eNod to the project.

The screenshot displays the MikroTik WinBox interface for configuring a switch. The left-hand pane shows a hierarchical tree structure. Under 'Channel Properties', 'Services', and 'Ethernet/IP Local Slaves' are expanded. The 'Device List' is also expanded, showing a device named '[003] Scale1 <EIP: 192.168.10.111>' which is highlighted with a red box. The right-hand pane shows the 'Properties' window for this device, with the 'Address Setting' tab selected and highlighted with a red box. This tab contains fields for 'Change Address', 'Address Server', 'DHCP for this device', 'Identified by', 'Identifier', 'Subnet Mask', and 'Gateway'. The 'IP Address' field is set to '192 . 168 . 10 . 116', the 'Subnet Mask' is '255 . 255 . 255 . 0', and the 'Gateway' is '0 . 0 . 0 . 0'. Other fields like 'DHCP for this device' (Disabled), 'Identified by' (MAC Address), and 'Identifier' (00-00-00-00-00-00) are also visible.

Now generate the project to build all the changes. Unity will create two variables for the implicit input and output data. If you transfer the project to PLC, the input data will be automatically updated.

10
257704-A NU-eNod4-T-FtBk_EIP_Unity_E.docx

3.2 Adding “DFB instructions” in the project.

This part must be done only once by PLC project.

3.2.1 Selection instructions

Various DFB instructions have been developed for eNo4T integration.

TeNod4TBlocMain.

This one is the base one and it is mandatory in all applications.

All other blocs need this one to communicate with the eNod.

TeNod4TBlocBackupRestore

This bloc allows backup and restoring full eNod4T configuration and calibration in PLC memory.

TeNod4TBlocCalibration

This bloc helps you to manage scale calibration in physical or theoretical mode.

TeNod4TBlocFilters

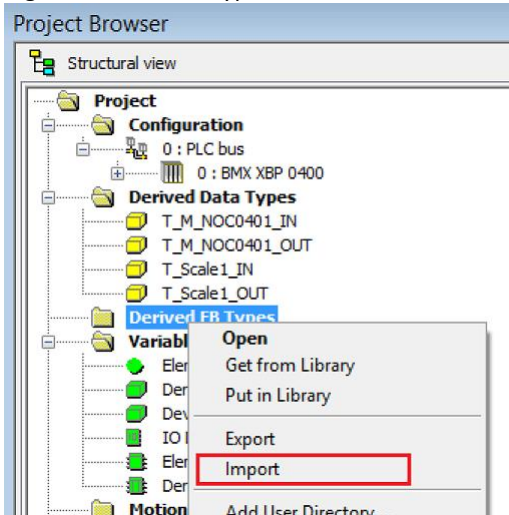
This bloc helps you to adjust eNod filters parameters.

TeNod4TBlocLegal

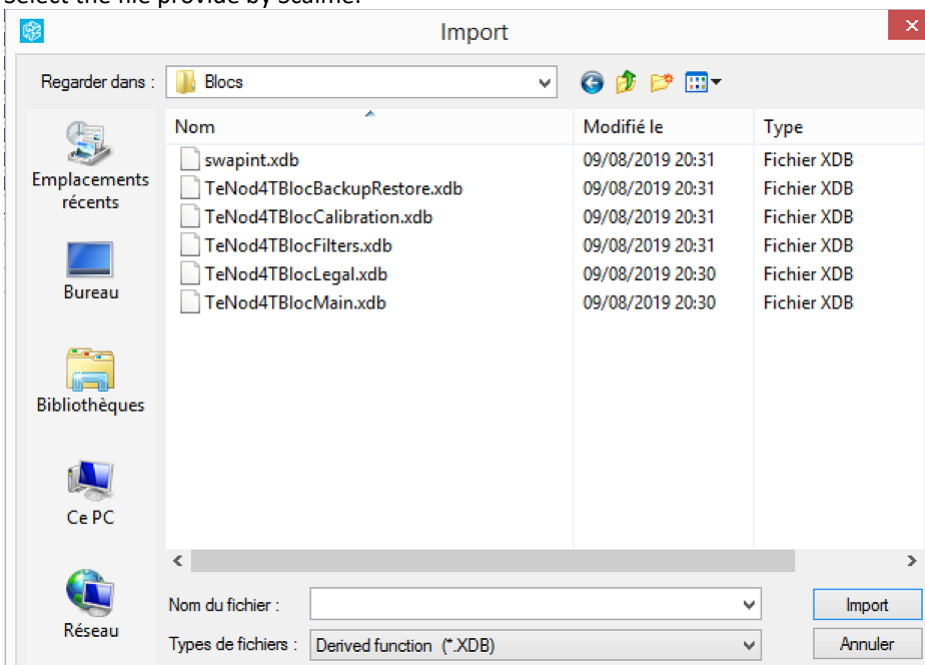
This bloc helps you to use legal functions of eNod4T scale.

3.2.2 Installing Add-on instructions

Right click on “DFB types” section and select “Import”.

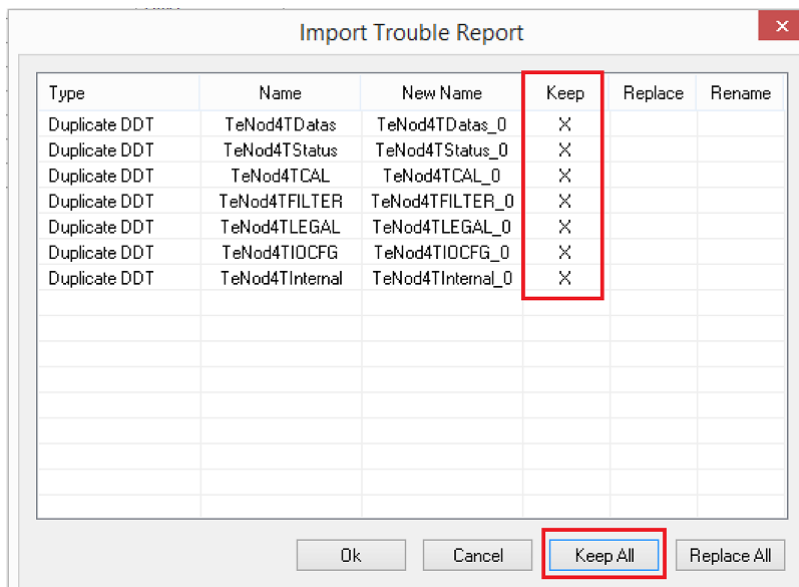


Select the file provide by Scaime.



Important: you must always import first the “main” bloc Add-on. This bloc includes all data types used by all blocs.

For the first import of the “main” bloc, you should not have any conflict warning. For the other blocs, you have the conflict window has some datatype are shared:



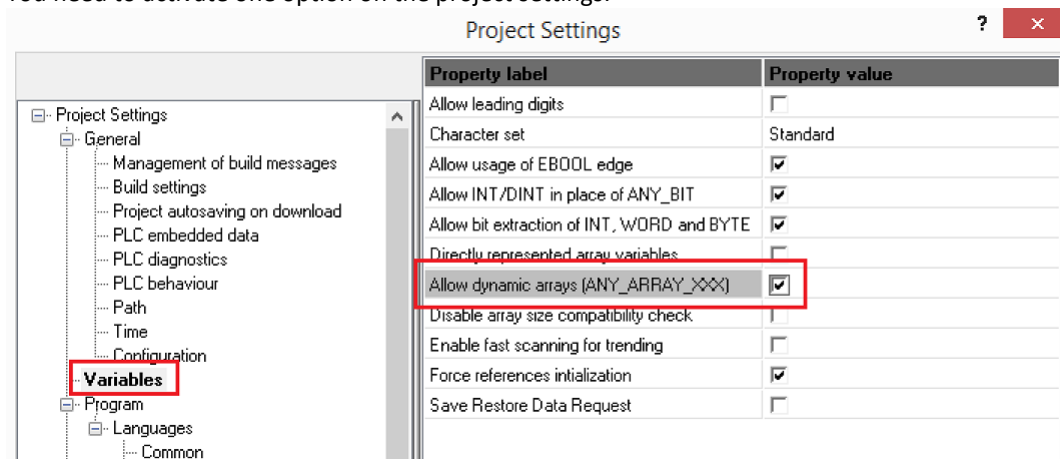
You must click on the “keep all” button to keep the existing ones.
And then confirm the importation of the instruction:

Repeat above instructions for all Add-on instructions you need in your project.

