OMRON

Application Library

OEN_Communication 1.05.00
OEN_CommunicationSecure 1.00.02

Sysmac Function Block Library for Modbus, NMEA0183, EtherCAT SDO, EIP Control, SMS and Modem Communication

User's Manual

Contents

Introduction

Thank you for using the Application Library: **OEN_Communication**
Use it when programming with the automation software Sysmac Studio.
This manual contains information that is necessary to use the Library with Sysmac Studio.
Hereinafter, the function blocks are described as FB, functions as FNs.

1.1. Notice

This manual describes the necessary information to use the Application Library. Refer also to the user's manuals for Application Library, the *Sysmac Studio Version1 Operation Manual* (Cat.No. W504)

Please read and understand this manual before using the Library. Keep this manual in a safe place where it will be available for reference during operation.

1.2. Terms and Conditions Agreement

1 NO WARRANTY

1） The functions and function block Library is distributed as a sample in the hope that it will be useful, but without any warranty. It is provided “as is” without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the function block is with you. Should the function block prove defective, you assume the cost of all necessary servicing, repair or correction.

2） In no event unless required by applicable law the author will be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the function block (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the function block to operate with any other programs), even if the author has been advised of the possibility of such damages.

2 LIMITATION OF LIABILITY

1) OMRON SHALL HAVE NO LIABILITY FOR DEFECT OF THE SOFTWARE.
2) OMRON SHALL HAVE NO LIABILITY FOR SOFTWARE PARTS DEVELOPED BY THE USER OR ANY THIRD PARTY USING THE FUNCTION BLOCK DESCRIBED ON THIS MANUAL.

3 APPLICABLE CONDITIONS

USER SHALL NOT USE THE SOFTWARE FOR THE PURPOSE THAT IS NOT PROVIDED IN THE ATTACHED USER MANUAL.

4 CHANGE IN SPECIFICATION

The software specifications and accessories may be changed at any time based on improvements and other reasons.

5 ERRORS AND OMISSIONS

The information in this manual has been carefully checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical, or proofreading errors, or omissions.

1.3. Safety Precautions

Definition of Precautionary Information

The following notation is used in this manual to provide precautions required to ensure safe usage of OEN_Communication Library.
The safety precautions that are provided are extremely important to safety. Always read and heed the information provided in all safety precautions.

The following notation is used.

⚠ WARNING	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury. Additionally, there may be severe property damage.

⚠ Caution	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or property damage.

Precautions for Safe Use

Indicates precautions on what to do and what not to do to ensure safe usage of the product.

Precautions for Correct Use

Indicates precautions on what to do and what not to do to ensure proper operation and performance.

Additional Information

Additional information to read as required.
This information is provided to increase understanding or make operation easier.

	The triangle symbol indicates precautions (including warnings). The specific operation is shown in the triangle and explained in text. This example indicates a general precaution.
	The filled circle symbol indicates operations that you must do. The specific operation is shown in the circle and explained in text. This example shows a general precaution for something that you must do.

Warning list

WARNING	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury. Additionally, there may be severe property damage.

Emergency stop circuits, interlock circuits, hardware limit and similar safety measures must be provided in external control circuits.
Using this FB in a device, confirm that the program and FB operate properly. Design a program so that safety measures such as fail-safe circuits are implemented outside of the FB

Caution list

Caution	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or property damage.

Confirming an operation of the control program, including this FB. Trial operation such as the concerned motor runs in low velocity is recommended.
Performing adjustment of the device controlled by the program with this FB, secure the safety of the machine.
Do not use this FB for the system with devices and versions not specified in this document. To use, contact your OMRON representative
If a Task Period Exceeded Error occurred by executing this FB, the CPU Unit shifts to an error state. Make sure to set the execution task period to an appropriate value by referring to the execution time of this FB.
Do not delete the instances from the program with online editing during an execution of this FB. Program communication will stop in error.
Make sure to set the input parameters of this FB appropriately in accordance with the actual device. Make settings as described in this manual.

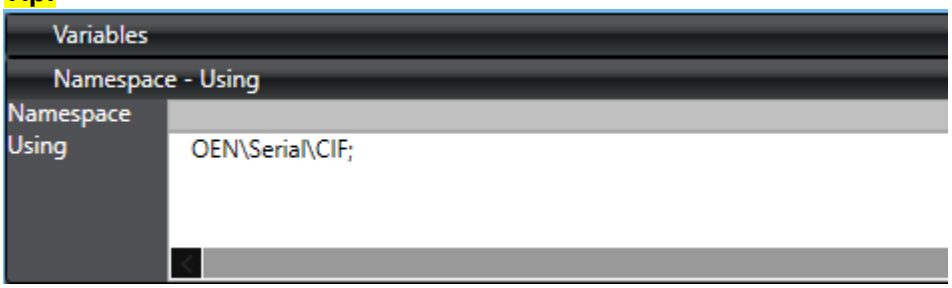
Functions and FunctionBlocks

Applications

The **OEN_Communication** is a set of functions and function blocks for Modbus, EtherCAT SDO and Modems Communication. If not notified, these function blocks are compatible with all Sysmac series PLCs having Firmware 1.18 or higher.

~~This Library is using features from~~
~~OEN_DataTypes~~
~~You must add it to have this library working properly.~~
(Future plan).

Tip!



If you add f.ex. the OEN\Serial\CIF namespace you can save yourself from entering the whole path when using Functions and Function Blocks.

Library Change Log

OEN_Communication

See details on each Function/FunctionBlock	
1.00.18	Added 4 new modbus function blocks. ModbusRTU Slave/Master. ModbusTCP Client/Server. The master/client are configurable based on a request list. The slave/server have an Accesslist to control R, W, RW properties.
1.00.19	Renamed a variable called "Dummy", due to a function in OEN_Toolbox called "Dummy"
1.00.20	Rebuilt FB's for EtherCat SDO handling to use dynamic ARRAY for NodeDat. Removed Input NoOfNodes.
1.00.21	Redesigned the NX_SendSMS, NX_RcvSMS, NX_ClearModemBuffer
1.00.22	Changed NX_ModbusRTU_Master and ModbusTCP_Client. Separated modbus addresses into local and remote. So that one master can be used for many similar slaves.
1.01.00	Added some F and FBs from OEN_BaseBlocks.
1.02.00	EIP_Control Update
1.03.00	Modbus_RTUMaster and Modbus_TCPClient update
1.03.00	Mapping example added to Modbus TCPServer
1.04.00	Added EIP_PortReset
1.05.00	Added NMEA function and an example for NX_SerialRcv

OEN_CommmunicationSecure

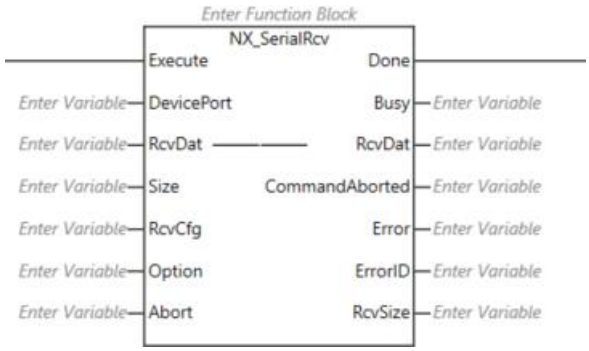
See details on each Function/FunctionBlock	
1.00.0	Contains RT100_SMSAlarm
1.00.1	RT100_SMSAlarm minor bugfix
1.00.2	RT100_SMSAlarm minor bugfix

1. NX_SerialRcv

A Function Block in the standard Sysmac Studio Library to be used when receiving data through a Serial Port. A more thorough description can be found in the manual.

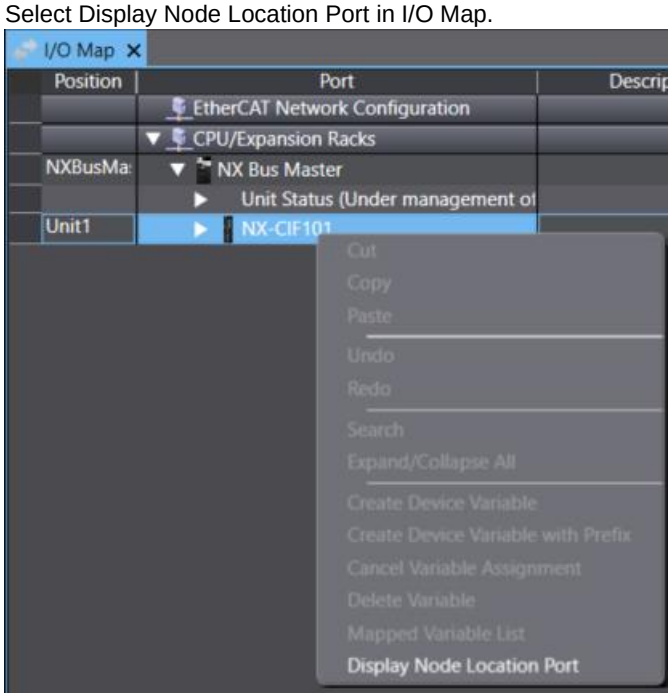
Remember to set BaudRate, DataFormat and other settings for the Serial Port.

1.1. FB Layout

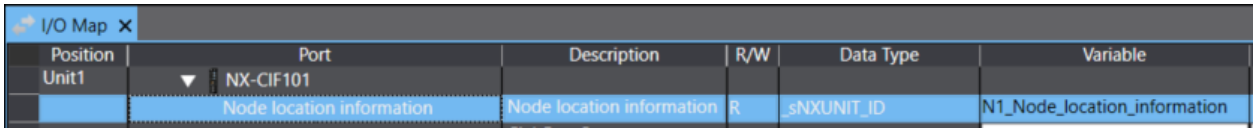


1.1. How to identify the Device Port

Select Display Node Location Port in I/O Map.

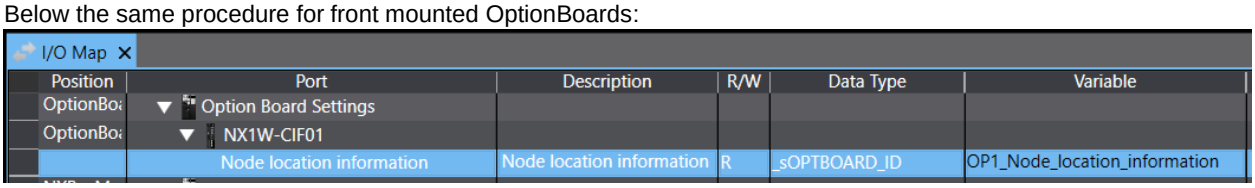


Then add a variable to the Node.



This variable is used to identify the Unit when used in the program. There is no difference if the NX Unit is mounted on NX-ECC coupler.

Below the same procedure for front mounted OptionBoards:

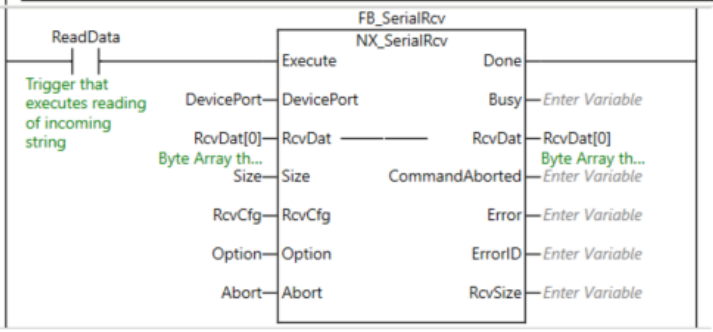



```
//Configure DevicePort variable to point at the correct port of our front mounted OptionBoard
DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceOptionBoard ;  //This is an OptionBoard
DevicePort.NxUnit:=OP1_Node_location_information;              //This is the OptionBoard
DevicePort.PortNo:=1;                                          //This is the port of the OptionBoard
```

1.2. Example

You need to specify the port you want to read from.
You need to specify how to identify the incoming message.
You need to set the Timeout.

```
1  //Configure DevicePort variable to point at the correct port of our CIF 101 NX Unit
2  DevicePort.DeviceType:=_eDEVICE_TYPE#_DeviceNXUnit;  //This is a NX Unit
3  DevicePort.NxUnit:=N1_Node_location_information;      //This is the NX Unit
4  DevicePort.PortNo:=1;                                 //This is the port of the NX Unit
5
6  //Configure how to identify the incoming message
7  RcvCfg.StartTrig:=_SERIAL_START_STARTCODE1;          //The message always starts with a fixed character
8  StringToAry('$$', RcvCfg.StartCode[0]);              //The message starts with $
9  RcvCfg.EndTrig:=_SERIAL_END_ENDCODE2;                //The message always ends with two fixed characters
10 StringToAry('$0D$0A', RcvCfg.EndCode[0]);            //The message ends with <CR><LF>
11 RcvCfg.RcvSizeCfg:=0;                                //Used only when .EndTrig=_SERIAL_END_RCV_SIZE
12
13 //Set Timeout for incoming message
14 Option.TimeOut:=0;                                   //Wait for incoming message indefinitely
15
```



ReadData can be a pulse.
RcvDat contains incoming message. Starting from Byte 0.
Size is the number of bytes in your RcvDat byte Array

When Done, you must trigger ReadData again.

```
//Configure how to identify the incoming message
RcvCfg.StartTrig:=_SERIAL_START_STARTCODE1;       //The message always starts with a fixed character
StringToAry('$$', RcvCfg.StartCode[0]);           //The message starts with $
RcvCfg.EndTrig:=_SERIAL_END_ENDCODE2;             //The message always ends with two fixed characters
StringToAry('$0D$0A', RcvCfg.EndCode[0]);         //The message ends with <CR><LF>
RcvCfg.RcvSizeCfg:=0;                             //Used only when .EndTrig=_SERIAL_END_RCV_SIZE
```

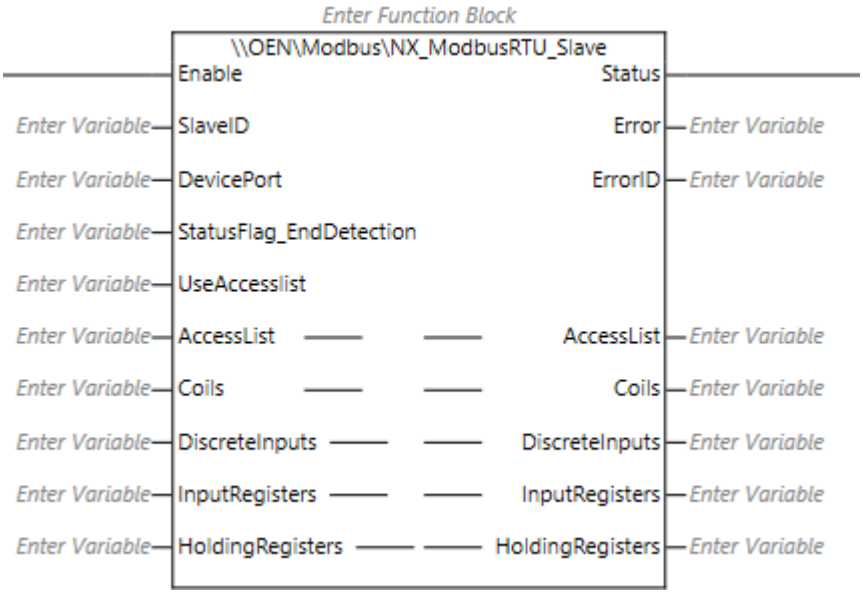
2. NX_ModbusRTU_Slave

Modbus RTU slave that are based on NX_SerialRcv, NX_SerialSend, NX_SerialBufClear function blocks in Sysmac studio.
For description regarding DevicePort input, see the help for the NX_Serial function blocks.
The input StatusFlag_EndDetection are used to check the silence period of 3.5 characters. See "Precautions for correct use".

The slave will respond to any valid Modbus requests.
If the request does try to write to an address set as read only (R), the slave will send a Modbus exception code 02.
If the request does try to read/write to a coil/register outside of the range of your ARRAY, the slave will send a Modbus exception code 02.

Supported Modbus functions codes:
- Fn01 Read Coils
- Fn02 Read discrete inputs
- Fn03 Read holding registers
- Fn04 Read input registers
- Fn05 Write single coil
- Fn06 Write single holding register
- Fn15 Write multiple coils
- Fn16 Write multiple holding registers
- Fn23 Read/Write multiple holding registers

2.1. FB Layout



2.2. Input Variables

Name	Data type	Description
Enable	BOOL	Enable the slave
SlaveID	UINT	Modbus address
DevicePort	_sDEVICE_PORT	Reference to the serial card.
StatusFlag_EndDetection	BOOL	To determine the end of the request from the master.
UseAccesslist	BOOL	FALSE: All registers are RW TRUE: The register access is determined from the accesslist

2.3. In-Out Variables

Name	Data type	Description
Accesslist	sModbusAccess[*] (Dynamic size)	List of address ranges to determine R/W/RW access.
Coils	BOOL[*] (Dynamic size)	ARRAY[10..19] OF BOOL will be modbus address 10 – 19.
DiscreteInputs	WORD[*] (Dynamic size)	ARRAY[10..19] OF WORD will be modbus address 10 – 19.
InputRegisters	WORD[*] (Dynamic size)	ARRAY[10..19] OF WORD will be modbus address 10 – 19.
HoldingRegisters	WORD[*] (Dynamic size)	ARRAY[10..19] OF WORD will be modbus address 10 – 19.

2.4. Output Variables

Name	Data Type	Description
Status	BOOL	True = Activated
Error	BOOL	
ErrorID	WORD	See ErrorID's for NX_SerialRcv, NX_SerialSend, NX_SerialBufClear in the Instructions Reference Manual

2.5. Revisions

Revision	In Library	Correction
1.0.20	1.00.22	

2.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

2.7. Example

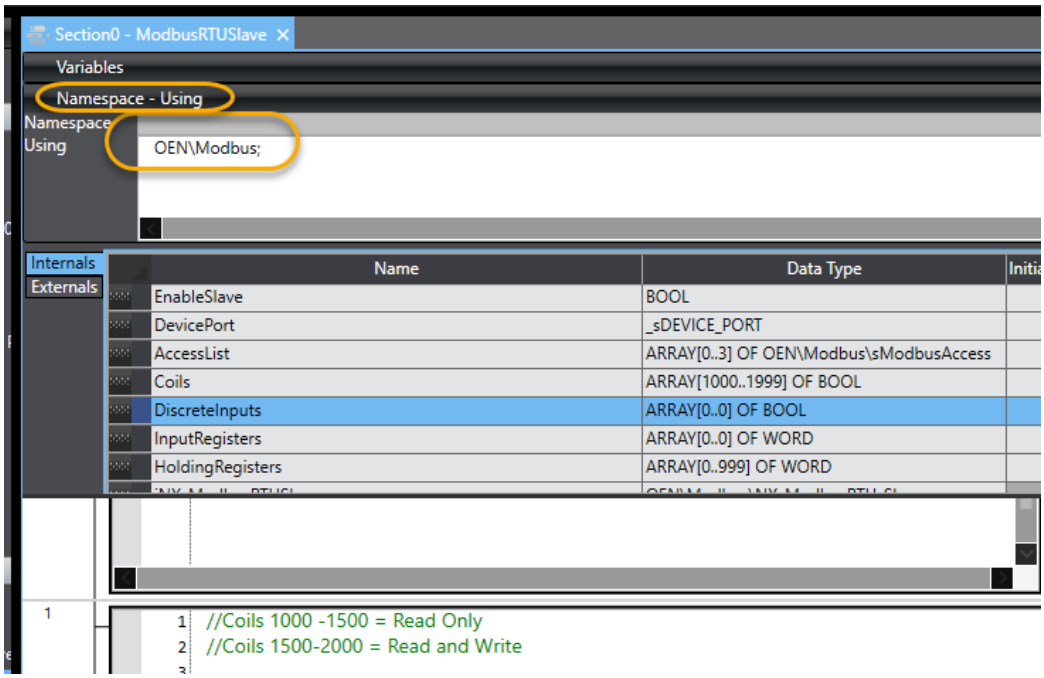
To control the read/write access:

Set the UseAccesslist to “TRUE”.
Create a variable E.G AccessList ARRAY[0..3] OF OEN\Modbus\sModbusAccess
The number of array elements of the In/Out AccessList are dynamic, so you can specify as many as you need.
Sample code for filling data into AccessList:

```
1  //Coils 1000 -1500 = Read Only
2  //Coils 1500-2000 = Read and Write
3
4  AccessList[0].AccessType := eAccess#R;
5  AccessList[0].RegisterType := eRegisterType#Coil;
6  AccessList[0].AddressArea.StartAddress := 1000;
7  AccessList[0].AddressArea.Count := 500;
8
9  AccessList[1].AccessType := eAccess#RW;
10 AccessList[1].RegisterType := eRegisterType#Coil;
11 AccessList[1].AddressArea.StartAddress := 1500;
12 AccessList[1].AddressArea.Count := 500;
```

If you want to grant read and write access to all registers, set the input UseAccesslist to “FALSE”.
Since the In/Out AccessList requires a variable, you can create a variable E.G. AccessList ARRAY[0..0] OF OEN\Modbus\sModbusAccess
The four In/Out Coils, DiscreteInputs, InputRegisters, HoldingRegisters do also require a variable.
If E.G. you don’t want to use DiscreteInputs, create a variable DiscreteInputs ARRAY[0..0] OF BOOL

