OMRON APPLICATION NOTE

Using NX_ReadObj and NX_WriteObj instruction with NX1P along with NX-TS attached to a local bus

This document explains how to:

- Hardware configuration
- Setup the NX_ReadObj to read a parameter value from a locally mounted NX-TS slice.
- Setup the NX_WriteObj to write a parameter value from a locally mounted NX-TS slice.
- Setup the NX_SaveParam to save a parameter value from a locally mounted NX-TS slice to non-volatile memory.
- Examples of code

Product(s):
NX-TS

Sysmac Studio Version: 1.24.2.2

Date: 9/11/2018

Table of Contents

1. Local NX-TS Hardware Connection 3
1.1 Hardware Connection 3

2. Using Sysmac Studio set up the card using CPU/Expansion Rack/ CPU Rack 4
2.1 "CPU/Expansion Racks"/ CPU Rack 4

3. Create Node Location Information in the I/O map and assign a variable to it 5
3.1 Locate Unit number for the NX-TS card 5
3.2 Create Node Location Port variable 6
3.2.1 Display Node Location Information 6
3.2.2 Create Node Location Port variable for the Node location information 7

4. Inserting NX_ReadObj instruction and assigning variables to the block 8
4.1 Inserting the instruction and assigning UnitProxy variable 8
4.2 Assigning the Obj variable and provide the attributes to the variable 9
4.2.1 Assign Index, Subindex and IsCompleteAccess variables to the structure 9
4.3 Assigning the TimeOut variable 10
4.4 Assigning the ReadDat variable 10
4.5 Example Code Showing NX_ReadObj 10

5. Inserting NX_WriteObj instruction and assigning variables to the block 11
5.1 Inserting the instruction and assigning UnitProxy variable 11
5.2 Assigning the Obj variable and provide the attributes to the variable 11
5.3 Assigning the TimeOut variable 11
5.4 Assigning the WriteDat variable 11
5.5 Example code showing NX_WriteObj 12

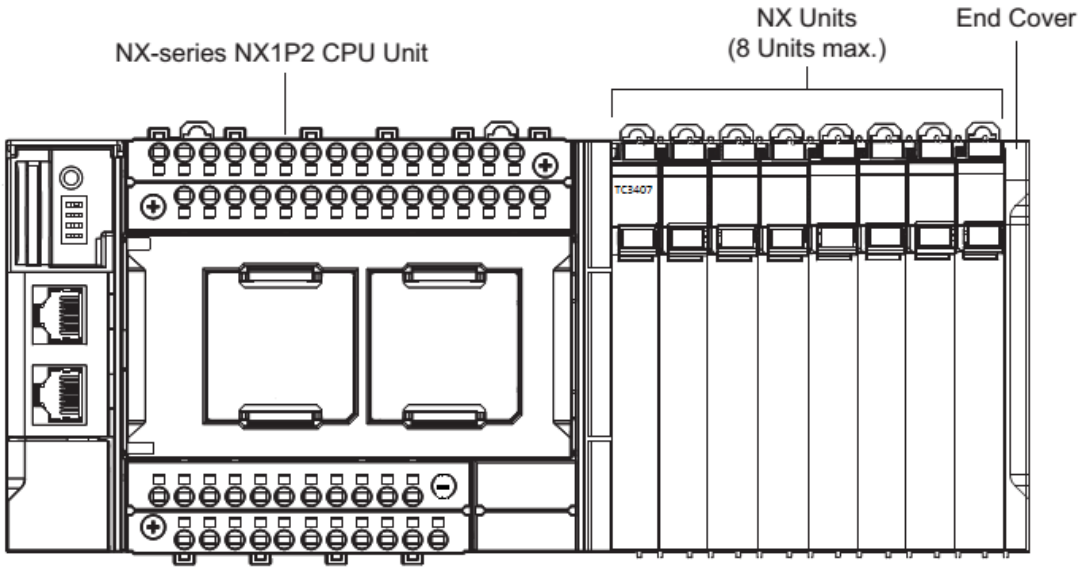
6. Saving data to non-volatile memory 13
6.1 Saving data for parameters labeled "Enabled by restarting" 13
6.1.1 Change the unit to write mode 13
6.1.2 Use the NX_WriteObj to write the value to the parameter 13
6.1.3 Save the parameter value to the unit 13
6.1.4 Restart the unit 13
6.1.5 Example code showing all the instructions joined into one action, ... 13
6.2 Saving data for parameters labeled "Enabled at all times" 15
6.2.1 Use the NX_WriteObj to write the value to the parameter 15
6.2.2 Save the parameter value to the unit 15
6.2.3 Example code showing all the instructions joined into one action, ... 15

1. Local NX-TS Hardware Connection

1.1 Hardware Connection

Attach the NX-TS card on the local bus see (Figure 1). The card can be inserted in any position on the local bus. The example below shows the card in unit #1 position and will be used in this app. note for example purpose.