During project compilation you may receive this warning:

```
[CodeCIPComm <DFB> : [TeNod4TBlocMain]] : (l: 180, c: 14) E1208 usage of dynamic arrays is disabled
[CodeCIPComm <DFB> : [TeNod4TBlocMain]] : (l: 182, c: 4) E1208 usage of dynamic arrays is disabled
```

You need to activate one option on the project settings:



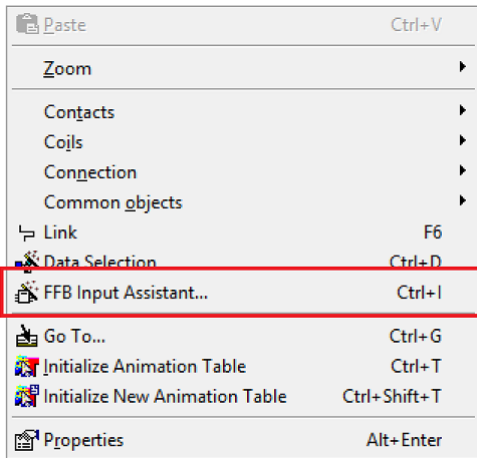
Then you can compile the project.

4 USING ADD-ON INSTRUCTIONS IN THE PROJECT

4.1 Main bloc “TeNod4TBlocMain”

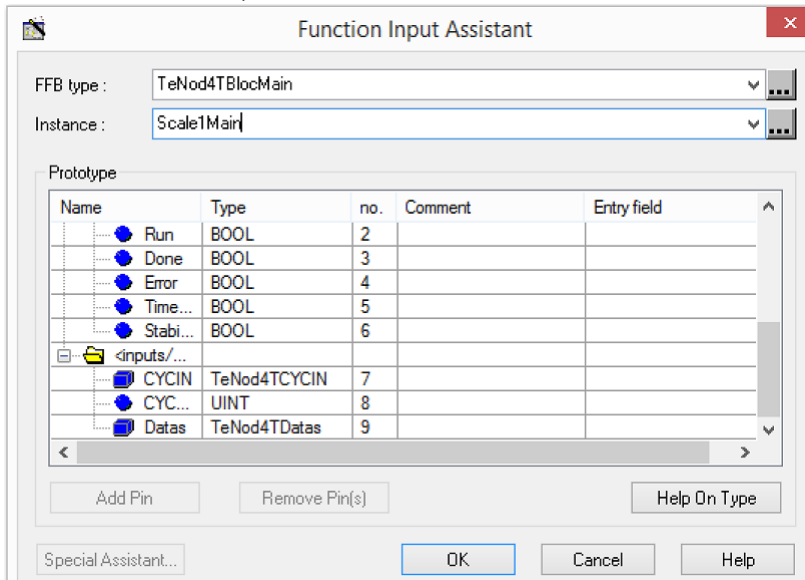
4.1.1 Creating the instance

Somewhere in your program, right click on the grid and select “FFB input assistant”.

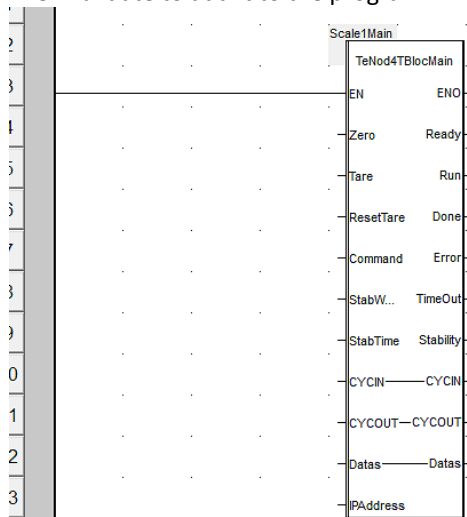


On the “FFB type” field select the TeNod4TBlocMain type.

On the instance name, enter a name for the instance:

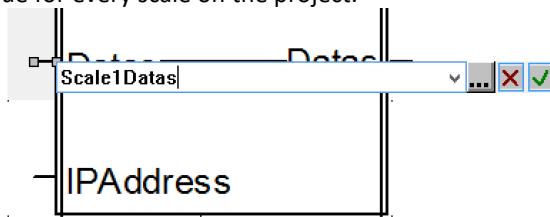


Then validate to add it to the program:

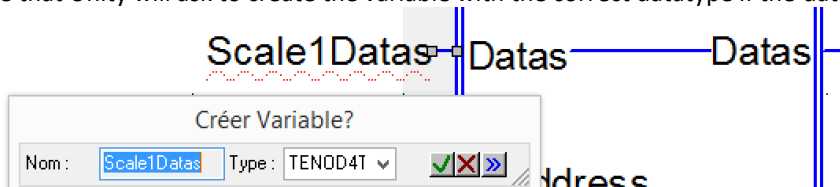


4.1.2 Select global data for the eNod.

This structure will receive all data from the eNod and will be a parameter for all instructions. It must be unique for every scale on the project.



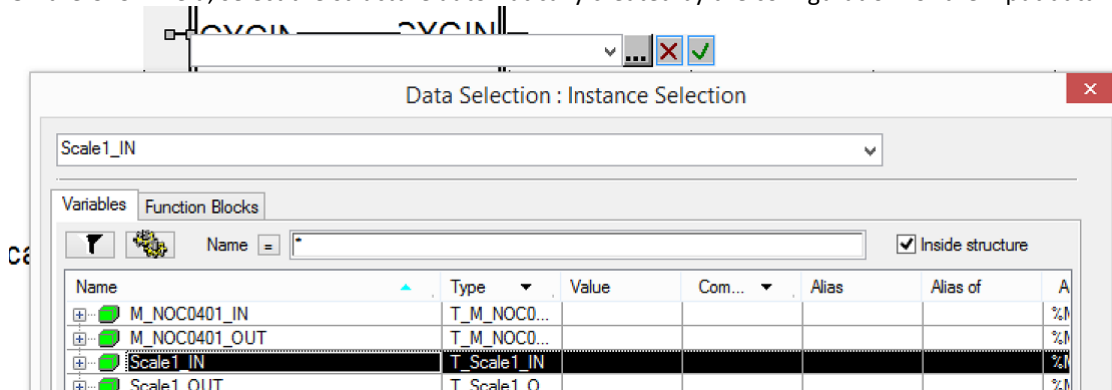
Note that Unity will ask to create the variable with the correct datatype if the data do not exists.



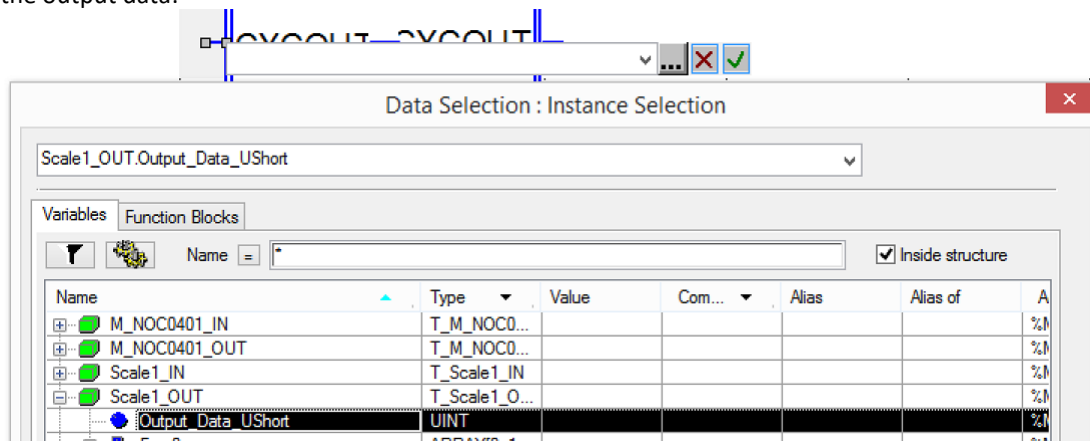
Just confirm the creation of the variable.