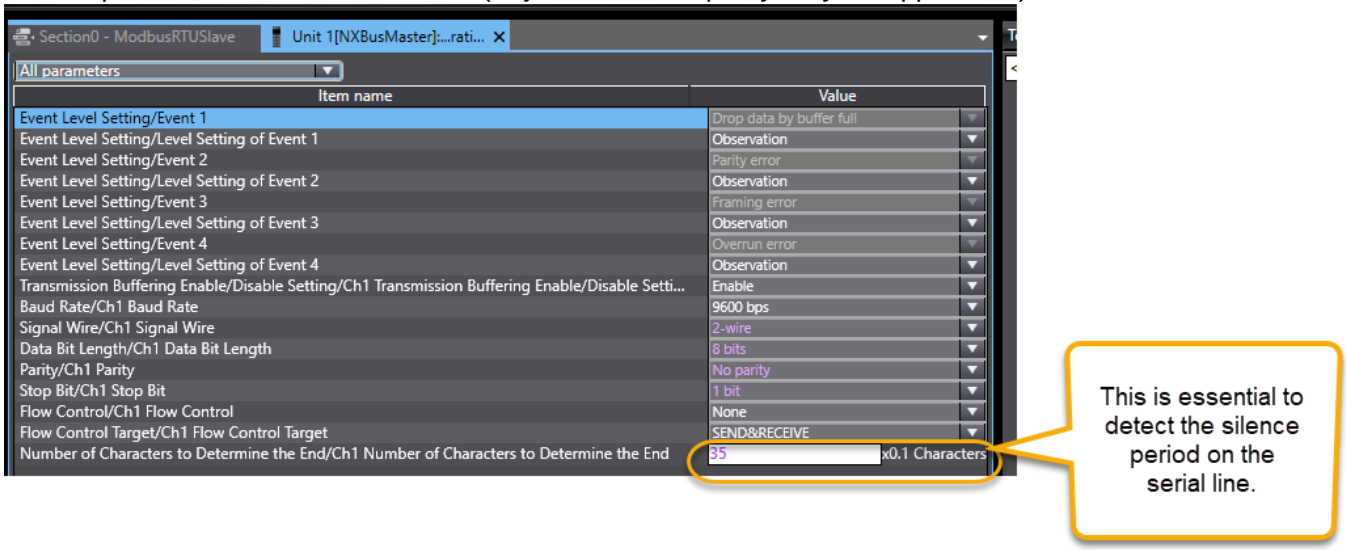
To avoid having to write OEN\Modbus to address the namespace when programming: Add OEN\Modbus in the “Namespace – using”:



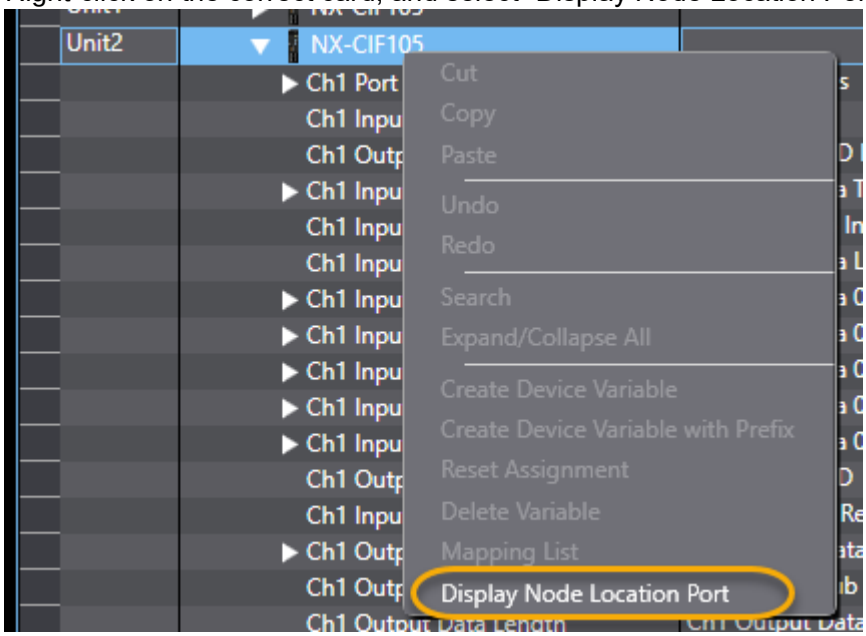
Precautions for correct use

The FB can't be used for serial option boards. (Mounted in the front slot of NX1P2)

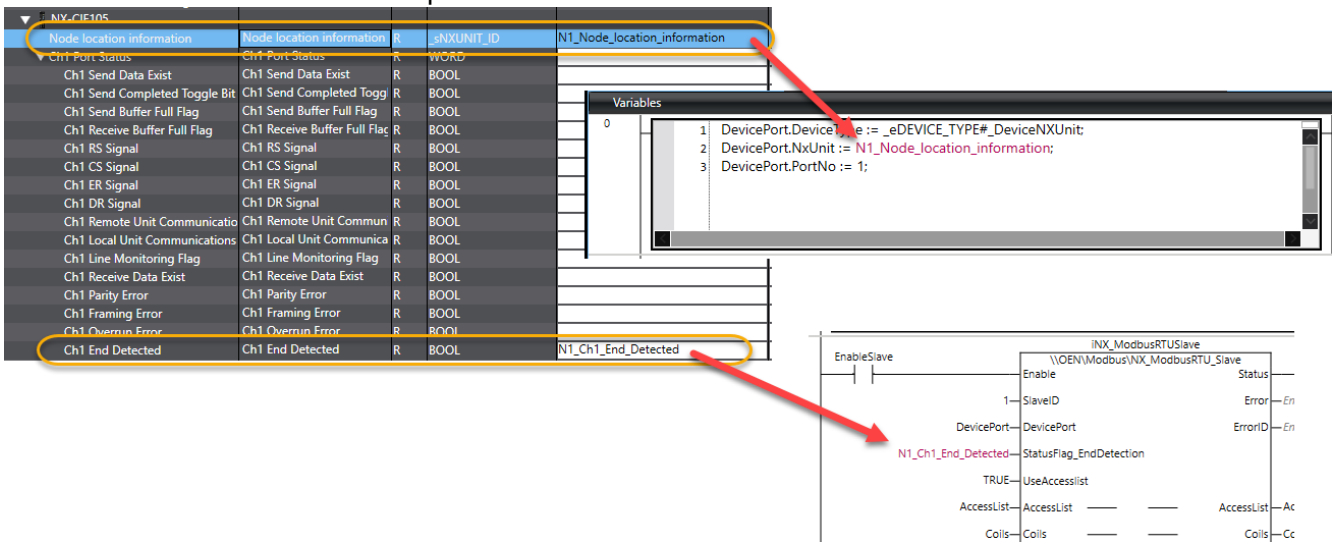
Set the parameters on the serial card: (Adjust Baud rate/parity for your application)



In the I/O Map:
Right-click on the correct card, and select "Display Node Location Port"



You need these two variables to operate the function block.



3. NX_ModbusRTU_Master

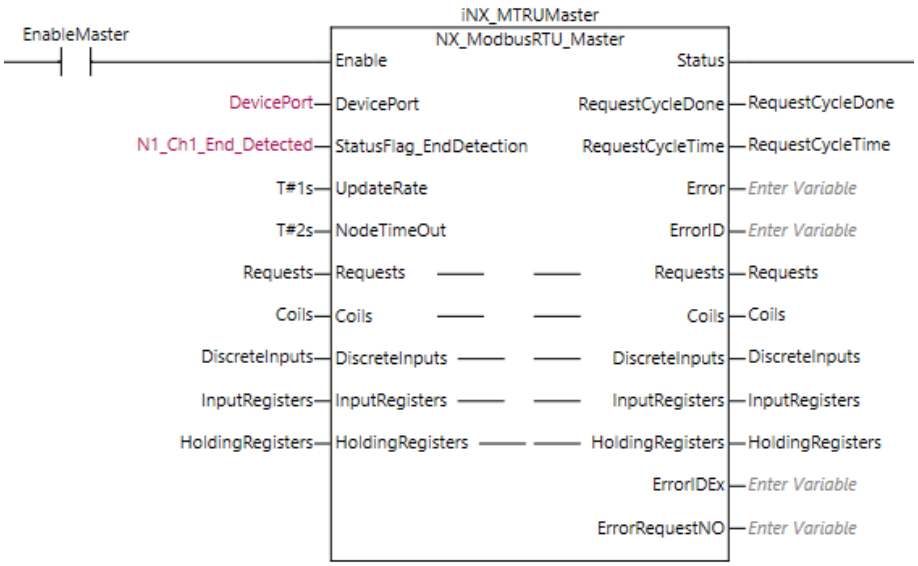
Modbus RTU slave that are based on NX_SerialBufClear, NX_SerialRcv, NX_SerialSend function blocks in Sysmac Studio.
For description regarding DevicePort input, see the help for the NX_Serial function blocks.
The input StatusFlag_EndDetection are used to check the silence period of 3.5 characters. See "Precautions for correct use".

The master will sequentially perform the requests when the member.Enable is set to TRUE.
How often the requests are performed are controlled by the Input "UpdateRate".
If one of the requests encounters an error, the error will be set to TRUE, and the value of the Array index for the request with errors on the Output "ErrorRequestNo".

Supported Modbus functions codes:

- Fn01 Read Coils
- Fn02 Read discrete inputs
- Fn03 Read holding registers
- Fn04 Read input registers
- Fn05 Write single coil
- Fn06 Write single holding register
- Fn15 Write multiple coils
- Fn16 Write multiple holding registers
- Fn23 Read/Write multiple holding registers

3.1. FB Layout



3.2. Input Variables

Name	Data type	Description
Enable	BOOL	Enable the slave
DevicePort	BOOL	Reference to the serial card.
SlaveID	_sDEVICE_PORT	Modbus address
StatusFlag_EndDetection	UINT	To determine the end of the response from the slave(s).
UpdateRate	TIME	Time between each poll of the request list.
NodeTimeOut	TIME	Timeout for each request.
Requests	TIME	List of requests to the slave(s)

3.3. In-Out Variables

Name	Data type	Description
Coils	BOOL[*] (Dynamic size)	ARRAY[10..19] OF BOOL will be modbus address 10 – 19.
DiscreteInputs	WORD[*] (Dynamic size)	ARRAY[10..19] OF WORD will be modbus address 10 – 19.
InputRegisters	WORD[*] (Dynamic size)	ARRAY[10..19] OF WORD will be modbus address 10 – 19.
HoldingRegisters	WORD[*] (Dynamic size)	ARRAY[10..19] OF WORD will be modbus address 10 – 19.

3.4. Output Variables

Name	Data Type	Description
Status	BOOL	True = Activated
RequestCycleDone	BOOL	True for one cycle, when polling the request list is completed.
RequestCycleTime	TIME	The time used for polling the request list.
Error	BOOL	
ErrorID	WORD	If Error ID = 16#0C10, you will find a modbus exception code in ErrorIDEx
ErrorIDEx	DWORD	Modbus exception code
ErrorRequestNo	DINT	The array index of the request that got an error.
ModbusError	STRING[50]	Modbus error as text message

3.5. Revisions

Revision	In Library	Correction
1.0.22	1.00.22	Separated Modbus addresses into local and remote. So that one master can be used for many similar slaves.
1.1.00	1.03.00	Added ModbusError output to show Modbus error message

3.6. Credits

	Name
Omron - Norway	Bjarte Myklebust
Omron - Norway	Kjell Baardsgaard

3.7. Example

How to set up the requests:

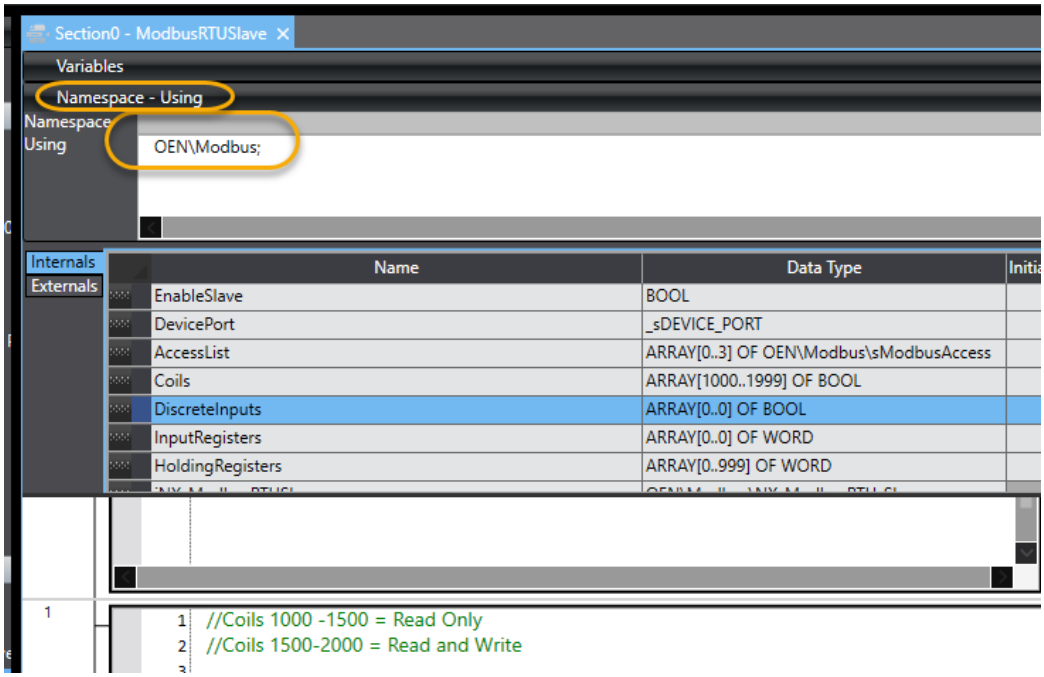
Create a variable for requests E.G. “Requests”, of the datatype ARRAY[X..Y] OF OEN\Modbus\sModbusReq.
The number of array elements of the In/Out “Requests” are dynamic, so you can spesify as many as you need.

Sample code for filling data into the “Request” variable:

```
//Read coils 1000-1009
Requests[0].Enable := TRUE;
Requests[0].FunctionCode := eFun#Fn01_ReadCoils;
Requests[0].NodeAdr := 1;
Requests[0].Read.StartAddressRemote := 1000;    //Address in the slave/server
Requests[0].Read.StartAddressLocal := 1000;   //Address in the master/client
Requests[0].Read.Count := 10;

//Read Holding registers 10-19
Requests[1].Enable := TRUE;
Requests[1].FunctionCode := eFun#Fn03_ReadHoldingRegisters;
Requests[1].NodeAdr := 1;
Requests[1].Read.StartAddressRemote := 10;   //Address in the slave/server
Requests[1].Read.StartAddressLocal := 10;       //Address in the master/client
Requests[1].Read.Count := 10;

//Writing Holding registers 20-29
Requests[2].Enable := TRUE;
Requests[2].FunctionCode := eFun#Fn05_WriteSingleCoil;
Requests[2].NodeAdr := 1;
Requests[2].Write.StartAddressRemote := 20;   //Address in the slave/server
Requests[2].Write.StartAddressLocal := 20;       //Address in the master/client
Requests[2].Write.Count := 10;
```

The four In/Out Coils, DiscreteInputs, InputRegisters, HoldingRegisters do also require a variable.
If E.G. you don’t want to use DiscreteInputs, create a variable DiscreteInputs ARRAY[0..0] OF BOOL

To avoid having to write OEN\Modbus to address the namespace when programming: Add OEN\Modbus in the “Namespace – using”:



Errors

The list of ErrorID’s are found in the “instructions reference manual” for the controller.
See the ErrorID’s for NX_SerialRcv, NX_SerialSend, NX_SerialBufClear.

If the “ErrorID” = 16#0C10 then the modbus exception code will be found in “ErrorIDEx”.

List of ErrorID’s in addition to the above:

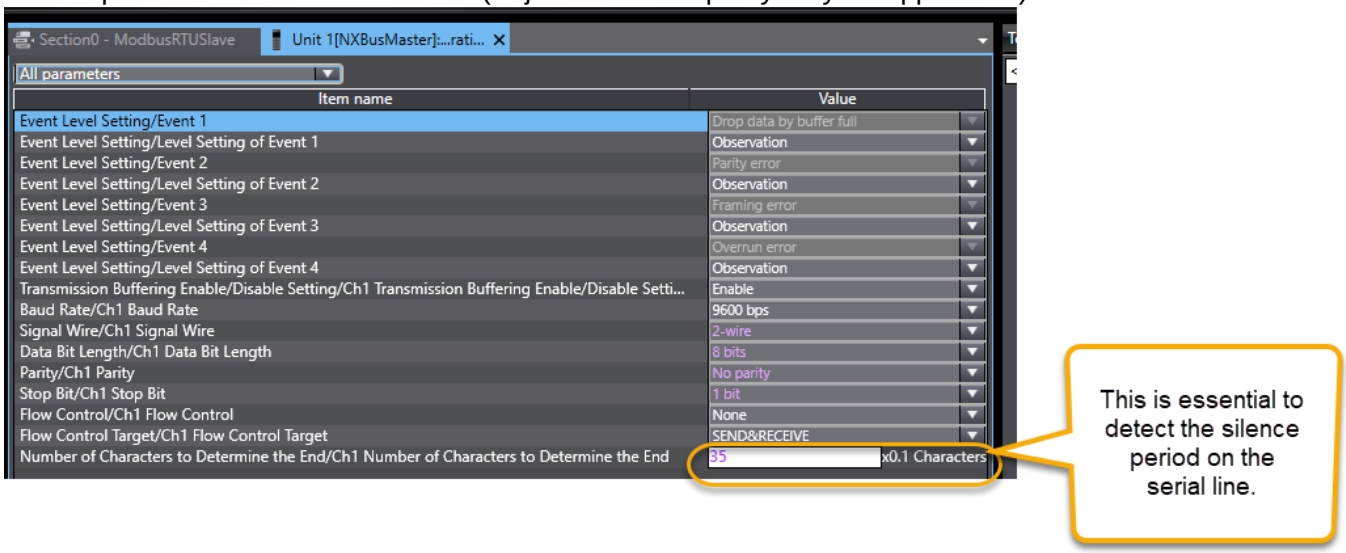
16#1001 Modbus address outside of Array boundary
16#1002 Invalid modbus function code
16#1004 Response with wrong function code
16#1005 Response with wrong size
16#1006 Wrong CRC

The output “ErrorRequestNo” will contain the value of the ARRAY index of the request that failed.
This value can only be trusted on the rising edge of the output “Error”.

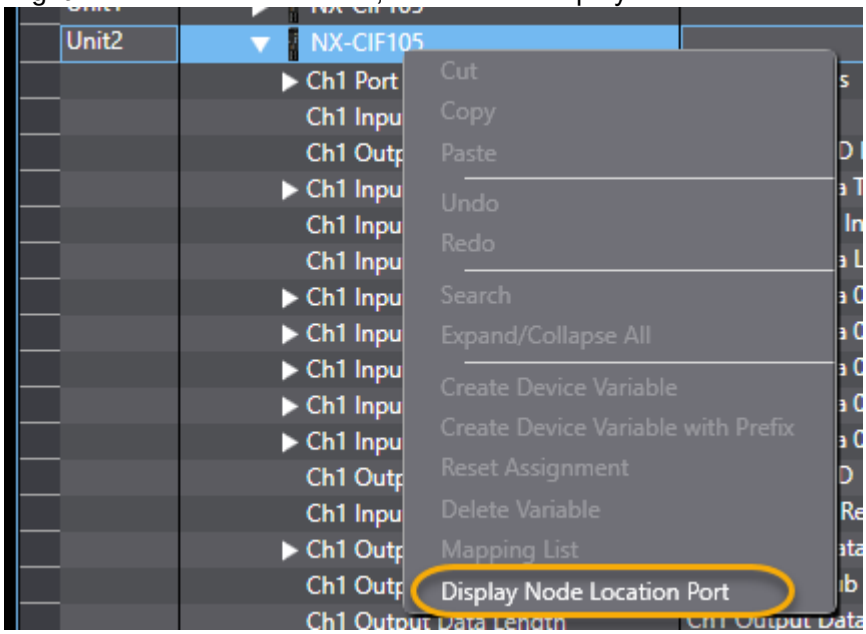
Precautions for correct use

The FB can't be used for serial option boards. (Mounted in the slot at front of NX1P)

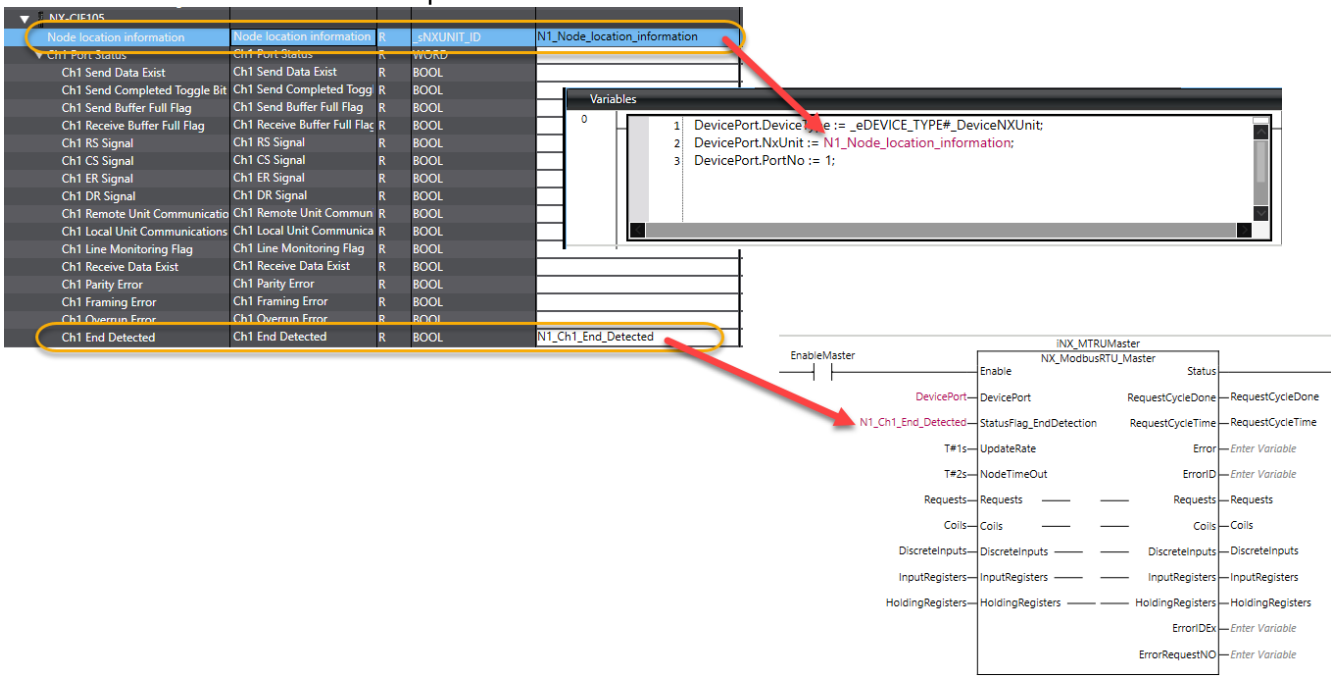
Set the parameters on the serial card: (Adjust Baud rate/parity for your application)



In the I/O Map:
Right-click on the correct card, and select "Display Node Location Port"



You need these two variables to operate the function block.



