Application Library

OEN_Communication 1.0.22

Sysmac Function Block Library for Modbus, EtherCAT SDO 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.

Library Change Log

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 do have 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. *NX_ModbusRTU_Slave*

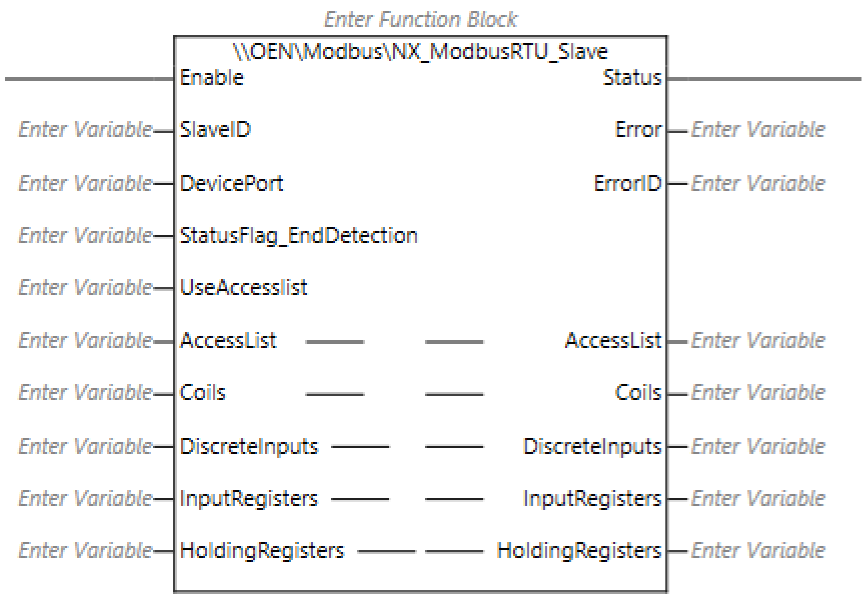
Modbus RTU slave that are based on NX_SerialRcv, NX_SerialSend, NX_SerialBufClear function blocks in Sysmac studio.
For description regarding DevicePort input, see the help for the NX_Serial function blocks.
The input StatusFlag_EndDetection are used to check the silence period of 3.5 characters. See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

1.1. FB Layout



1.2. 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

1.3. 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.

1.4. 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

1.5. Revisions

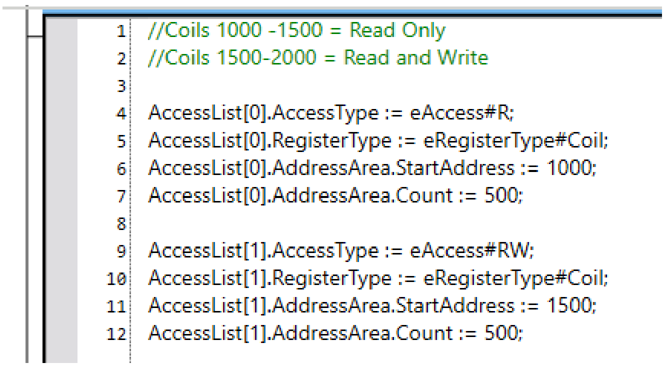
Revision	In Library	Correction
1.0.20	1.00.22	

1.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

1.7. Example

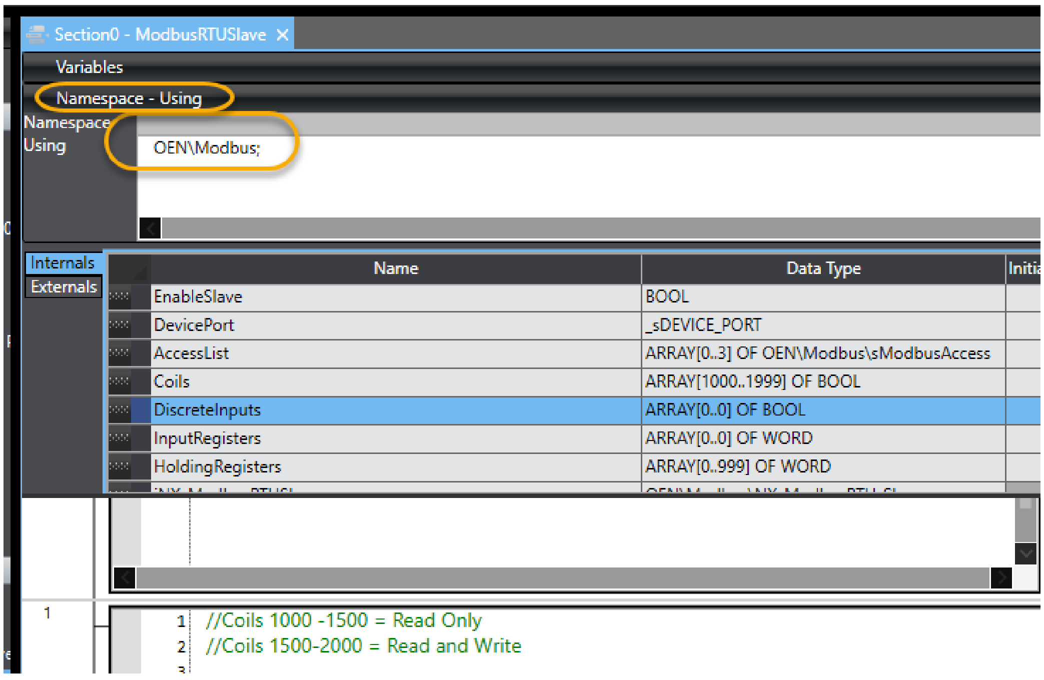
To control the read/write access:
Set the UseAccesslist to “TRUE”.
Create a variable E.G AccessList ARRAY[0..3] OF OEN\Modbus\sModbusAccess
The number of array elements of the In/Out AccessList are dynamic, so you can specify as many as you need.
Sample code for filling data into AccessList:



```
//Coils 1000 -1500 = Read Only
//Coils 1500-2000 = Read and Write

AccessList[0].AccessType := eAccess#R;
AccessList[0].RegisterType := eRegisterType#Coil;
AccessList[0].AddressArea.StartAddress := 1000;
AccessList[0].AddressArea.Count := 500;

AccessList[1].AccessType := eAccess#RW;
AccessList[1].RegisterType := eRegisterType#Coil;
AccessList[1].AddressArea.StartAddress := 1500;
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”:



```
OEN\Modbus;
```

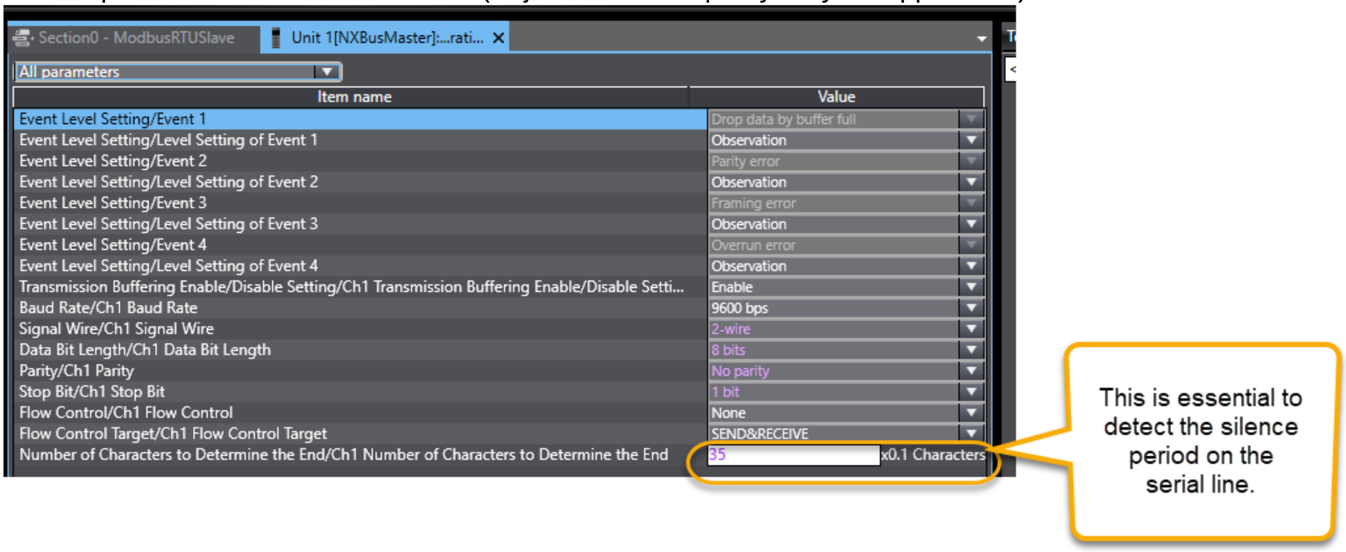
Name	Data Type
EnableSlave	BOOL
DevicePort	_sDEVICE_PORT
AccessList	ARRAY[0..3] OF OEN\Modbus\sModbusAccess
Coils	ARRAY[1000..1999] OF BOOL
DiscreteInputs	ARRAY[0..0] OF BOOL
InputRegisters	ARRAY[0..0] OF WORD
HoldingRegisters	ARRAY[0..999] OF WORD

```
//Coils 1000 -1500 = Read Only
//Coils 1500-2000 = Read and Write
```

Precautions for correct use

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

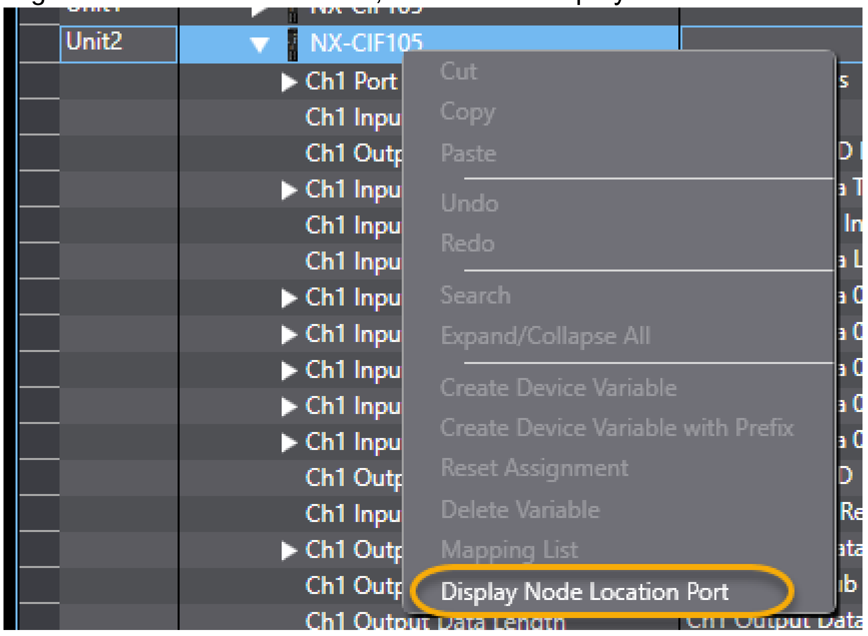
Set the parameters on the serial card: (Adjust Baud rate/parity for your application)



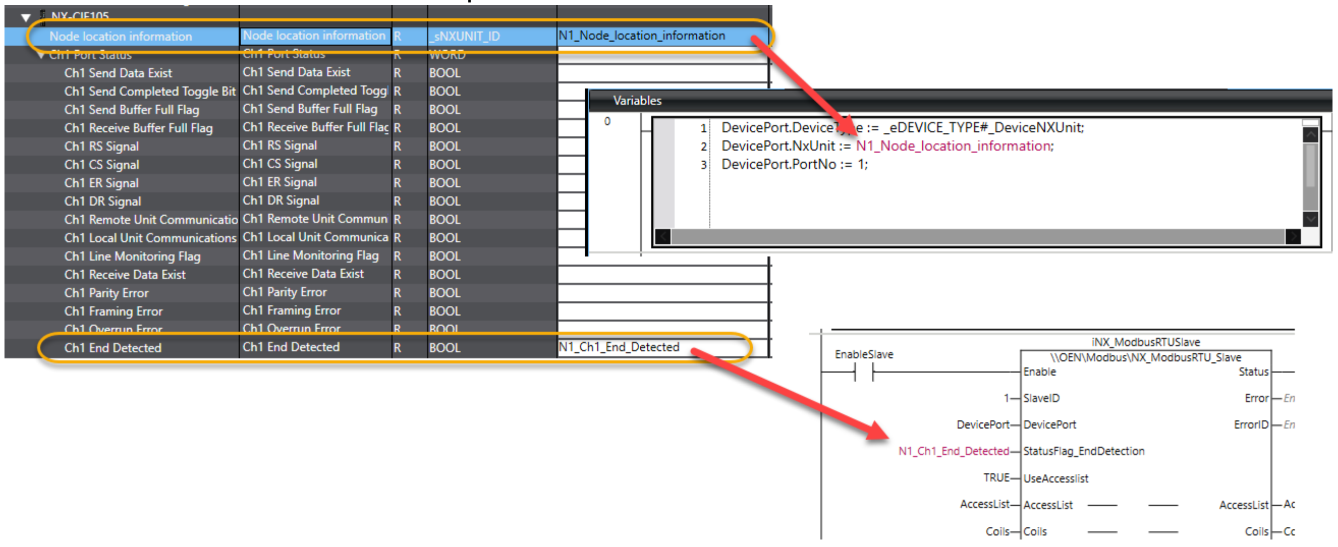
Item name	Value
Event Level Setting/Event 1	Drop data by buffer full
Event Level Setting/Level Setting of Event 1	Observation
Event Level Setting/Event 2	Parity error
Event Level Setting/Level Setting of Event 2	Observation
Event Level Setting/Event 3	Framing error
Event Level Setting/Level Setting of Event 3	Observation
Event Level Setting/Event 4	Overrun error
Event Level Setting/Level Setting of Event 4	Observation
Transmission Buffering Enable/Disable Setting/Ch1 Transmission Buffering Enable/Disable Setti...	Enable
Baud Rate/Ch1 Baud Rate	9600 bps
Signal Wire/Ch1 Signal Wire	2-wire
Data Bit Length/Ch1 Data Bit Length	8 bits
Parity/Ch1 Parity	No parity
Stop Bit/Ch1 Stop Bit	1 bit
Flow Control/Ch1 Flow Control	None
Flow Control Target/Ch1 Flow Control Target	SEND&RECEIVE
Number of Characters to Determine the End/Ch1 Number of Characters to Determine the End	35 x0.1 Characters

This is essential to detect the silence period on the serial line.

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



You do need these two variables to operate the function block.



NX-CIF105				
Node location information	Node location information	R	_sNXUNIT_ID	N1_Node_location_information
Ch1 Port Status	Ch1 Port Status	R	WORD	
Ch1 Send Data Exist	Ch1 Send Data Exist	R	BOOL	
Ch1 Send Completed Toggle Bit	Ch1 Send Completed Toggl	R	BOOL	
Ch1 Send Buffer Full Flag	Ch1 Send Buffer Full Flag	R	BOOL	
Ch1 Receive Buffer Full Flag	Ch1 Receive Buffer Full Flag	R	BOOL	
Ch1 RS Signal	Ch1 RS Signal	R	BOOL	
Ch1 CS Signal	Ch1 CS Signal	R	BOOL	
Ch1 ER Signal	Ch1 ER Signal	R	BOOL	
Ch1 DR Signal	Ch1 DR Signal	R	BOOL	
Ch1 Remote Unit Communicatio	Ch1 Remote Unit Commun	R	BOOL	
Ch1 Local Unit Communications	Ch1 Local Unit Communica	R	BOOL	
Ch1 Line Monitoring Flag	Ch1 Line Monitoring Flag	R	BOOL	
Ch1 Receive Data Exist	Ch1 Receive Data Exist	R	BOOL	
Ch1 Parity Error	Ch1 Parity Error	R	BOOL	
Ch1 Framing Error	Ch1 Framing Error	R	BOOL	
Ch1 Overrun Error	Ch1 Overrun Error	R	BOOL	
Ch1 End Detected	Ch1 End Detected	R	BOOL	N1_Ch1_End_Detected

Variables