4.1.3 Selecting implicit data from hardware configuration

On the CYCIN field, select the structure automatically created by the configuration for the input data:



On the CYCOUT field, select the "Output_Data_UShort" field of the structure automatically created by the configuration for the output data:



4.1.4 Enter the Ethernet/IP link to the eNod.

On the IPAddress field, you must enter the link to the eNod. The address must follow this format '`<rack>.<module>.0{<ipaddress>}UNC.CIP`' where

- Rack: PLC rack where the NOC card is installed
- Module: Module position of the NOC card on the rack
- Ippaddress: IP address of the eNod4t.

See Unity help file for ADDM instruction for more information.

For example, a full IP link is:

'0.1.0{192.168.10.116}UNC.CIP'

IPAddress

4.1.5 Using the instruction

At this moment the configuration and basic program is done.

You can transfer the program to PLC to test communication

4.1.5.1.1 Implicit data.

Implicit input data is decoded on the CYCIN and CYCINStatus fields:

Scale1Ddatas		TeNod4TDatas
CYCIN		TeNod4TCYCIN
RefreshCounter	5876253	DINT
IOStatus	0	UINT
MeasureStatus	16	UINT
Gross	50	DINT
Tare	0	DINT
Net	50	DINT
ADPoints	5935	DINT
CommandEcho	0	UINT
CommandResult	0	UINT
CYCINStatus		TeNod4TStatus
b00Reserved	0	BOOL
b01Reserved	0	BOOL
b02ErrorCode0	0	BOOL
b03ErrorCode1	0	BOOL
b04NoMotion	1	BOOL
b05ZeroWeight	0	BOOL
b06EEPROMFail...	0	BOOL
b07Reserved	0	BOOL
b08Input1ON	0	BOOL
b09Input2ON	0	BOOL
b10Output1ON	0	BOOL
b11Output2ON	0	BOOL
b12Output3ON	0	BOOL
b13Output4ON	0	BOOL
b14TareActive	0	BOOL
b15Reserved	0	BOOL

4.1.5.1.2 eNod configuration information.

By default, the main bloc copies the full configuration from eNod to data structure. This is done only once, when the eNod is online the first time.

Scale1Dats		TeNod4TDats
CYCStatus		TeNod4TStatus
CALDAT		TeNod4TCAL
MeasureRange	4000	DINT
CalibSegment	1	INT
ResAlign1	0	INT
CalibSegment1	200	DINT
CalibSegment2	300	DINT
CalibSegment3	0	DINT
SensorSens	200000	DINT
StepMes	1	INT
ResAlign2	0	INT
CalibZero	-2510	DINT
CalibCoeff1	16.7075	REAL
CalibCoeff2	16.7075	REAL
CalibCoeff3	16.7075	REAL
CalibSpanCorr	1000000	DINT
gValueCalib	9805470	DINT
qValueLocal	9805470	DINT
ZeroOffset	0	DINT
FixedTare	0	DINT
SensorRef	0	DINT
SensorTol	30	INT
FILDAT		TeNod4TFILTER
LEGAL		TeNod4TLEGAL
IOCFG		TeNod4TIOCFG

CALDAT section of the data structure contains all calibration data

FILDAT section of the structure contains all filter configuration data

IOCFG section of the structure contains all I/O configuration data

LEGAL section of the structure contains legal configuration data and legal DSD results.

All structures details can be found on the Scaime documentation “eNod4T Ethernet” chapter 16 “Ethernet/IP register map”. The structures keep the same names and order than the register map objects. Details of every register can be found in the same documentation.

4.1.5.1.3 Bloc parameters and commands

The bloc contains direct command parameters.

Inputs parameters and commands:

“Zero”, execute a zero command on the scale

“Tare”, execute a tare command on the scale

“ResetTare”, suppress an existing tare

“Command”, send a command to the scale. List of available commands are described Scaime documentation.

“StabWeight”, weight tolerance used for weight stability calculation. It is in scale units.

“StabTime”, delay of stability status when weight stays within the weight tolerance. It is in milliseconds units.

Outputs values:

“Ready”, thus status indicates a new command can be done. This includes “Zero”, “Tare”, “ResetTare” and “Command” inputs parameters. If you use other add-on instructions, the “Ready” status can also change when other blocs send commands to the scale. It is recommended to test “Ready” status before sending a command.

“Run”, it is ON when a command is in progress

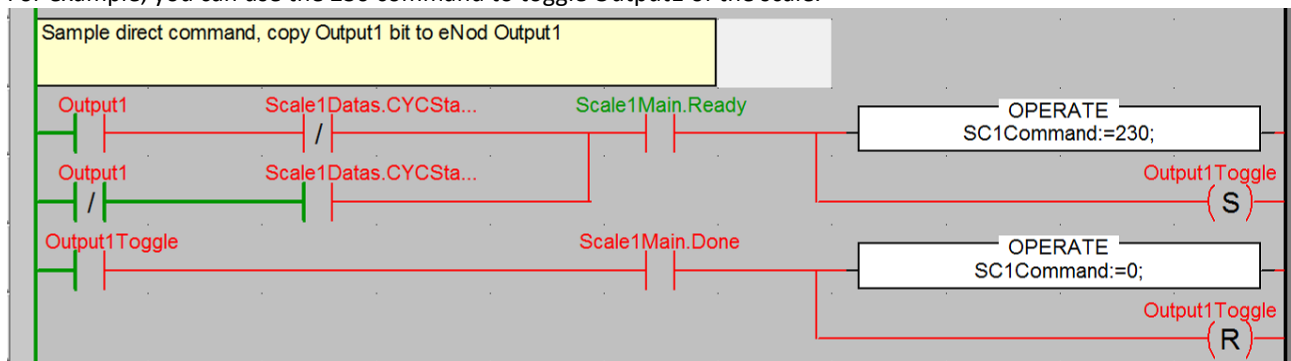
“Done”, it is ON when a command is finished successfully. “Done” stays ON until the command is reset.

“Error”, it is ON when a command in finished with an error or cannot be executed. “Error” stays ON until the command is reset.

“Timeout”, it is ON when there is no implicit communication with the eNod during more than 500ms. Note than some command can result with a “TimeOut” error like “RESET” command.

“Stability”, calculated stability using implicit net weight, “StabWeight” and “StabTime” parameters.

For example, you can use the 230 command to toggle Output1 of the scale.



Not, this will work only if the output 1 is defined as “remote” on the scale configuration.

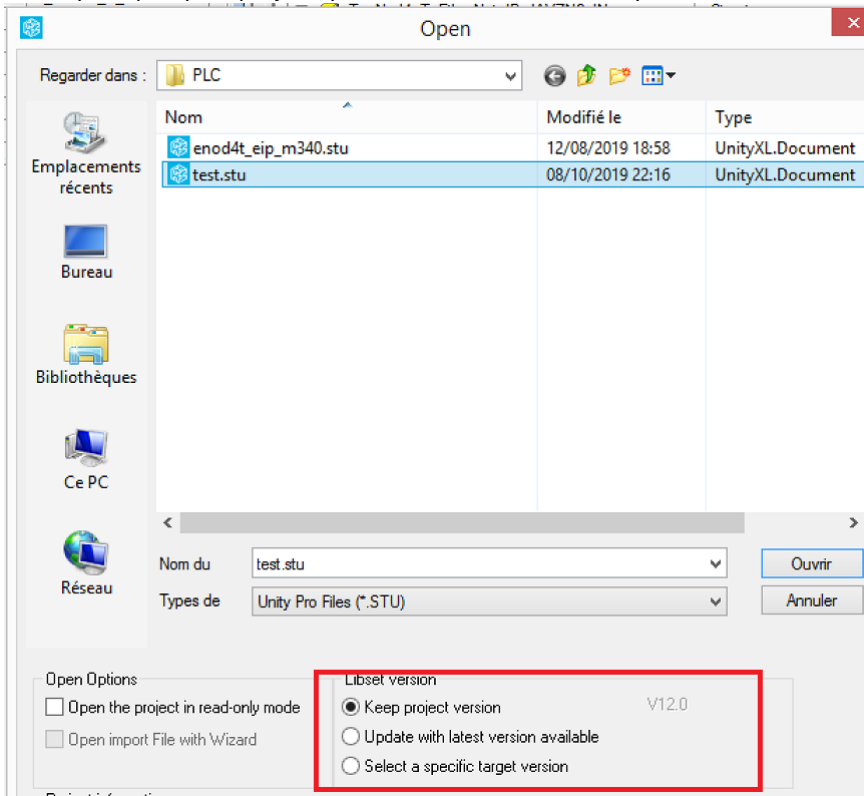
4.1.6 Compatibility with different Unity versions

Library version.