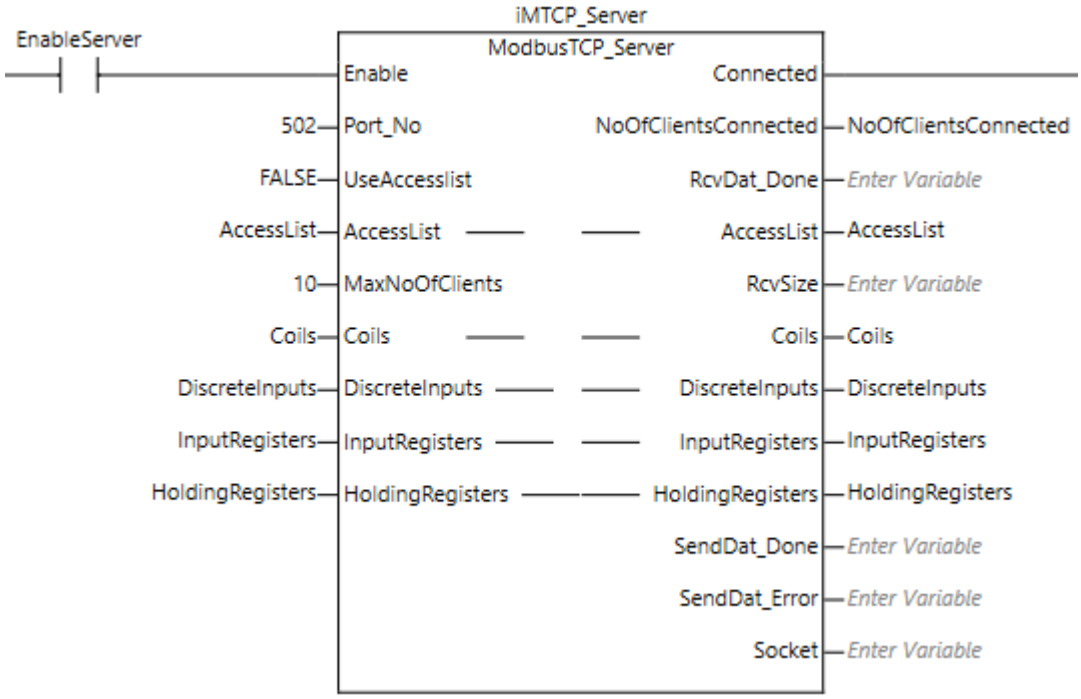
4. ModbusTCP_Server

Modbus TCP Server are based on TCP socket FB's: SktTCPAccept, SktGetTCPStatus, SktTCPRcv, SktTCPSend, SktTCPClose.

The server will respond to any valid modbus requests.
If the request does try to write to an address set as read only (R), the server will send a modbus exception code 02.
If the request does try to read/write to a coil/register outside of the range of your ARRAY, the server will send a modbus exception code 02.
If the Client requests a function code that the server does not support, the modbus exception code 01 will be sent.

Supported modbus functions codes:
- Fn01 Read Coils
- Fn02 Read discrete inputs
- Fn03 Read holding registers
- Fn04 Read input registers
- Fn05 Write single coil
- Fn06 Write single holding register
- Fn15 Write multiple coils
- Fn16 Write multiple holding registers
- Fn23 Read/Write multiple holding registers

4.1. FB Layout



4.2. Input Variables

Name	Data type	Valid Range	Description
Enable	BOOL		Enable the slave
Port_No	UINT		TCP port for the server.
UseAccesslist	BOOL		FALSE: All registers are RW TRUE: The register access is determined from the accesslist
MaxNoOfClients	UINT	1-10	To limit the number of clients.

4.3. In-Out Variables

Name	Data type	Description
Accesslist	sModbusAccess[*] (Dynamic size)	List of address ranges to determine R/W/RW access.
Coils	BOOL[*] (Dynamic size)	ARRAY[10..19] OF BOOL becomes modbus address 10 – 19.
DiscreteInputs	WORD[*] (Dynamic size)	ARRAY[10..19] OF WORD becomes modbus address 10 – 19.
InputRegisters	WORD[*] (Dynamic size)	ARRAY[10..19] OF WORD becomes modbus address 10 – 19.
HoldingRegisters	WORD[*] (Dynamic size)	ARRAY[10..19] OF WORD becomes modbus address 10 – 19.

4.4. Output Variables

Name	Data Type	Description
Connected	BOOL	At least one client is connected.
NoOfClientsConnected	UINT	Number of connected clients
RcvDat_Done	BOOL[0..9]	True when data received from the client.
RcvSize	UINT[0..9]	The size of data received from the client.
SendDat_Done	BOOL[0..9]	True when the server sent response to the client.
SendDat_Error	BOOL[0..9]	True when the server failed to send response to the client.
Socket	sSOCKET[0..9]	Socket details for each client.

4.5. Revisions

Revision	In Library	Correction
1.0.20	1.00.22	

4.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

4.7. Example

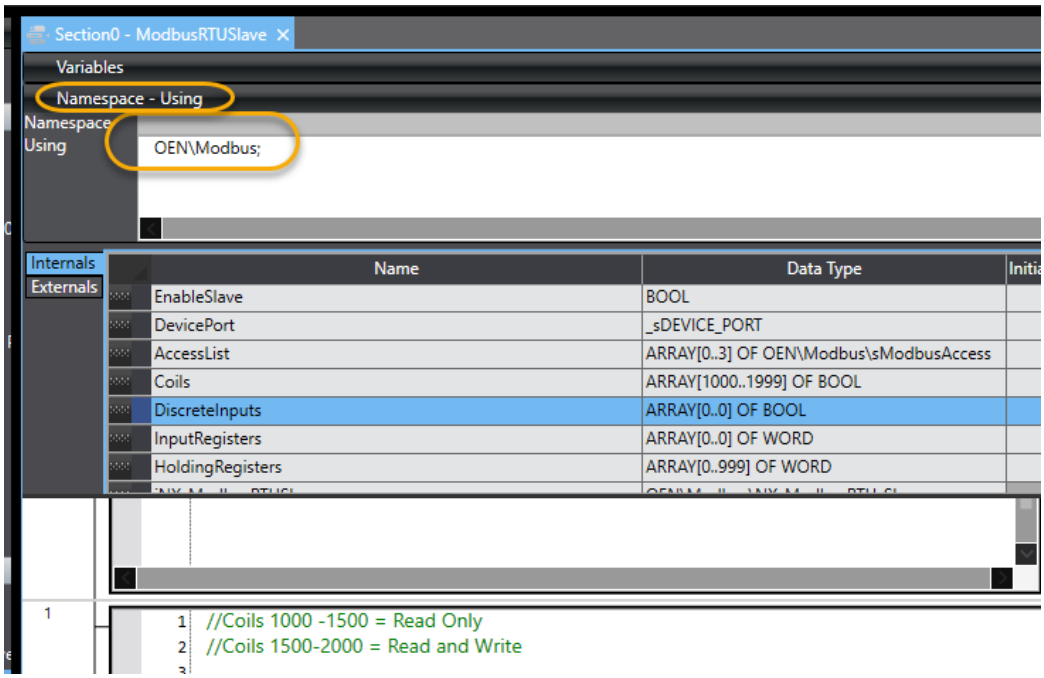
To control the read/write access :

Set the UseAccesslist to “TRUE”.
Create a variable E.G AccessList ARRAY[0..3] OF OEN\Modbus\sModbusAccess
The number of array elements of the In/Out AccessList are dynamic, so you can specify as many as you need.
Sample code for filling data into the accesslist:

```
1  //Coils 1000 -1500 = Read Only
2  //Coils 1500-2000 = Read and Write
3
4  AccessList[0].AccessType := eAccess#R;
5  AccessList[0].RegisterType := eRegisterType#Coil;
6  AccessList[0].AddressArea.StartAddress := 1000;
7  AccessList[0].AddressArea.Count := 500;
8
9  AccessList[1].AccessType := eAccess#RW;
10 AccessList[1].RegisterType := eRegisterType#Coil;
11 AccessList[1].AddressArea.StartAddress := 1500;
12 AccessList[1].AddressArea.Count := 500;
```

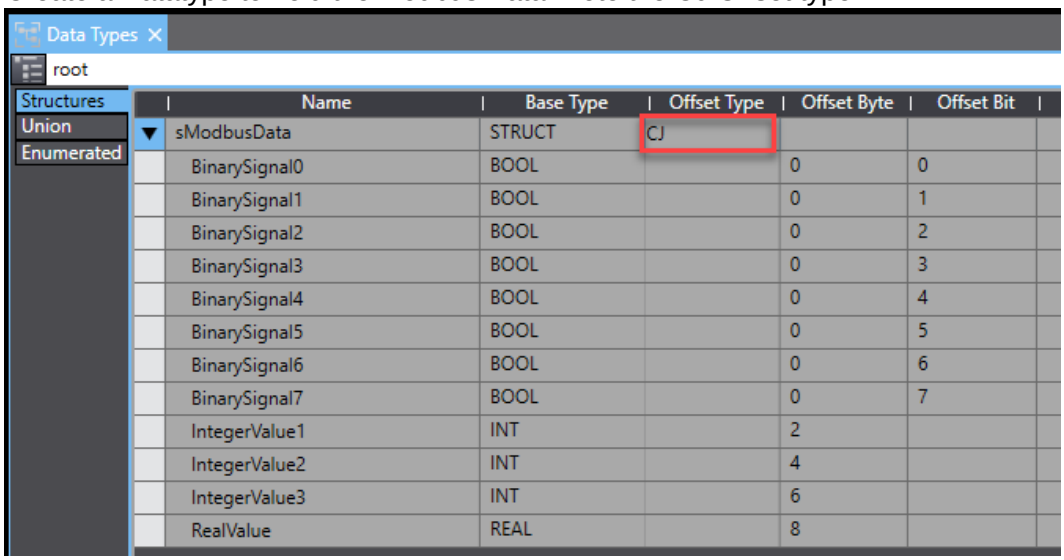
If you want to grant read and write access to all registers, set the input UseAccesslist to “FALSE”.
Since the In/Out AccessList requires a variable, you can create a variable E.G. AccessList ARRAY[0..0] OF OEN\Modbus\sModbusAccess
The four In/Out Coils, DiscreteInputs, InputRegisters, HoldingRegisters do also require a variable.
If E.G. you don’t want to use DiscreteInputs, create a variable DiscreteInputs ARRAY[0..0] OF BOOL

To avoid having to write OEN\Modbus to address the namespace when programming: Add OEN\Modbus in the “Namespace – using”:



How to Map a Structured variable into Holding Registers

Create a Datatype to hold the Modbus Data. Note the CJ Offset type.



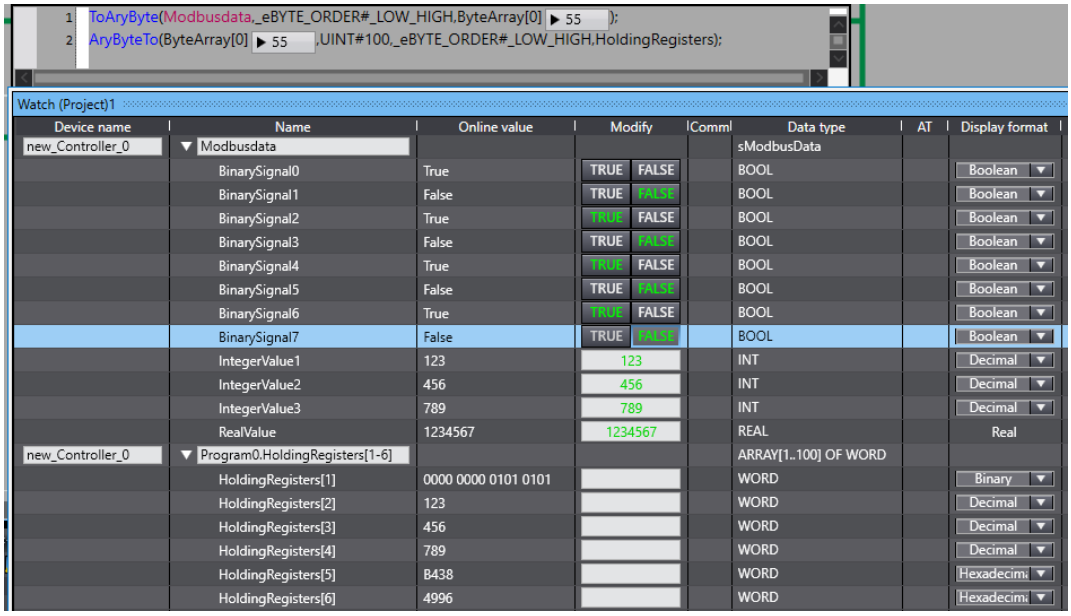
Data Types

root

Structures | Union | Enumerated

Name	Base Type	Offset Type	Offset Byte	Offset Bit
sModbusData	STRUCT	CJ		
BinarySignal0	BOOL		0	0
BinarySignal1	BOOL		0	1
BinarySignal2	BOOL		0	2
BinarySignal3	BOOL		0	3
BinarySignal4	BOOL		0	4
BinarySignal5	BOOL		0	5
BinarySignal6	BOOL		0	6
BinarySignal7	BOOL		0	7
IntegerValue1	INT		2	
IntegerValue2	INT		4	
IntegerValue3	INT		6	
RealValue	REAL		8	

Add the following 2 commands into the program to convert the structure into an array of WORD. Monitor Global variable and the corresponding Modbus HoldingRegisters. Note the Display Format for HoldingRegisters and the Hex values that corresponds to the REAL value.



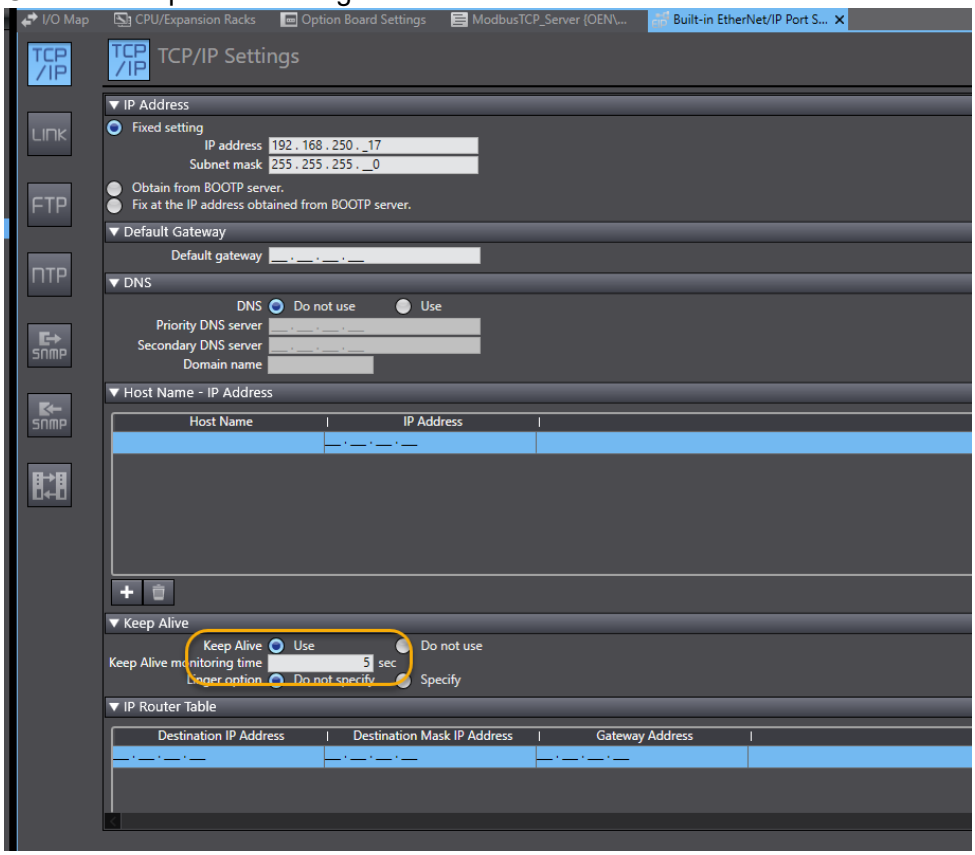
```
1  ToAryByte(Modbusdata,_eBYTE_ORDER#_LOW_HIGH,ByteArray[0] ▶ 55 );
2  AryByteTo(ByteArray[0] ▶ 55 ,UINT#100,_eBYTE_ORDER#_LOW_HIGH,HoldingRegisters);
```

Watch (Project)1

Device name	Name	Online value	Modify	Comm	Data type	AT	Display format
new_Controller_0	Modbusdata				sModbusData		
	BinarySignal0	True	TRUE FALSE		BOOL		Boolean
	BinarySignal1	False	TRUE FALSE		BOOL		Boolean
	BinarySignal2	True	TRUE FALSE		BOOL		Boolean
	BinarySignal3	False	TRUE FALSE		BOOL		Boolean
	BinarySignal4	True	TRUE FALSE		BOOL		Boolean
	BinarySignal5	False	TRUE FALSE		BOOL		Boolean
	BinarySignal6	True	TRUE FALSE		BOOL		Boolean
	BinarySignal7	False	TRUE FALSE		BOOL		Boolean
	IntegerValue1	123	123		INT		Decimal
	IntegerValue2	456	456		INT		Decimal
	IntegerValue3	789	789		INT		Decimal
	RealValue	1234567	1234567		REAL		Real
new_Controller_0	Program0.HoldingRegisters[1-6]				ARRAY[1..100] OF WORD		
	HoldingRegisters[1]	0000 0000 0101 0101			WORD		Binary
	HoldingRegisters[2]	123			WORD		Decimal
	HoldingRegisters[3]	456			WORD		Decimal
	HoldingRegisters[4]	789			WORD		Decimal
	HoldingRegisters[5]	B438			WORD		Hexadecim
	HoldingRegisters[6]	4996			WORD		Hexadecim

Precautions for correct use

Use the “Keep Alive” settings for the Ethernet card:



When using “Keep alive” the TCP socket will send a keep alive message to the client, to check if the client is still responding. If the client does not respond within the “Keep Alive monitoring time” the socket will close, and reopen for new connection for the client.

This will also prevent that one client occupies several sockets/connections on the server.

5. *ModbusTCP_Client*

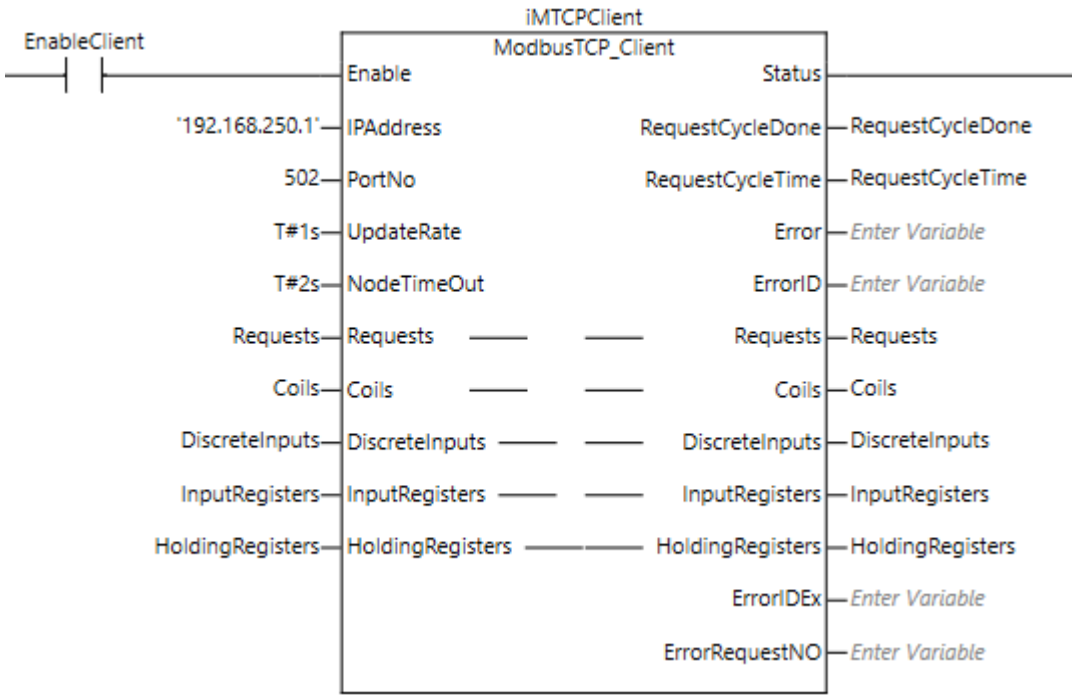
Modbus TCP Client that are based on TCP socket FB's: SktTCPConnect, SktGetTCPStatus, SktTCPRcv, SktTCPSend, SktTCPClose.