```
DevicePort.DeviceType := _eDEVICE_TYPE#_DeviceNXUnit;
DevicePort.NxUnit := N1_Node_location_information;
DevicePort.PortNo := 1;
```

2. *NX_ModbusRTU_Master*

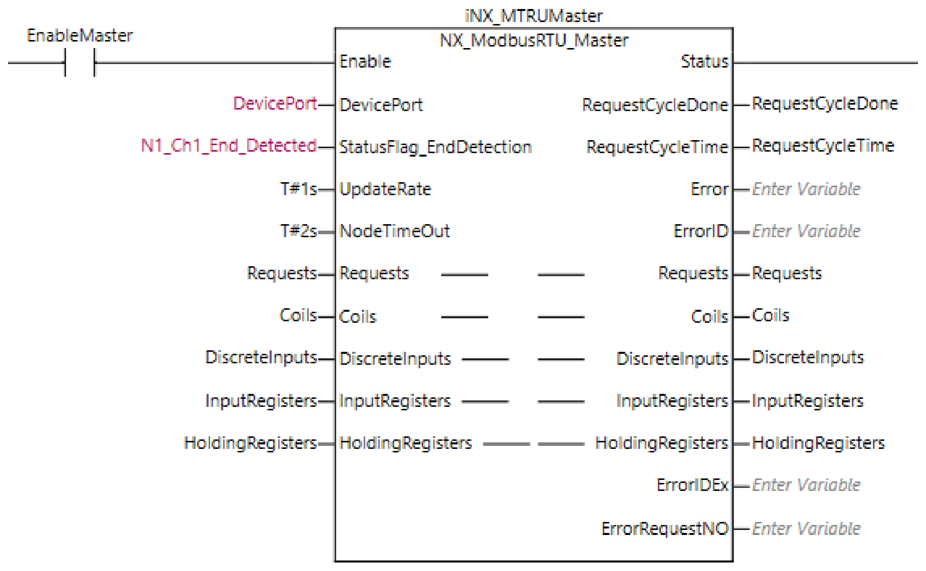
Modbus RTU slave that are based on NX_SerialBufClear, NX_SerialRcv, NX_SerialSend function blocks in Sysmac Studio.
For description regarding DevicePort input, see the help for the NX_Serial function blocks.
The input StatusFlag_EndDetection are used to check the silence period of 3.5 characters. See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

The master will sequentially perform the requests 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".

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
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	BOOL	Time between each poll of the request list.
NodeTimeOut	TIME	Timeout for each request.
Requests	TIME	List of requests to the slave(s)

2.3. 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.

2.4. 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.

2.5. 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.

2.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

2.7. Example

How to set up the requests:

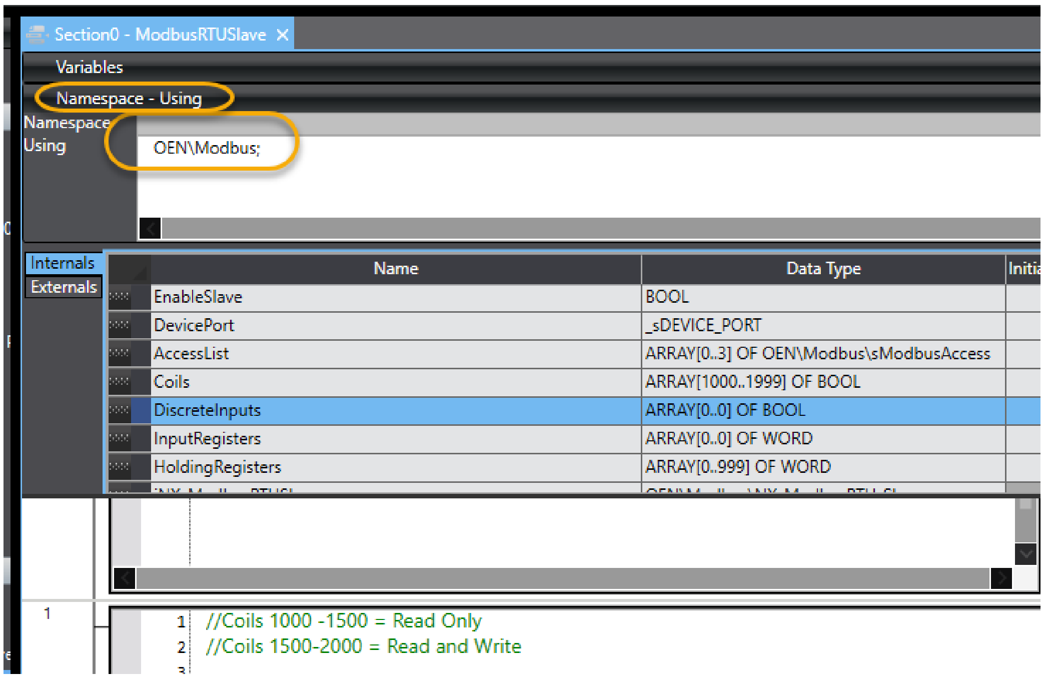
Create a variable for requests E.G. “Requests”, of the datatype ARRAY[X..Y] OF OEN\Modbus\sModbusReq.
The number of array elements of the In/Out “Requests” are dynamic, so you can spesify as many as you need.

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":



Namespace Using: OEN\Modbus;

Name	Data Type
EnableSlave	BOOL
DevicePort	_sDEVICE_PORT
AccessList	ARRAY[0..3] OF OEN\Modbus\sModbusAccess
Coils	ARRAY[1000..1999] OF BOOL
DiscreteInputs	ARRAY[0..0] OF BOOL
InputRegisters	ARRAY[0..0] OF WORD
HoldingRegisters	ARRAY[0..999] OF WORD