Unity keep the library version sued to create the project even if you modify it with a more recent version.

These blocs are compatible with library version 6.0 or more recent.

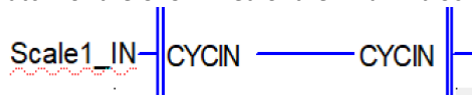
When you open your project, you can see the actual library version used for the project:



EDS importation compatibility.

All Unity versions do not compile the EDS files with the same result.

The blocs are done for Unity 14. But if you compile them with an older version of Unity, you may have an error of type mismatch for the CYCIN filed of the "main" bloc.



To correct this, you need to adapt the datatype T_eNod4TDatasIN imported with the bloc.

This datatype by default is:

T_eNod4TDatasIN
Input_Data_Long
Input_Data_UShort
Input_Data_UShortA
Input_Data_LongB
Input_Data_LongC
Input_Data_LongD
Input_Data_LongE
Input_Data_UShortF
Input_Data_UShortG

You need to compare it to the type created when the scale has been added to the material configuration. The type name is T_XXXXXX_IN where XXXXXX is the name given to the scale.

If the type mismatch you need to update the type T_eNod4TDatasIN to match the one you have.

For example, in Unity 12 the type created with EDS importation is:

T_eNod4_T_EtherNet_IP_VHGQGGQ_IN	<Struct>
Input_Data_Long	DINT
Input_Data_UShort	UINT
Input_Data_UShortH	UINT
Input_Data_LongI	DINT
Input_Data_LongJ	DINT
Input_Data_LongK	DINT
Input_Data_LongL	DINT
Input_Data_UShortM	UINT
Input_Data_UShortN	UINT

To be able to compile the blocs, the T_eNod4TDatasIN must be changed to:

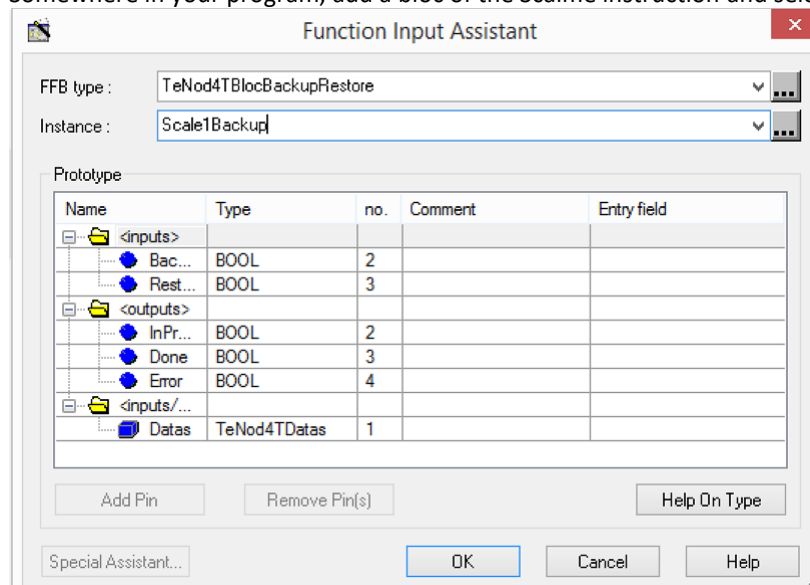
T_eNod4TDatasIN	<Struct>
Input_Data_Long	DINT
Input_Data_UShort	UINT
Input_Data_UShortH	UINT
Input_Data_LongI	DINT
Input_Data_LongJ	DINT
Input_Data_LongK	DINT
Input_Data_LongL	DINT
Input_Data_UShortM	UINT
Input_Data_UShortN	UINT

4.2 “TeNod4TBlocBackupRestore” to restore and backup configuration

This bloc allows backup and restoring full configuration and calibration data of the scale.

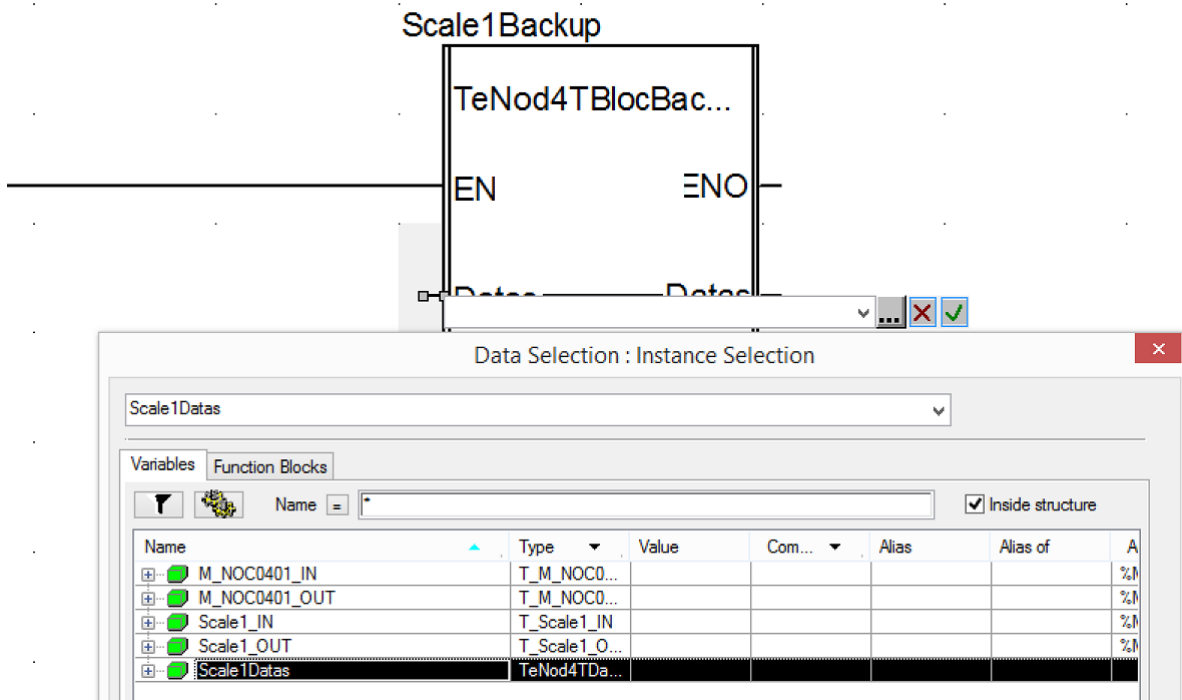
4.2.1 Creating the instance

Somewhere in your program, add a bloc of the Scaime instruction and select an instance name:



4.2.2 Select global data for the eNod.

This structure must be the one created for the main bloc.



4.2.3 Using the instruction

4.2.3.1.1 Bloc parameters and commands

The bloc contains direct command parameters.

Inputs parameters and commands:

“Backup”, execute a copy of all parameters from the scale to the PLC

“Restore”, execute a copy of all parameters from the PLC to the scale.

Outputs values:

“InProgress”, it is ON when a command is in progress

“Done”, it is ON when a command is finished successfully. “Done” stays ON until the command is reset.

“Error”, it is ON when a command in finished with an error or cannot be executed. “Error” stays ON until the command is reset.