The Client will sequentially perform the requests with the member .Enable set to true.
How often the requests are performed are controlled by the Input "UpdateRate".
If one of the requests encounters an error, the error will be set to true, and the value of the Array index for the request with errors on the Output "ErrorRequestNo".
For error codes see the section "Errors".

Supported modbus functions codes:

Fn01 Read Coils
Fn02 Read discrete inputs
Fn03 Read holding registers
Fn04 Read input registers
Fn05 Write single coil
Fn06 Write single holding register
Fn15 Write multiple coils
Fn16 Write multiple holding registers
Fn23 Read/Write multiple holding registers

5.1. FB Layout



5.2. Input Variables

Name	Data type	Description
Enable	BOOL	Enable the slave
IPAddress	STRING[20]	The IP address of the server
Port_No	UINT	The TCP port number of the server.
UpdateRate	TIME	Time between each poll of the request list.
NodeTimeOut	TIME	Timeout for each request.
Requests	BOOL	List of requests to the slave(s)

5.3. In-Out Variables

Name	Data type	Description
Coils	BOOL[*] (Dynamic size)	ARRAY[10..19] OF BOOL will be modbus address 10 – 19.
DiscreteInputs	WORD[*] (Dynamic size)	ARRAY[10..19] OF WORD will be modbus address 10 – 19.
InputRegisters	WORD[*] (Dynamic size)	ARRAY[10..19] OF WORD will be modbus address 10 – 19.
HoldingRegisters	WORD[*] (Dynamic size)	ARRAY[10..19] OF WORD will be modbus address 10 – 19.

5.4. Output Variables

Name	Data Type	Description
Status	BOOL	True = Activated
RequestCycleDone	BOOL	True for one cycle, when polling the request list is completed.
RequestCycleTime	TIME	The time used for polling the request list.
Error	BOOL	
ErrorID	WORD	If Error ID = 16#0C10, you will find a modbus exception code in ErrorIDEx
ErrorIDEx	DWORD	Modbus exception code
ErrorRequestNo	DINT	The array index of the request that got an error.
ModbusError	STRING[50]	Modbus error as text message

5.5. Revisions

Revision	In Library	Correction
1.0.22	1.00.22	Separated modbus addresses into local and remote. So that one Client can be used for many similar servers.
1.1.00	1.03.00	Added ModbusError output to show Modbus error message

5.6. Credits

	Name
Omron - Norway	Bjarte Myklebust
Omron - Norway	Kjell Baardsgaard

5.7. Example

How to set up the requests :

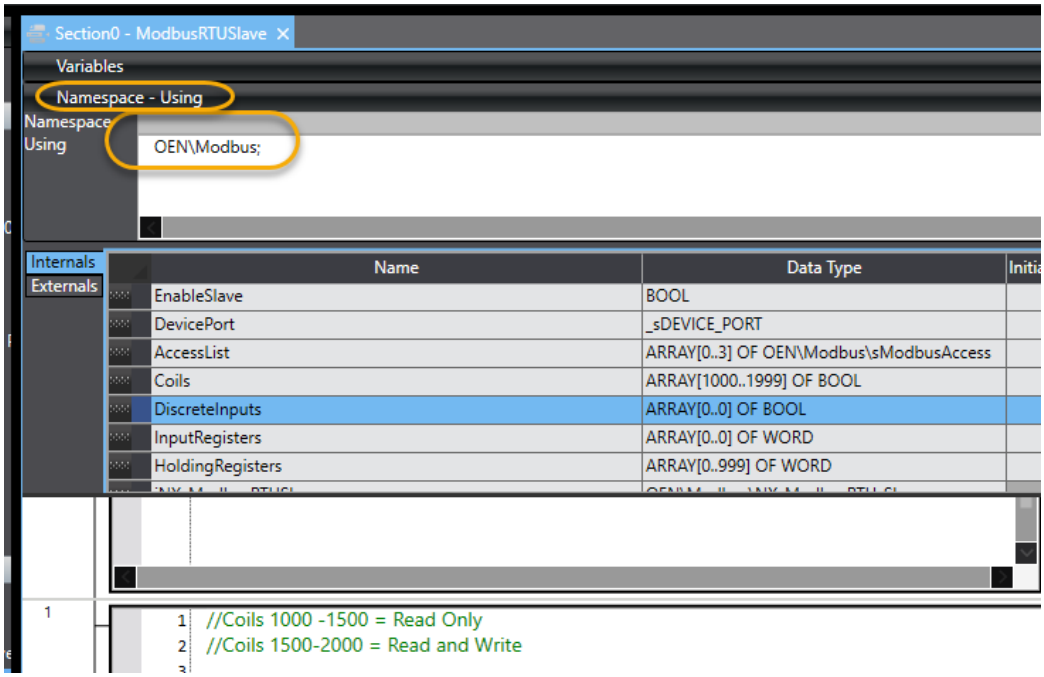
Create a variable for requests E.G. “Requests”, of the datatype ARRAY[X..Y] OF OEN\Modbus\sModbusReq.
The number of array elements of the In/Out “Requests” are dynamic, so you can spesify as many as you need.

Sample code for filling data into the “Request” variable:

```
//Read coils 1000-1009
Requests[0].Enable := TRUE;
Requests[0].FunctionCode := eFun#Fn01_ReadCoils;
Requests[0].NodeAdr := 1;
Requests[0].Read.StartAddressRemote := 1000;    //Address in the slave/server
Requests[0].Read.StartAddressLocal := 1000;   //Address in the master/client
Requests[0].Read.Count := 10;

//Read Holding registers 10-19
Requests[1].Enable := TRUE;
Requests[1].FunctionCode := eFun#Fn03_ReadHoldingRegisters;
Requests[1].NodeAdr := 1;
Requests[1].Read.StartAddressRemote := 10;   //Address in the slave/server
Requests[1].Read.StartAddressLocal := 10;       //Address in the master/client
Requests[1].Read.Count := 10;

//Writing Holding registers 20-29
Requests[2].Enable := TRUE;
Requests[2].FunctionCode := eFun#Fn05_WriteSingleCoil;
Requests[2].NodeAdr := 1;
Requests[2].Write.StartAddressRemote := 20;   //Address in the slave/server
Requests[2].Write.StartAddressLocal := 20;       //Address in the master/client
Requests[2].Write.Count := 10;
```

The four In/Out Coils, DiscreteInputs, InputRegisters, HoldingRegisters do also require a variable.
If E.G. you don’t want to use DiscreteInputs, create a variable DiscreteInputs ARRAY[0..0] OF BOOL

To avoid having to write OEN\Modbus to address the namespace when programming: Add OEN\Modbus in the “Namespace – using”:



Errors

The list of ErrorID’s are found in the “instructions reference manual” for the controller.
See the ErrorID’s for TCP socket FB’s: SktTCPConnect, SktGetTCPStatus, SktTCPRcv, SktTCPSend, SktTCPClose.

If the “ErrorID” = 16#0C10 then the modbus exception code will be found in “ErrorIDEx”.

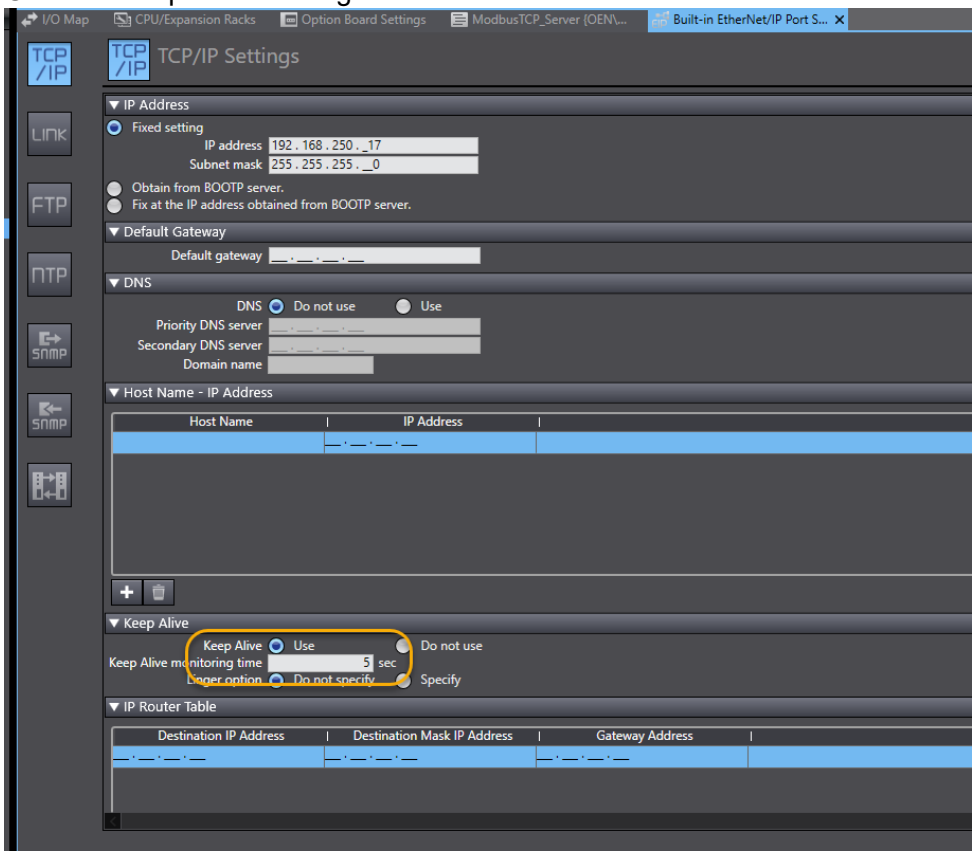
List of ErrorID’s in addition to the above:

16#1001 Modbus address outside of Array boundary
16#1002 Invalid modbus function code
16#1004 Response with wrong function code
16#1005 Response with wrong size
16#1006 Wrong CRC
16#1007 Mismatch of TransactionID between request and response
16#1008 Mismatch of SlaveID between request and response
16#1009 Mismatch of function code between request and response
16#1010 The expected byte size in response is wrong
16#1011 Mismatch of address between request and response
16#1012 Too many bytes in packet. (Byte size is > 2000 bytes)
16#1013 Unknown function code

The output “ErrorRequestNo” will contain the value of the ARRAY index of the request that failed. This value can only be trusted on the rising edge of the output “Error”.

Precautions for correct use

Use the “Keep Alive” settings for the Ethernet card:



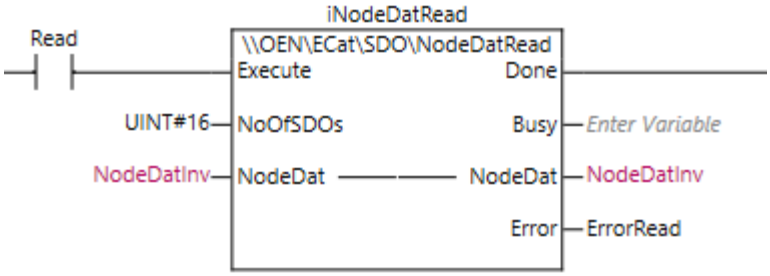
When using “Keep alive” the TCP socket will send a keep alive message to the client, to check if the client is still responding. If the client does not respond within the “Keep Alive monitoring time” the socket will close, and reopen for new connection for the client.

This will also prevent that one client occupies several sockets/connections on the server.

6. NodeDatRead

Reads all the information that are prepared in the NodeDat structure sequentially.
When all nodes with (InUse = TRUE), and all SDO's on each node (InUse = TRUE), the Busy goes to FALSE, and Done = TRUE if successful, or Error = TRUE if not successful completion.
The function block will check the _EC_MBXSlavTbl[NodeAdr] before reading SDO's from the node.
If there is an error in one or more nodes, the FB will stop with an error, and will not proceed.

6.1. FB Layout



6.2. Input Variables

Name	Data type	Valid Range	Description
Execute	BOOL		Start reading on rising edge.
NoOfSDOs	UINT	1-40	Put in the highest number of SDOs in use, to eliminate unnecessary looping.

6.3. In-Out Variables

Name	Data type	Description
NodeDat	OEN\ECat\SDO\sNodeDat[*] (Dynamic size)	One array index for each node. The index does not reflect the node address.

6.4. Output Variables

Name	Data Type	Description
Done	BOOL	TRUE at least one cycle after successfully completion. Or as long as Execute is TRUE. According to PLC Open standard
Busy	BOOL	TRUE while busy with reading.
Error	BOOL	TRUE at least one cycle after successfully completion. Or as long as Execute is TRUE. According to PLC Open standard

6.5. Revisions

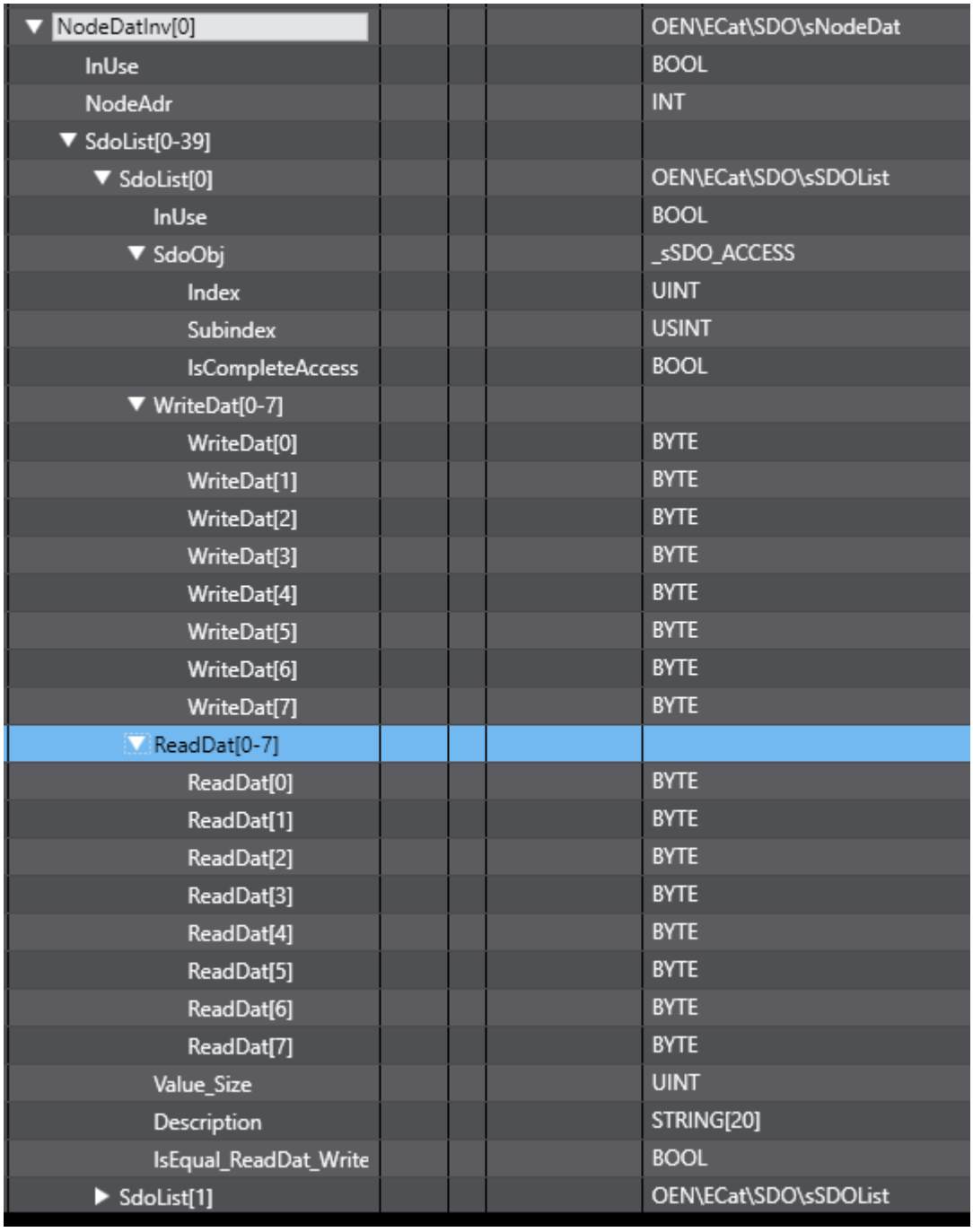
Revision	In Library	Correction
1.0.1	1.00.22	Replaced NodeDat with dynamic ARRAY.Removed Input NoOfNodes

6.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

6.7. Example

Picture of OEN\ECat\SDO\sNodeDat:



Example, set up NodeDat for MX2 inverter:

```
i := 0;
NodeDat[i].NodeAdr := 4;
NodeDat[i].InUse := TRUE;

j := 0;
NodeDat[i].SdoList[j].Description := 'A001 Freq. Ref. Sel';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3012;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#26;
Value_WORD := 16#4;
ToAryByte(In := Value_WORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 1;
NodeDat[i].SdoList[j].Description := 'A002 Run Cmd Sel';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3012;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#27;
Value_WORD := 16#4;
ToAryByte(In := Value_WORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 2;
NodeDat[i].SdoList[j].Description := 'A044 Control Method';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3012;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#63;
Value_WORD := 16#3;
ToAryByte(In := Value_WORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 3;
NodeDat[i].SdoList[j].Description := 'A131 Acc Curve';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3012;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#CA;
Value_WORD := 16#1;
ToAryByte(In := Value_WORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 4;
NodeDat[i].SdoList[j].Description := 'A132 Dec Curve';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3012;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#CB;
Value_WORD := 16#1;
ToAryByte(In := Value_WORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 5;
NodeDat[i].SdoList[j].Description := 'F002 Acc time';
NodeDat[i].SdoList[j].SdoObj.Index := 16#4011;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#26;
Value_DWORD := 400;
ToAryByte(In := Value_DWORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 4;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 6;
NodeDat[i].SdoList[j].Description := 'F003 Dec time';
NodeDat[i].SdoList[j].SdoObj.Index := 16#4011;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#28;
Value_DWORD := 500;
ToAryByte(In := Value_DWORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 4;
NodeDat[i].SdoList[j].InUse := TRUE;
```

```
j := 7;
NodeDat[i].SdoList[j].Description := 'H030 R1';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3015;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#50;
Value_DWORD := 22387;(* 22,387 *)
ToAryByte(In := Value_DWORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 8;
NodeDat[i].SdoList[j].Description := 'H031 R2';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3015;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#52;
Value_DWORD := 8192; (* 8,192 *)
ToAryByte(In := Value_DWORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 9;
NodeDat[i].SdoList[j].Description := 'H032 L';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3015;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#54;
Value_DWORD := 15185; (* 151,85 *)
ToAryByte(In := Value_DWORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 10;
NodeDat[i].SdoList[j].Description := 'B041 Torque Limit 1';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3013;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#52;
Value_WORD := 60; (* 60% *)
ToAryByte(In := Value_WORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 11;
NodeDat[i].SdoList[j].Description := 'B042 Torque Limit 2';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3013;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#53;
Value_WORD := 60; (* 60% *)
ToAryByte(In := Value_WORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 12;
NodeDat[i].SdoList[j].Description := 'B043 Torque Limit 3';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3013;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#54;
Value_WORD := 60; (* 60% *)
ToAryByte(In := Value_WORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 13;
NodeDat[i].SdoList[j].Description := 'B044 Torque Limit 4';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3013;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#55;
Value_WORD := 60; (* 60% *)
ToAryByte(In := Value_WORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;

j := 14;
NodeDat[i].SdoList[j].Description := 'B083 Carrier Freq';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3013;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#7D;
Value_WORD := 110; (* 11 kHz *)
ToAryByte(In := Value_WORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
```

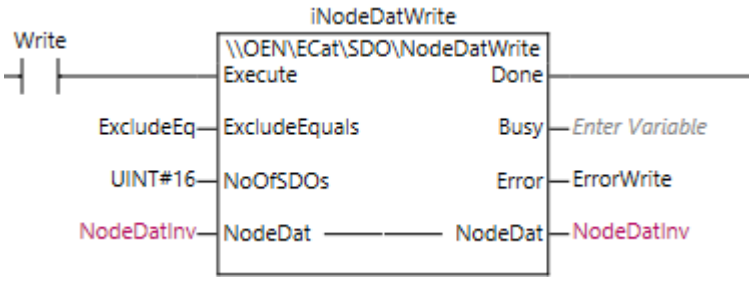
```
NodeDat[i].SdoList[j].InUse := TRUE;

j := 15;
NodeDat[i].SdoList[j].Description := 'H002 Motor par Auto';
NodeDat[i].SdoList[j].SdoObj.Index := 16#3015;
NodeDat[i].SdoList[j].SdoObj.Subindex := 16#2D;
Value_WORD := 2; (* Autotuned motorparameters *)
ToAryByte(In := Value_WORD, AryOut := NodeDat[i].SdoList[j].WriteDat[0]);
NodeDat[i].SdoList[j].Value_Size := 2;
NodeDat[i].SdoList[j].InUse := TRUE;
```

7. NodeDatWrite

Writes all the information that are prepared in the NodeDat structure sequentially.
If the Input ExcludeEquals = TRUE, the member IsEqual_ReadDat_WriteDat will determine if the SDO is written or not. Use the function OEN\ECAT\SDO\NodeDatCmp_Read_Write to set IsEqual_ReadDat_WriteDat for all the SDO's.
When all nodes with (InUse = TRUE), and all SDO's on each node (InUse = TRUE), the Busy goes to FALSE, and Done = TRUE if successful, or Error = TRUE if not successful completion.
The function block will check the _EC_MBXSlavTbl[NodeAdr] before reading SDO's from the node.
If there is error in one or more nodes, the FB will stop with an error, and will not proceed.