Figure 1

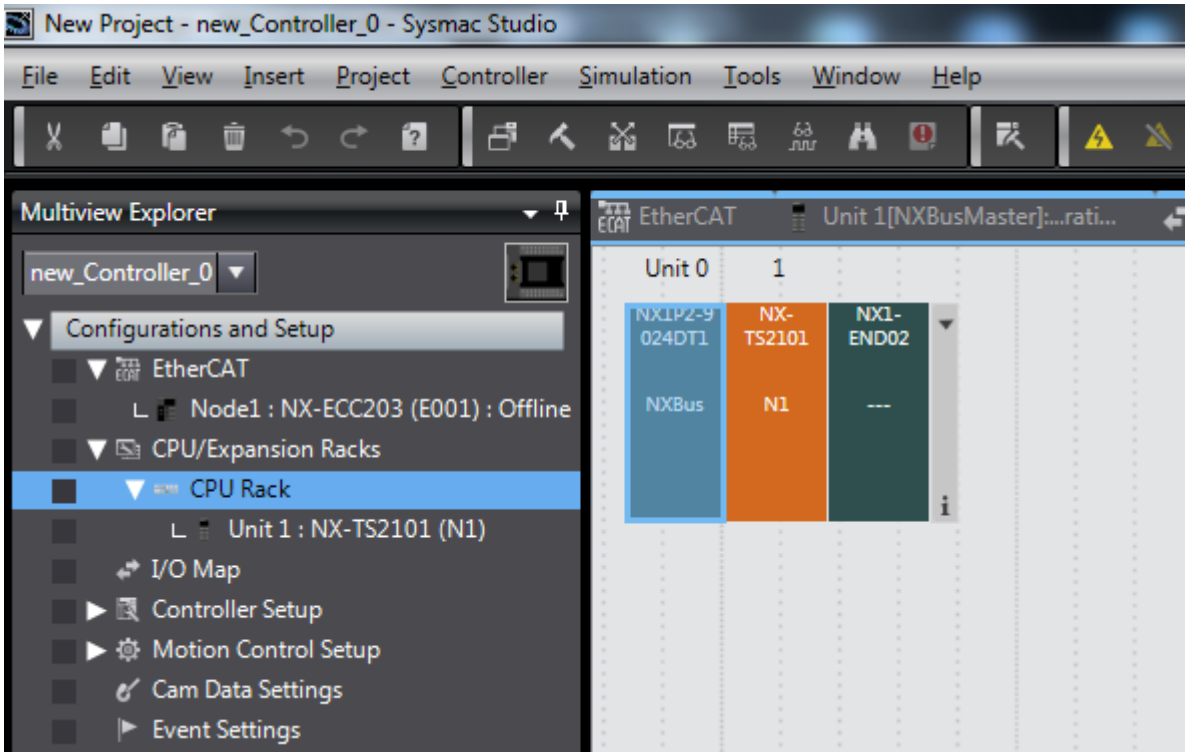


2. Using Sysmac Studio set up the card using CPU/Expansion Rack/ CPU Rack

2.1 "CPU/Expansion Racks"/ CPU Rack

This application note will not go into great detail in assigning the card on the rack. Please see NX1P manual for details on how to configure the rack. However (Figure 2) shows how the card is attached for this application note so the user can see how the addressing works when using the instruction. The NX-TS card will be mounted in Unit 1 position.

Figure 2

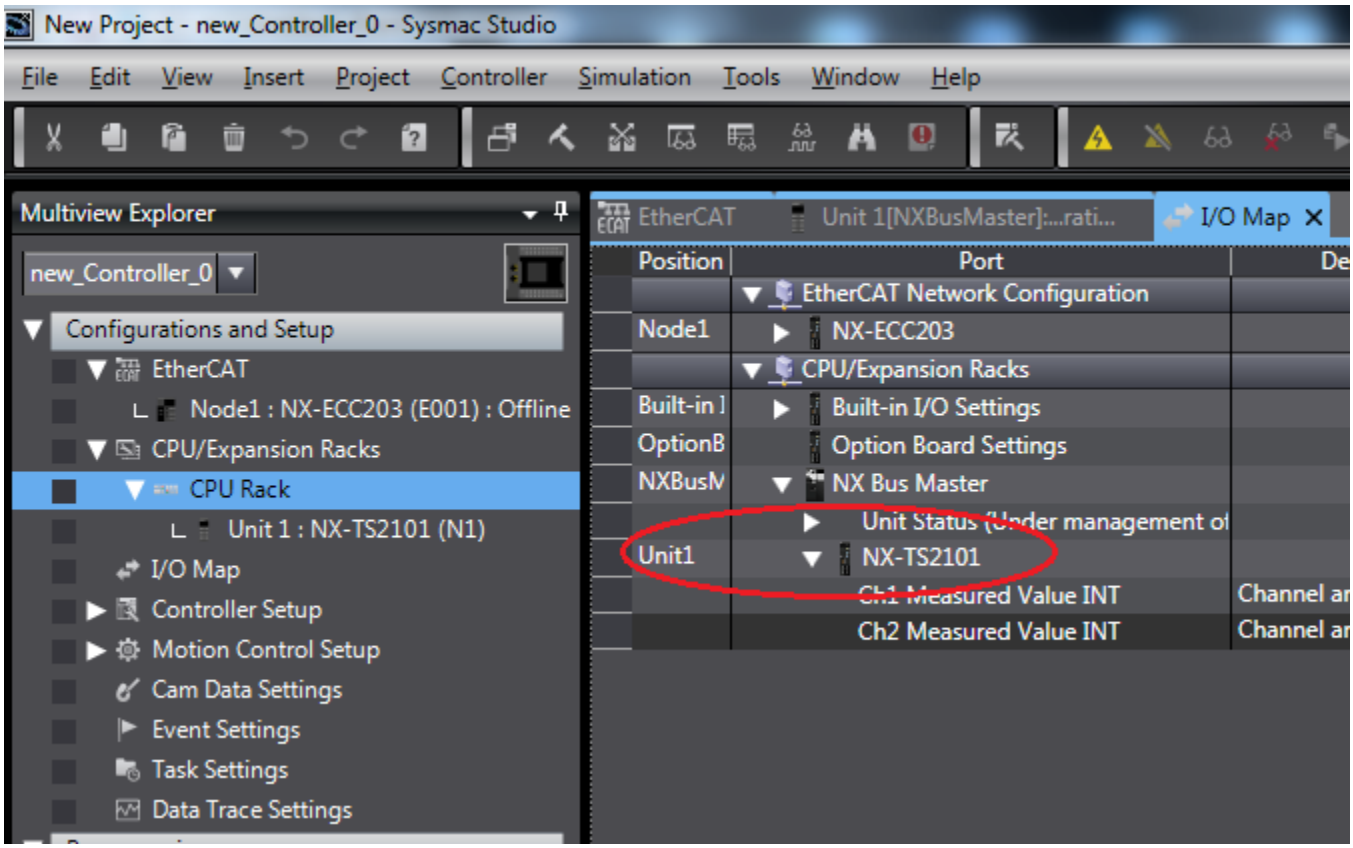


3. Create Node Location Information in the I/O map and assign a variable to it

3.1 Locate Unit number for the NX-TS card

Once the slice is registered as being part of the CPU rack we need to assign a variable using the Node Location Information. To do this double click on “I/O Map” in the Multiview Explorer widow to view the I/O map. Within the I/O Map the user will need to scroll down to the location where the card is mounted. In this example we are using Unit #1 shown in (Figure 3) below:

Figure 3

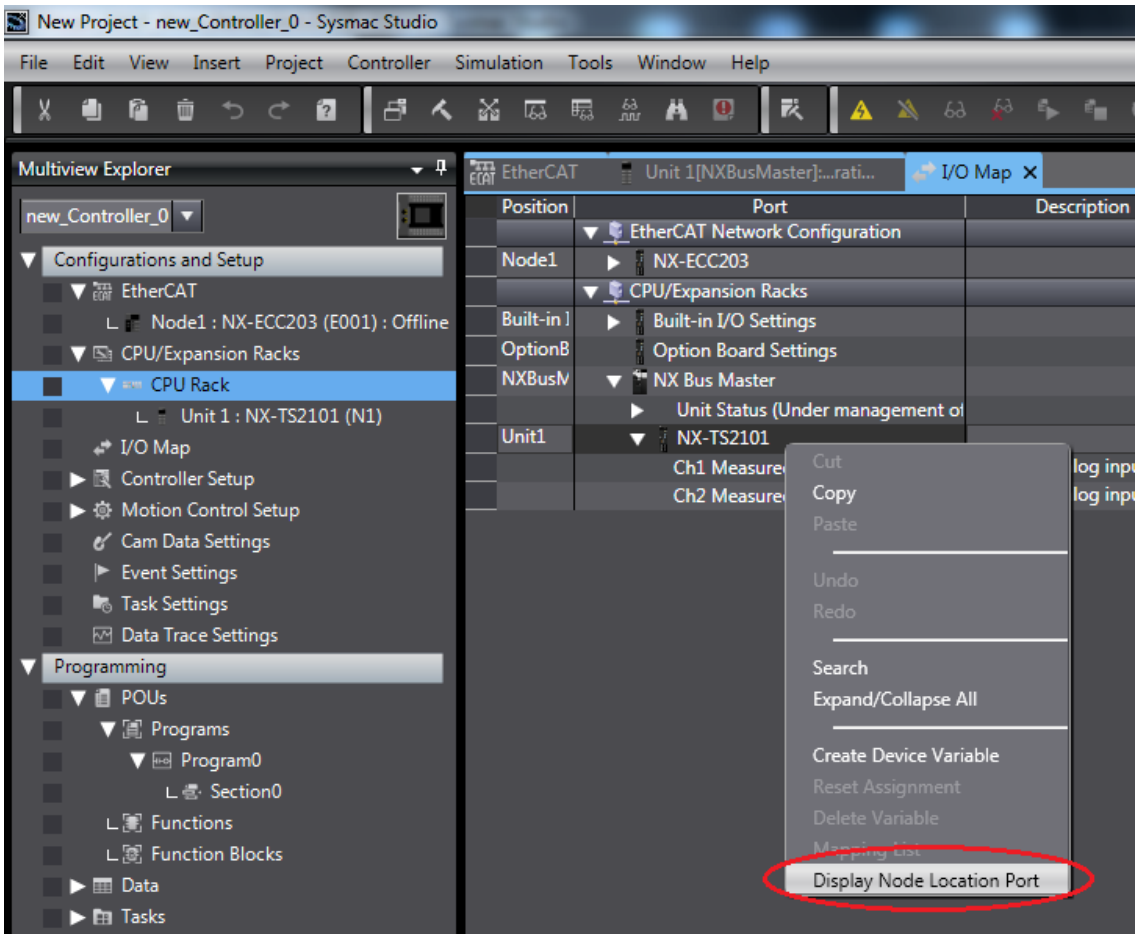


3.2 Create Node Location Port variable

3.2.1 Display Node Location Information

If the user selects the NX-TS card they will be using and then right clicks on the card, this will provide a pull down to where the user can display the Node Location Port. Please select “Display Node Location Port” from the pull down shown in (Figure 4) below.

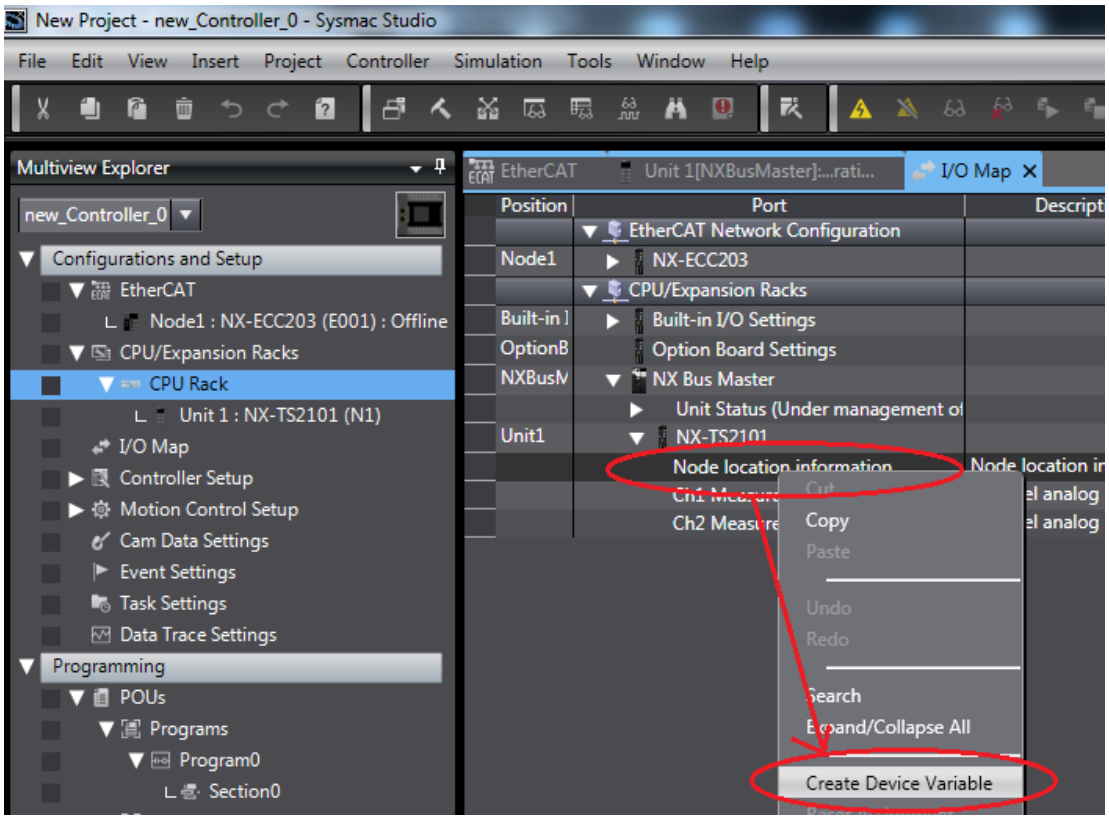
Figure 4



3.2.2 Create Node Location Port variable for the Node location information

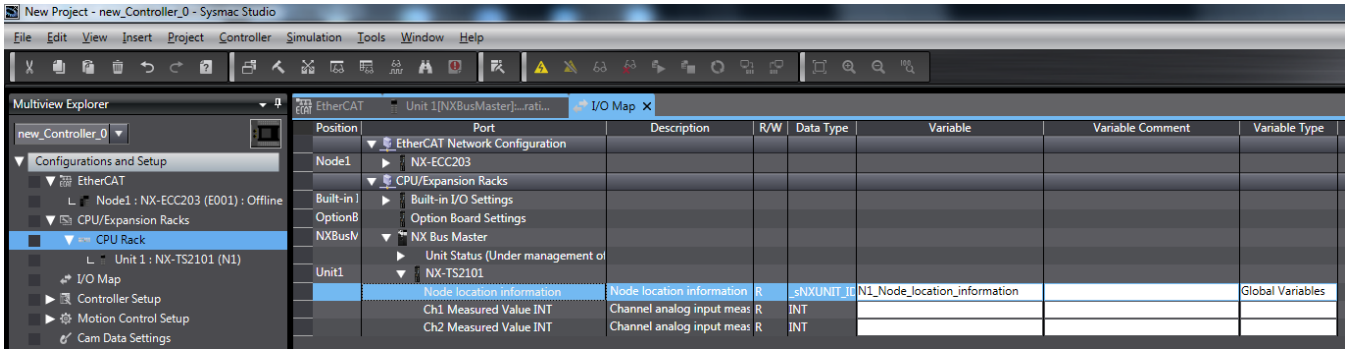
Once the display node location port is selected the software will add the “Node location information” to the I/O Map. If you then select “Node location information” and right click on it the software will provide another popup window to “Create Device Variable”. The user can allow the system to assign a variable or if they choose they can enter a variable for it by manually typing the variable name. Below (Figure 5) shows the procedure allowing the system to create the variable.

Figure 5



Below (Figure 6) shows the variable once it has been created. The variable will be assigned as a global variable type. This variable will then be used in the instruction further below.