4.2.3.1.2 Backup of the configuration

To copy the information, just set to 1 the input “Backup” of the bloc. This will copy the whole part of the structure members

CAL for calibration data

FIL for filter data

IOCFG for input/output configuration data

LEGAL for legal configuration and data

Note that all existing data on the structure will be erased. The backup also read the current DSD values of the legal structure.

Backup can be done while the scale is in use.

4.2.3.1.3 Restoring the configuration

To restore the configuration, just set to 1 the input “Restore” of the bloc. This will send to the scale all configuration data:

CAL for calibration data. The last saved calibration will be sent, so the scale will be calibrated with the last calibration data after the restore

FIL for filter data

IOCFG for input/output configuration data

LEGAL, only the legal configuration fields will be restored: ZeroConfig, Stability, Decimal, Unit and HMName.

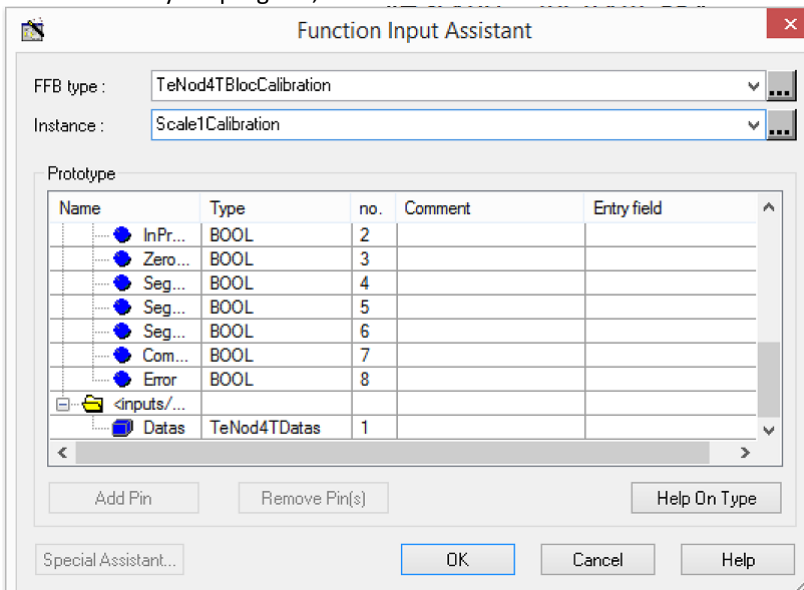
After the restore, an EEPROM save command is sent to save the data. Then a RESET command is sent to restart the eNod with the restored configuration. During this moment the communication will be temporally lost.

4.3 “TeNod4TBlocCalibration” to calibrate a scale

This bloc allows to calibrate the scale.

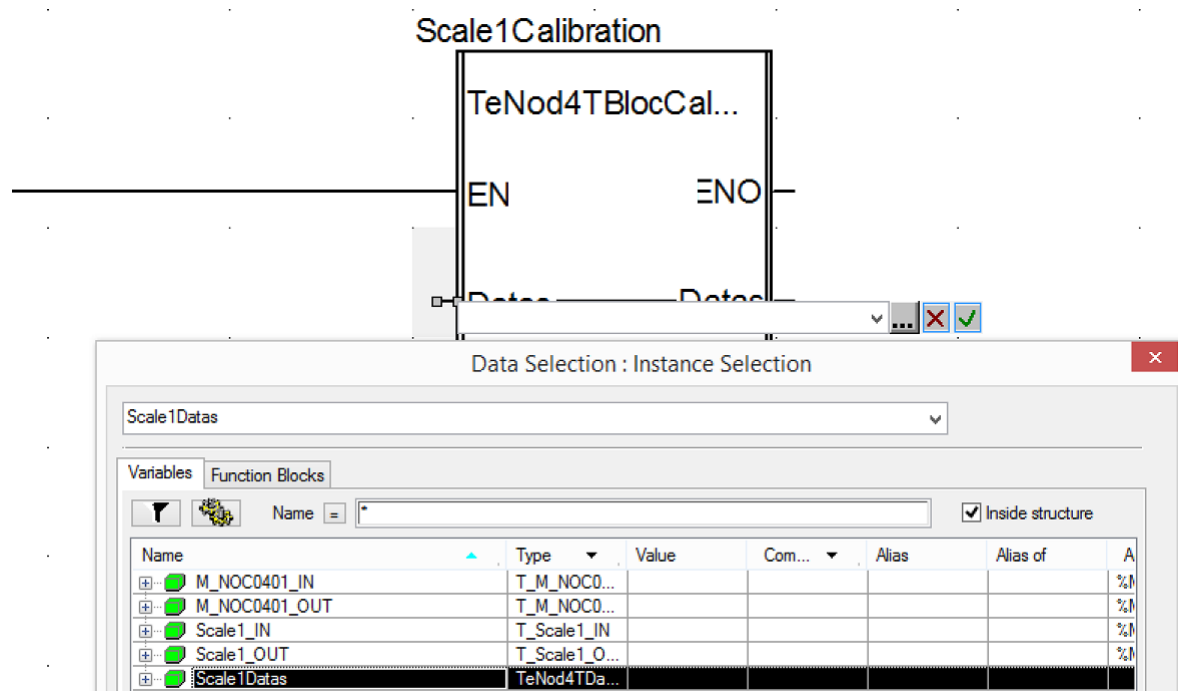
4.3.1 Creating the instance

Somewhere in your program, add a bloc of the Scaime instruction and select an instance name:



4.3.2 Select global data for the eNod.

This structure must be the one created for the main bloc.



4.3.3 Using the instruction

4.3.3.1.1 Bloc parameters and commands

The bloc contains direct command parameters.

Inputs parameters and commands:

"NumSegment", number of calibration segment for calibration

"ValSegment1", weight value in scale unit of the first calibration segment

"ValSegment2", weight value in scale unit of the second calibration segment

"ValSegment3", weight value in scale unit of the third calibration segment

"Start", start of a physical calibration procedure.

"Zero", execute to zero point of the calibration procedure

"Segment1", execute the first point of the calibration procedure

"Segment2", execute the second point of the calibration procedure

"Segment3", execute the third point of the calibration procedure

"Save", save the calibration procedure. This will save calibration point to EEPROM and copy the calibration data to PLC memory.

"Cancel", cancel the physical calibration in progress.

"Theoretical", execute a theoretical calibration.

"ZeroAdjust", execute a zero adjustment command.

Outputs values:

"InProgress", it is ON when a command is in progress. InProgress stays ON until the "Save" command is executed indicating a calibration is in progress.

"ZeroDone", it is ON when the "zero" is finished successfully. "ZeroDone" stays ON until the command is reset.

"Seg1Done", it is ON when the "Seg1Done" is finished successfully. "Seg1Done" stays ON until the command is reset.

"Seg2Done", it is ON when the "Seg2Done" is finished successfully. "Seg2Done" stays ON until the command is reset.

"Seg3Done", it is ON when the "Seg3Done" is finished successfully. "Seg3Done" stays ON until the command is reset.

"Complete", it is ON when the "Save" is finished successfully. "Complete" stays ON until the command is reset. When "Complete" is ON, "InProgress" is reset.

"Error", it is ON when a command in finished with an error or cannot be executed. "Error" stays ON until the command is reset.

Calibration must respect a command sequence; valid command sequences are:

Start, Zero, Segment1, Save

Start, Zero, Segment1, Segment2, Save

Start, Zero, Segment1, Segment2, Segment3, Save

Start, Segment1, Save

ZeroAdjust, Save

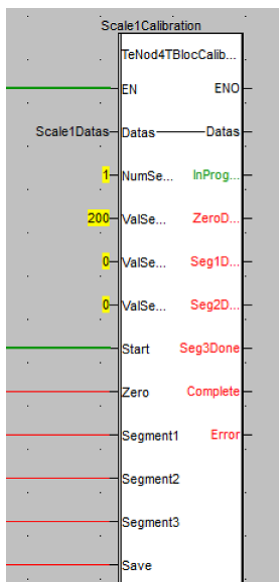
Theoretical, Save

Full physical calibration

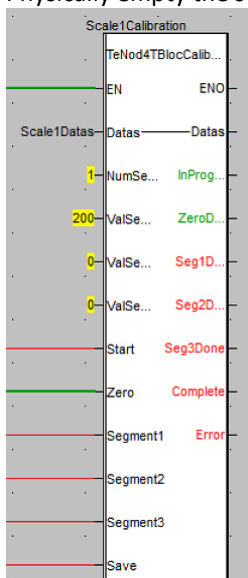
Calibration can be done using an HMI program that uses the input parameters of the bloc. You can also do it directly using Logix Studio by writing bloc parameters.