See example code: "Example, set up NodeDat for MX2 inverter:"

7.1. FB Layout



7.2. Input Variables

Name	Data type	Valid Range	Description
Execute	BOOL		Start writing on rising edge.
ExcludeEquals	BOOL		Writing only the SDO's that have IsEqual_ReadDat_WriteDat = FALSE. To check for differences, use the OEN\ECAT\SDO\NodeDatCmp_Read_Write
NoOfSDOs	UINT	1-40	Put in the highest number of SDOs in use, to eliminate unnecessary looping.

7.3. In-Out Variables

Name	Data type	Description
NodeDat	OEN\ECat\SDO\sNodeDat[*] (Dynamic size)	One array index for each node. The index does not reflect the node address.

7.4. Output Variables

Name	Data Type	Description
Done	BOOL	TRUE at least one cycle after successfully completion. Or as long as Execute is TRUE. According to PLC Open standard
Busy	BOOL	TRUE while busy with reading.
Error	BOOL	TRUE at least one cycle after successfully completion. Or as long as Execute is TRUE. According to PLC Open standard

7.5. Revisions

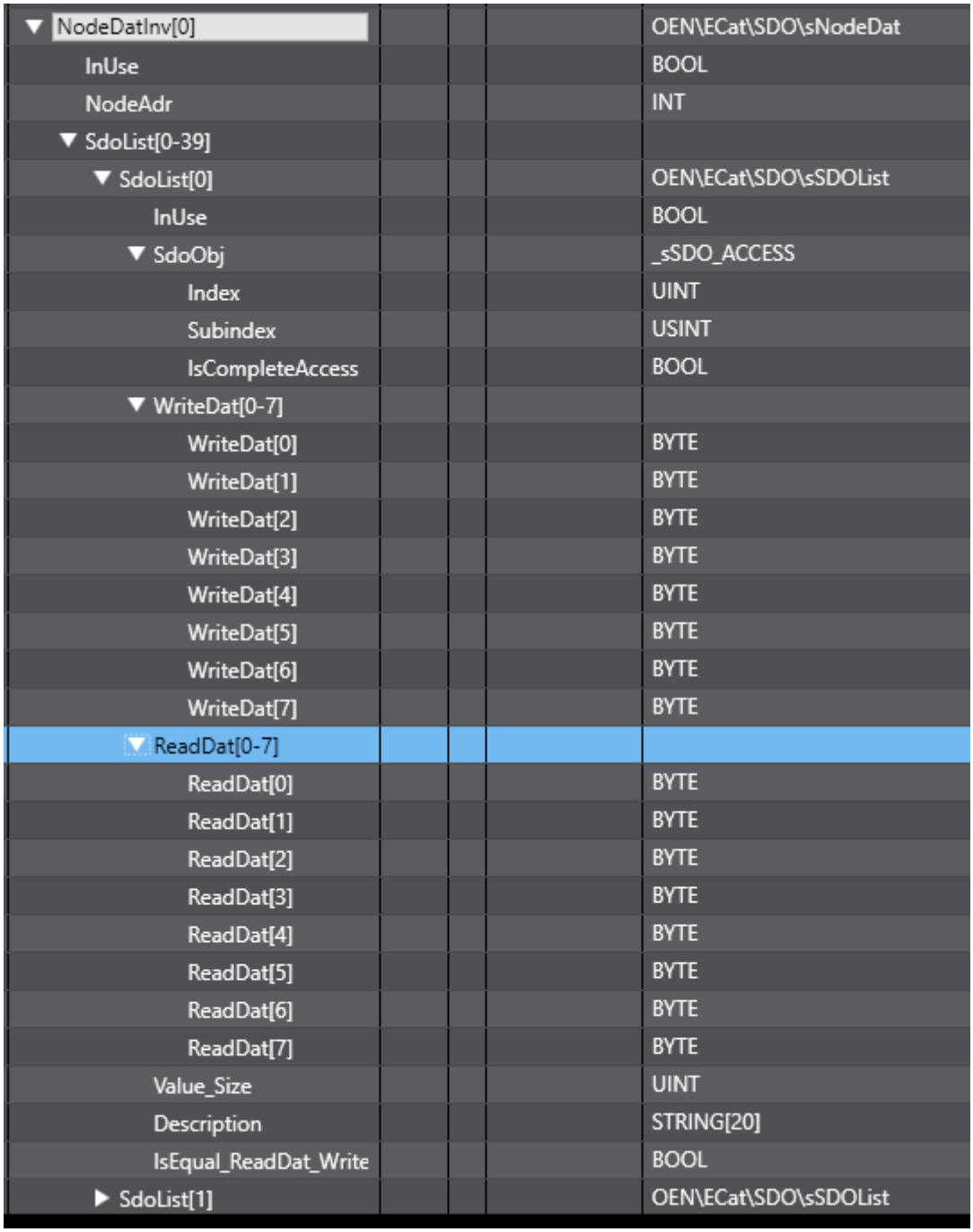
Revision	In Library	Correction
1.0.1	1.00.22	Replaced NodeDat with dynamic ARRAY. Removed Input NoOfNodes

7.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

7.7. Example

Picture of OEN\ECat\SDO\sNodeDat:



8. NodeDatCmp_Read_Write

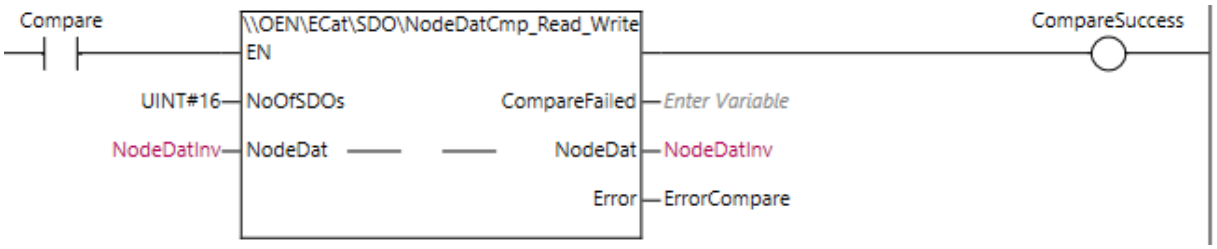
Compares all the ReadDat values with the WriteDat values for all nodes and all SDO's.

The return value will be set to TRUE if there are no differences found in any of the SDO's.
If there are differences, the Output CompareFailed output will be TRUE.
In addition the member .IsEqual_ReadDat_WriteDat will be set for each SDO.

So when using NodeWriteDat function block with the ExcludeEquals = TRUE, only the differences will be written to the nodes.

For example code: "Example, set up NodeDat for MX2 inverter:"

8.1. FB Layout



8.2. Input Variables

Name	Data type	Valid Range	Description
Execute	BOOL		Start reading on rising edge.
NoOfSDOs	UINT	1-40	Put in the highest number of SDOs in use, to eliminate unnecessary looping.

8.3. In-Out Variables

Name	Data type	Description
NodeDat	OEN\ECat\SDO\sNodeDat[*] (Dynamic size)	One array index for each node. The index does not reflect the node address.

8.4. Output Variables

Name	Data Type	Description
(Return)	BOOL	TRUE if compare is successful. (All ReadDat values are equal with the value set in WriteDat)
CompareFailed	BOOL	TRUE if compare is unsuccessful. (ReadDat values are not equal with the value set in WriteDat)
Error	BOOL	TRUE if NoOfSDOs is greater than 40.

8.5. Revisions

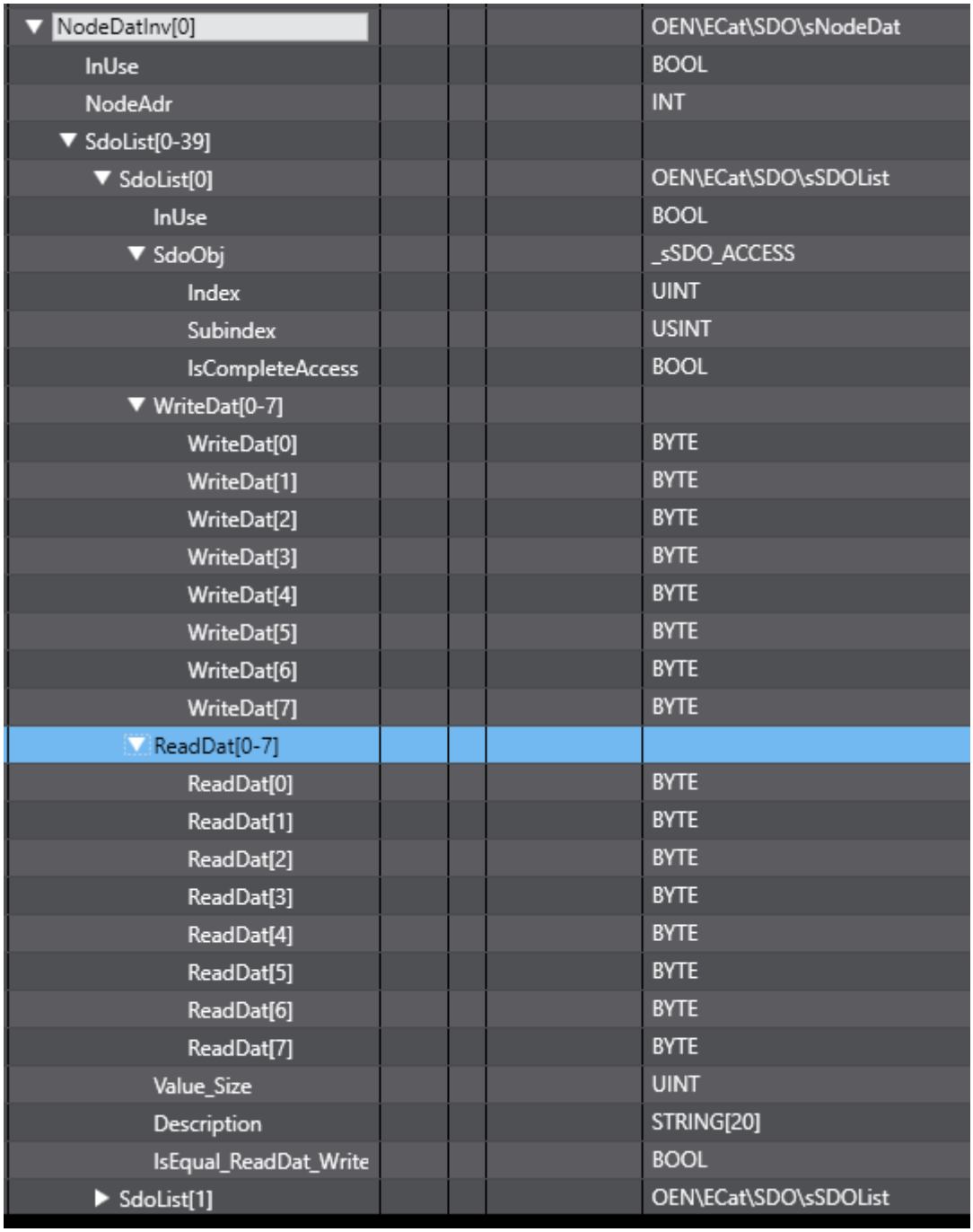
Revision	In Library	Correction
1.0.1	1.00.22	Replaced NodeDat with dynamic ARRAY. Removed Input NoOfNodes

8.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

8.7. Example

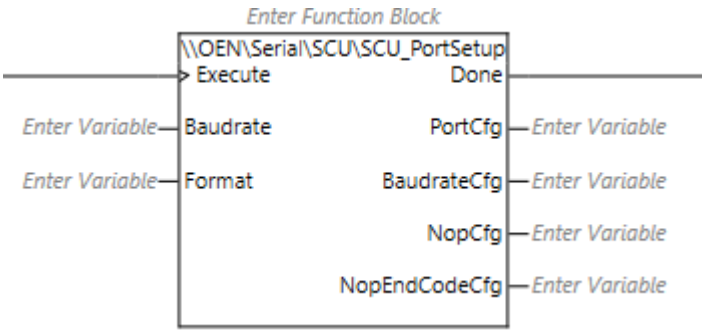
Picture of OEN\ECat\SDO\sNodeDat:



9. SCU_PortSetup

A function block to simplify the setup of an SCU port.
All 4 outputs on the block must be connected to the SCU module's variables in IOMap.
NX CPUs cannot be used.

9.1. FB Layout



9.2. Input Variables

Name	Data type	Valid Range	Description
Execute	BOOL		Enable function
Baudrate	DINT	9600..230400	
Format	STRING[4]	7E1,7E2, 8N1,8E1	

StartCode is Default.
EndCode is set to <CR><LF>.

9.3. Output Variables

Name	Data Type	Description
Done	BOOL	Setup done
PortCfg	WORD	Connect this to the I/O Map variable
BaudrateCfg	USINT	Connect this to the I/O Map variable
NopCfg	WORD	Connect this to the I/O Map variable
NopEndCodeCfg	USINT	Connect this to the I/O Map variable

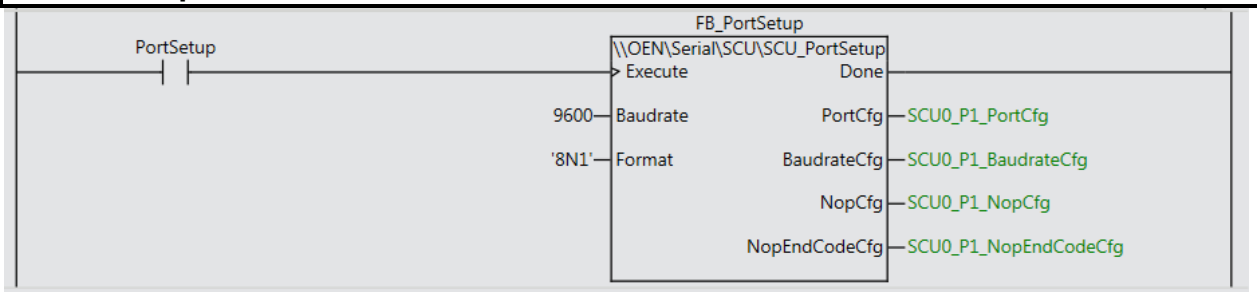
9.4. Revisions

Revision	In Library	Correction
0.1.0	1.01.00	

9.5. Credits

	Name
Omron - Norway	Kjell Baardsgaard

9.1. Example

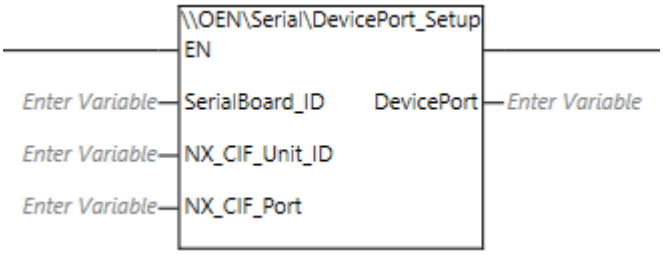


[00]	CJ1W-SCU22 (Serial Communication Unit)				
	Com_UnitSta	Serial Communication Unit	R	WORD	SCU0_Com_UnitSta
	P1_PortCfg	Port1: Port Settings	RW	WORD	SCU0_P1_PortCfg
	P1_BaudrateCfg	Port1: Baud Rate	RW	USINT	SCU0_P1_BaudrateCfg
	P1_SendDelayCfg	Port1: Send Delay Settings	RW	WORD	SCU0_P1_SendDelayCfg
	P1_HlkCfg	Port1: Host-Linlk Protocol S	RW	WORD	SCU0_P1_HlkCfg
	P1_PmrSgwTimeoutCfg	Port1: Serial Gateway Time	RW	WORD	SCU0_P1_PmrSgwTimeoutCfg
	P1_PmrTransCfg	Port1: Protocol macro Tran	RW	WORD	SCU0_P1_PmrTransCfg
	P1_PmrMaxDatSzCfg	Port1: Maximum Number o	RW	UINT	SCU0_P1_PmrMaxDatSzCfg
	P1_NopCfg	Port1: No-Protocol Settings	RW	WORD	SCU0_P1_NopCfg
	P1_NopStartCodeCfg	Port1: No-protocol Start Co	RW	USINT	SCU0_P1_NopStartCodeCfg
	P1_NopEndCodeCfg	Port1: No-protocol End Co	RW	USINT	SCU0_P1_NopEndCodeCfg

10. *DevicePort_Setup*

DevicePort_Setup simplifies the setup of function blocks that use serial ports. Use *DisplayNodeLocationPort* in I/O Map to create a port ID. Remember to select protocol, baud, parity and stopbit for the actual port in the Port Setup screen.

10.1. FN Layout



10.2. Input Variables

Name	Data type	Valid Range	Description
EN	BOOL		Enable function
SerialBoardID	_sOPTBOARD_ID		Serial board ID created in I/O Map or
NX_CIF_Unit_ID	_sNXUNIT_ID		…NX Unit ID created in I/O Map
NX_CIF_Port	USINT	0..2	NX Unit Port Number

10.3. Output Variables

Name	Data Type	Description
Return	BOOL	EN=TRUE
DevicePort	_sDEVICE_PORT	DevicePort Setup Structure

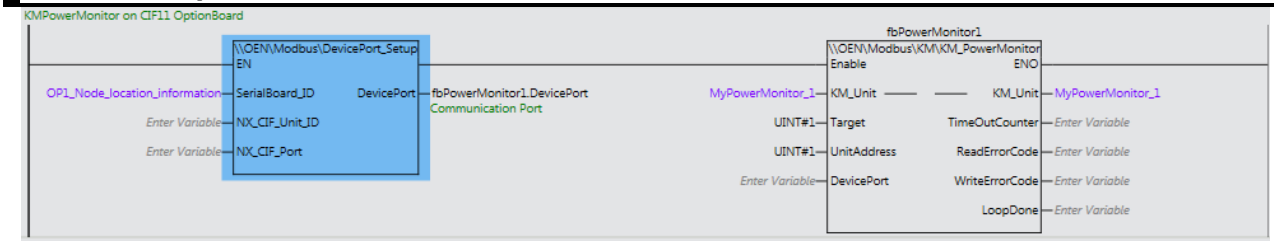
10.4. Revisions

Revision	In Library	Correction
1.2.0	1.01.00	

10.5. Credits

	Name
Omron - Norway	Kjell Baardsgaard

10.6. Example

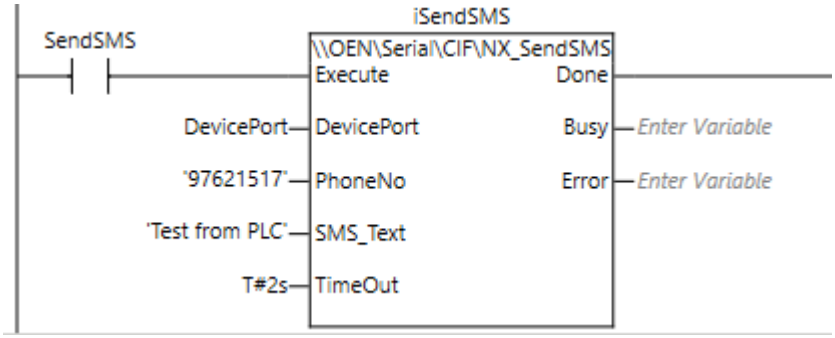

KMPowerMonitor on CIF11 OptionBoard
\\OEN\Modbus\DevicePort_Setup
EN
OP1_Node_location_information
SerialBoard_ID
DevicePort
fbPowerMonitor1.DevicePort
Communication Port
Enter Variable
NX_CIF_Unit_ID
Enter Variable
NX_CIF_Port
fbPowerMonitor1
\\OEN\Modbus\KM\KM_PowerMonitor
Enable
ENO
MyPowerMonitor_1
KM_Unit
KM_Unit
MyPowerMonitor_1
UINT#1
Target
TimeOutCounter
Enter Variable
UINT#1
UnitAddress
ReadErrorCode
Enter Variable
Enter Variable
DevicePort
WriteErrorCode
Enter Variable
LoopDone
Enter Variable

11. *NX_SendSMS*

The function block is using AT commends to communicate with the modem.