Figure 6

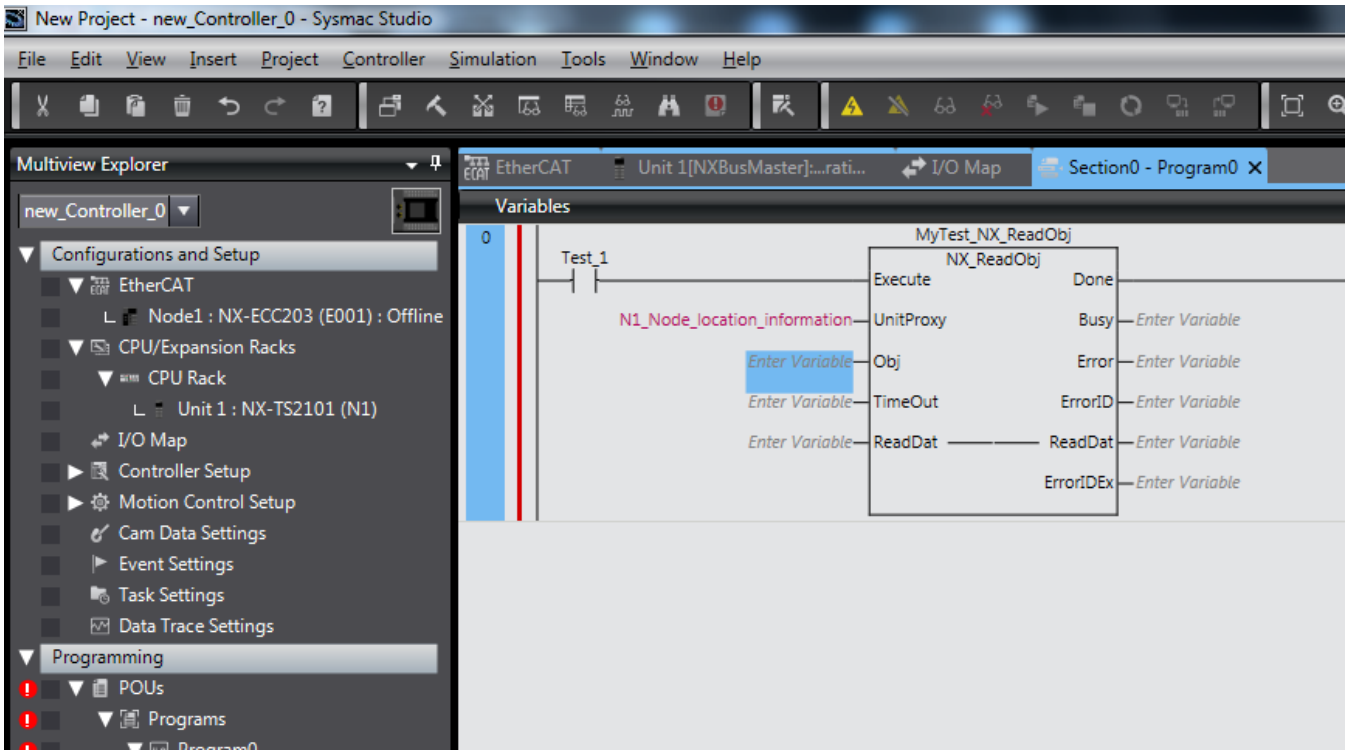


4. Inserting NX_ReadObj instruction and assigning variables to the block

4.1 Inserting the instruction and assigning UnitProxy variable

Open the section that the code will be used and insert the NX_ReadObj instruction in the rung it will be used. Provide a name for the instruction. Assign the variable that was created in section 3 above for the UnitProxy variable of the instruction (Figure 7).

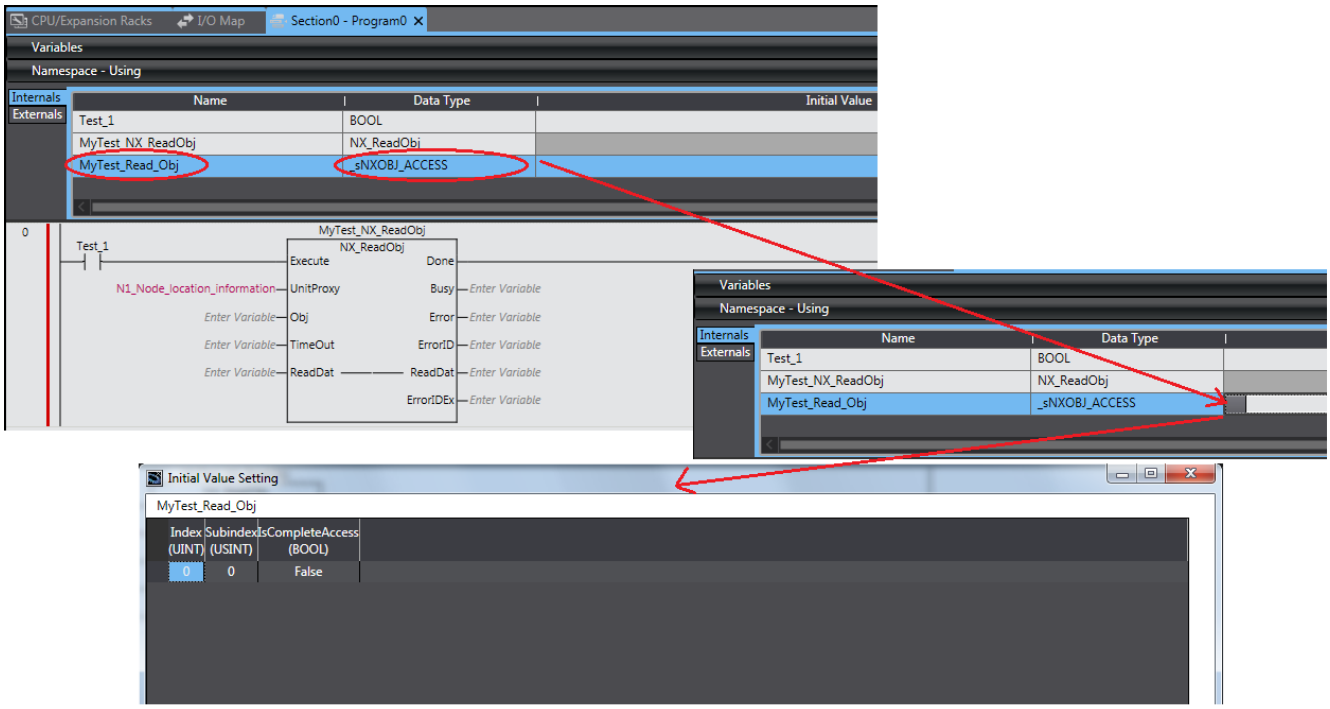
Figure 7



4.2 Assigning the Obj variable and provide the attributes to the variable

Determine the name that will be used for the Obj variable and enter it into the local variable table. Once the variable has been entered assign a “Data Type” of “_sNXOBJ_ACCESS” to it. Once the structure has been assigned click on the left hand side of the Initial Value to assign variable to the structure to open the Initial Value Setting window (Figure 8).