```
//Coils 1000 -1500 = Read Only
//Coils 1500-2000 = Read and Write
```

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:

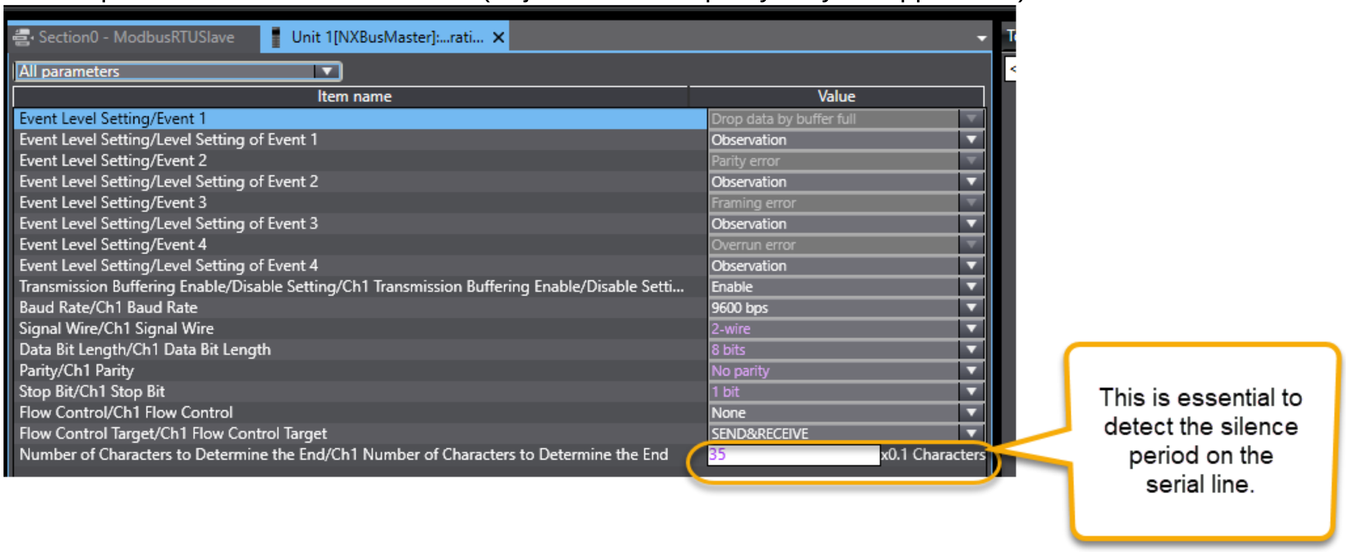
16#1001	Modbus address outside of Array boundary
16#1002	Invalid modbus function code
16#1004	Response with wrong function code
16#1005	Response with wrong size
16#1006	Wrong CRC

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)

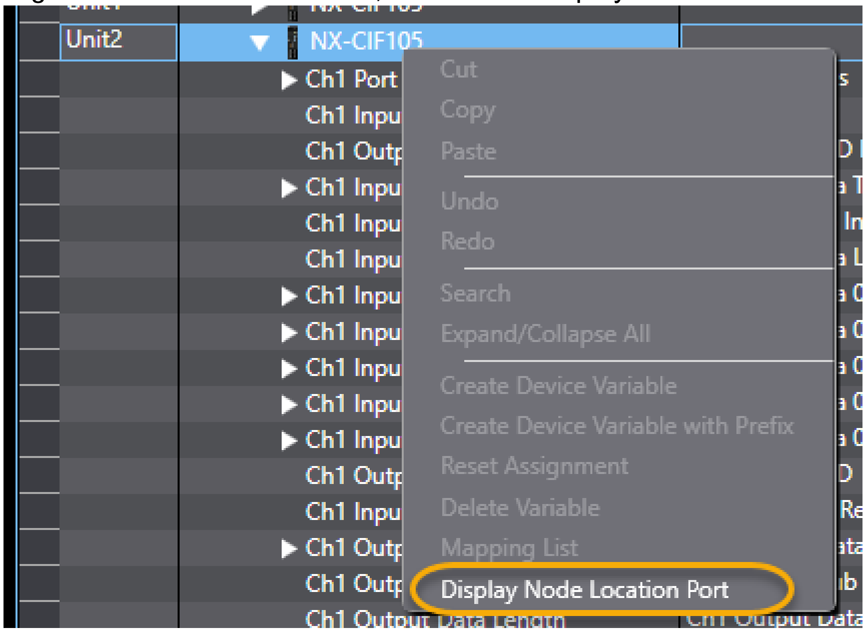
Set the parameters on the serial card: (Adjust Baud rate/parity for your application)



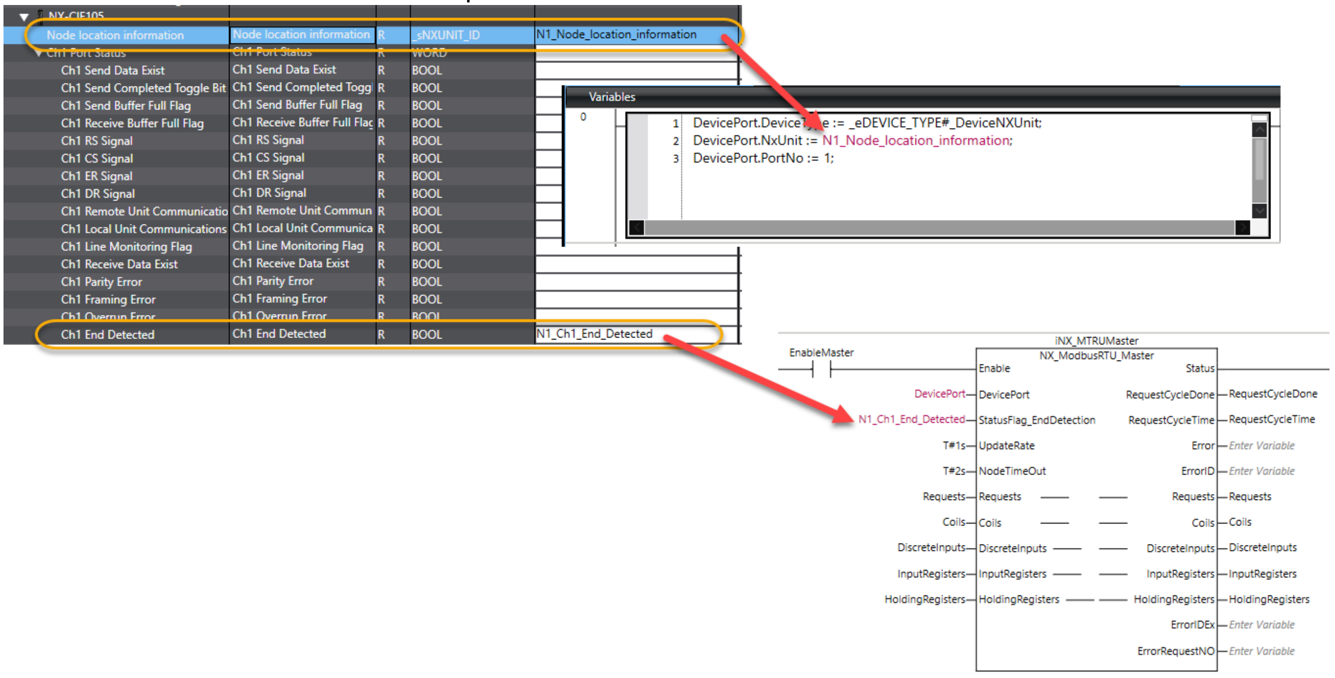
Item name	Value
Event Level Setting/Event 1	Drop data by buffer full
Event Level Setting/Level Setting of Event 1	Observation
Event Level Setting/Event 2	Parity error
Event Level Setting/Level Setting of Event 2	Observation
Event Level Setting/Event 3	Framing error
Event Level Setting/Level Setting of Event 3	Observation
Event Level Setting/Event 4	Overrun error
Event Level Setting/Level Setting of Event 4	Observation
Transmission Buffering Enable/Disable Setting/Ch1 Transmission Buffering Enable/Disable Setti...	Enable
Baud Rate/Ch1 Baud Rate	9600 bps
Signal Wire/Ch1 Signal Wire	2-wire
Data Bit Length/Ch1 Data Bit Length	8 bits
Parity/Ch1 Parity	No parity
Stop Bit/Ch1 Stop Bit	1 bit
Flow Control/Ch1 Flow Control	None
Flow Control Target/Ch1 Flow Control Target	SEND&RECEIVE
Number of Characters to Determine the End/Ch1 Number of Characters to Determine the End	35 x0.1 Characters

This is essential to detect the silence period on the serial line.

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



You do need these two variables to operate the function block.



NX-CIF105				
Node location information	Node location information	R	_sNXUNIT_ID	N1_Node_location_information
Ch1 Port Status	Ch1 Port Status	R	WORD	
Ch1 Send Data Exist	Ch1 Send Data Exist	R	BOOL	
Ch1 Send Completed Toggle Bit	Ch1 Send Completed Toggl	R	BOOL	
Ch1 Send Buffer Full Flag	Ch1 Send Buffer Full Flag	R	BOOL	
Ch1 Receive Buffer Full Flag	Ch1 Receive Buffer Full Flag	R	BOOL	
Ch1 RS Signal	Ch1 RS Signal	R	BOOL	
Ch1 CS Signal	Ch1 CS Signal	R	BOOL	
Ch1 ER Signal	Ch1 ER Signal	R	BOOL	
Ch1 DR Signal	Ch1 DR Signal	R	BOOL	
Ch1 Remote Unit Communicatio	Ch1 Remote Unit Commun	R	BOOL	
Ch1 Local Unit Communications	Ch1 Local Unit Communica	R	BOOL	
Ch1 Line Monitoring Flag	Ch1 Line Monitoring Flag	R	BOOL	
Ch1 Receive Data Exist	Ch1 Receive Data Exist	R	BOOL	
Ch1 Parity Error	Ch1 Parity Error	R	BOOL	
Ch1 Framing Error	Ch1 Framing Error	R	BOOL	
Ch1 Overrun Error	Ch1 Overrun Error	R	BOOL	
Ch1 End Detected	Ch1 End Detected	R	BOOL	N1_Ch1_End_Detected

Variables