"ATE0" to turn off echo.
"AT+CMGF=1" to put the modem into "SMS Text mode".
"AT+CMGS="PHONE NUMBER""
Then the SMS text are sent.

11.1. FB Layout



11.2. Input Variables

Name	Data type	Description
Execute	BOOL	Start reading on raising edge.
DevicePort	_sDevicePort	See the reference manual for NX_SerialSend, NX_SerialRcv for help
PhoneNo	STRING[256]	The receiver phone number, can also use country code: '+4797621517'
SMS_Text	STRING[256]	The SMS tekst
TimeOut	TIME	Timeout on the serial line operation. The timeout when reading 'OK' from the modem to confirm that the SMS has been sent are hardcoded to 40s.

11.3. In-Out Variables

Name	Data type	Description

11.4. Output Variables

Name	Data Type	Description
Done	BOOL	TRUE at least one cycle after successfully completion. Or as long as Execute is TRUE. According to PLC Open standard
Busy	BOOL	TRUE while busy with reading.
Error	BOOL	TRUE at least one cycle after successfully completion. Or as long as Execute is TRUE. According to PLC Open standard

11.5. Revisions

Revision	In Library	Correction
1.0.21	1.00.22	

11.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

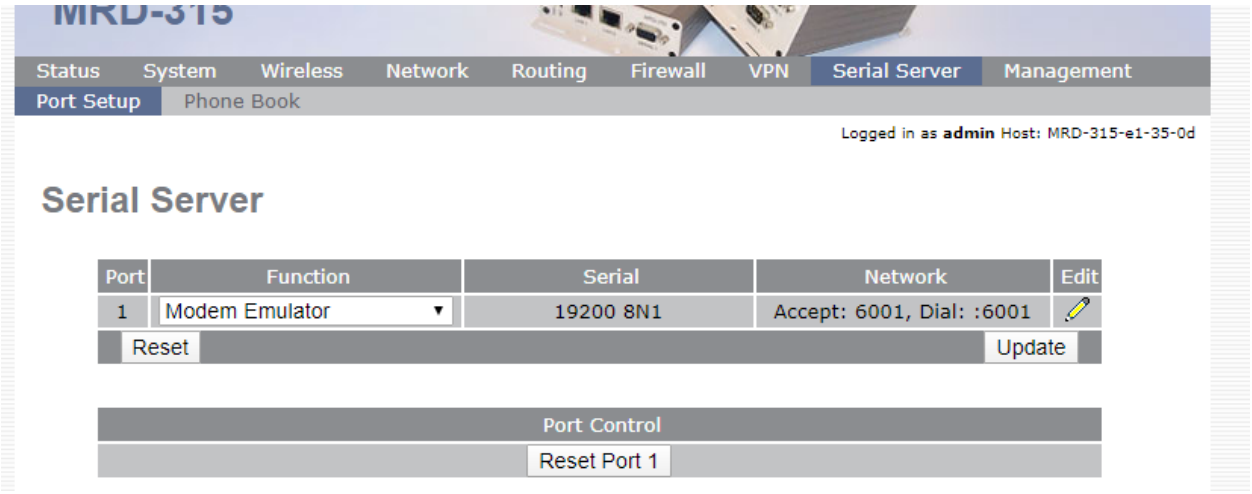
11.7. Example

Settings used in Westermo MRD-315:

SMS

SMS Triggers			
Action	Enabled	Match on	Trigger
System			
Status query	☐	Exact	Query status
Reboot	☐	Exact	Reboot
Wireless			
Packet mode	☐	Exact	Mode packet
CSD mode	☐	Exact	Mode CSD
VPN			
VPN control	☐	Starts with	VPN
Unhandled SMS Control			
Forward to email distribution list			☐
Forward to SMS distribution list			☐
Forward to serial ports			☑
Reset			Update

SMS Access Control				
Label	Phone Number	Action	Edit	Delete
Default policy		Accept	Update	
Add new access control				

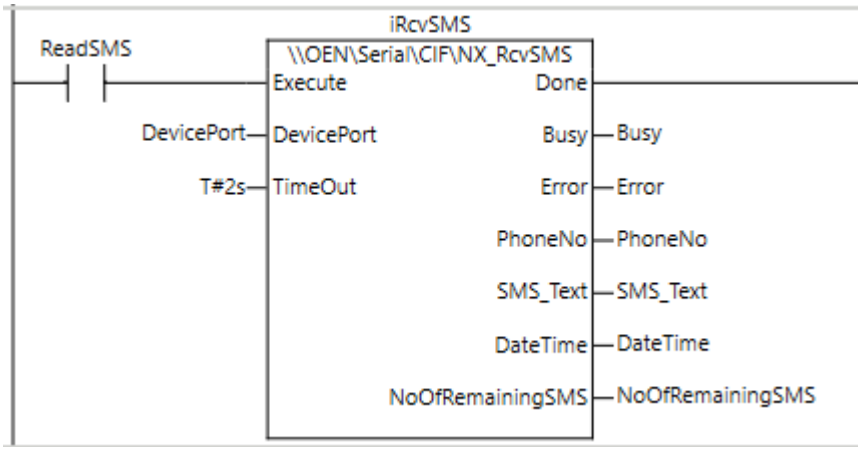

MRD-315
Status
System
Wireless
Network
Routing
Firewall
VPN
Serial Server
Management
Port Setup
Phone Book
Logged in as admin Host: MRD-315-e1-35-0d
Serial Server
Port
Function
Serial
Network
Edit
1
Modem Emulator
19200 8N1
Accept: 6001, Dial: :6001
Reset
Update
Port Control
Reset Port 1

12. NX_RcvSMS

The function block is using AT commends to communicate with the modem.

“ATE0” to turn off echo.
“AT+CMGL="ALL"“ to read all the messages in the modem. (Both READ and UNREAD messages) The function block will extract only the first message.
“AT+CMGD=X,0” to delete the read message from the modems buffer. The X is the buffernumber the message was stored in.
“AT+CPMS="SM"” to read the number of remaining messages in the buffer of the modem.

12.1. FB Layout



12.2. Input Variables

Name	Data type	Description
Execute	BOOL	Start reading on raising edge.
DevicePort	_sDevicePort	See the reference manual for NX_SerialSend, NX_SerialRcv for help
TimeOut	TIME	Timeout on the serial line operation.

12.3. In-Out Variables

Name	Data type	Description

12.4. Output Variables

Name	Data Type	Description
Done	BOOL	TRUE at least one cycle after successfully completion. Or as long as Execute is TRUE. According to PLC Open standard
Busy	BOOL	TRUE while busy with reading.
Error	BOOL	TRUE at least one cycle after successfully completion. Or as long as Execute is TRUE. According to PLC Open standard
PhoneNo	STRING[256]	The senders phone number
SMS_Text	STRING[256]	The received message.
DateTime	STRING[50]	Date and Time raw from the message.
NoOfRemainingSMS	INT	The number of SMS in the buffer of the modem after the current message was read.

12.5. Revisions

Revision	In Library	Correction
1.0.21	1.00.22	Complete redesign

12.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

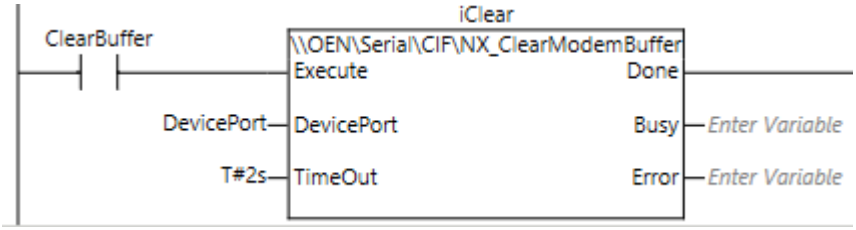
12.7. Example

13. NX_ClearModemBuffer

The function block is using AT commends to communicate with the modem.
The purpose is to clear all the messages stored in the modem.
Typically before sending a message, and waiting for a specific response.
(In case some has sendt some rubbish SMS etc.)

"ATE0" to turn off echo.
"AT+CMGD=1,4" to delete all the messages in the modem.

13.1. FB Layout



13.2. Input Variables

Name	Data type	Description
Execute	BOOL	Start reading on raising edge.
DevicePort	_sDevicePort	See the reference manual for NX_SerialSend, NX_SerialRcv for help
TimeOut	TIME	Timeout on the serial line operation.

13.3. In-Out Variables

Name	Data type	Description

13.4. Output Variables

Name	Data Type	Description
Done	BOOL	TRUE at least one cycle after successfully completion. Or as long as Execute is TRUE. According to PLC Open standard
Busy	BOOL	TRUE while busy with reading.
Error	BOOL	TRUE at least one cycle after successfully completion. Or as long as Execute is TRUE. According to PLC Open standard

13.5. Revisions

Revision	In Library	Correction
1.0.21	1.00.22	

13.6. Credits

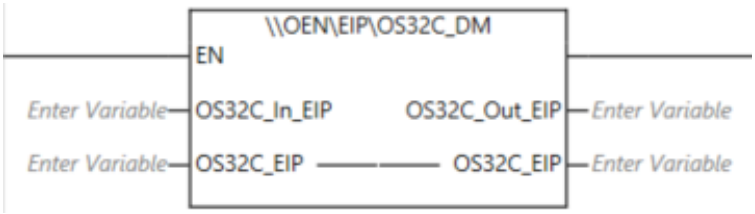
	Name
Omron - Norway	Bjarte Myklebust

13.7. Example

14. OS32C_DM

Reading data from safety scanner OS32C-DM via EthernetIP.

14.1. FN Layout



14.2. Input Variables

Name	Data type	Description
EN	BOOL	Enable function
OS32C_In_EIP	OEN\nEIP\sOS32C_In_EIP	EIP Network data from OS32C

14.3. In-Out Variables

Name	Data type	Description
OS32C_EIP		

14.4. Output Variables

Name	Data Type	Description
Return	BOOL	EN is TRUE
OS32C_Out_EIP	OEN\nEIP\sOS32C_Out_EIP	EIP Network data to OS32C

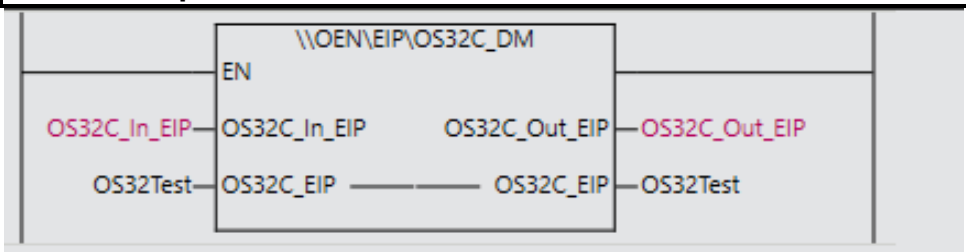
14.5. Revisions

Revision	In Library	Correction
1.0.0	1.01.00	

14.6. Credits

	Name
Omron - Norway	Kjell Baardsgaard

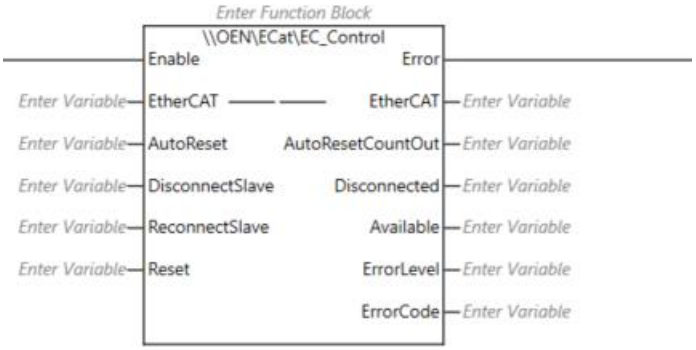
14.7. Example



15. *EC_Control*

With this function block, you can reset the EtherCAT network in case of errors, as well as disconnect nodes out and out of the loop.
The EtherCAT output is a data structure where you can read the status of up to 192 nodes.

15.1. FB Layout



15.2. Input Variables

Name	Data type	Description
Enable	BOOL	Enable function
AutoReset	BOOL	Activate AutoReset if you want to automatically reset errors on the network when slaves fall out and reconnect.
DisconnectSlave	UINT	Set DisconnectSlave to a node number and the node will be disconnected immediately. EtherCAT.CTRL.DisconnectSlave is automatically set to 0 when the slave is offline. Offline slaves can be disconnected from the network without giving an error message.
ReconnectSlave	UINT	Set ReconnectSlave to a node number and the node will be connected immediately. EtherCAT.CTRL.ReconnectSlave is automatically set to 0 when the slave is connected.
Reset	BOOL	With a pulse on Reset, you can manually reset errors if you do not want to use AutoReset.

15.3. In-Out Variables

Name	Data type	Description
EtherCAT	OEN\nECAT\sEC_Network	Structure to hold Network data

15.4. Output Variables

Name	Data Type	Description
Error	BOOL	EtherCAT Error, see ErrorLevel and ErrorCode
AutoresetCountOut	BOOL	AutoResetCountOut indicates that the selected number of resets has been achieved without the error being corrected. The reset interval is 1s.
Disconnected	UINT	Number of Nodes in StandBy.
Available	UINT	Number of Nodes Online+OnlineStandBy
ErrorLevel	UINT	0: No error, 2: Partial Fault, 3: Minor Fault
ErrorCode	DWORD	See GetECError Error Codes in Manual W503

15.5. Optional Settings

Name	Data type	Default	Description
<EtherCAT>.CTRL.AutoresetCount	UINT	#3	May be set to another value if one does not want to use 3.
<EtherCAT>.CTRL.AutoresetDelay		1s	May be set to another value if one does not want to use 1s.

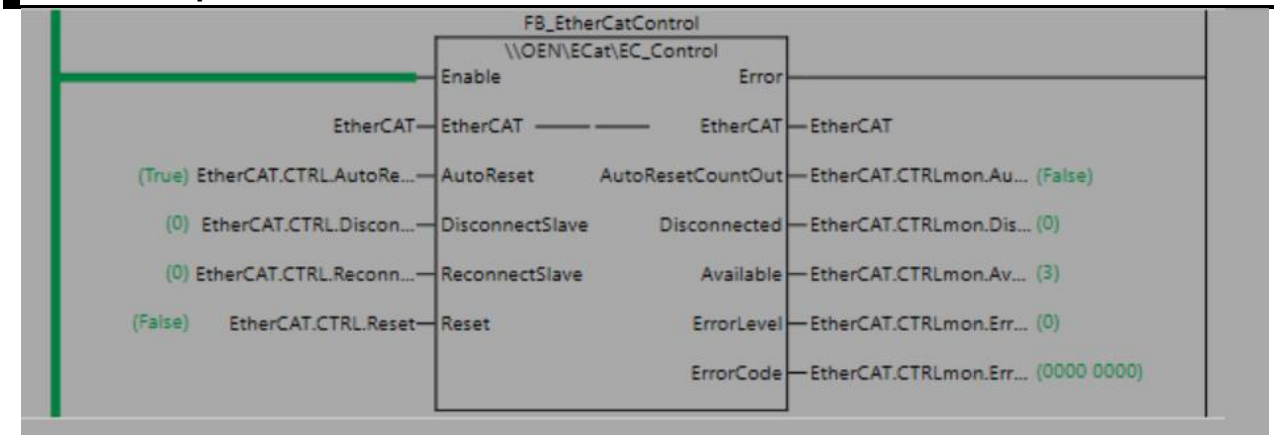
15.6. Revisions

Revision	In Library	Correction
3.3.0	1.01.0	

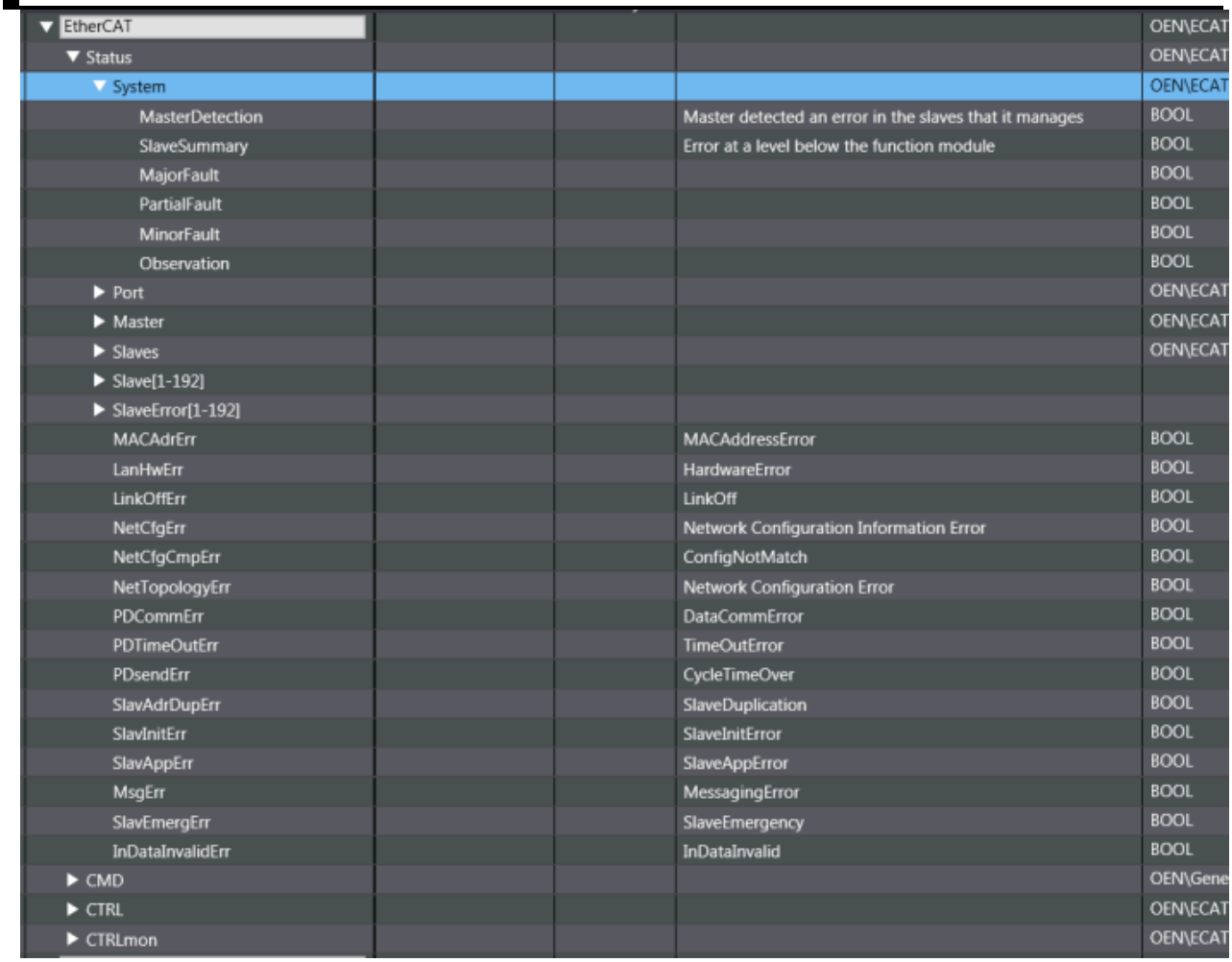
15.7. Credits

	Name
Omron - Norway	Kjell Baardsgaard

15.8. Example



15.9. Structure

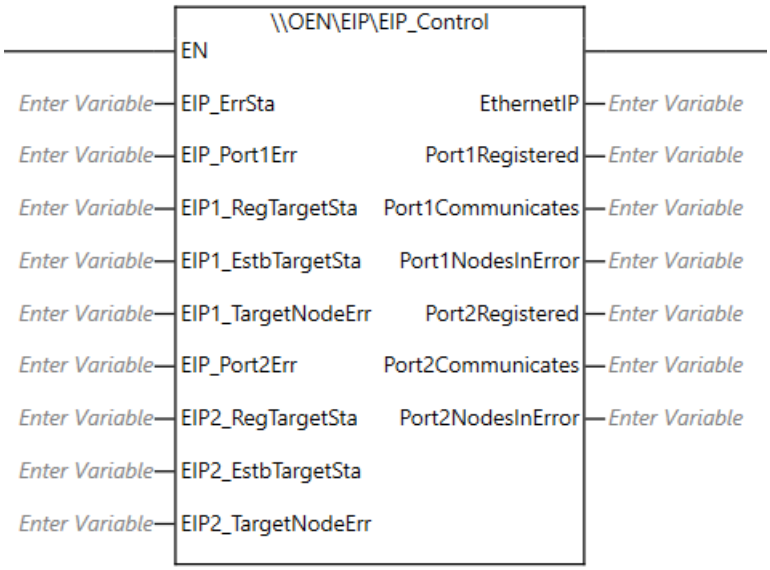


▼ EtherCAT				OEN\ECAT
▼ Status				OEN\ECAT
▼ System				OEN\ECAT
MasterDetection			Master detected an error in the slaves that it manages	BOOL
SlaveSummary			Error at a level below the function module	BOOL
MajorFault				BOOL
PartialFault				BOOL
MinorFault				BOOL
Observation				BOOL
▶ Port				OEN\ECAT
▶ Master				OEN\ECAT
▶ Slaves				OEN\ECAT
▶ Slave[1-192]				
▶ SlaveError[1-192]				
MACAdrErr			MACAddressError	BOOL
LanHwErr			HardwareError	BOOL
LinkOffErr			LinkOff	BOOL
NetCfgErr			Network Configuration Information Error	BOOL
NetCfgCmpErr			ConfigNotMatch	BOOL
NetTopologyErr			Network Configuration Error	BOOL
PDCommErr			DataCommError	BOOL
PDTimeOutErr			TimeOutError	BOOL
PDsendErr			CycleTimeOver	BOOL
SlavAdrDupErr			SlaveDuplication	BOOL
SlavInitErr			SlaveInitError	BOOL
SlavAppErr			SlaveAppError	BOOL
MsgErr			MessagingError	BOOL
SlavEmergErr			SlaveEmergency	BOOL
InDataInvalidErr			InDataInvalid	BOOL
▶ CMD				OEN\Gene
▶ CTRL				OEN\ECAT
▶ CTRLmon				OEN\ECAT

16. *EIP_Control*

This Function shows the status of the EIP communication in a data structure. Registered, Communicates, and NodesInError display the number of nodes defined, that communicate and that have error status. Applicable for NJ, NX102 and NX1P2.