Figure 8



4.2.1 Assign Index, Subindex and IsCompleteAccess variables to the structure

If the user would just like to use the instruction to read just one parameter they can use the Initial Value Setting to pre-can the attributes. If the instruction is intended to read multiple parameters, the attributes can be loaded via the program. Since “_sNXOBJ_ACCESS” is a structure it can be accessed using the following:

“Attribute Name”.Index
“Attribute Name”.Subindex
“Attribute Name”.IsCompleteAccess

To determine the values to store in these attributes the user would need to check the NX-TS manual (W556-E1). Section 6.4 shows the Index and Subindex values for each parameter. For the Index number this needs to be sent as a UINT format, the manual shows the HEX value for the Index.

Ch1 Lower Offset Value (Two-point Correction) has the following Index: 5011 HEX so UINT#16#5011 would be the value entered.

Same as for the Subindex value which needs to be a USINT:

Ch1 Lower Offset Value (Two-point Correction) has the following Subindex: 01 HEX so USINT#16#1 would be the value entered.

For the IsCompleteAccess attribute this needs to remain FALSE (Access data for the specified subindex)

4.3 Assigning the TimeOut variable

TimeOut specifies the timeout time. If a response does not return within the timeout time, it is assumed that communications failed. In that case, the Unit data is not saved. Timeout time if 0 is set, the timeout time is 2.0 s. This can be set from 0 to 60,000 and units are in mSec. Typically using the default time of 0, works for most applications. This can be assigned as UINT#0.

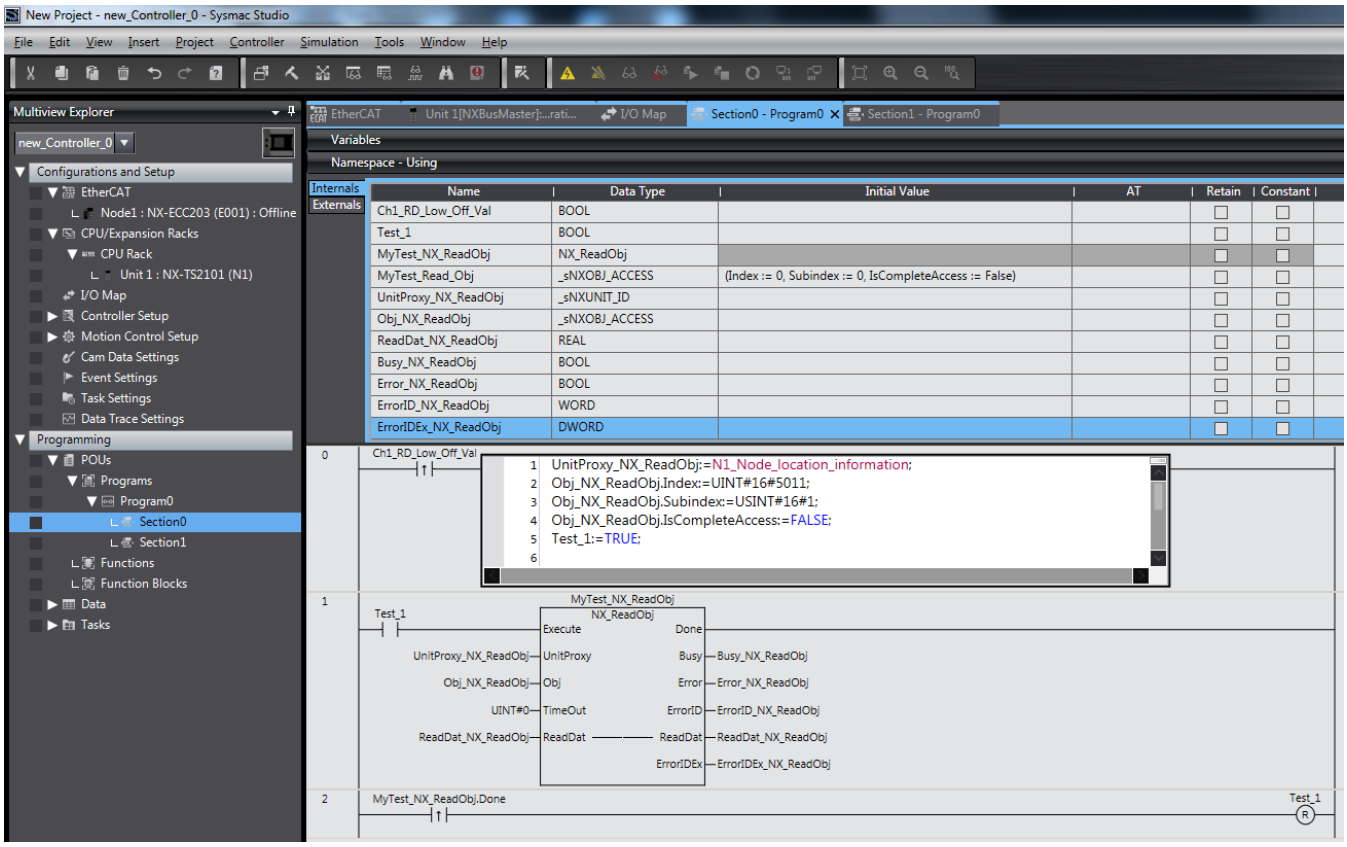
4.4 Assigning the ReadDat variable

Determine the name that will be used for the ReadDat variable and enter it next to the ReadDat position. Depending on the parameter will determine the data type need to set for the variable. Section A-3-2 in the NX-TS manual (W556) provides information on the different data types for each parameter.

4.5 Example Code Showing NX_ReadObj

(Figure 9) shows an example of the NX_ReadOBJ instruction in code using inline structured text to load the variables for the _sNXOBJ_ACCESS structure.

Figure 9



5. Inserting NX_WriteObj instruction and assigning variables to the block

5.1 Inserting the instruction and assigning UnitProxy variable

The procedure is the same as the procedure listed in section 4.1. If the slice is the same unit number as section 4.1 then the same variable can be used for the instruction. Otherwise a new variable would need to be created for the unit number in question (Section 3).

5.2 Assigning the Obj variable and provide the attributes to the variable

The procedure is the same as the procedure listed in section 4.2.

5.3 Assigning the TimeOut variable

The procedure is the same as the procedure listed in section 4.3.

5.4 Assigning the WriteDat variable

When writing to a parameter the user needs to refer to the NX-TS User Manual (W566) Section A-3 for details on the data type for the parameter going to be written to. The user needs to confirm that the parameter is a RW (Read/Write) variable. If the parameter is a read only the instruction will produce an error when trying to write a value. Depending on the variable type will then determine how the value being written is needed to be converted to send it in the correct format. Also confirm the Data Range and the Unit to the parameter being written to. If the value falls outside of the data range an error will be produced when the value is written. It is also important to verify the units as well. So enough digits are added to the value when writing the value.