```
DevicePort.DeviceType := _eDEVICE_TYPE#_DeviceNXUnit;
DevicePort.NxUnit := N1_Node_location_information;
DevicePort.PortNo := 1;
```

3. *ModbusTCP_Server*

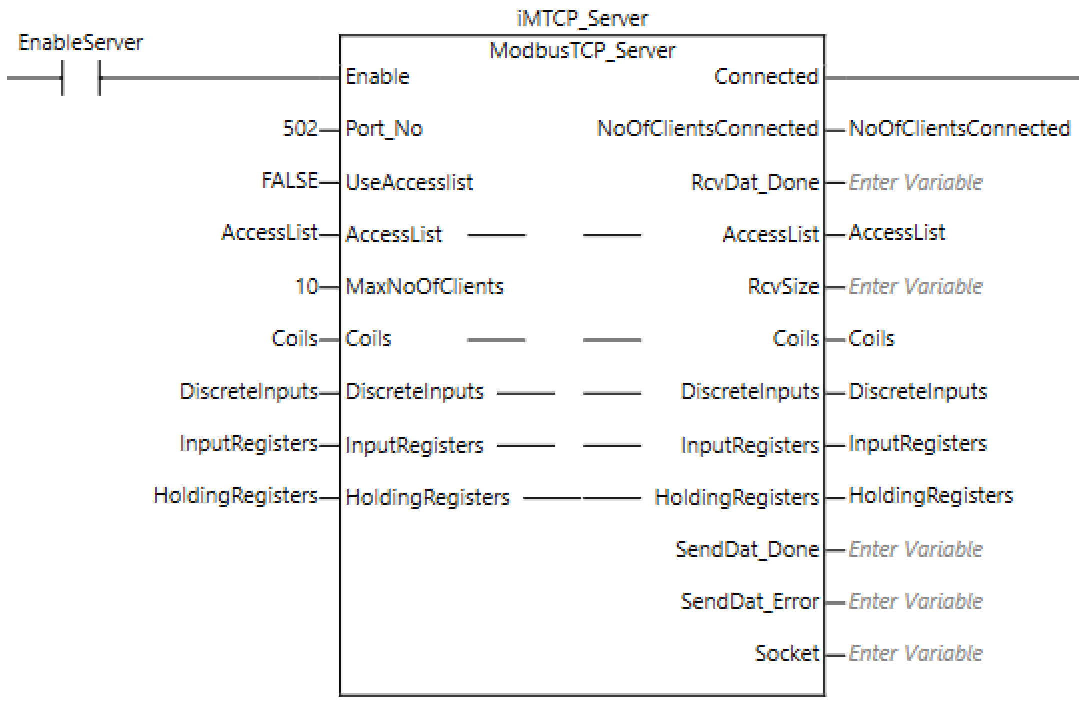
Modbus TCP Server are based on TCP socket FB's: SktTCPAccept, SktGetTCPStatus, SktTCPRcv, SktTCPSend, SktTCPClose.

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

3.1. FB Layout



3.2. 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.

3.3. 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.

3.4. 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.

3.5. Revisions

Revision	In Library	Correction
1.0.20	1.00.22	

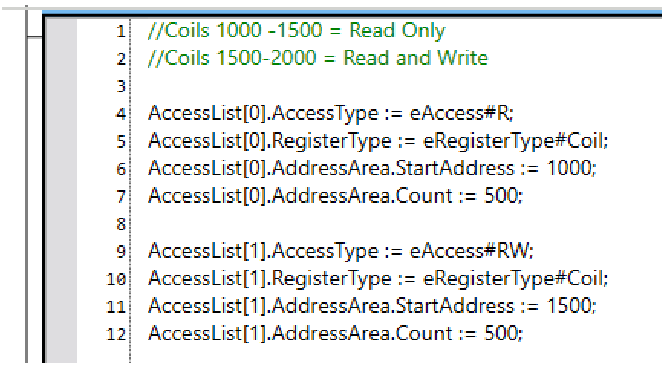
3.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

3.7. Example

To control the read/write access :

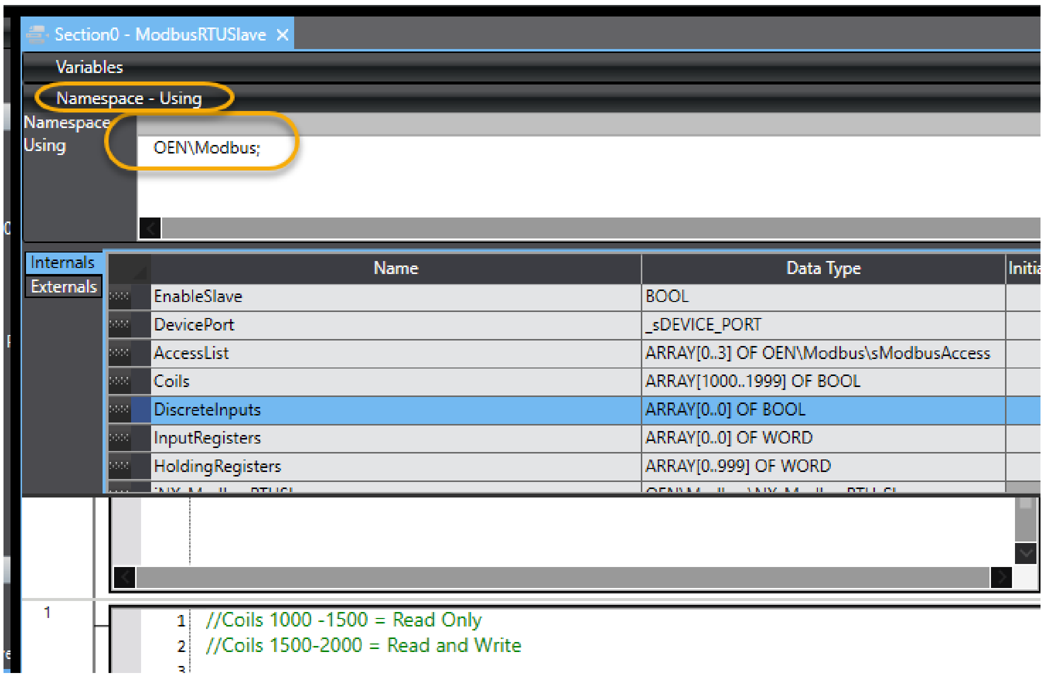
Set the UseAccesslist to “TRUE”.
Create a variable E.G AccessList ARRAY[0..3] OF OEN\Modbus\sModbusAccess
The number of array elements of the In/Out AccessList are dynamic, so you can spesify as many as you need.
Sample code for filling data into the accesslist:



```
//Coils 1000 -1500 = Read Only
//Coils 1500-2000 = Read and Write

AccessList[0].AccessType := eAccess#R;
AccessList[0].RegisterType := eRegisterType#Coil;
AccessList[0].AddressArea.StartAddress := 1000;
AccessList[0].AddressArea.Count := 500;

AccessList[1].AccessType := eAccess#RW;
AccessList[1].RegisterType := eRegisterType#Coil;
AccessList[1].AddressArea.StartAddress := 1500;
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”:



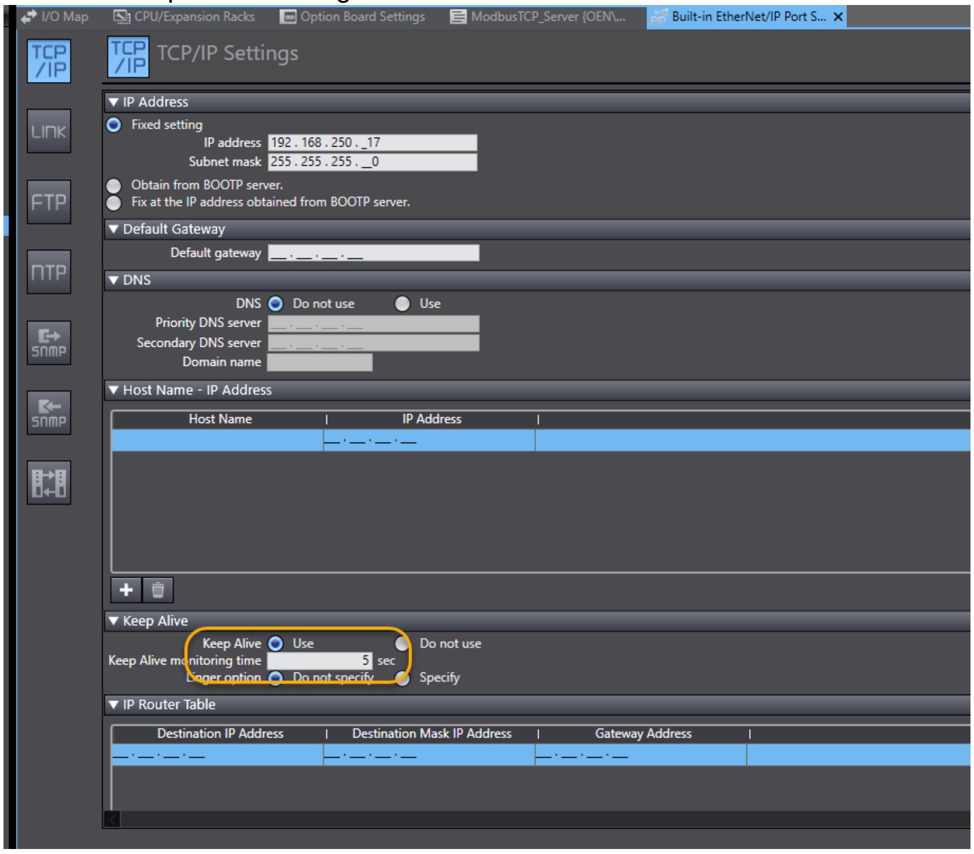
```
OEN\Modbus;
```

Name	Data Type
EnableSlave	BOOL
DevicePort	_sDEVICE_PORT
AccessList	ARRAY[0..3] OF OEN\Modbus\sModbusAccess
Coils	ARRAY[1000..1999] OF BOOL
DiscreteInputs	ARRAY[0..0] OF BOOL
InputRegisters	ARRAY[0..0] OF WORD
HoldingRegisters	ARRAY[0..999] OF WORD

```
//Coils 1000 -1500 = Read Only
//Coils 1500-2000 = Read and Write
```

Precautions for correct use

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



TCP/IP Settings

▼ IP Address

[✓ Fixed setting]
IP address 192 . 168 . 250 . _17
Subnet mask 255 . 255 . 255 . _0
Obtain from BOOTP server.
Fix at the IP address obtained from BOOTP server.

▼ Default Gateway

Default gateway __ . __ . __ . __

▼ DNS

DNS [✓ Do not use] Use
Priority DNS server __ . __ . __ . __
Secondary DNS server __ . __ . __ . __
Domain name

▼ Host Name - IP Address

Host Name	IP Address
	__ . __ . __ . __

▼ Keep Alive

Keep Alive [✓ Use] Do not use
Keep Alive monitoring time 5 sec
Linger option [✓ Do not specify] Specify

▼ IP Router Table

Destination IP Address	Destination Mask IP Address	Gateway Address
__ . __ . __ . __	__ . __ . __ . __	__ . __ . __ . __

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.

4. *ModbusTCP_Client*

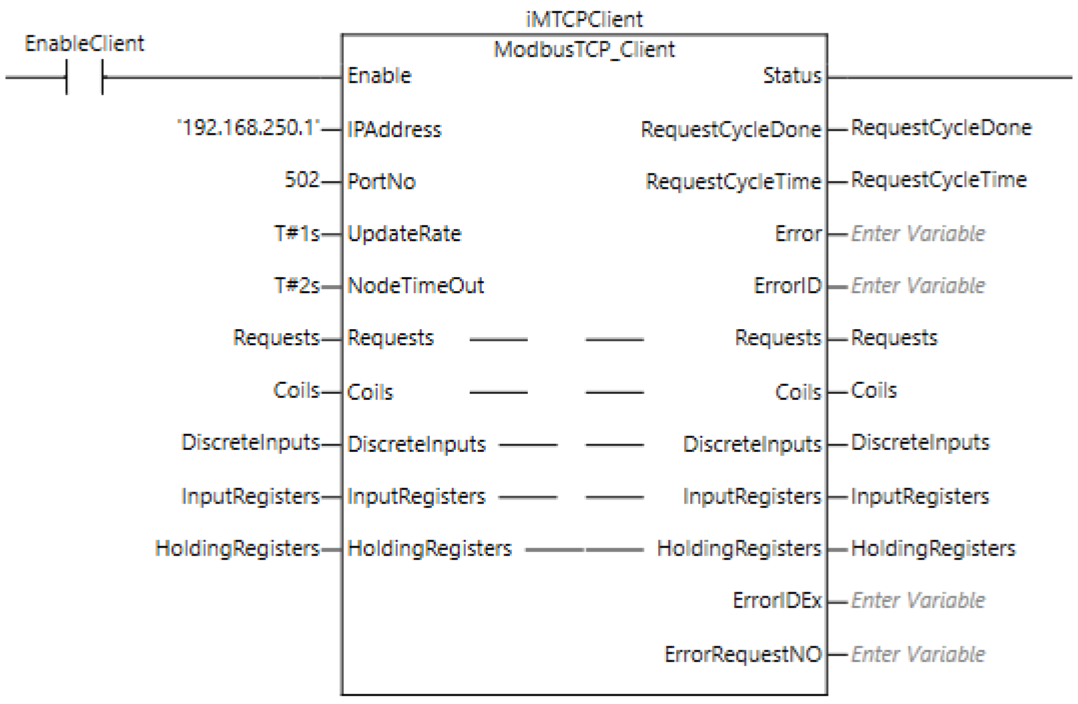
Modbus TCP Client that are based on TCP socket FB's: SktTCPConnect, SktGetTCPStatus, SktTCPRcv, SktTCPSend, SktTCPClose.

