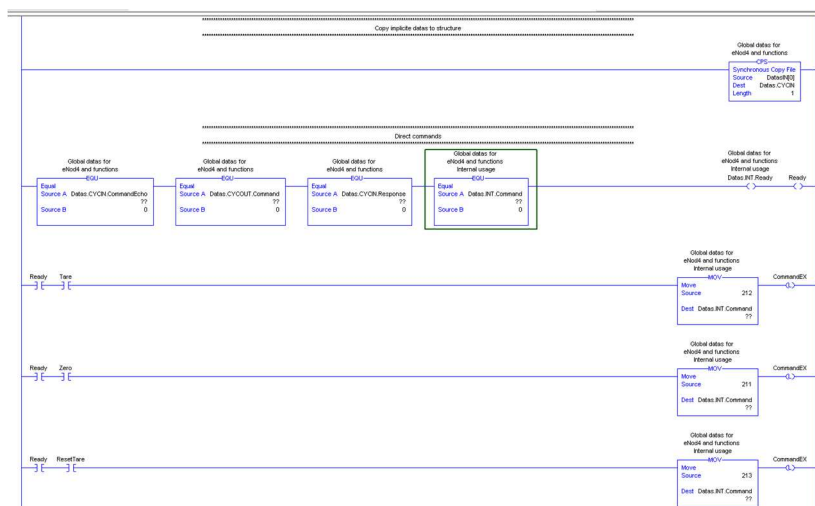




Function Block for eNod4-T EIP

For Logix Designer IDE, Rockwell Automation



1 GENERAL PRESENTATION	3
2 INSTALLING EDS FILE IN LOGIX DESIGNER	4
3 ADDING ENOD4 IN THE PROJECT	7
3.1 Adding eNod4 on the Ethernet configuration	7
3.2 Adding “ADD-ON instruction” in the project	9
3.2.1 Select instructions	9
3.2.2 Installing Add-on instructions.....	9
4 USING ADD-ON INSTRUCTIONS IN THE PROJECT.....	11
4.1 Main bloc “TeNod4TBlocMain”	11
4.1.1 Creating the instance	11
4.1.2 Select global data for the eNod4.....	11
4.1.3 Selecting implicit data from hardware configuration	12
4.1.4 Creating MSG structure for read operations	12
4.1.5 Creating MSG structure for write operations.	14
4.1.6 Using the instruction	16
4.2 “TeNod4TBlocBackupRestore” to restore and backup configuration	18
4.2.1 Creating the instance	18
4.2.2 Select global data for the eNod4.....	19
4.2.3 Using the instruction	19
4.3 “TeNod4TBlocCalibration” to calibrate a scale	20
4.3.1 Creating the instance	20
4.3.2 Select global data for the eNod4.....	21
4.3.3 Using the instruction	21
4.4 “TeNod4TBlocFilters” to read and write filter configuration.....	24
4.4.1 Creating the instance	24
4.4.2 Select global data for the eNod4.....	25
4.4.3 Using the instruction	25
4.5 “TeNod4TBlocLegal” to manage DSD legal values.....	26
4.5.1 Creating the instance	26
4.5.2 Select global data for the eNod4.....	26
4.5.3 Using the instruction	27
5 COMMUNICATION ERROR CODES	29
6 BASIC COMMUNICATION.....	30

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 Logix designer is an open source ladder code that eases the integration of an eNod4 with Rockwell/Allen Bradley PLC. This tool integrates pre-programmed ladder functions for Logix designer. 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 Logix designer and communicate with an eNod4-T Ethernet/IP through a Rockwell/Allen Bradley PLC.

2 INSTALLING EDS FILE IN LOGIX DESIGNER

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

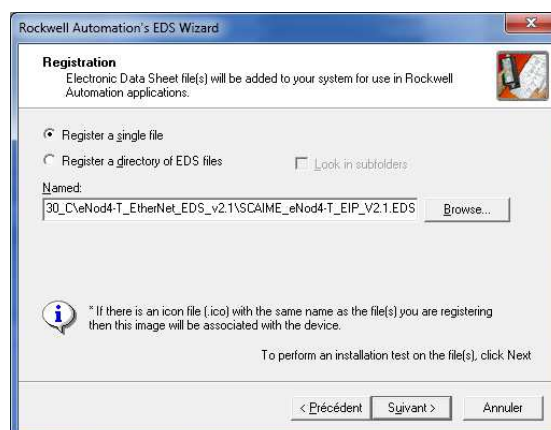
In “Tool” menu select “EDS hardware installation tool”.



Click “next” to continue.



Select “Register an EDS file”



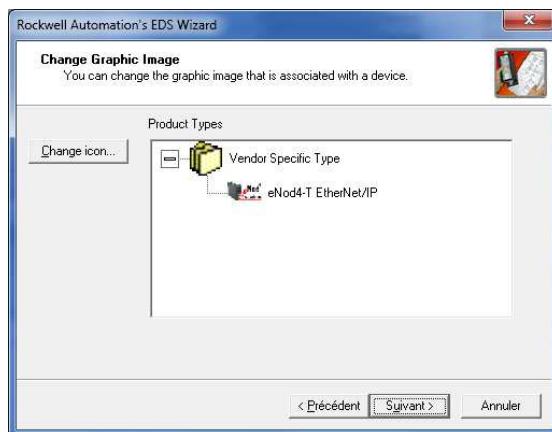
Select “Register a single file”.

Use “Browse” button to select the EDS file to be imported.

For new configurations select the eNod4 EDS file (you can download it on Scaime Website).



Click “Next” to start importation.



On this step you can change the icon for eNod4. Just click “next” to continue.



Click “next” to continue.

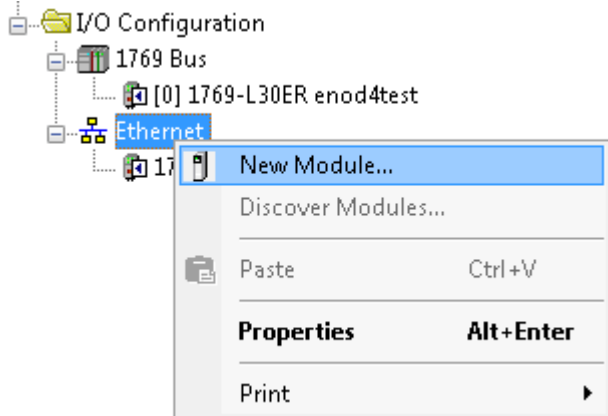


Click “close” to finish the procedure.

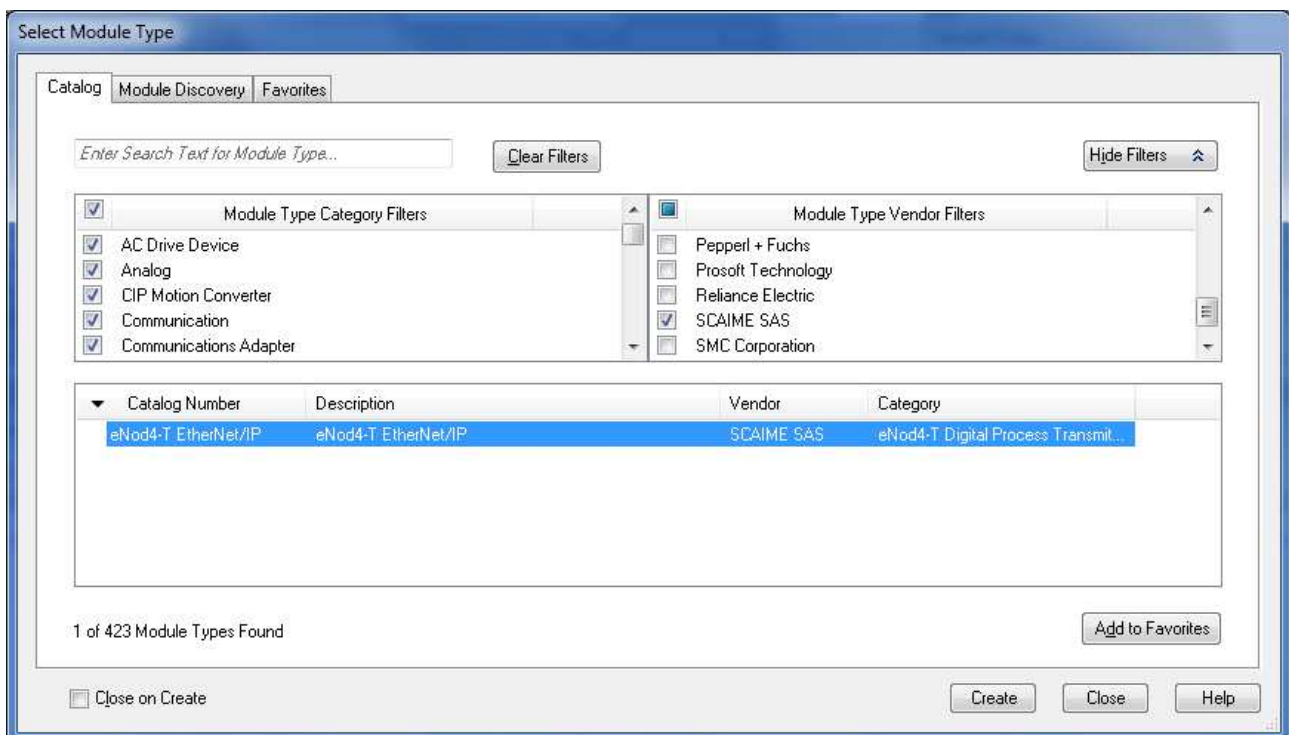
3 ADDING ENOD4 IN THE PROJECT

3.1 Adding eNod4 on the Ethernet configuration

Right click on Ethernet network on the hardware configuration. Then select “New module”.