Before starting procedure, the inputs parameters NumSegment and ValSegment1 must be set. ValSegment2 must be set if NumSegment is 2 or 3. ValSegment3 must be set if NumSegment is 3.

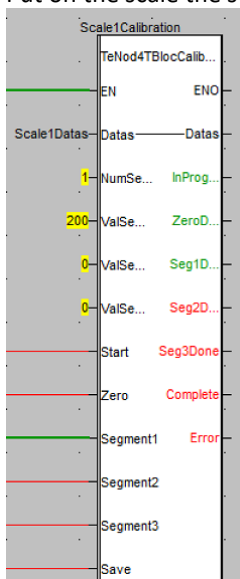
Start the procedure with "Start" command.



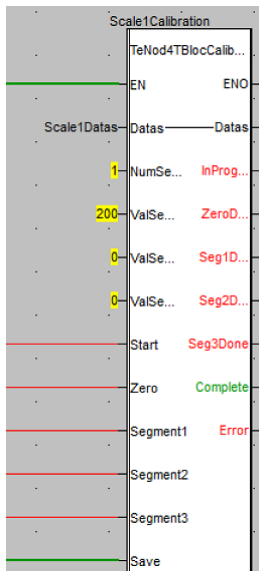
Physically empty the scale and send the “Zero” command



Put on the scale the standard load you need according the “ValSegment1” value and then send “Segment1” command



In case you have a 2 or 3 segments calibration, use Segment2 and Segment3 commands.
When it is finished, use save command

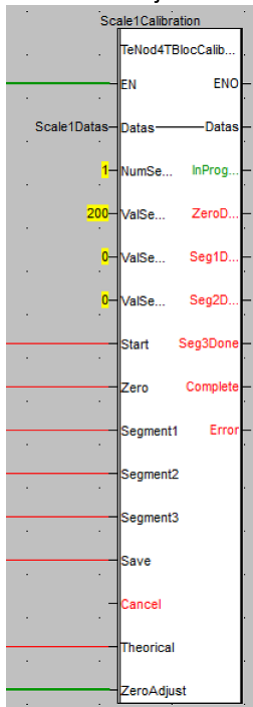


Note that ValSegmentX can be modified just before the corresponding command SegmentX is sent.

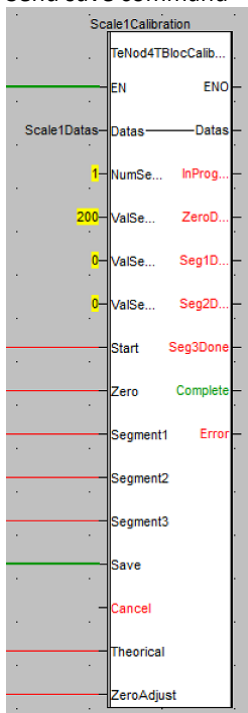
4.3.3.1.2 Zero adjust calibration

Calibration can be done using an HMI program that uses the input parameters of the bloc. You can also do it directly using Logix Studio by writing bloc parameters.

Send “ZeroAdjust” command



Send save command

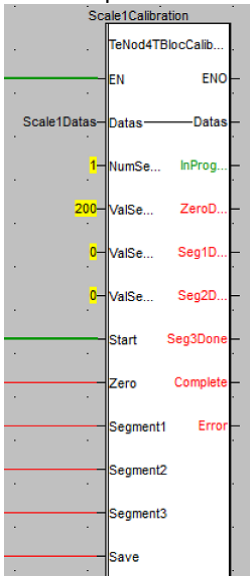


4.3.3.1.3 Span adjust calibration

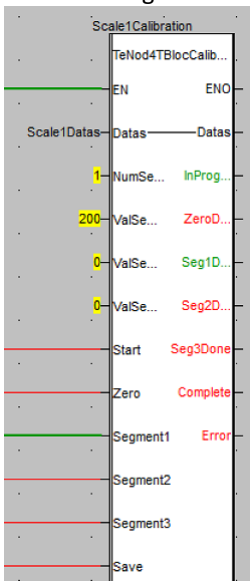
Calibration can be done using an HMI program that uses the input parameters of the bloc. You can also do it directly using Logix Studio by writing bloc parameters.

For this procedure, NumSegment must be set to 1 even if the original calibration procedure is 2 or 3 segments.

Start the procedure with “Start” command.



Send the “Segment1” command



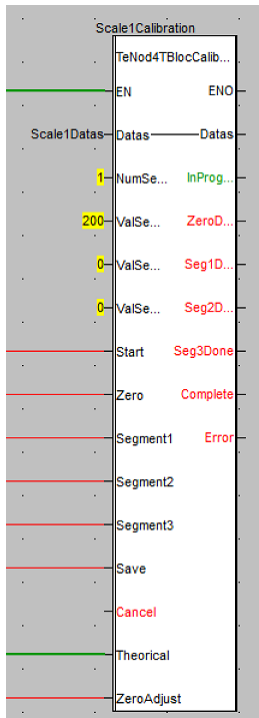
Send the “Save” command

Scale1Calibration	
TeNod4TBlocCalib...	
EN	ENO
Scale1Datas	Datas
1	NumSe... InProg...
200	ValSe... ZeroD...
0	ValSe... Seg1D...
0	ValSe... Seg2D...
Start	Seg3Done
Zero	Complete
Segment1	Error
Segment2	
Segment3	
Save	

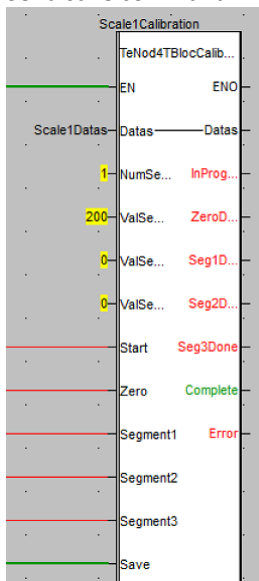
4.3.3.1.4 Theoretical calibration

Calibration can be done using an HMI program that uses the input parameters of the bloc. You can also do it directly using Logix Studio by writing bloc parameters.

Send “Theoretical” command



Send save command



4.3.3.1.5 Cancel calibration

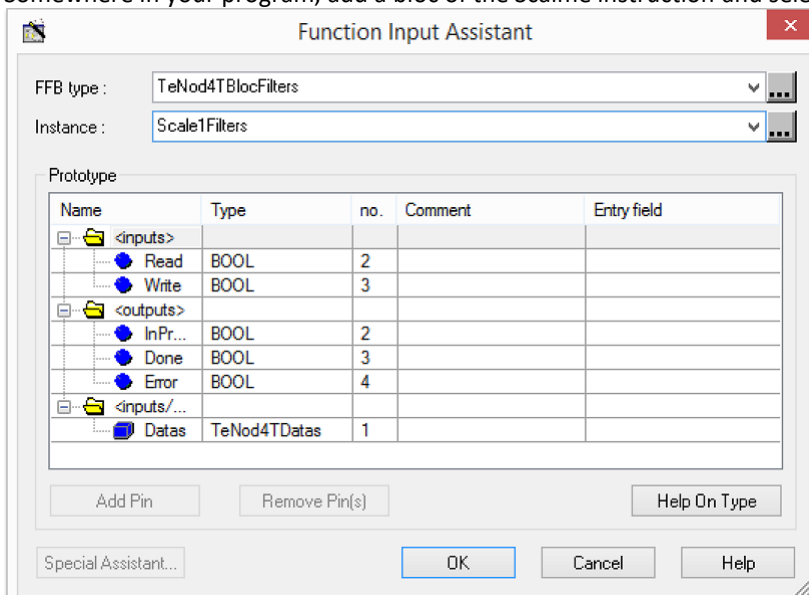
Calibration can be cancelled anytime until the save command is sent. “Cancel” command will reset the eNod and calibration will return to the last calibration values.