For example: if a value of 10.0 is needed to be written to the Ch1 Lower Offset Value (Two-point Correction). Looking at the information on this parameter the NX-TS2101 has 0.1°C (Figure 10)

Figure 10

Model	Number of points	Input type	Conversion time	Resolution	I/O refreshing method	Reference
NX-TS2101	2 points	Thermocouple	250 ms/Unit	0.1°C max. *1	Free-Run refreshing	P. A-6
NX-TS2102			10 ms/Unit	0.01°C max.		P. A-7
NX-TS2104			60 ms/Unit	0.001°C max.		P. A-8
NX-TS2201		Resistance thermometer (Pt100/Pt1000, three-wire) *2	250 ms/Unit	0.1°C max.		P. A-9
NX-TS2202			10 ms/Unit	0.01°C max.		P. A-10
NX-TS2204			60 ms/Unit	0.001°C max.		P. A-11

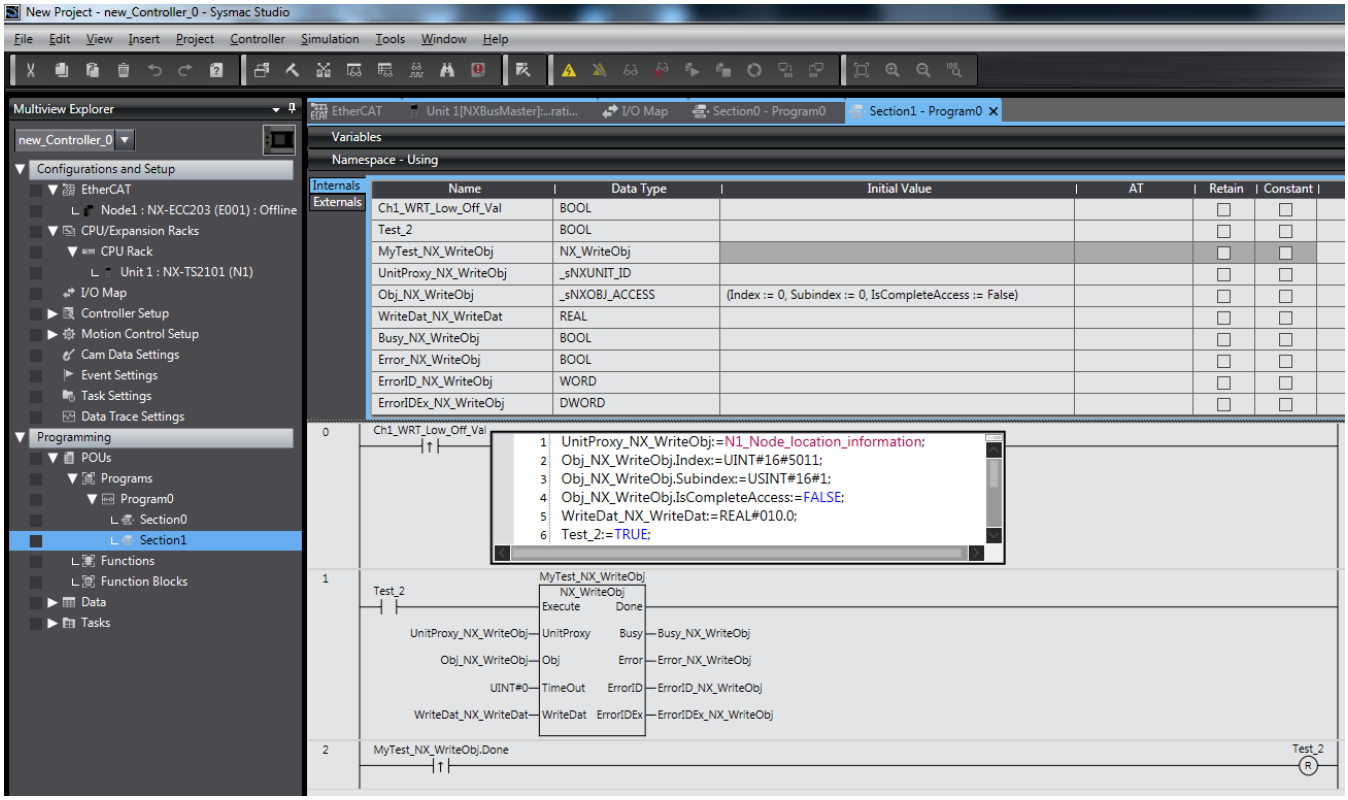
Index (hex)	Subindex (hex)	Object name	Default value	Data range	Unit	Data type	Access	I/O allocation	Data attribute
5011	01	Ch1 Lower Offset Value (Two-point Correction)	0	-400 to 5000	°C or °F	REAL	RW	Not possible	N

The value that is needed to be written is in an REAL data type so the value needed to be sent would then be REAL#010.0. To send a value of 10.1 would then be REAL#010.1.

5.5 Example code showing NX_WriteObj

(Figure 11) shows an example of the NX_WriteOBJ instruction in code using inline structured text to load the variables for the _sNXOBJ_ACCESS structure. The value of 10.0 will be written to the parameter.

Figure 11



6. Saving data to non-volatile memory

6.1 Saving data for parameters labeled “Enabled by restarting”

There are basically two types of data used in the NX-TS, ones mark as Y: “Enabled by restarting” and ones marked as N: “Enabled at all times”. If the user refers to the User Manual (W566) Section A-3-1 and goes to the parameter there will be a column “Data Attribute”, this will indicate how the parameter needs to be saved. For parameters marked as “Enabled by restarting” there will be four steps to write and save the data to non-volatile memory. **Keep in mind that this procedure will stop the card while writing the data.**

6.1.1 Change the unit to write mode

The first step will be to use the NX_ChangeWriteMode instruction which will set the EtherCAT Coupler Unit or NX Unit to a mode that allows writing data. The setting for this instruction is basically assigning the UnitProxy value for the unit. The procedure is the same as the procedure listed in section 4.1. If the slice is the same unit number as section 4.1 then the same variable can be used for the instruction. Otherwise a new variable would need to be created for the unit number in question (Section 3).

6.1.2 Use the NX_WriteObj to write the value to the parameter

Refer to section 5 for details on using the NX_WriteObj instruction.

6.1.3 Save the parameter value to the unit

Using the NX_SaveParam instruction will save the value written to the parameter/unit. The setting for this instruction is basically assigning the UnitProxy value for the unit. The procedure is the same as the procedure listed in section 4.1. If the slice is the same unit number as section 4.1 then the same variable can be used for the instruction. Otherwise a new variable would need to be created for the unit number in question (Section 3).