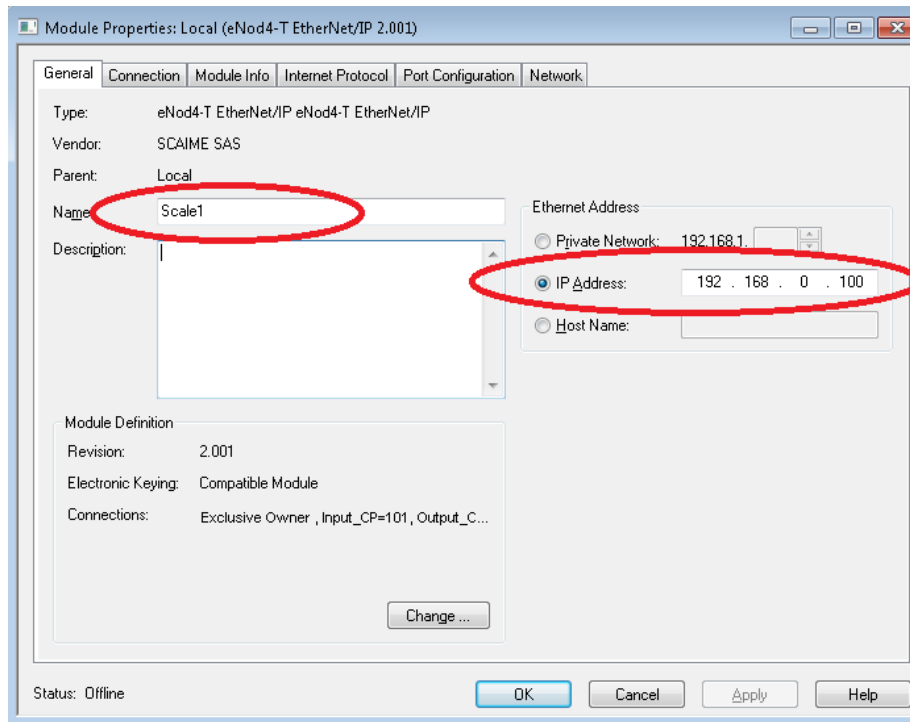


Select “Scaime SAS” group on the vendor list, and select eNod4-T from the list. Click on “create”.

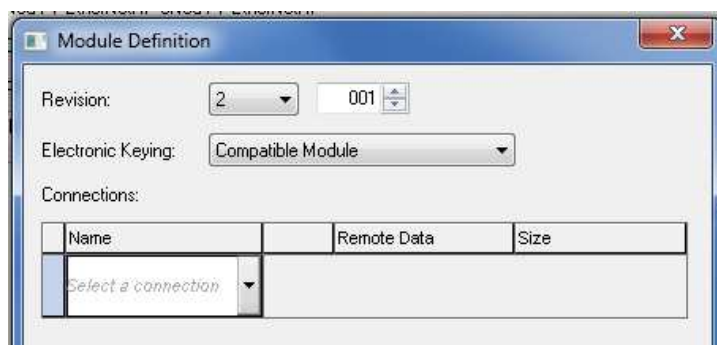


On the node configuration window, enter the following data:

- “name”: name of the eNod4 on the Ethernet configuration. This name will be referenced later during message’s definition.
- “IP address”: IP address of the eNod4

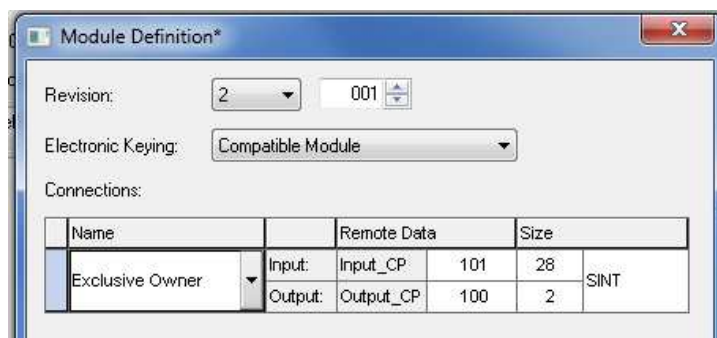


Before closing this Window, click on “Change” button of the module definition.

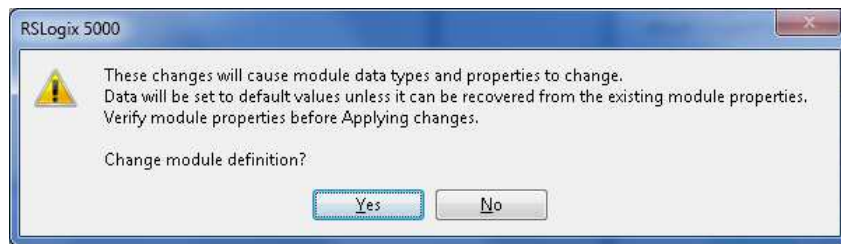


Select the “exclusive Owner” connection type.

Important: do not change data size or data type if you plan to use Add-On instructions delivered by Scaime.



Click OK to confirm the connection. You receive a warning at this point that is normal and confirm by clicking on “yes”.



Click OK to complete the hardware configuration.

At this moment the connection is effective and implicit data exchange is in place.

By default, data IN and OUT are stored in predefined byte arrays.

eNod1Eth:I	{...}	{...}		_054A:eNod4_TEtherNetIP_3740F51B:I:0
eNod1Eth:I.ConnectionFaulted	0		Decimal	BOOL
eNod1Eth:I.RunMode	0		Decimal	BOOL
+ eNod1Eth:I.Data	{...}	{...}	Decimal	SINT[28]
eNod1Eth:O	{...}	{...}		_054A:eNod4_TEtherNetIP_7CE0A29D...
+ eNod1Eth:O.Data	{...}	{...}	Decimal	SINT[2]

To get more “user friendly” structures, follow the next instructions.

3.2 Adding “ADD-ON instruction” in the project

This part must be done only once by PLC project.

3.2.1 Select instructions

Various add-on instructions have been developed for eNod4 integration.

TeNod4TBlocMain.

This one is the basic one and is mandatory in all applications.

All other blocs need this one to communicate with the eNod4.

TeNod4TBlocBackupRestore

This bloc allows backup and restoring full eNod4 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 eNod4 filters parameters.

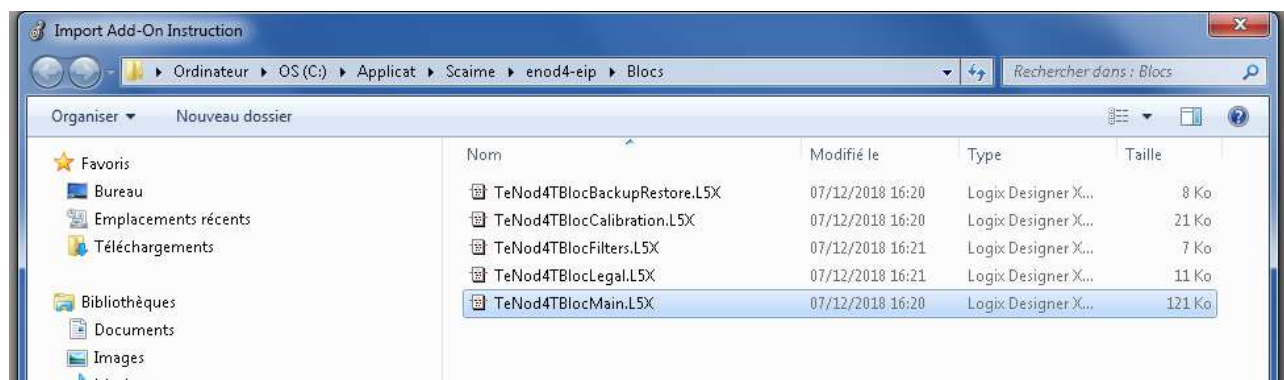
TeNod4TBlocLegal

This bloc helps you to use legal functions of eNod4 scale.