4.4 “TeNod4TBlocFilters” to read and write filter configuration

This bloc allows to read and write filter configuration of the scale.

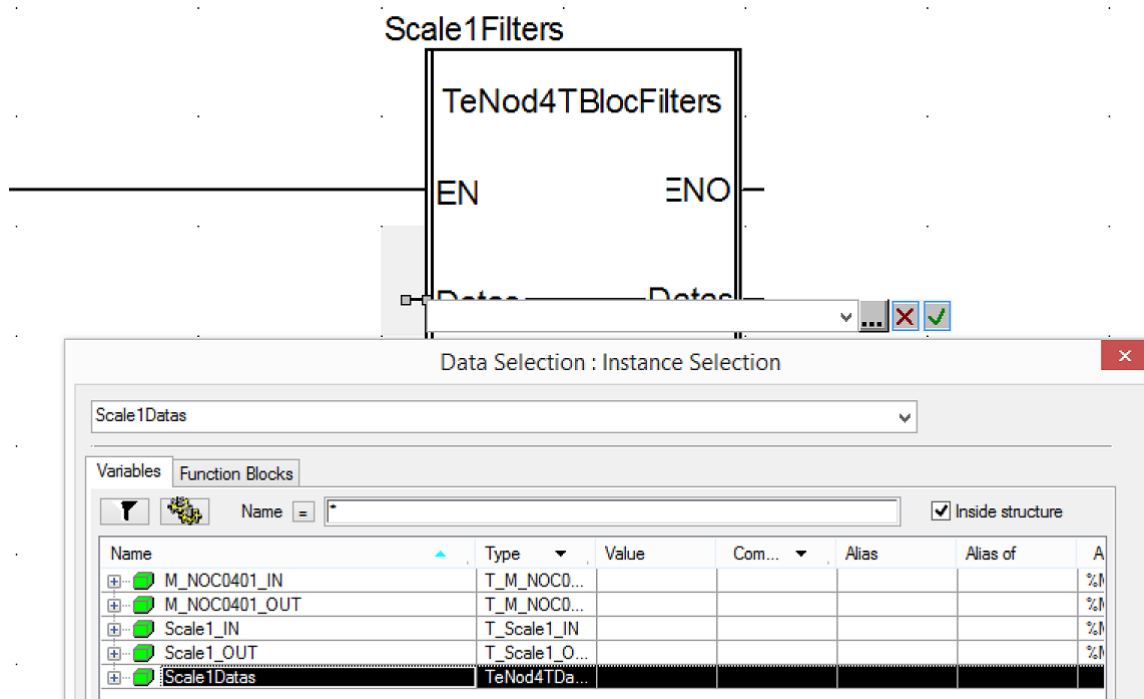
4.4.1 Creating the instance

Somewhere in your program, add a bloc of the Scaime instruction and select an instance name:



4.4.2 Select global data for the eNod.

This structure must be the one created for the main bloc.



4.4.3 Using the instruction

Bloc parameters and commands

The bloc contains direct command parameters.

Inputs parameters and commands:

“Read”, execute a copy of all parameters from the scale to the PLC

“Write”, execute a copy of all parameters from the PLC to the scale.

Outputs values:

“InProgress”, it is ON when a command is in progress

“Done”, it is ON when a command is finished successfully. “Done” stays ON until the command is reset.

“Error”, it is ON when a command is finished with an error or cannot be executed. “Error” stays ON until the command is reset.

This bloc uses the FIL section of the data structure.

4.4.3.1.1 Read the filter configuration

To copy the information, just set to 1 the input “Read” of the bloc. This will copy the filter configuration to FIL member of the.

4.4.3.1.2 Set filter configuration

First you need to set the desired filter configuration on the FIL section of the eNod data structure.

Then, you set the “Write” command of the bloc.

New configuration is directly is use. The command also saves the new configuration to EEPROM.

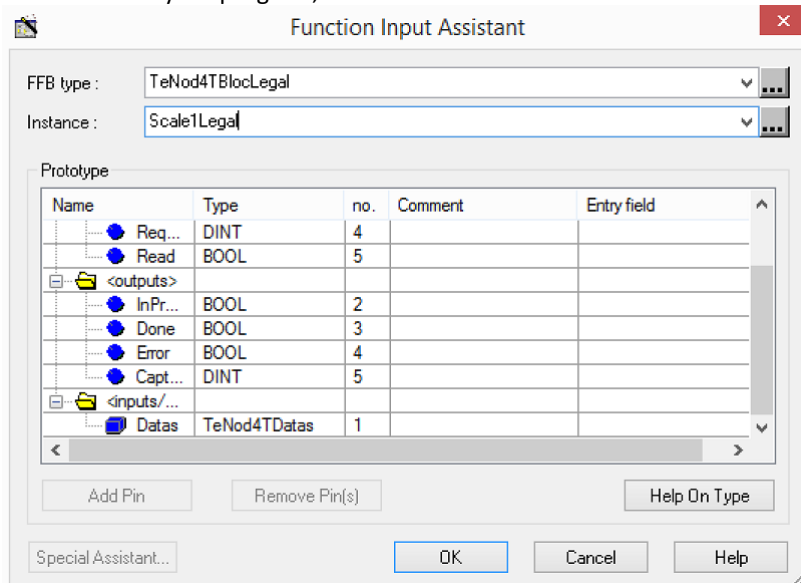
Note that if the A/D parameters values are changed, the changed will be used on the next reboot or reset of the eNod

4.5 “TeNod4TBlocLegal” to manage DSD legal values

This bloc allows to create DSD values and read them

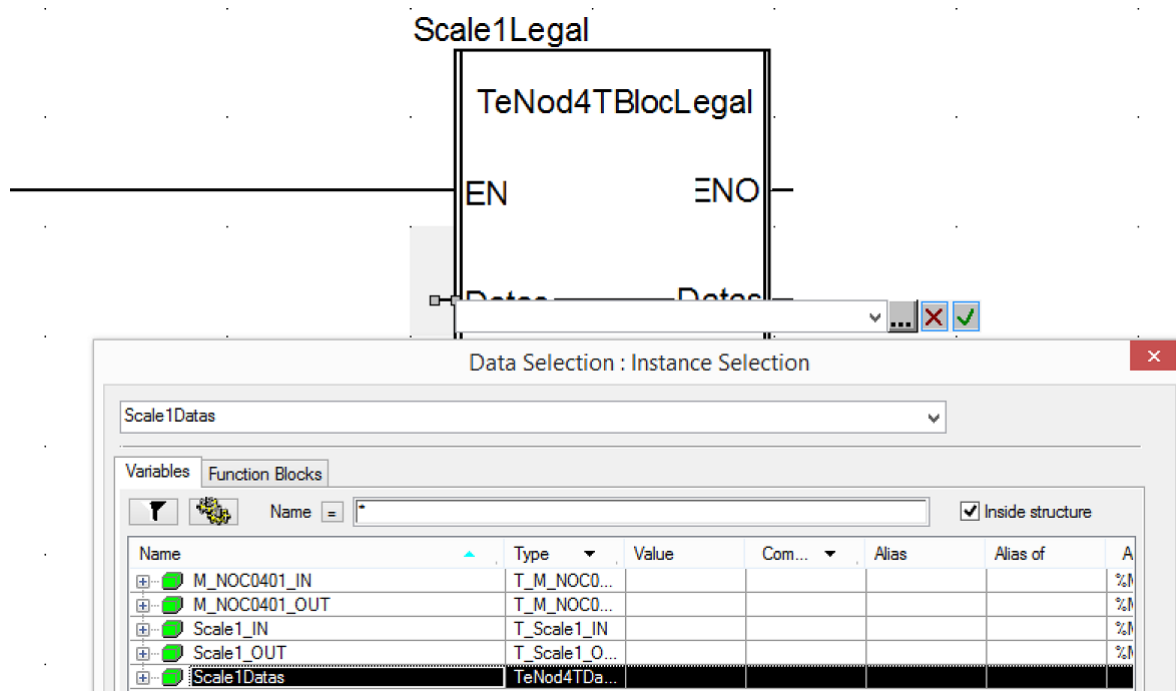
4.5.1 Creating the instance

Somewhere in your program, add a bloc of the Scaime instruction and select an instance name:



4.5.2 Select global data for the eNod.

This structure must be the one created for the main bloc.