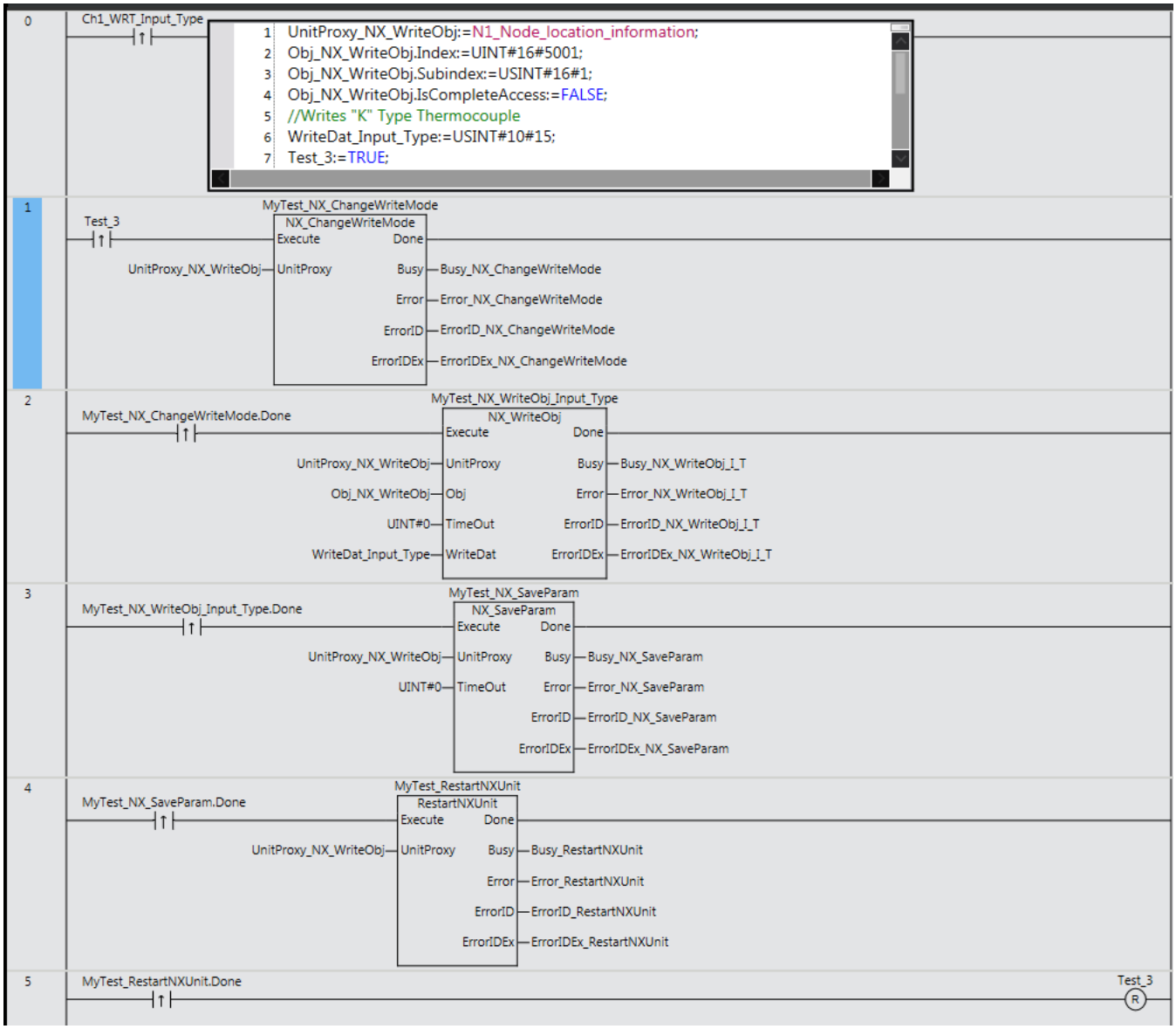
6.1.4 Restart the unit

Using the RestartNXUnit instruction will then perform a restart the controller leaving the module/channels in the state before the change to write mode done in step 6.1.1 above. The setting for this instruction is basically assigning the UnitProxy value for the unit. The procedure is the same as the procedure listed in section 4.1. If the slice is the same unit number as section 4.1 then the same variable can be used for the instruction. Otherwise a new variable would need to be created for the unit number in question (Section 3).

6.1.5 Example code showing all the instructions joined into one action,

(Figure 12) shows an example of the NX_ChangeWriteMode, NX_WriteObj, NX_SaveParam and the RestartNXUnit instructions joined to do a complete write and save of parameter values. The example shows how to change the Input Type of Unit #1 Channel 1. This code can be used for any parameter and any unit that needs to be restarted to save the value written. The user would just need to change the UnitProxy_NX_ReadObj, OBJ_Input_NX_ReadObj.Index and OBJ_Input_NX_ReadObj.Subindex values found in the structure text box. Names for each instruction can be changed based on parameter as well. If the names for the instruction are changed make sure to change the names of the “xxx.Done” bits as well.

Figure 12


0
Ch1_WRT_Input_Type
1 UnitProxy_NX_WriteObj:=N1_Node_location_information;
2 Obj_NX_WriteObj.Index:=UINT#16#5001;
3 Obj_NX_WriteObj.Subindex:=USINT#16#1;
4 Obj_NX_WriteObj.IsCompleteAccess:=FALSE;
5 //Writes "K" Type Thermocouple
6 WriteDat_Input_Type:=USINT#10#15;
7 Test_3:=TRUE;
1
MyTest_NX_ChangeWriteMode
NX_ChangeWriteMode
Test_3
Execute
Done
UnitProxy_NX_WriteObj
UnitProxy
Busy
Busy_NX_ChangeWriteMode
Error
Error_NX_ChangeWriteMode
ErrorID
ErrorID_NX_ChangeWriteMode
ErrorIDEx
ErrorIDEx_NX_ChangeWriteMode
2
MyTest_NX_WriteObj_Input_Type
NX_WriteObj
MyTest_NX_ChangeWriteMode.Done
Execute
Done
UnitProxy_NX_WriteObj
UnitProxy
Busy
Busy_NX_WriteObj_I_T
Obj_NX_WriteObj
Obj
Error
Error_NX_WriteObj_I_T
UINT#0
TimeOut
ErrorID
ErrorID_NX_WriteObj_I_T
WriteDat_Input_Type
WriteDat
ErrorIDEx
ErrorIDEx_NX_WriteObj_I_T
3
MyTest_NX_SaveParam
NX_SaveParam
MyTest_NX_WriteObj_Input_Type.Done
Execute
Done
UnitProxy_NX_WriteObj
UnitProxy
Busy
Busy_NX_SaveParam
UINT#0
TimeOut
Error
Error_NX_SaveParam
ErrorID
ErrorID_NX_SaveParam
ErrorIDEx
ErrorIDEx_NX_SaveParam
4
MyTest_RestartNXUnit
RestartNXUnit
MyTest_NX_SaveParam.Done
Execute
Done
UnitProxy_NX_WriteObj
UnitProxy
Busy
Busy_RestartNXUnit
Error
Error_RestartNXUnit
ErrorID
ErrorID_RestartNXUnit
ErrorIDEx
ErrorIDEx_RestartNXUnit
5
MyTest_RestartNXUnit.Done
Test_3
R

6.2 Saving data for parameters labeled “Enabled at all times”

Again if the user refers to the User Manual (W566) Section A-3-1 and goes to the parameter there will be a column “Data Attribute”, this will indicate how the parameter needs to be saved. For parameters marked as N: “Enabled at all times” there will be two steps to write and save the data to non-volatile memory. **Keep in mind that this procedure will NOT stop the card while writing the data.**

6.2.1 Use the NX_WriteObj to write the value to the parameter

Refer to section 5 for details on using the NX_WriteObj instruction.