3.2.2 Installing Add-on instructions

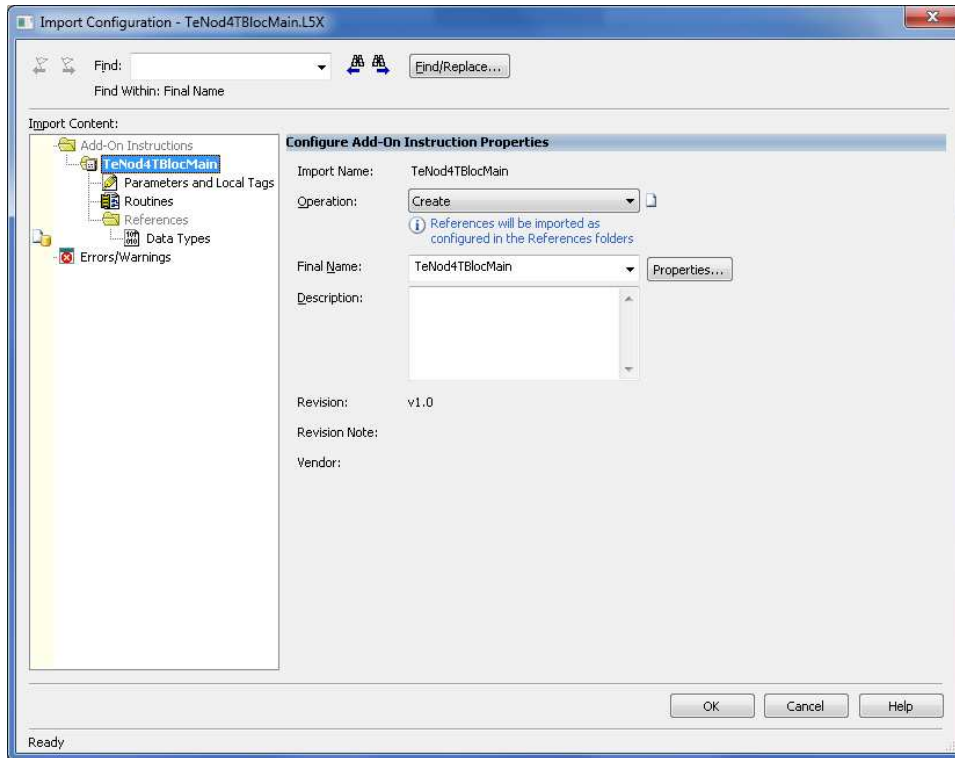
Right click on “ADD-ON instruction” section and select “Import”.

Select the file provided by Scaime.



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

Then confirm the importation of the instruction:



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

When importing the blocs, you may receive warning about data types:

```
Importing...
Name collision: 'TeNod4TFILTER' not defined in import file (existing data type used)
Name collision: 'TeNod4TInternal' not defined in import file (existing data type used)
Name collision: 'TeNod4TStatus' not defined in import file (existing data type used)
Name collision: 'TeNod4TIOCFG' not defined in import file (existing data type used)
Name collision: 'TeNod4TDatas' not defined in import file (existing data type used)
Name collision: 'TeNod4TLEGAL' not defined in import file (existing data type used)
Creating add-on instruction: 'TeNod4TBlocBackupRestore'
Import complete - 0 error(s), 6 warning(s)
```

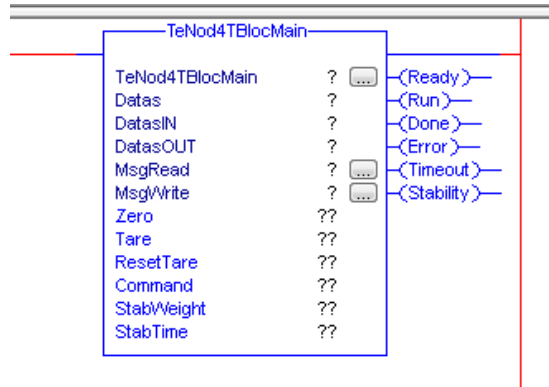
This is just a warning that says that the blocs use data types already imported by the “main” bloc.

4 USING ADD-ON INSTRUCTIONS IN THE PROJECT

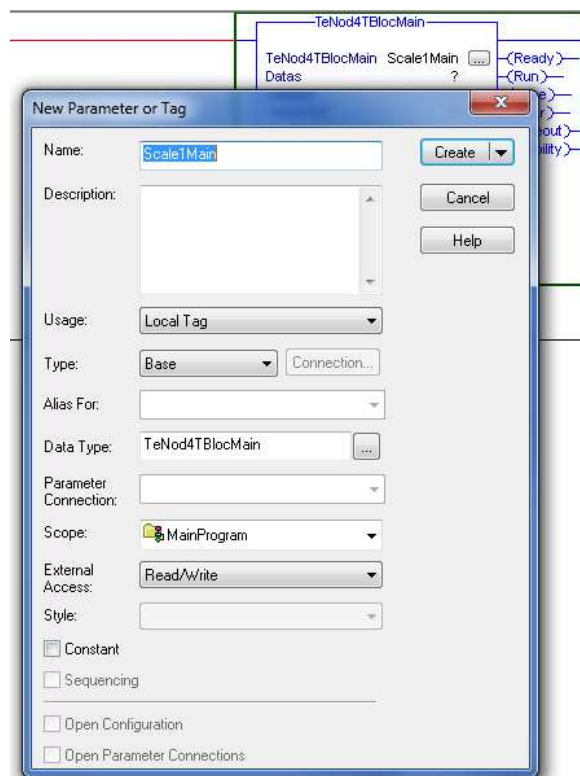
4.1 Main bloc “TeNod4TBlocMain”

4.1.1 Creating the instance

Somewhere in your program, add the TeNod4TBlocMain.

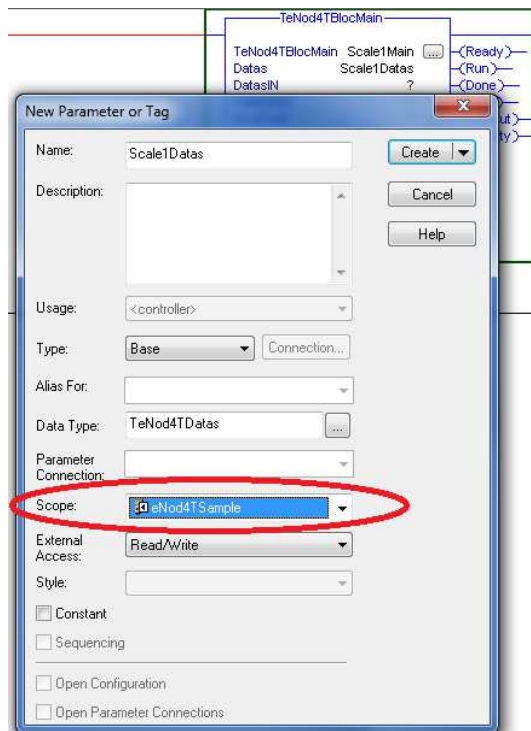


Select and create a name for the instance:



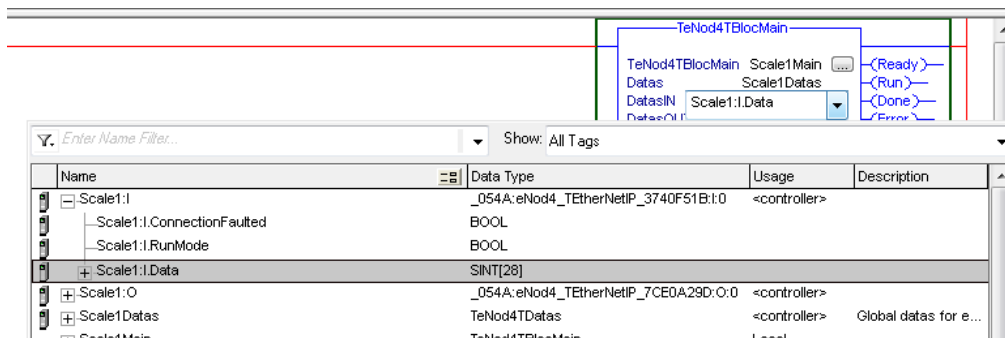
4.1.2 Select global data for the eNod4.

This structure will receive all data from the eNod4 and will be a parameter for all instructions. It must be unique on the project and it must be at controller level scope.

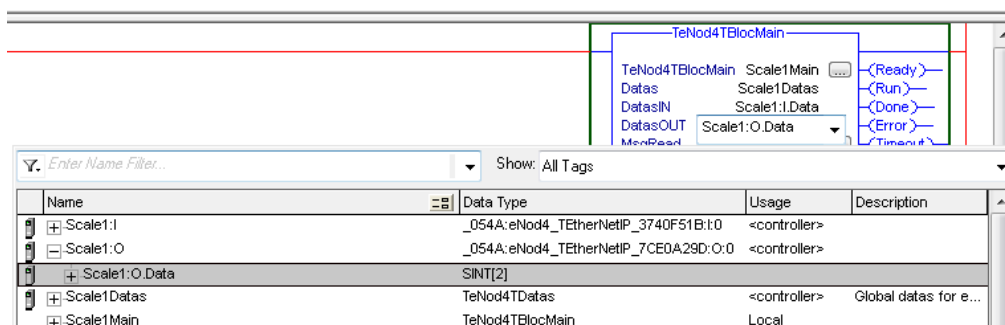


4.1.3 Selecting implicit data from hardware configuration

On the DatasIN field, select the Data[] array automatically created by the configuration for the input data:



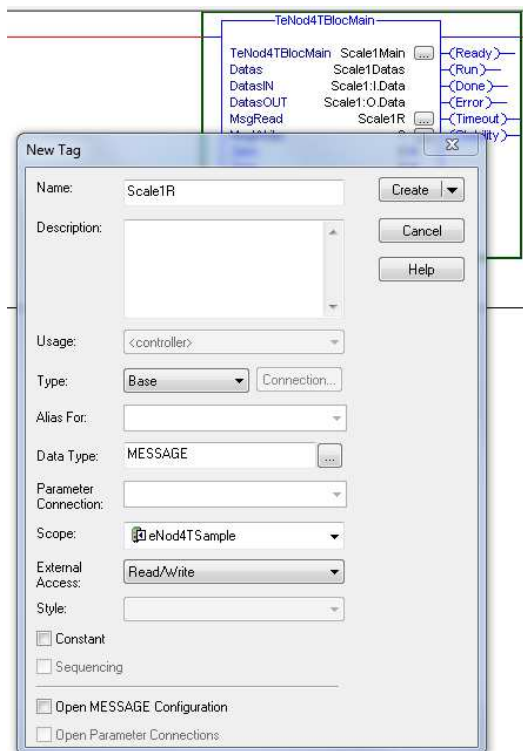
On the DatasOUT field, select the Data[] array automatically created by the configuration for the output data:



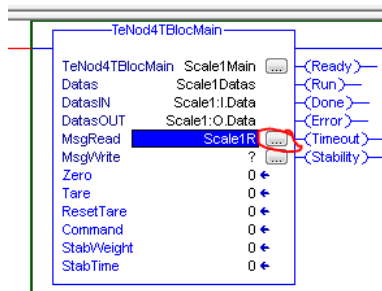
4.1.4 Creating MSG structure for read operations

This structure will be used by the instruction for explicit data read.

It must be unique for every eNod4 of the project and it must be at controller level.

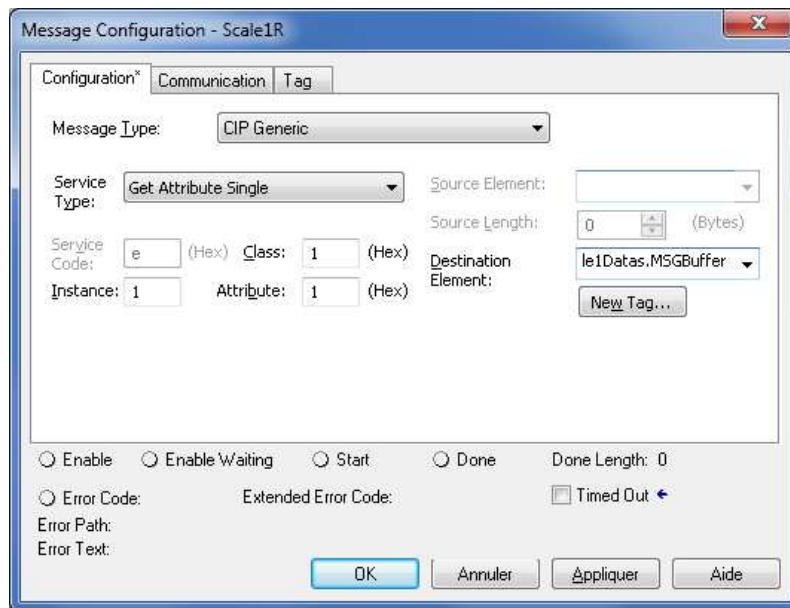


After the structure has been created, click on the configuration button to update MSG data.

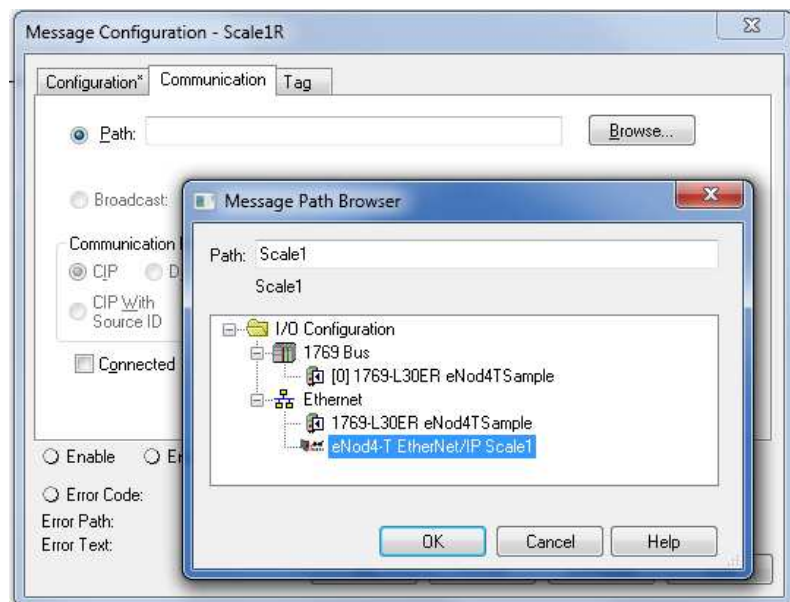


On the message configuration tab, update the following information:

- Message type: leave "CIP generic" messages
- Service type: select "Get Attribute Single"
- Class: enter 1
- Instance: enter 1
- Attribute: enter 1
- Destination element: select the "MSGBuffer" field from the eNod4 structure data created before.



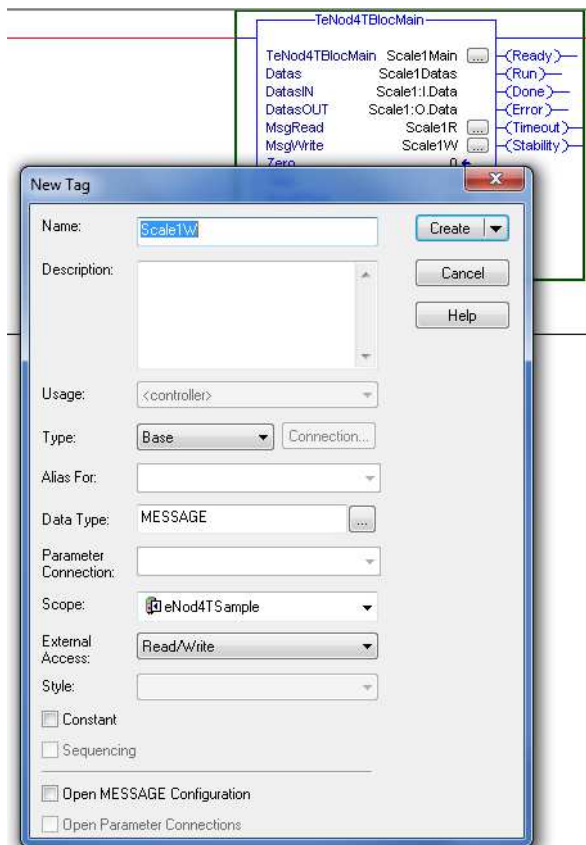
On the “communication” tab select the eNod4 from the hardware configuration.



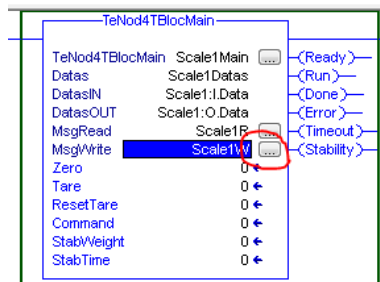
Click OK and OK to confirm the MSG configuration.

4.1.5 Creating MSG structure for write operations.

This structure will be used by the instruction for explicit data write.
It must be unique for every eNod4 of the project and it must be at controller level.

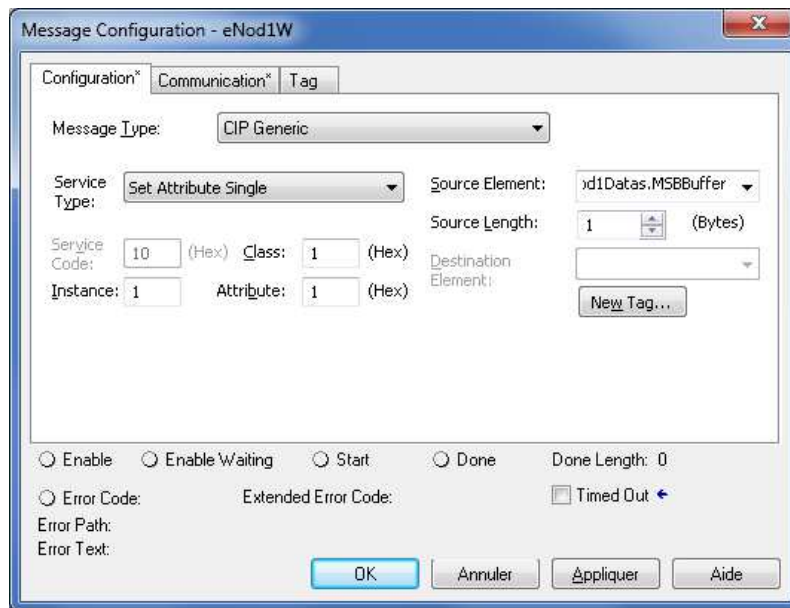


After the structure has been created, click on the configuration button to update MSG data.