4.5.3 Using the instruction

4.5.3.1.1 Bloc parameters and commands

The bloc contains direct command parameters.

Inputs parameters and commands:

“CaptureRead”, create a DSD measure and copy it to PLC memory.

“CaptureOnly”, create a DSD measure but to not read it on the scale. Measure can be read later. “CaptureOnly” is faster than “CaptureRead” command and does not erase the DSDRead data.

“RequestedID”, parameter of the “Read” command

“Read”, reads “RequestedID” from scale memory and copies it to PLC memory.

Outputs values:

“InProgress”, it is ON when a command is in progress

“Done”, it is ON when a command is finished successfully. “Done” stays ON until the command is reset.

“Error”, it is ON when a command is finished with an error or cannot be executed. “Error” stays ON until the command is reset.

“CapturedID”, the last ID generated by “CaptureRead” or “CaptureOnly”.

The “CaptureRead” or the “Read” commands read the scale memory and copy the DSD values on the LEGAL members of the data structure:

Scale1DData		TeNod4TDData
CYCStatus		TeNod4TStatus
CALDAT		TeNod4TCAL
FILDAT		TeNod4TFILTER
LEGAL		TeNod4TLEGAL
eNodVersion	24787	INT
ModbusCfg	1793	INT
LegalVersion	2	BYTE
LegalSwitch	0	BYTE
LegalCount	0	BYTE
ResAlign1	0	BYTE
LegalChecksum	0	INT
ZeroConfig	0	INT
Stability	2	BYTE
Decimal	3	BYTE
UNIT	'kg'	string[4]
ResAlign2	0	BYTE
ResAlign3	0	BYTE
DSDLastID	22	DINT
DSDOldestID	1	DINT
DSDCurrentID	19	DINT
DSDReadID	19	DINT
DSDReadNet	262	DINT
DSDReadTare	0	DINT
DSDReadStatus	7298	INT
DSDReadCheckSum	-7579	INT
HMName	'EFGH'	string[4]

4.5.3.1.2 Capture a DSD value

Set the command “CaptureRead” or “CaptureOnly” of the bloc. When the command is done, the generated ID is displayed on the bloc:

5 COMMUNICATION ERROR CODES

In case a command returns an error, it can be for various reasons.

For direct command, refer to Scaime documentation for command conditions.

In some cases, “main” bloc cannot read or write data on the scale. This is mainly due to data format. As a lot of registers can be read or written, the data structure contains the current read or written register.

If a communication error occurs the bit “**Datas.INTER.MSGError**” is set.

In this case check the “**Datas.MSGGroup**” data field. If value is from 1 to 99, it is a read command. If value is from 101 to 199 it is a write command.

Then check the following fields:

Datas.MSGClass: It is the CIP class name of the register

Datas.MSGAttribute: It is the CIP attribute value of the register

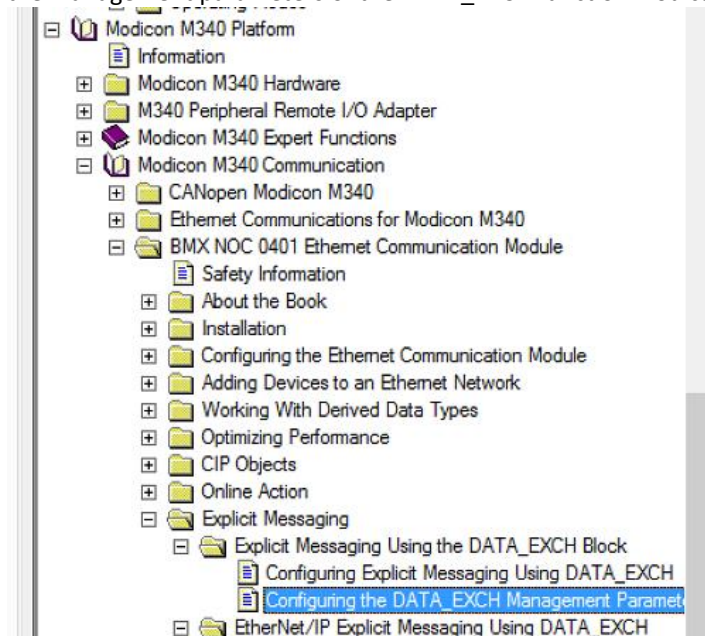
With these two values, refer to chapter 16 to get the register name.

For example, class 105 attribute 2 is the number of segments in calibration procedure.

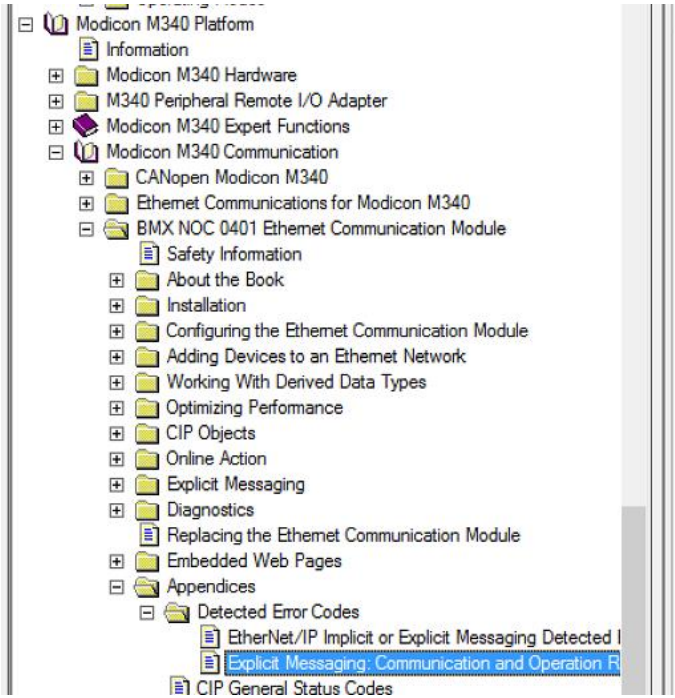
There are two levels of error codes stored in MSGError and CIPError.

The first level (MSGError) is an error returned by the DATA_EXCH function.

If this field is not null, the PLC cannot establish the communication with the eNod. The error code is the second word of the management parameters of the DATA_EXCH function. You can find the error in the Unity help file:



If the first level is null, the communication is established but the CIP message failed. The error code is stored in CIPError field. The CIP errors codes are from the ODVA standard. You can find the error code list in Unity help file:



6 BASIC COMMUNICATION

If you do not need any calibration or parameters changes of the eNod by the PLC, you can use this basic procedure that only read implicit data.

First you need to configure the scale on the Ethernet network as describe on §2-A.

All data is present on the cyclic structure automatically created by Unity.

Scale1_IN		T_Scale1_IN
Input_Data_Long	728203	DINT
Input_Data_UShort	0	UINT
Input_Data_UShortH	16	UINT
Input_Data_LongI	50	DINT
Input_Data_LongJ	0	DINT
Input_Data_LongK	50	DINT
Input_Data_LongL	5954	DINT
Input_Data_UShortM	0	UINT
Input_Data_UShortN	0	UINT

Structure member name signification is:

Input_Data_Long	Refresh counter
Input_Data_UShort	Digital input and output status
Input_Data_UShortH	Measure status
Input_Data_LongI	Gross weight
Input_Data_LongJ	Tare
Input_Data_LongK	Net weight
Input_Data_LongL	Filtered ADC points
Input_Data_UShortM	eNod command echo
Input_Data_UShortN	eNod command result