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

4.1. FB Layout



4.2. 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)

4.3. 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.

4.4. 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.

4.5. 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.

4.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

4.7. Example

How to set up the requests :

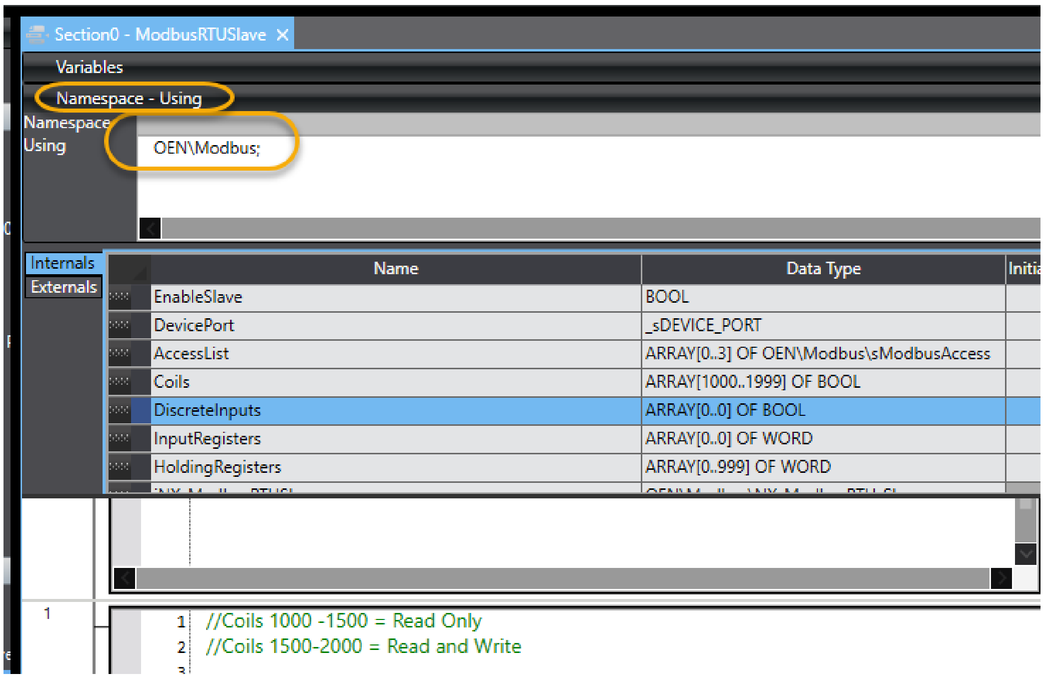
Create a variable for requests E.G. “Requests”, of the datatype ARRAY[X..Y] OF OEN\Modbus\sModbusReq.
The number of array elements of the In/Out “Requests” are dynamic, so you can spesify as many as you need.

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”:


Section0 - ModbusRTUSlave

Variables

Namespace - Using

Namespace Using: OEN\Modbus;

Internals | Externals

Name	Data Type
EnableSlave	BOOL
DevicePort	_sDEVICE_PORT
AccessList	ARRAY[0..3] OF OEN\Modbus\sModbusAccess
Coils	ARRAY[1000..1999] OF BOOL
DiscreteInputs	ARRAY[0..0] OF BOOL
InputRegisters	ARRAY[0..0] OF WORD
HoldingRegisters	ARRAY[0..999] OF WORD

```
//Coils 1000 -1500 = Read Only
//Coils 1500-2000 = Read and Write
```

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”.

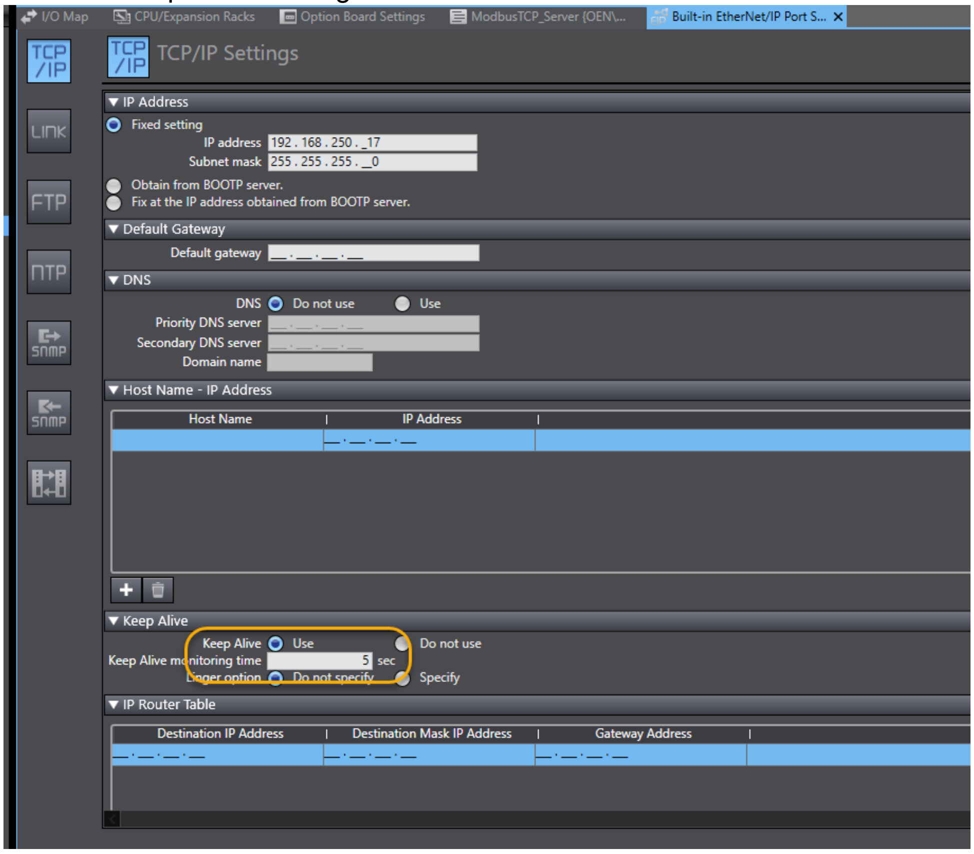
List of ErrorID’s in addition to the above:

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:



TCP/IP Settings

▼ IP Address

[✓ Fixed setting]
IP address 192 . 168 . 250 . _17
Subnet mask 255 . 255 . 255 . _0
Obtain from BOOTP server.
Fix at the IP address obtained from BOOTP server.