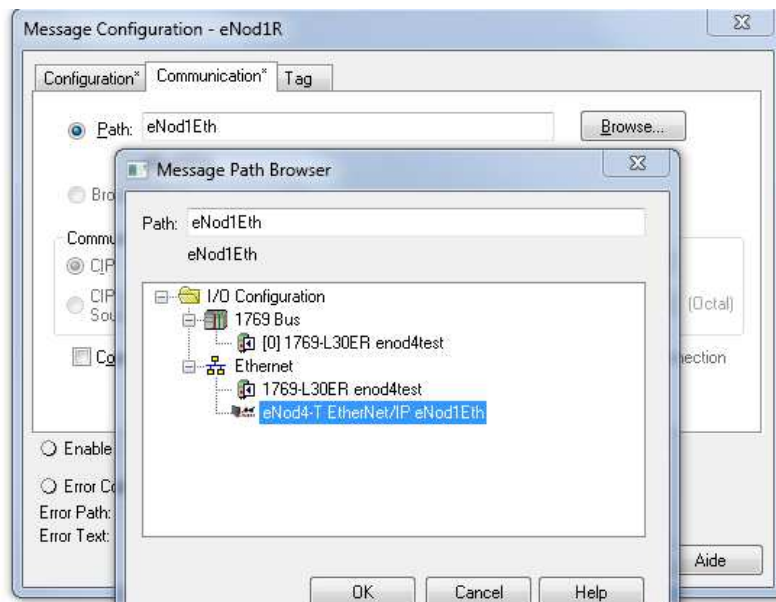


On the message configuration tab, update the following information:

- Message type: leave "CIP generic" messages
- Service type: select "Set Attribute Single"
- Class: enter 1
- Instance: enter 1
- Attribute: enter 1
- Source element: select the "MSGBuffer" field from the eNod4 structure datas created before.



On the “communication” tab select the eNod4 from the hardware configuration.



Click OK and OK to confirm the MSG configuration.

4.1.6 Using the instruction

At this moment the configuration and basic program is done.

You can transfer the program to PLC to test communication

4.1.6.1 Implicit data.

Implicit input data are stored on the field CYCIN of the structure.

Implicit input data can be updated on the field CYCOUT of the structure.

[-] Scale1Ddatas	{...}	
[-] Scale1Ddatas.CYCIN	{...}	
+ Scale1Ddatas.CYCIN.eNodCounter	17102015	
+ Scale1Ddatas.CYCIN.MeasureStatus	{...}	
+ Scale1Ddatas.CYCIN.Gross	1492	
+ Scale1Ddatas.CYCIN.Tare	500	
+ Scale1Ddatas.CYCIN.Net	992	
+ Scale1Ddatas.CYCIN.ADCPoints	22565	
+ Scale1Ddatas.CYCIN.CommandEcho	0	
+ Scale1Ddatas.CYCIN.Response	0	
[-] Scale1Ddatas.CYCOUT	{...}	
+ Scale1Ddatas.CYCOUT.Command	0	
+ Scale1Ddatas.CAL	{...}	
+ Scale1Ddatas.FIL	{...}	
+ Scale1Ddatas.LEGAL	{...}	

You can use the CYCOUT.Command field to send commands to the eNod4. Command values are described on eNod4 software user manual.

Response will be received on the CYCIN.Response value according to Scaime documentation.

4.1.6.2 eNod4 configuration information

By default, the main bloc copies the full configuration from eNod4 to data structure.

This is done only once, when the eNod4 is online the first time.

[-] Scale1Ddatas	{...}	
+ Scale1Ddatas.CYCIN	{...}	
+ Scale1Ddatas.CYCOUT	{...}	
[-] Scale1Ddatas.CAL	{...}	
+ Scale1Ddatas.CAL.MeasureRange	4000	
+ Scale1Ddatas.CAL.CalibSegments	3	
+ Scale1Ddatas.CAL.CalibSegment1	2000	
+ Scale1Ddatas.CAL.CalibSegment2	3000	
+ Scale1Ddatas.CAL.CalibSegment3	3500	
+ Scale1Ddatas.CAL.SensorSens	200000	
+ Scale1Ddatas.CAL.Step	2	
+ Scale1Ddatas.CAL.CalibZero	-2361	
+ Scale1Ddatas.CAL.CalibCoeff1	1.6703	
+ Scale1Ddatas.CAL.CalibCoeff2	1.9882	
+ Scale1Ddatas.CAL.CalibCoeff3	2.3316	
+ Scale1Ddatas.CAL.CalibSpanCorr	1000000	
+ Scale1Ddatas.CAL.gValueCalib	9805470	
+ Scale1Ddatas.CAL.gValueLocal	9805470	
+ Scale1Ddatas.CAL.ZeroOffset	0	
+ Scale1Ddatas.CAL.FixedTare	0	
+ Scale1Ddatas.CAL.SensorRef	0	
+ Scale1Ddatas.CAL.SensorTol	30	
+ Scale1Ddatas.FIL	{...}	

CAL section of the data structure contains all calibration data

FIL 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 eNod4 software user manual (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.

4.1.6.3 Bloc parameters and commands

The bloc contains direct command parameters.

Inputs parameters and commands:

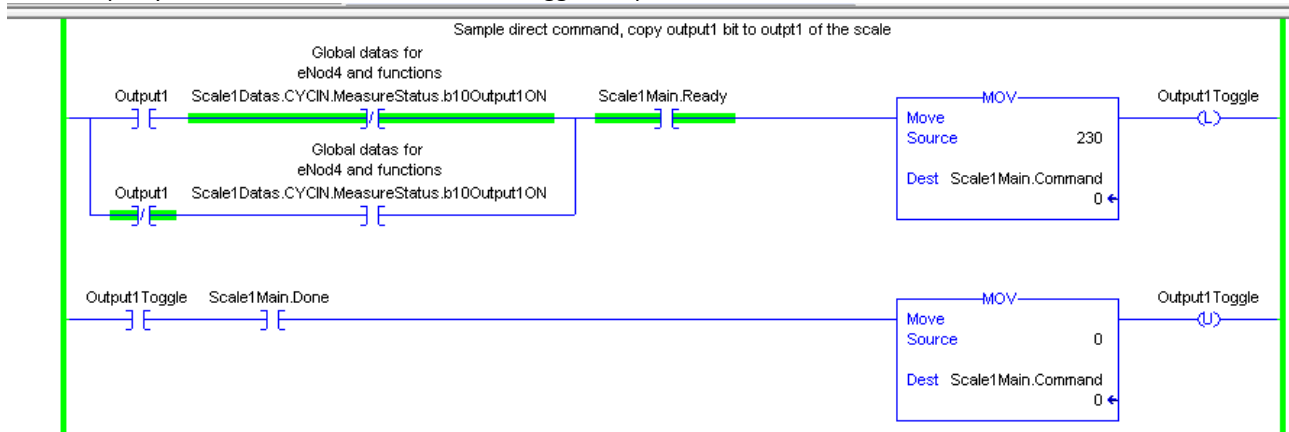
- "Zero", executes a zero command on the scale
- "Tare", executes a tare command on the scale
- "ResetTare", suppress an existing tare

- “Command”, send a command to the scale. List of available commands are described in eNod4 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 tolerances. 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 eNod4 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.



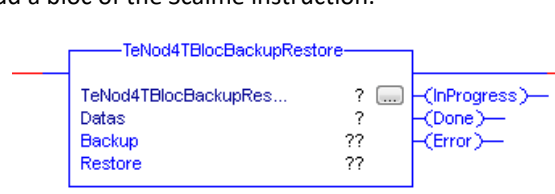
Not, this will work only if the output 1 is defined as “remote” on the scale configuration.

4.2 “TeNod4TBlocBackupRestore” to restore and backup configuration

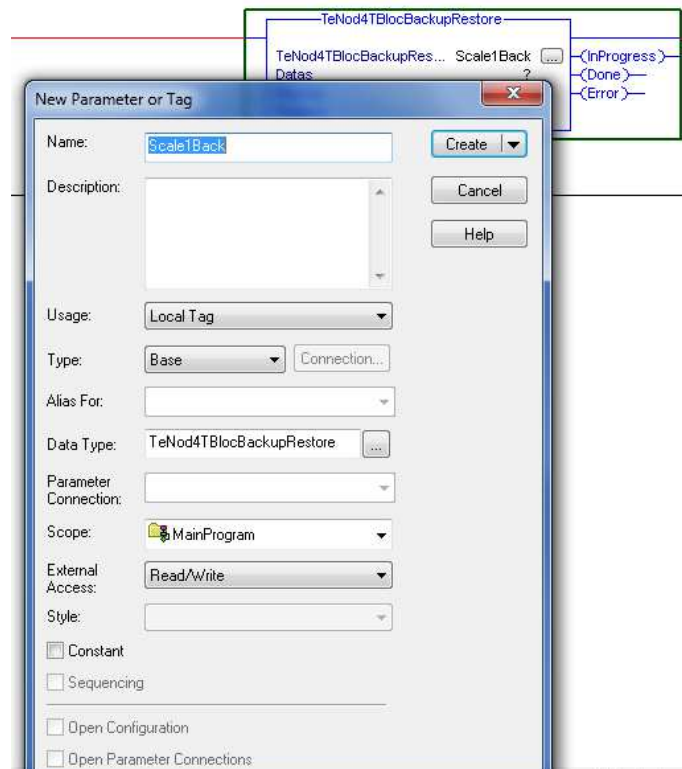
This bloc allows backup and restoring full configuration and calibration data of the scale.