16.1. FN Layout



16.2. Input Variables

Name	Data type	Description
EN	BOOL	Enable function
EIP_ErrSta	WORD	System variable name starting with _EIP
EIP_Port1Err	WORD	System variable name starting with _EIP
EIP1_RegTargetSta	ARRAY[0..255] OF BOOL	System variable name starting with _EIP
EIP1_EstbTargetSta	ARRAY[0..255] OF BOOL	System variable name starting with _EIP
EIP1_TargetNodeErr	ARRAY[0..255] OF BOOL	System variable name starting with _EIP
EIP_Port2Err	WORD	System variable name starting with _EIP
EIP2_RegTargetSta	ARRAY[0..255] OF BOOL	System variable name starting with _EIP
EIP2_EstbTargetSta	ARRAY[0..255] OF BOOL	System variable name starting with _EIP
EIP2_TargetNodeErr	ARRAY[0..255] OF BOOL	System variable name starting with _EIP

16.3. Output Variables

Name	Data Type	Description
Return	BOOL	EN=TRUE
EthernetIP	OEN\nEIP\sEIP_Network	Structure containing status for all EIP Nodes
Port1Registered	USINT	Number of Nodes registered
Port1Communicates	USINT	Number of Nodes Communicating
Port1NodesInError	USINT	Number of Nodes in Error
Port2Registered	USINT	Number of Nodes registered
Port2Communicates	USINT	Number of Nodes Communicating
Port2NodesInError	USINT	Number of Nodes in Error

16.1. Optional Settings

Name	Data type	Default	Description
<EthernetIP>.Cmd.Reset	BOOL		Not in use

16.2. Revisions

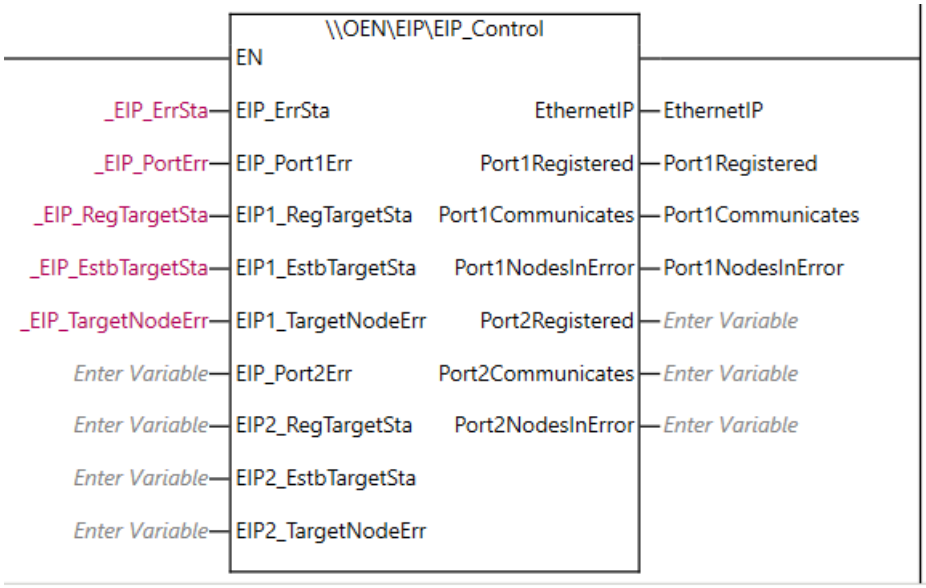
Revision	In Library	Correction
2.0.0	1.01.0	
3.0.0	1.05.0	Added Port#2 for NX102

16.3. Credits

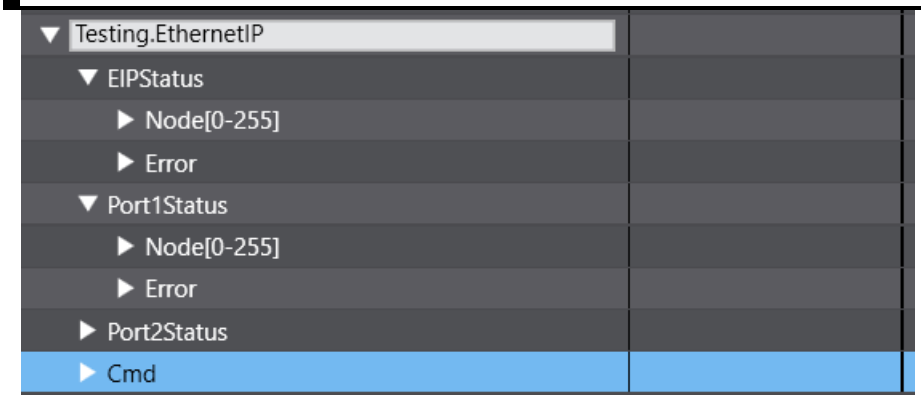
	Name
Omron - Norway	Kjell Baardsgaard

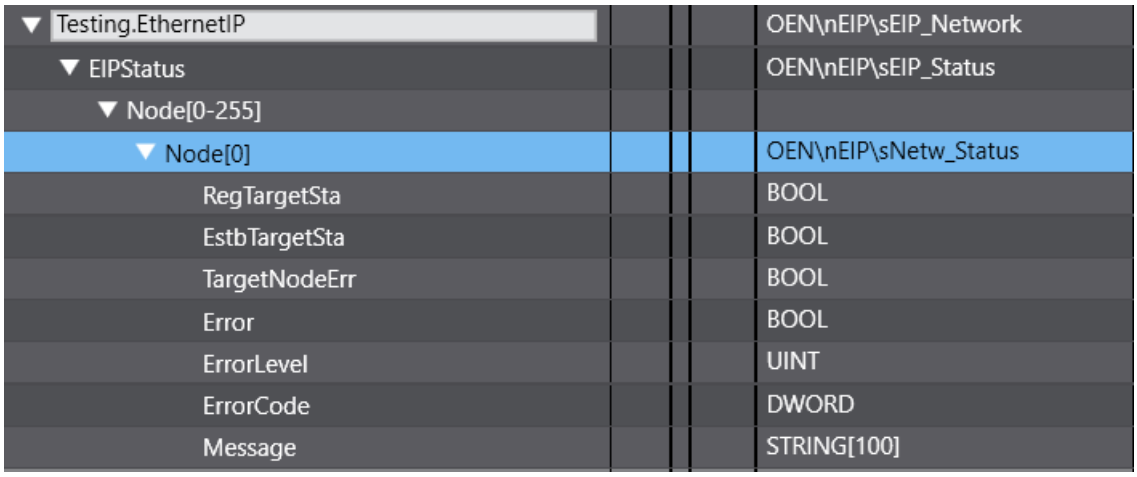
16.4. Example

An example using NX1P2



16.5. Structure

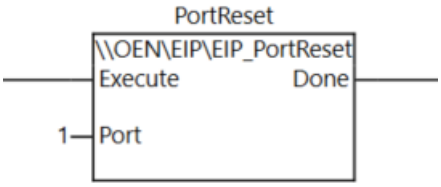




17. *EIP_PortReset*

This Function Block sends a Reset command to the chosen Ethernet/IP-port.

17.1. FB Layout



17.2. Input Variables

Name	Data type	Description
Execute	BOOL	Send Reset Command
Port	UINT	Ethernet/IP-port number 1 or 2

17.3. Output Variables

Name	Data Type	Description
Done	BOOL	Completed

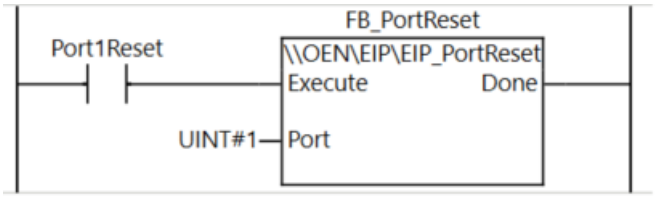
17.4. Revisions

Revision	In Library	Correction
1.0.0	1.04.0	

17.5. Credits

	Name
Omron - Norway	Kjell Baardsgaard

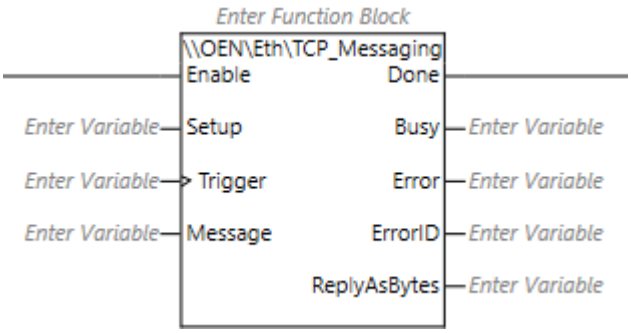
17.6. Example



18. TCP_Messaging

An FB to send custom data over TCP.

18.1. FB Layout



18.2. Input Variables

Name	Data type	Description
Enable	BOOL	Enable function
Setup	OEN\nEth\sSetup	TCP Server configuration setup data
Trigger	BOOL	Trigger to send Message
Message	STRING[256]	Message to be send

18.3. Output Variables

Name	Data Type	Description
Done	BOOL	Message successfully send
Busy	BOOL	Connecting, Sending and waiting for reply
Error	BOOL	Error connecting or sending
ErrorID	WORD	Connecting or sending error code
ReplyAsBytes	ARRAY[0..255] OF BYTE	Reply from connected device

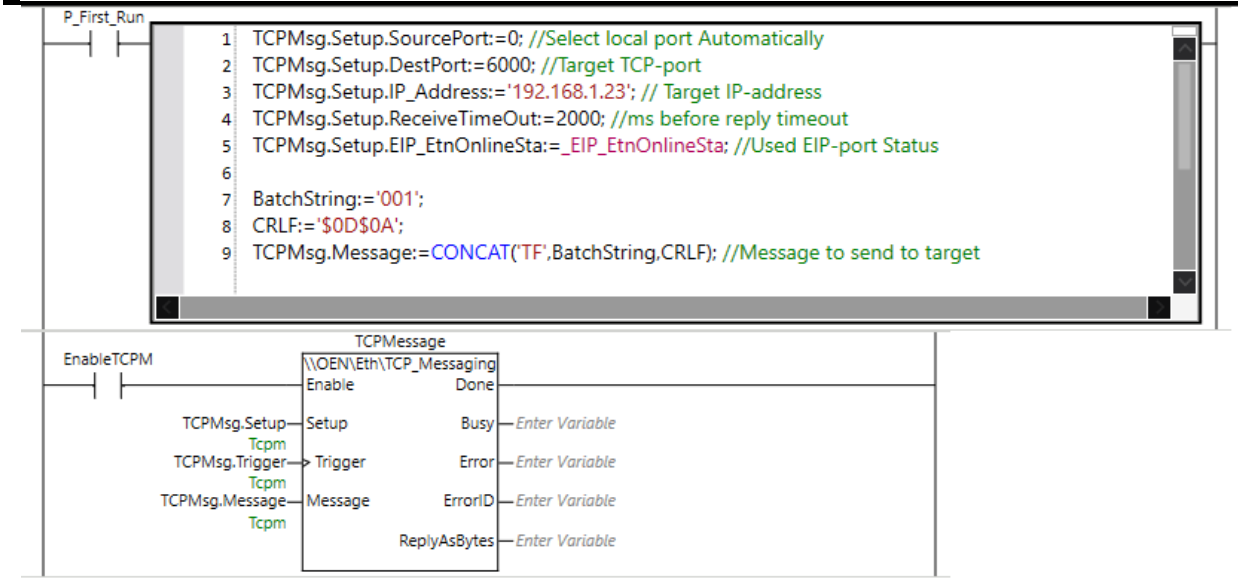
18.4. Revisions

Revision	In Library	Correction
1.1.0	1.01.0	

18.5. Credits

	Name
Omron - Norway	Kjell Baardsgaard

18.6. Example

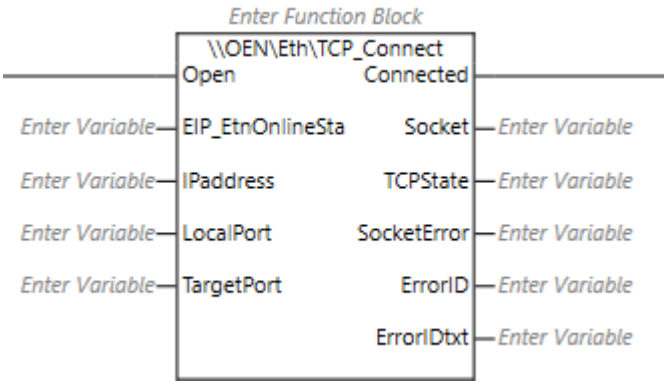

P_First_Run
1 TCPMsg.Setup.SourcePort:=0; //Select local port Automatically
2 TCPMsg.Setup.DestPort:=6000; //Target TCP-port
3 TCPMsg.Setup.IP_Address:='192.168.1.23'; // Target IP-address
4 TCPMsg.Setup.ReceiveTimeOut:=2000; //ms before reply timeout
5 TCPMsg.Setup.EIP_EtnOnlineSta:=_EIP_EtnOnlineSta; //Used EIP-port Status
6
7 BatchString:='001';
8 CRLF:='$0D$0A';
9 TCPMsg.Message:=CONCAT('TF',BatchString,CRLF); //Message to send to target
TCPMessage
EnableTCPM
\\OEN\Eth\TCP_Messaging
Enable
Done
TCPMsg.Setup
Tcpm
Setup
Busy
Enter Variable
TCPMsg.Trigger
Tcpm
Trigger
Error
Enter Variable
TCPMsg.Message
Tcpm
Message
ErrorID
Enter Variable
ReplyAsBytes
Enter Variable

19. *TCP_Connect*

A function block to simplify communication via SocketService. Open, Close, ClearBuf and GetTCPStatus are used in this, and the input Open can be used to open/close a Socket. E.g. ModbusTCP can then be run via the specified Socket. If you set LocalPort=0, it will find a free port automatically.

A socket is a kind of serial port. You open a socket and then you send and receive data on it. The CPU can have many sockets open on the same physical ethernet port, so this is much more rational than before, where one often had to have a serial port for each device to communicate with.

19.1. FB Layout



19.2. Input Variables

Name	Data type	Valid Range	Description
Open	BOOL		Enable function
EIP_EtnOnlineSta	BOOL		EIP port Status. Use System variable
IPAddress	STRING[200]		Example '192.168.250.3'
LocalPort	UINT		0=Auto Assignment
TargetPort	UINT		Remote TCP Target Port

19.3. Output Variables

Name	Data Type	Description
Connected	BOOL	Successfully connected
Socket	_sSocket	Socket status for monitoring purpose
TCPState	_eCONNECTION_STATE	Connection state as eNum
SocketError	BOOL	Socket error
ErrorID	WORD	Error Code
ErrorIDtxt	STRING[100]	ErrorID as readable message

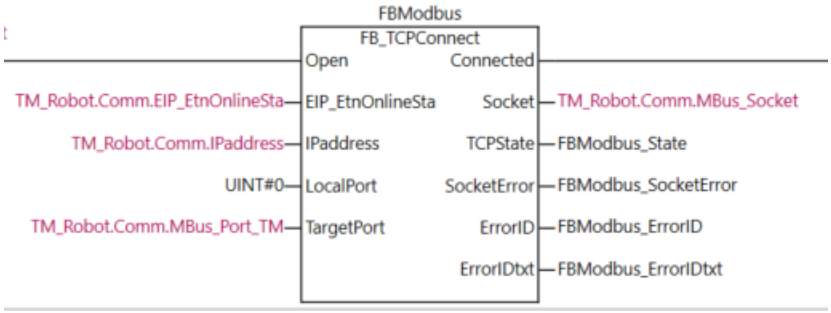
19.4. Revisions

Revision	In Library	Correction
2.2.0	1.01.00	Counts 3 consecutive Port Errors before closing and reopening

19.5. Credits

	Name
Omron - Norway	Kjell Baardsgaard

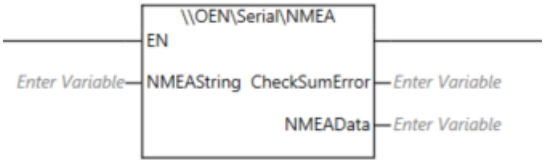
19.6. Example



20. NMEA 0183

This function adds NMEA-data to a structured variable.
It supports GGA, GLL, WPL and RMC

20.1. FN Layout



20.2. Input Variables

Name	Data type	Description
NMEAString	STRING[199]	String data captured via serial port

20.3. Output Variables

Name	Data Type	Description
CheckSumError	BOOL	CheckSum Error in incoming string
NMEAData	OEN\nSerial\nNMEA\sNMEA	Structured variable that holds NMEA-data

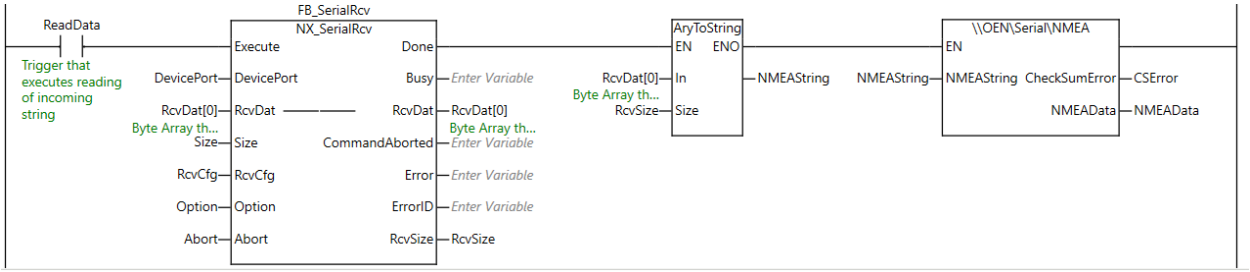
20.4. Revisions

Revision	In Library	Correction
1.0.0	1.05.0	

20.5. Credits

	Name
Omron - Norway	Kjell Baardsgaard

20.6. Example

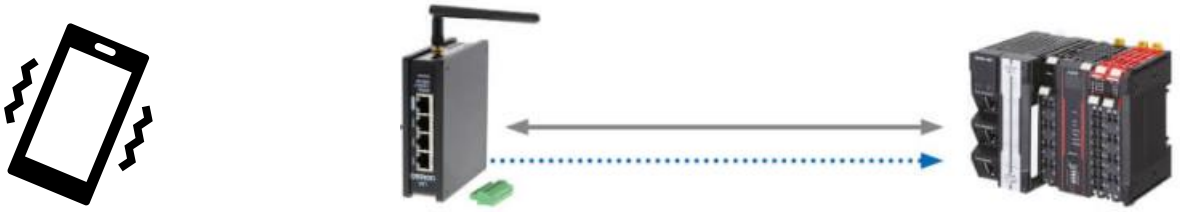


See how to configure NX_SerialRcv earlier in this document.

21. RT100_SMSAlarm

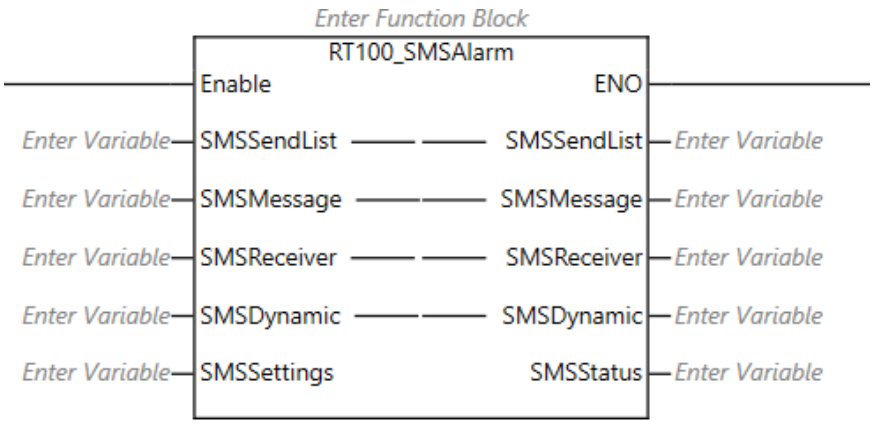
A Function Block that uses Omron RT100-4G router to send and receive SMS-Messages to/from multiple receivers. Delays between messaging each receiver can be set for each alarm. Number of receivers and alarms can be up to 9999 but [NumOfReceivers+1 * NumOfAlarms+1] cannot exceed 65535. System stops sending the message if Alarm=off or Disabled=on or Ack_Received=on in SMSDynamic.
Alarms can be acknowledged from receivers, and this acknowledge will be replied to from the CPU.
Receivers can belong to a group, and messages can be sent to all members of a group simultaneously.
With a little programming effort, commands can be sent from the phone to change CPU variables.