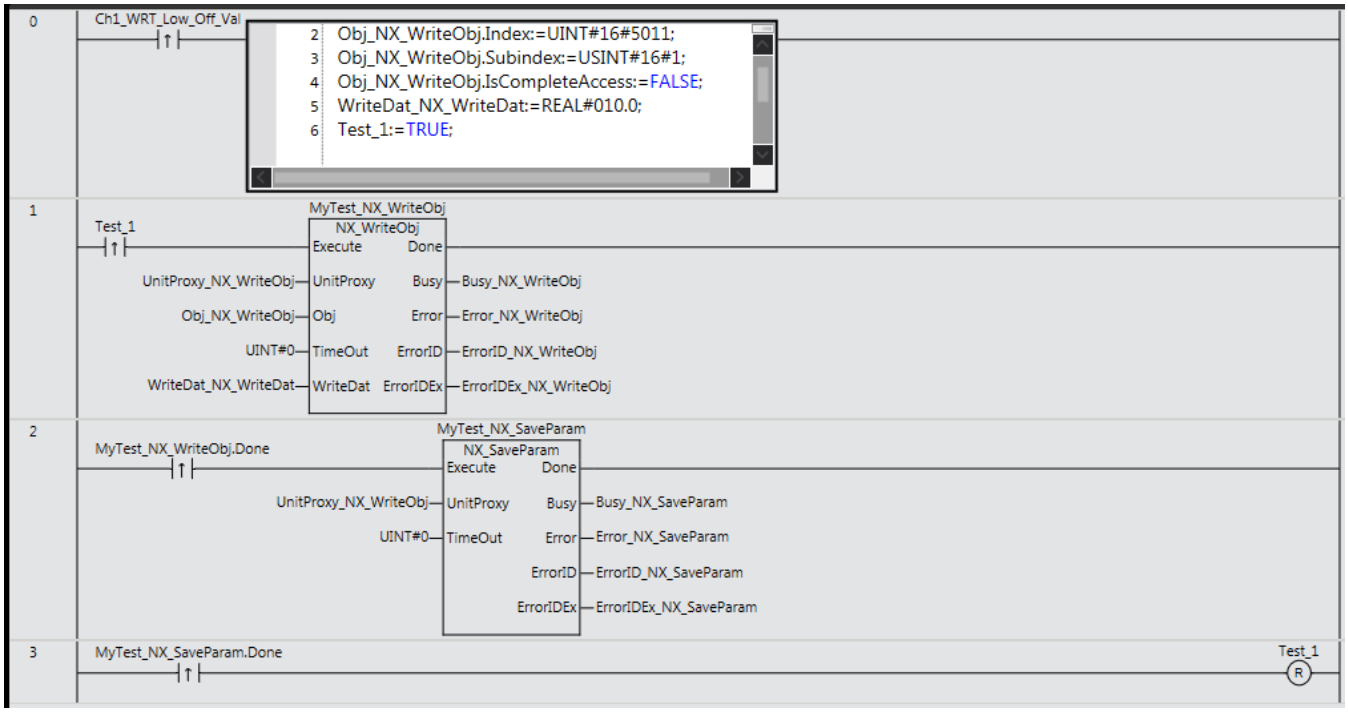
6.2.2 Save the parameter value to the unit

Using the NX_SaveParam instruction will save the value written to the parameter/unit. The setting for this instruction is basically assigning the UnitProxy value for the unit. The procedure is the same as the procedure listed in section 4.1. If the slice is the same unit number as section 4.1 then the same variable can be used for the instruction. Otherwise a new variable would need to be created for the unit number in question (Section 3).

6.2.3 Example code showing all the instructions joined into one action,

(Figure 13) shows an example of the NX_WriteObj and the NX_SaveParam instructions joined to do a complete write and save of parameter values. The example shows how to change the Ch1 Lower Offset Value (Two-point Correction) of Unit #1 Channel 1. This code can be used for any parameter and any unit that needs to be restarted to save the value written. The user would just need to change the UnitProxy_NX_ReadObj, OBJ_Input_NX_ReadObj.Index and OBJ_Input_NX_ReadObj.Subindex values found in the structure text box. Names for each instruction can be changed based on parameter as well. If the names for the instruction are changed make sure to change the names of the “xxx.Done” bits as well.

Figure 13



OMRON AUTOMATION AMERICAS HEADQUARTERS • Chicago, IL USA • 847.843.7900 • 800.556.6766 • automation.omron.com

OMRON CANADA, INC. • HEAD OFFICE
Toronto, ON, Canada • 416.286.6465 • 866.986.6766 • automation.omron.com

OMRON ELECTRONICS DE MEXICO • HEAD OFFICE
Ciudad de México • 52.55.5901.4300 • 01.800.386.6766 • mela@omron.com

OMRON ELECTRONICS DE MEXICO • SALES OFFICE
San Pedro Garza García, N.L. • 81.12.53.7392 • 01.800.386.6766 • mela@omron.com

OMRON ELECTRONICS DE MEXICO • SALES OFFICE
Eugenio Garza Sada,León, Gto • 01.800.386.6766 • mela@omron.com

OMRON ELETRÔNICA DO BRASIL LTDA • HEAD OFFICE
São Paulo, SP, Brasil • 55 11 5171-8920 • automation.omron.com

OMRON ARGENTINA • SALES OFFICE
Buenos Aires, Argentina • +54.11.4521.8630 • +54.11.4523.8483
mela@omron.com

OTHER OMRON LATIN AMERICA SALES
+54.11.4521.8630 • +54.11.4523.8483 • mela@omron.com

Authorized Distributor:

Controllers & I/O
• Machine Automation Controllers (MAC) • Motion Controllers
• Programmable Logic Controllers (PLC) • Temperature Controllers • Remote I/O

Robotics
• Industrial Robots • Mobile Robots

Operator Interfaces
• Human Machine Interface (HMI)

Motion & Drives
• Machine Automation Controllers (MAC) • Motion Controllers • Servo Systems
• Frequency Inverters

Vision, Measurement & Identification
• Vision Sensors & Systems • Measurement Sensors • Auto Identification Systems

Sensing
• Photoelectric Sensors • Fiber-Optic Sensors • Proximity Sensors
• Rotary Encoders • Ultrasonic Sensors

Safety
• Safety Light Curtains • Safety Laser Scanners • Programmable Safety Systems
• Safety Mats and Edges • Safety Door Switches • Emergency Stop Devices
• Safety Switches & Operator Controls • Safety Monitoring/Force-guided Relays

Control Components
• Power Supplies • Timers • Counters • Programmable Relays
• Digital Panel Meters • Monitoring Products

Switches & Relays
• Limit Switches • Pushbutton Switches • Electromechanical Relays
• Solid State Relays

Software
• Programming & Configuration • Runtime

AppNote_NX-I/O_0001.0_NX-TS

Note: Specifications are subject to change.

Printed in U.S.A.

Printed on recycled paper.