4.2.1 Creating the instance

Somewhere in your program, add a bloc of the Scaime instruction.

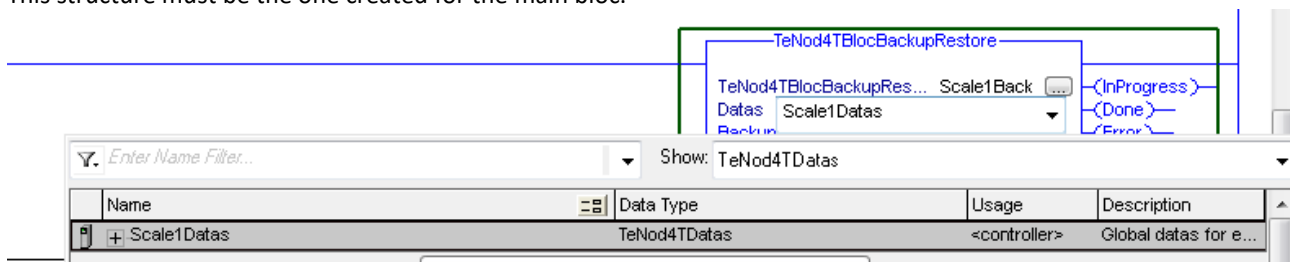


Select and create a name for the instance:



4.2.2 Select global data for the eNod4.

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



4.2.3 Using the instruction

4.2.3.1 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.

4.2.3.2 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 reads the current DSD values of the legal structure.

Backup can be done while the scale is in use.

4.2.3.3 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.

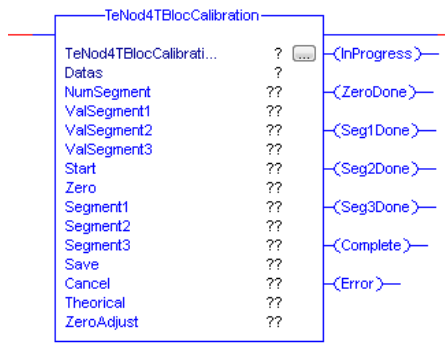
After the restore, an EEPROM save command is sent to save the data. Then a RESET command is sent to restart the eNod4 with the restored configuration. During this moment the communication will be temporally lost.

4.3 “TeNod4TBlocCalibration” to calibrate a scale

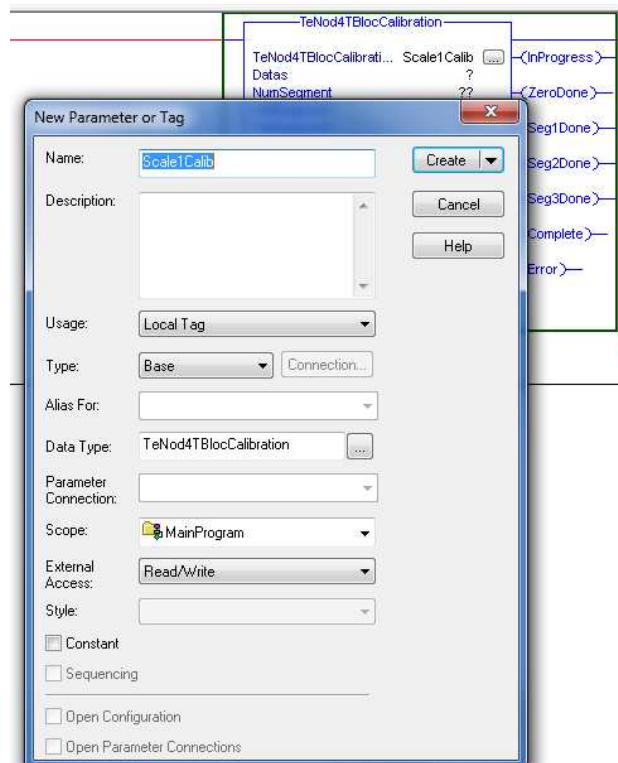
This bloc allows to calibrate the scale.

4.3.1 Creating the instance

Somewhere in your program, add a bloc of the Scaime instruction.

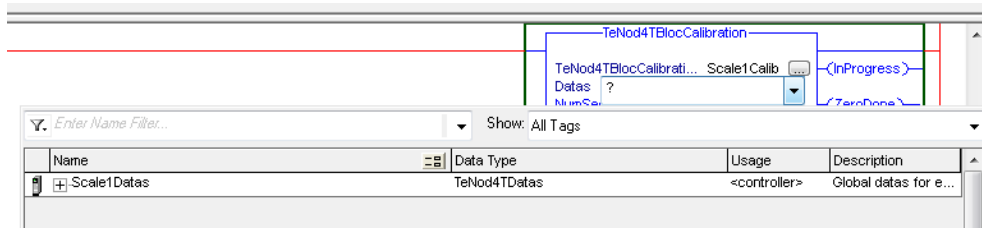


Select and create a name for the instance:



4.3.2 Select global data for the eNod4.

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



4.3.3 Using the instruction

4.3.3.1 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", starts of a physical calibration procedure.
- "Zero", executes to zero point of the calibration procedure
- "Segment1", executes the first point of the calibration procedure
- "Segment2", executes the second point of the calibration procedure
- "Segment3", executes the third point of the calibration procedure
- "Save", saves the calibration procedure. This will save calibration point to EEPROM and copy the calibration data to PLC memory.
- "Cancel", cancels the physical calibration in progress.
- "Theoretical", executes a theoretical calibration.
- "ZeroAdjust", executes 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
- ZeroAdjut, Save
- Theoretical, Save

4.3.3.2 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.

The screenshot shows the 'TeNod4TBlocCalibration' window. The 'Start' button is highlighted in blue. The status bar on the right shows 'InProgress' for the main process and 'ZeroDone' for the first segment.

Command	Value	Status
NumSegment	1	(ZeroDone)
ValSegment1	200	
ValSegment2	0	(Seg1 Done)
ValSegment3	0	
Start	1	(Seg2Done)
Zero	0	
Segment1	0	(Seg3Done)
Segment2	0	
Segment3	0	(Complete)
Save	0	
Cancel	0	(Error)
Theoretical	0	
ZeroAdjust	0	

Physically empty the scale and send the “Zero” command

The screenshot shows the 'TeNod4TBlocCalibration' window. The 'Zero' button is highlighted in blue. The status bar on the right shows 'InProgress' for the main process and 'ZeroDone' for the first segment.

Command	Value	Status
NumSegment	1	(ZeroDone)
ValSegment1	200	
ValSegment2	0	(Seg1 Done)
ValSegment3	0	
Start	0	(Seg2Done)
Zero	1	(Seg3Done)
Segment1	0	
Segment2	0	(Complete)
Segment3	0	
Save	0	
Cancel	0	(Error)
Theoretical	0	
ZeroAdjust	0	

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

The screenshot shows the 'TeNod4TBlocCalibration' window. The 'Segment1' button is highlighted in blue. The status bar on the right shows 'InProgress' for the main process and 'ZeroDone' for the first segment.

Command	Value	Status
NumSegment	1	(ZeroDone)
ValSegment1	200	
ValSegment2	0	(Seg1 Done)
ValSegment3	0	
Start	0	(Seg2Done)
Zero	0	
Segment1	1	(Seg3Done)
Segment2	0	
Segment3	0	(Complete)
Save	0	
Cancel	0	(Error)
Theoretical	0	
ZeroAdjust	0	

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

The screenshot shows the 'TeNod4TBlocCalibration' window. The 'Save' button is highlighted in blue. The status bar on the right shows 'InProgress' for the main process and 'ZeroDone' for the first segment.

Command	Value	Status
NumSegment	1	(ZeroDone)
ValSegment1	200	
ValSegment2	0	(Seg1 Done)
ValSegment3	0	
Start	0	(Seg2Done)
Zero	0	
Segment1	0	(Seg3Done)
Segment2	0	
Segment3	0	(Complete)
Save	1	
Cancel	0	(Error)
Theoretical	0	
ZeroAdjust	0	

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

4.3.3.3 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

The screenshot shows the 'TeNod4TBlocCalibration' HMI screen. The 'Scale1Calib' button is highlighted. The 'ZeroAdjust' parameter is set to 1. The status indicators on the right show 'InProgress' for the calibration process and 'ZeroDone' for the zero adjustment step.

Parameter	Value	Status
TeNod4TBlocCalibrati...	Scale1Calib	InProgress
Datas	Scale1Datas	
NumSegment	1	ZeroDone
ValSegment1	200	
ValSegment2	0	Seg1Done
ValSegment3	0	
Start	0	Seg2Done
Zero	0	
Segment1	0	Seg3Done
Segment2	0	
Segment3	0	Complete
Save	0	
Cancel	0	Error
Theoretical	0	
ZeroAdjust	1	

Send save command

The screenshot shows the 'TeNod4TBlocCalibration' HMI screen. The 'Save' button is highlighted. The 'ZeroAdjust' parameter is set to 1. The status indicators on the right show 'Complete' for the calibration process and 'ZeroDone' for the zero adjustment step.