This FB is using HTTP_GET from HTTP Library and RT1SendSMS from RT1_MailSmsSend Library.



Requires RT100-4G and Sysmac Controller Firmware 1.46 or higher.

21.1. FB Layout



21.2. Input Variables

Name	Data type	Description
Enable	BOOL	Enable function
SMSSettings	OEN\nEth\nSMS\sSMSSettings	Setup for RT100 before you start

21.3. In-Out Variables

Name	Data type	Description
SMSSendList	ARRAY[*,*] OF OEN\nEth\nSMS\sSMSSendList	2Dim ReceiversID and delays between each message.
SMSMessage	ARRAY[*] OF OEN\nEth\nSMS\sSMSMessages	Messages. Do not use national characters.
SMSReceiver	ARRAY[*] OF OEN\nEth\nSMS\sSMSReceivers	Phone number per Receiver and initials to identify receiver
SMSDynamic	ARRAY[*] OF OEN\nEth\nSMS\sSMSDynamic	Dynamic data. Contains Alarm bit and other useful data for each alarm.

NOTE: index[0] variables must be defined but are reserved for special features.

21.4. Output Variables

Name	Data Type	Description
ENO	BOOL	Enable status
SMSStatus	OEN\nEth\nSMS\sSMSSettings	Signals to check while running

21.5. Structures

Name	Base Type	Offset Type	Comment
▼ sSMSSendList	STRUCT	NJ	Upper Size as [Messages, Receivers]
ReceiverID	UINT		ReceiverID in Prioritized Order for Message. [0] Reserved
Delay_SV	TIME		Delay before Messaging next Receiver when UnAcked for Message
▼ sSMSReceivers	STRUCT	NJ	Upper Size as number of Receivers
Phone	STRING[25]		Receiver Phone Number. [0] Reserved
INITs	STRING[4]		3 letters to identify Receiver. [0] Reserved
Group	UINT		Optional group number
▼ sSMSMessages	STRUCT	NJ	Upper Size as number of Messages
Message	STRING[152]		SMS Text Message. [0] reserved for Acknowledge reply
▼ sSMSSettings	STRUCT	NJ	
IP_RT100	STRING[16]		RT100 DEV-port IP-address
TLSSession	STRING[17]		TLS Session Name
MessageGap	TIME		Minimum time between each message
▼ sSMSDynamic	STRUCT	NJ	Upper Size as Messages
Alarm	BOOL		W Alarm activated. Write to set Alarm
Disabled	BOOL		W Do not send SMS. Write to Disable messaging
MessageSend	BOOL		R Message sending has started
Ack_Received	BOOL		RW Receiver has Acked. Write to override SMS Acknowledge
Ack_Returned	BOOL		R Ack returned to Receiver
Delay_PV	TIME		R Remaining Delay before sending SMS to next receiver
SendListIndex	UINT		R ActiveReceiver
Ack_Receiver	STRING[4]		R Receiver INITs that Acked
▼ sSMSStatus	STRUCT	NJ	
ArraySizeError	BOOL		Check manual for setting correct Array Size
ErrorRT100	BOOL		Check IP, settings and connection to RT100
NoActiveAlarms	BOOL		No active Alarms detected
NewMessage	BOOL		Switches state for every new incoming message
ProgStep	OEN\nEth\nSMS\eSteps		Monitor Program steps
SMSCommand	STRING[256]		Incoming message for your own features

Program your own features with **SMSCommand**. It can for example be used to:
change user phone, change group members, change message texts, change one or more setpoints or trigger an “alarm (event)” to get a message containing data from production. You should include a password into the message string to avoid unauthorized commands.
Tip! Separate your command data with comma and use the Function SubDelimiter to move data into a predefined Structure. Use NewMessage to detect new incoming message. A message must begin with 4 digits and two ** to be considered as a valid message. Else it will be discarded. To acknowledge an alarm, the next 3 characters (INITs) must be found in the SMSReceiver table.

21.6. Revisions

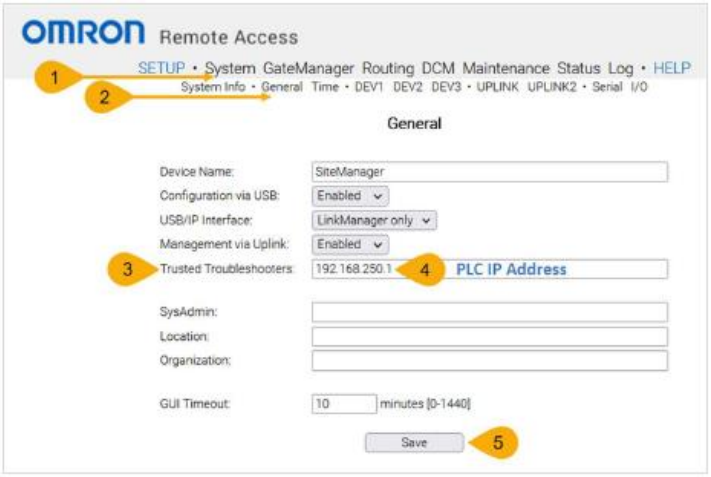
Revision	In Library	Correction
1.0.0	1.00.0	
1.0.1	1.00.1	Bugfix related to .MessageSend
1.0.2	1.00.2	Bugfix related to .FB Error

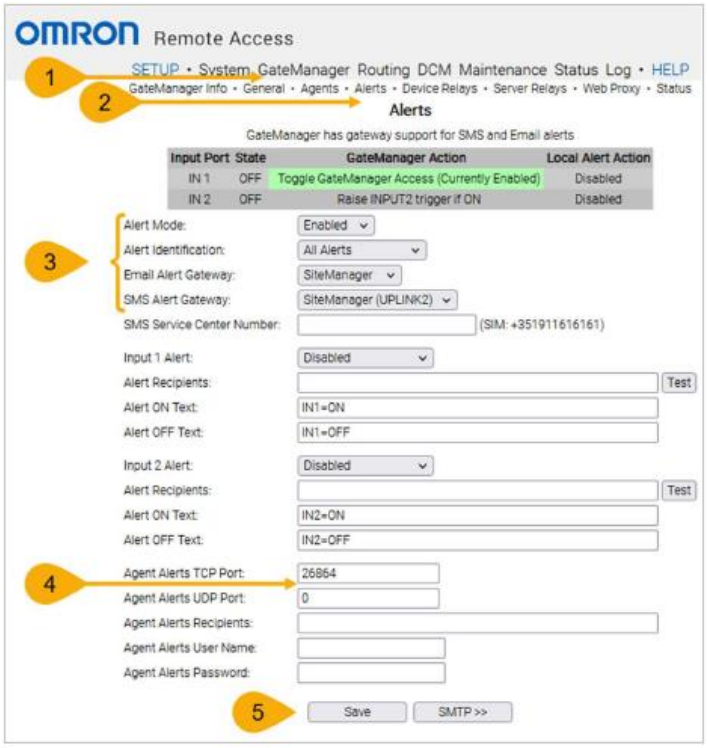
21.7. Credits

	Name
Omron - Norway	Kjell Baardsgaard
Omron - Italy	Paulo Carvalho
Omron - Italy	Francesco Matassa

21.8. Important Settings for RT100 and CPU

RT100 Settings

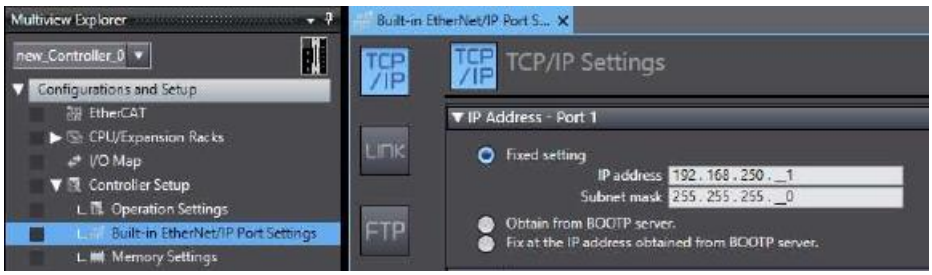




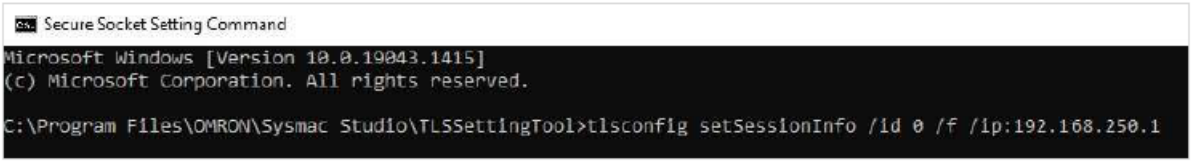
At 3: Check SMS Service Center Number for your own service provider.

Add these two agents

CPU Ethernet settings

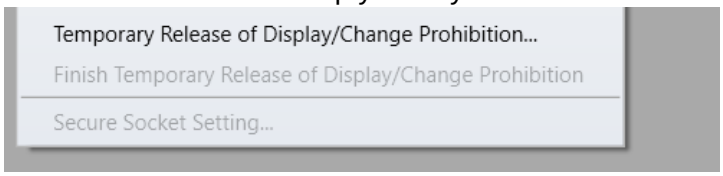


Create secure ‘TLSSession0’ in the controller using CMD-window and the Controller IP.



If you don’t get Success, then try with DIPSW #1=ON and #2=ON. Set to OFF when Success.

From Firmware 1.60 and up you may use *Secure Socket Setting…* in SysmacStudio instead.



21.9. Example

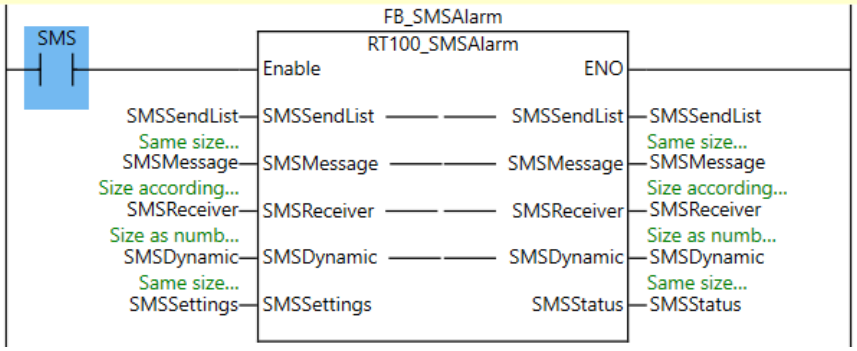
Adding a *Namespace Using* lets you skip Namespace inside FB.
Number of messages and receivers are not fixed, but make sure all array sizes correspond to this example and they all start with 0.

Name	Data Type	Initial Value	AT	Retain
SMSMessage	ARRAY[0..9] OF OEN\nEth\nSMS\sSMSMessages	[(Message := 'Your Alarm Acknowledge has been received'), (Message := 'Message from NX102'), (Message := 'Message		☑
SMSSendList	ARRAY[0..9,0..5] OF OEN\nEth\nSMS\sSMSSendList	[7((ReceiverID := 0, Delay_SV := T#0S)), (ReceiverID := 5, Delay_SV := T#10S), 5((ReceiverID := 0, Delay_SV := T#0S)), (Rec		☑
SMSReceiver	ARRAY[0..5] OF OEN\nEth\nSMS\sSMSReceivers	[(Phone := '', INITs := '', Group := 0), (Phone := '90133000', INITs := 'KJB', Group := 0), (Phone := '12345678', INITs := 'MrX		☑
SMSSettings	OEN\nEth\nSMS\sSMSSettings	(IP_RT100 := '192.168.250.200', TLSSession := 'TLSSession0', MessageGap := T#3S)		☑
SMSDynamic	ARRAY[0..9] OF OEN\nEth\nSMS\sSMSDynamic			☑
SMSStatus	OEN\nEth\nSMS\sSMSStatus			☐

Note! Initial Value column is an easy way to add content to variables. If you tick Retain, variables will not be overwritten with Initial Values unless you chose to Clear All Retain Variables when transferring program to the CPU.

0

```
This example has 9 different messages and 5 different receivers

SMSMessage contains the Message for each alarm x.
In SMSDynamic there is [x].Alarm when activated, starts sending Message to receivers
SendList contains receiversID and delay between each receiver
 for each alarm
SMSReceiver contains the Phonenumbers

Receivers can Acknowledge the alarm and the CPU will reply to the Acknowledge
```



SMSMessage[0..9]

	Message (STRING[152])	
[0]	'Your Alarm Acknowledge has been received'	
[1]	'Message from NX102'	
[2]	'Message 2'	
[3]	''	

Message[0] is always Alarm Acknowledge reply to Receiver.
Add your messages as Initial Values or create messages from the program if you want to add variables.

SMSSendList[0..9,0..5]

	[*, 0] ReceiverID (UINT)	[*, 0] Delay_SV (TIME)	[*, 1] ReceiverID (UINT)	[*, 1] Delay_SV (TIME)	[*, 2] ReceiverID (UINT)	[*, 2] Delay_SV (TIME)	[*, 3] ReceiverID (UINT)	[*, 3] Delay_SV (TIME)	[*, 4] ReceiverID (UINT)	[*, 4] Delay_SV (TIME)	[*, 5] ReceiverID (UINT)	[*, 5] Delay_SV (TIME)
[0, *]	0	T#0S	0	T#0S	0	T#0S	0	T#0S	0	T#0S	0	T#0S
[1, *]	0	T#0S	1	T#10S	2	T#0S	0	T#0S	0	T#0S	0	T#0S
[2, *]	0	T#0S	2	T#5S	1	T#0S	0	T#0S	0	T#0S	0	T#0S
[3, *]	0	T#0S	0	T#0S	0	T#0S	0	T#0S	0	T#0S	0	T#0S

Send List contains ReceiverID (see below) and a delay before sending same message to next receiver. Columns 0 and Row 0 are not used (Reserved). Time can be minutes or hours of course.

SMSReceiver[0..5]

	Phone (STRING[25])	INITs (STRING[4])	Group (UINT)
[0]	''	''	0
[1]	'90133000'	'KJB'	0
[2]	'12345678'	'MrX'	0
[3]	''	''	0

SMSReceiver contains Phone and Initials for the receiver. RowNumber=ReceiverID.
You can define a receiverID as a group by leaving the Phone='' and Group=0. Receivers belonging to that group will get the message simultaneously. Put INITs like Gr3 or something similar to identify the group.
Receiver[0] is not used (Reserved).

SMSSettings

IP_RT100 (STRING[16])	TLSSession (STRING[17])	MessageGap (TIME)
'192.168.250.200'	'TLSSession0'	T#3S

Check SMSStatus before trying to trigger an alarm.

SMS Message example from CPU when SMSDynamic.Alarm is set TRUE:

0001**KJB Message from NX102 (AlarmNumber ReceiverInitials Message)
Agent: MyNX102 (CPU Name from RT100 Agent setup)
SM: MyRT100 (the Name you set on your RT100)

SMSDynamic.MessageSend is now TRUE

You can Copy SMS and reply to sender to Acknowledge the alarm. Keep at least the first 9 characters to identify alarm number and receiver.Do not forget the **.

SMSDynamic.Ack_Received is now TRUE
SmsDynamic.Ack_Receiver is set to your Initials.

Soon after you will get the following message:

0001 Your Alarm Acknowledge has been received
Agent: MyNX102
SM: MyRT100

SMSDynamic.Ack_Returned is now TRUE.

All bits are set to FALSE when SMSDynamic.Alarm is set to FALSE

22. Template

text

21.7 FN Layout

21.8 Input Variables

Name	Data type	Valid Range	Default	Description
EN	BOOL		FALSE	Enable function

21.9 In-Out Variables

Name	Data type	Description

21.10 Output Variables

Name	Data Type	Description
	BOOL	

21.11 Revisions

Revision	In Library	Correction
1.0.0	1.00.0	

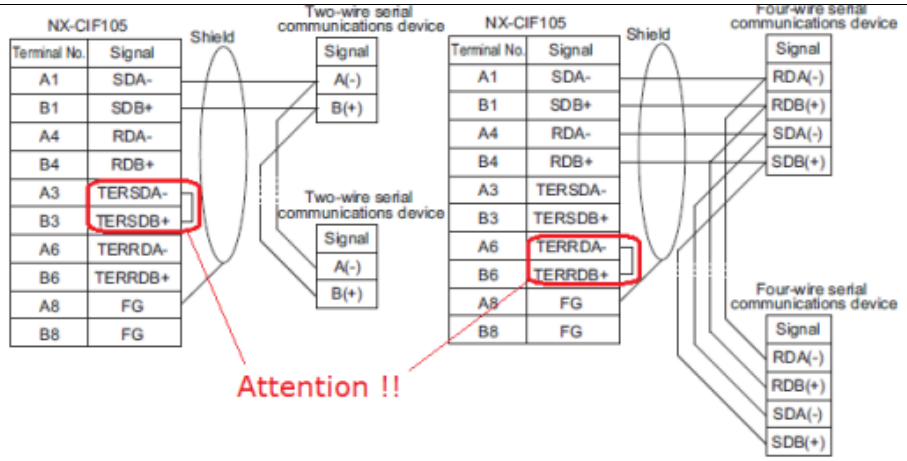
21.12 Credits

	Name
Omron - Norway	

21.13 Example

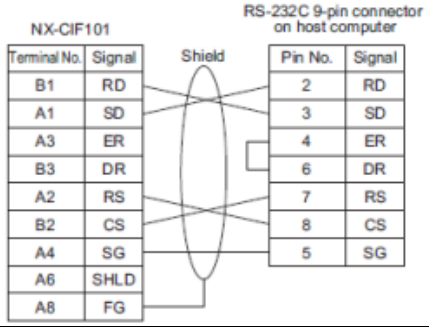
Appendix for Serial Communication

NX-CIF105



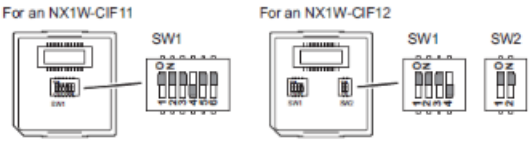
*1. For a two-wire connection, terminating resistance is turned ON when TERSDA- is connected to TERSDB+.
*2. For a four-wire connection, terminating resistance is turned ON when TERRDA- is connected to TERRDB+.
*3. The SG terminals are internally connected to the 0-V line inside the Unit. It is normally not necessary to connect the SG terminals. However, it may be possible to increase noise immunity by connecting the communications cables to the SG terminals.

NX-CIF101



Sometimes A2 can be connected to B2.

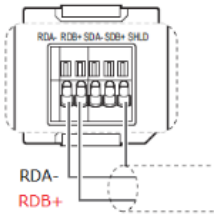
NX1W-CIF11
NX1W-CIF12

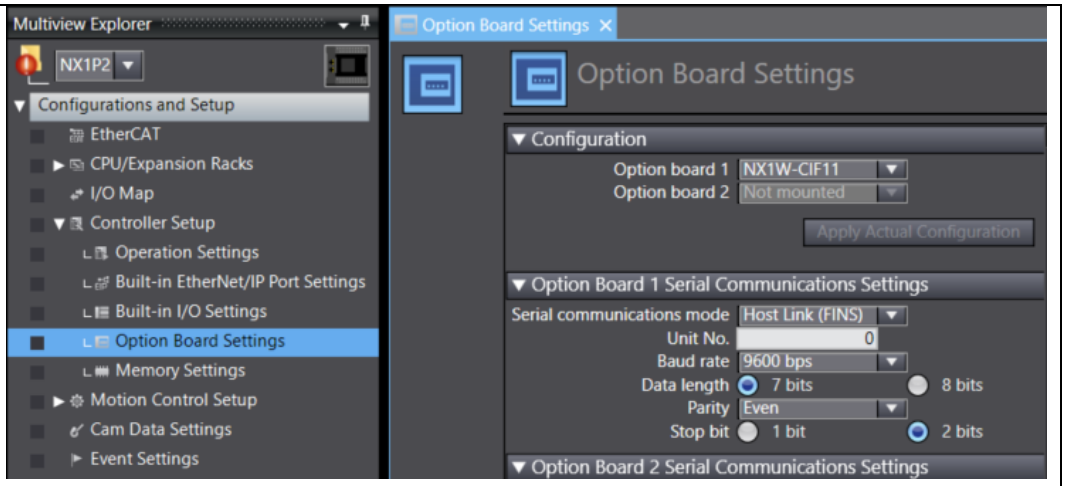


CIF11		CIF12		Setting	Setting description
SW	No.	SW	No.		
SW1	1	SW1	1	ON	With terminating resistance
	2		2	ON	Two-wire type
	3		3	ON	Two-wire type
	4		4	OFF	(Not used)
	5	SW2	1	ON	With RS control for receive data
	6		2	ON	With RS control for send data

RS485 configuration

RS-422A/485 Option Board
NX1W-CIF11/CIF12



NX1W-CIF01 **NX1W-CIF11** **NX1W-CIF12**	 **Serial Communications Mode:** **Host Link (FINS)** when communicating with CPUs supporting Omron HostLink (FINS). Hostlink C-Mode is not supported. (ex NT-displays) **Modbus-RTU Master** when CPU is a Modbus Master using the NX_ModbusRTU commands from Sysmac Studio default Toolbox. **No-Protocol** for all other communication protocols + all the serial Function Blocks described in this documentation.