Parameter	Value	Status
TeNod4TBlocCalibrati...	Scale1Calib	InProgress
Datas	Scale1Datas	
NumSegment	1	ZeroDone
ValSegment1	200	
ValSegment2	0	Seg1Done
ValSegment3	0	
Start	0	Seg2Done
Zero	0	
Segment1	0	Seg3Done
Segment2	0	
Segment3	0	Complete
Save	1	
Cancel	0	Error
Theoretical	0	
ZeroAdjust	0	

4.3.3.4 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.

The screenshot shows the 'TeNod4TBlocCalibration' HMI screen. The 'Start' button is highlighted. The 'ZeroAdjust' parameter is set to 1. The status indicators on the right show 'InProgress' for the calibration process and 'ZeroDone' for the zero adjustment step.

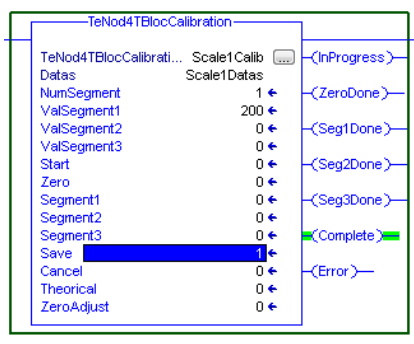
Parameter	Value	Status
TeNod4TBlocCalibrati...	Scale1Calib	InProgress
Datas	Scale1Datas	
NumSegment	1	ZeroDone
ValSegment1	200	
ValSegment2	0	Seg1Done
ValSegment3	0	
Start	1	Seg2Done
Zero	0	
Segment1	0	Seg3Done
Segment2	0	
Segment3	0	Complete
Save	0	
Cancel	0	Error
Theoretical	0	
ZeroAdjust	0	

Send the “Segment1” command

The screenshot shows the 'TeNod4TBlocCalibration' HMI screen. The 'Segment1' button is highlighted. The 'ZeroAdjust' parameter is set to 1. The status indicators on the right show 'InProgress' for the calibration process and 'ZeroDone' for the zero adjustment step.

Parameter	Value	Status
TeNod4TBlocCalibrati...	Scale1Calib	InProgress
Datas	Scale1Datas	
NumSegment	1	ZeroDone
ValSegment1	200	
ValSegment2	0	Seg1Done
ValSegment3	0	
Start	1	Seg2Done
Zero	0	
Segment1	1	Seg3Done
Segment2	0	
Segment3	0	Complete
Save	0	
Cancel	0	Error
Theoretical	0	
ZeroAdjust	0	

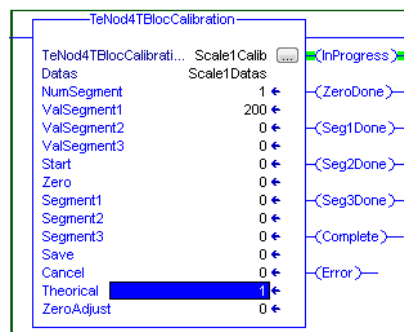
Send the “Save” command



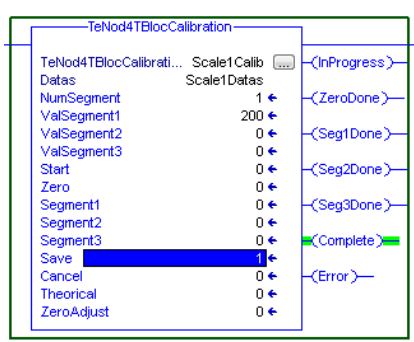
4.3.3.5 Theoretical calibration

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



4.3.3.6 Cancel calibration

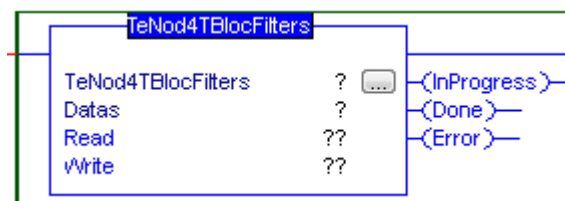
Calibration can be cancelled anytime until the save command is sent. "Cancel" command will reset the eNod4 and calibration will return to the last calibration values.

4.4 "TeNod4TBlocFilters" to read and write filter configuration

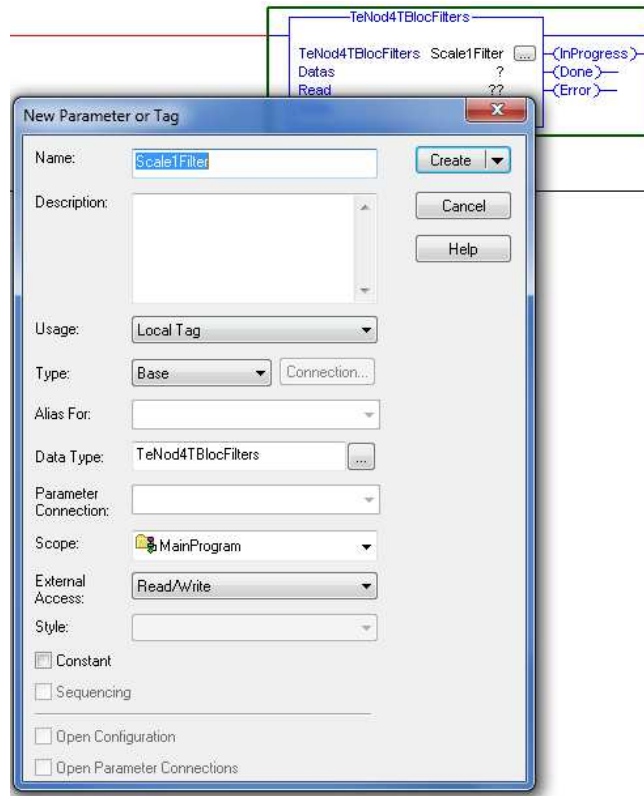
This bloc allows to read and write filter configuration of the scale.

4.4.1 Creating the instance

Somewhere in your program, add a bloc of the Scaime instruction.

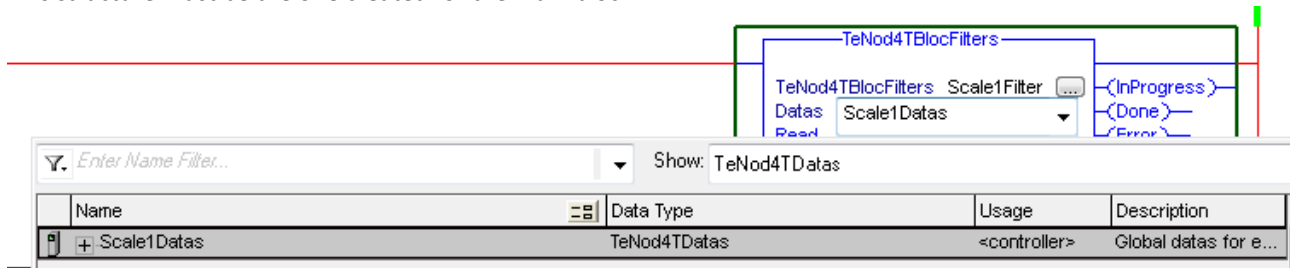


Select and create a name for the instance:



4.4.2 Select global data for the eNod4.

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



4.4.3 Using the instruction

4.4.3.1 Bloc parameters and commands

The bloc contains direct command parameters.

Inputs parameters and commands:

- “Read”, executes a copy of all parameters from the scale to the PLC
- “Write”, executes 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.

4.4.3.2 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 data structure.

4.4.3.3 Set filter configuration

First you need to set the desired filter configuration on the FIL section of the eNod4 data structure. Then, you set the "Write" command of the bloc. New configuration is directly used. The command also saves the new configuration to EEPROM.

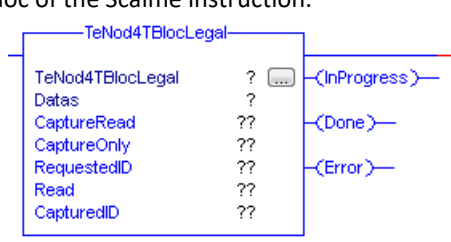
Note that if the A/D parameters values are changed, the changes will be used on the next reboot or reset of the eNod4.

4.5 "TeNod4TBlocLegal" to manage DSD legal values

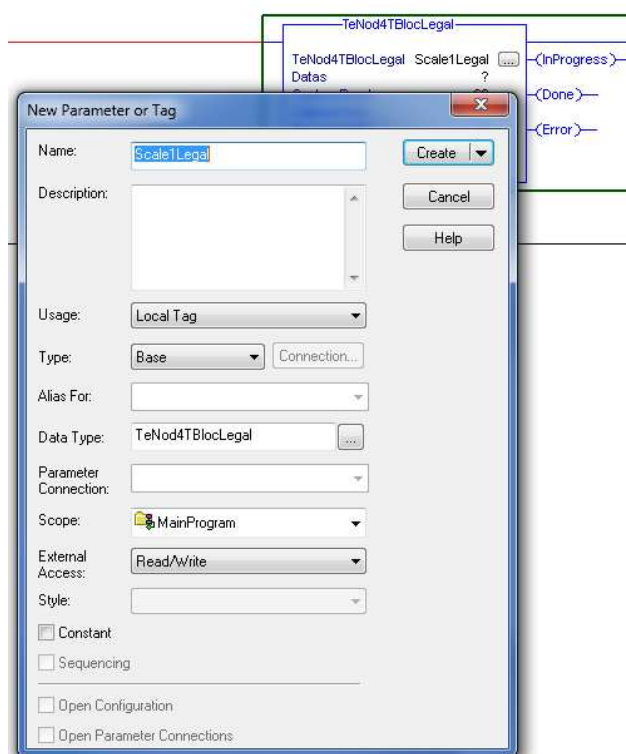
This bloc allows to create DSD values and read them.

4.5.1 Creating the instance

Somewhere in your program, add a bloc of the Scaime instruction.



Select and create a name for the instance:



4.5.2 Select global data for the eNod4.

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

The screenshot shows the TeNod4TBlocLegal bloc in the background. In the foreground, there is a search bar with the text "Enter Name Filter..." and a "Show:" dropdown set to "TeNod4TDatas". Below this is a table with the following data:

Name	Data Type	Usage	Description
Scale1Datas	TeNod4TDatas	<controller>	Global datas for e...

4.5.3 Using the instruction

4.5.3.1 Bloc parameters and commands

The bloc contains direct command parameters.

Inputs parameters and commands:

- “CaptureRead”, creates a DSD measure and copies it to PLC memory.
- “CaptureOnly”, creates a DSD measure but does 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:

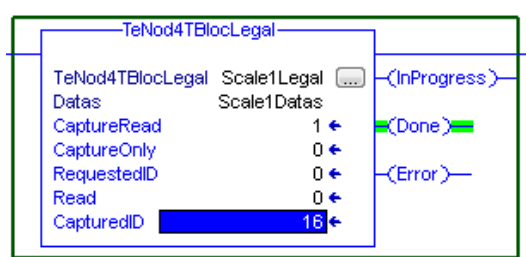
- “InProgress”, it is ON when a command is in progress
- “Done”, it is ON when a command is finished successfully. “Done” stays ON until the command is reset.
- “Error”, it is ON when a command is finished with an error or cannot be executed. “Error” stays ON until the command is reset.
- “CapturedID”, the last ID generated by “CaptureRead” or “CaptureOnly”.

The “CaptureRead” or the “Read” command reads the scale memory and copies the DSD values on the LEGAL members of the data structure:

+ Scale1Datas.LEGAL.DSDReadID	16
+ Scale1Datas.LEGAL.DSDReadNet	62
+ Scale1Datas.LEGAL.DSDResdTare	0
+ Scale1Datas.LEGAL.DSDReadStatus	3202
+ Scale1Datas.LEGAL.DSDReadChecksum	-3280

4.5.3.2 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:



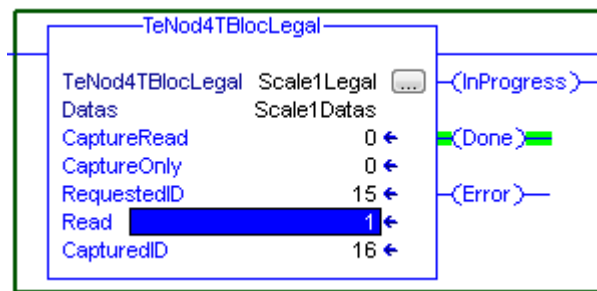
The last ID is also on the structure on the field

+ Scale1Datas.LEGAL.DSDLastID	16
-------------------------------	----

If you have used the “CaptureRead” command, the DSD information are located on the DSDRead part of the LEGAL section of the structure.

4.5.3.3 Read a previous DSD value

Set “RequestedID” value with the ID you want to read, Set the command “Read” to get DSD information from the scale to PLC memory:



The DSD information are located on the DSDRead part of the LEGAL section of the structure:

+ Scale1Datas.LEGAL.DSDReadID	15
+ Scale1Datas.LEGAL.DSDReadNet	54
+ Scale1Datas.LEGAL.DSDResdTare	0
+ Scale1Datas.LEGAL.DSDReadStatus	3202
+ Scale1Datas.LEGAL.DSDReadChecksum	-3271

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.INT.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

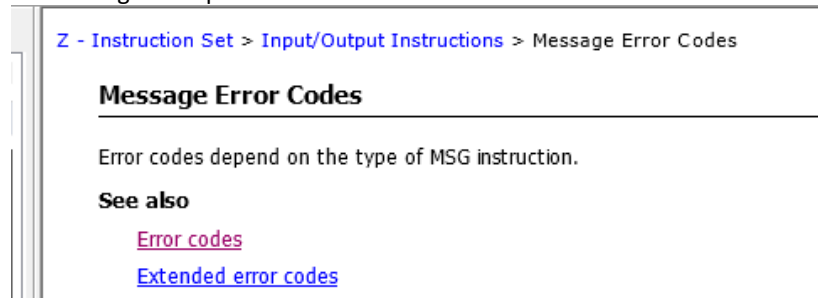
With these two values, refer to chapter 16 to get the register name.

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

The errors codes are returned by the standard MSG instruction. Error codes of the MSG result are located in the following fields:

- **Datas.MSGError**: is the error code
- **Datas.MSGErrorEx**: is the extended error code

For the values refer Studio Designer help file:

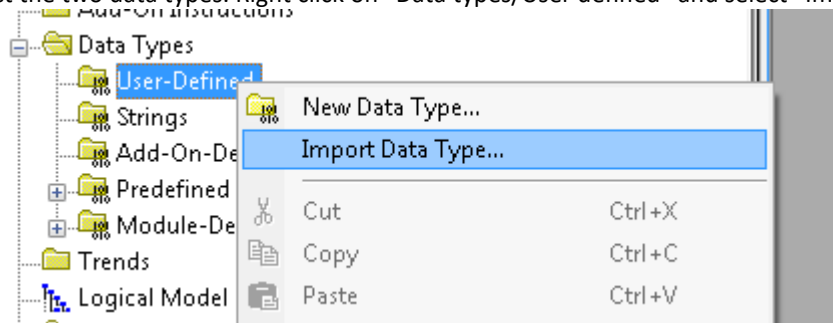


6 BASIC COMMUNICATION

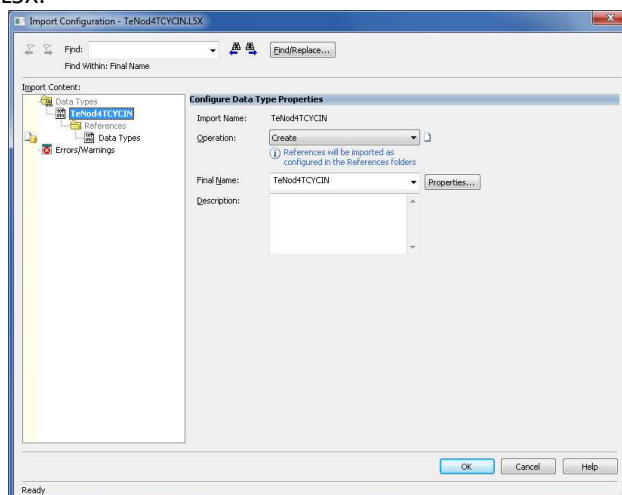
If you do not need any calibration or parameter change on the eNod4 by the PLC, you can use this basic procedure that only read implicit data.

First you need to configure the scale on the Ethernet network as described on previous chapters.

Import on the project the two data types. Right click on "Data types/User defined" and select "Import data type..."



Select the file TeNod4TCYCIN.L5X.



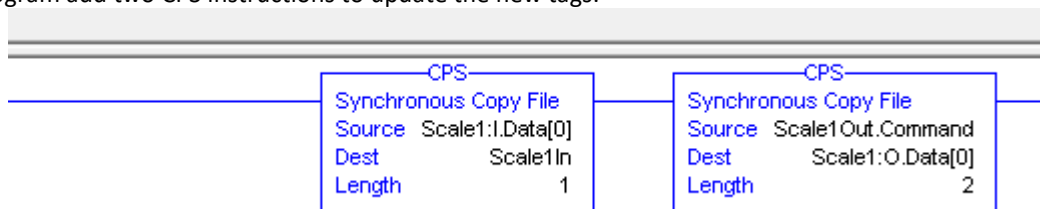
Click OK to confirm.

Repeat the same procedure with the file TeNod4TCYCOUT.L5X

On the program add two tags, one for the input type structure and one for the output type structure:

+	Scale1:U			U04A:eNod4_EtherNetIP_7CE0A23...
+	Scale1In			TeNod4TCYCIN
+	Scale1Out			TeNod4TCYCOUT

On the program add two CPS instructions to update the new tags:



If you load the program on the PLC, IN structure is updated with implicit data:

[-] Scale1In	{...}	
+ Scale1In.eNodCounter	5535343	
+ Scale1In.MeasureStatus	{...}	
+ Scale1In.Gross	28	
+ Scale1In.Tare	0	
+ Scale1In.Net	28	
+ Scale1In.ADCPoints	2404	
+ Scale1In.CommandEcho	0	
+ Scale1In.Response	0	

You can use the OUT structure to send basic command to the scale like zero, tare, etc.

[-] Scale1Out	{...}	{...}		TeNod4TCYCOUT
+ Scale1Out.Command	0		Decimal	INT