▼ Default Gateway

Default gateway __ . __ . __ . __

▼ DNS

DNS [✓ Do not use] Use
Priority DNS server __ . __ . __ . __
Secondary DNS server __ . __ . __ . __
Domain name

▼ Host Name - IP Address

Host Name	IP Address
	__ . __ . __ . __

▼ Keep Alive

Keep Alive [✓ Use] Do not use
Keep Alive monitoring time 5 sec
Linger option [✓ Do not specify] Specify

▼ IP Router Table

Destination IP Address	Destination Mask IP Address	Gateway Address
__ . __ . __ . __	__ . __ . __ . __	__ . __ . __ . __

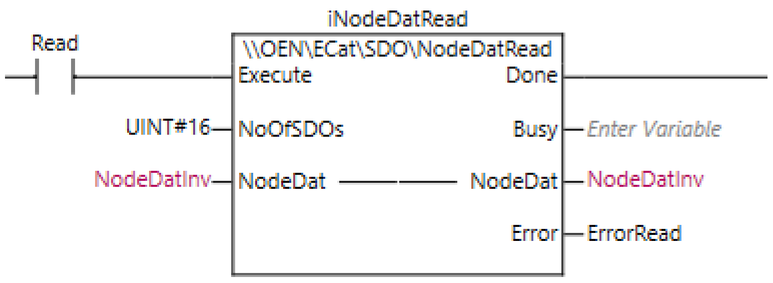
When using “Keep alive” the TCP socket will send a keep alive message to the client, to check if the client is still responding. If the client does not respond within the “Keep Alive monitoring time” the socket will close, and reopen for new connection for the client.

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

5. *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.

5.1. FB Layout



5.2. 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.

5.3. 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.

5.4. 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

5.5. Revisions

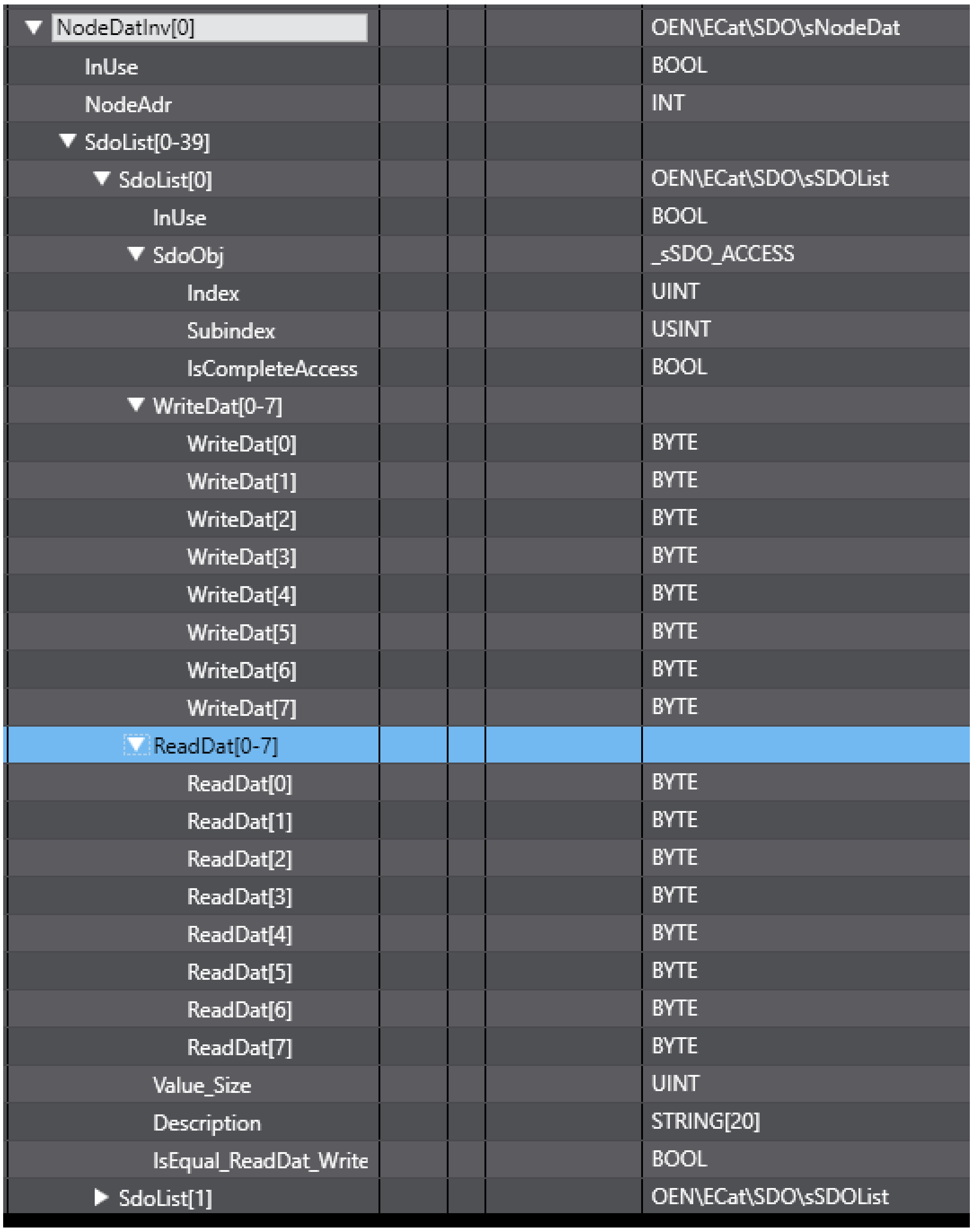
Revision	In Library	Correction
1.0.1	1.00.22	Replaced NodeDat with dynamic ARRAY.Removed Input NoOfNodes

5.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

5.7. Example

Picture of OEN\ECat\SDO\sNodeDat:



▼ NodeDatInv[0]				OEN\ECat\SDO\sNodeDat
InUse				BOOL
NodeAdr				INT
▼ SdoList[0-39]				
▼ SdoList[0]				OEN\ECat\SDO\sSDOList
InUse				BOOL
▼ SdoObj				_sSDO_ACCESS
Index				UINT
Subindex				USINT
IsCompleteAccess				BOOL
▼ WriteDat[0-7]				
WriteDat[0]				BYTE
WriteDat[1]				BYTE
WriteDat[2]				BYTE
WriteDat[3]				BYTE
WriteDat[4]				BYTE
WriteDat[5]				BYTE
WriteDat[6]				BYTE
WriteDat[7]				BYTE
▼ ReadDat[0-7]				
ReadDat[0]				BYTE
ReadDat[1]				BYTE
ReadDat[2]				BYTE
ReadDat[3]				BYTE
ReadDat[4]				BYTE
ReadDat[5]				BYTE
ReadDat[6]				BYTE
ReadDat[7]				BYTE
Value_Size				UINT
Description				STRING[20]
IsEqual_ReadDat_Write				BOOL
▶ SdoList[1]				OEN\ECat\SDO\sSDOList

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;
```

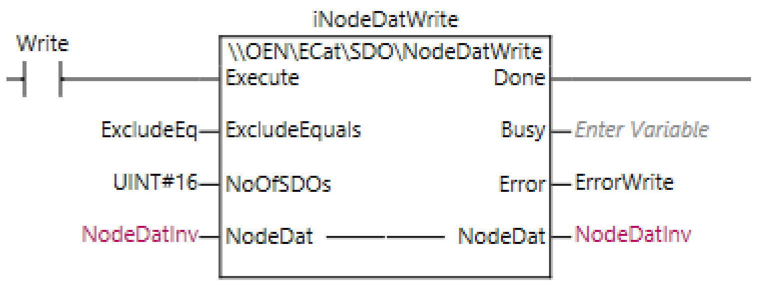
```
NodeDat[i].SdoList[j].InUse := TRUE;

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;
```

6. *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:"

6.1. FB Layout



6.2. 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.

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

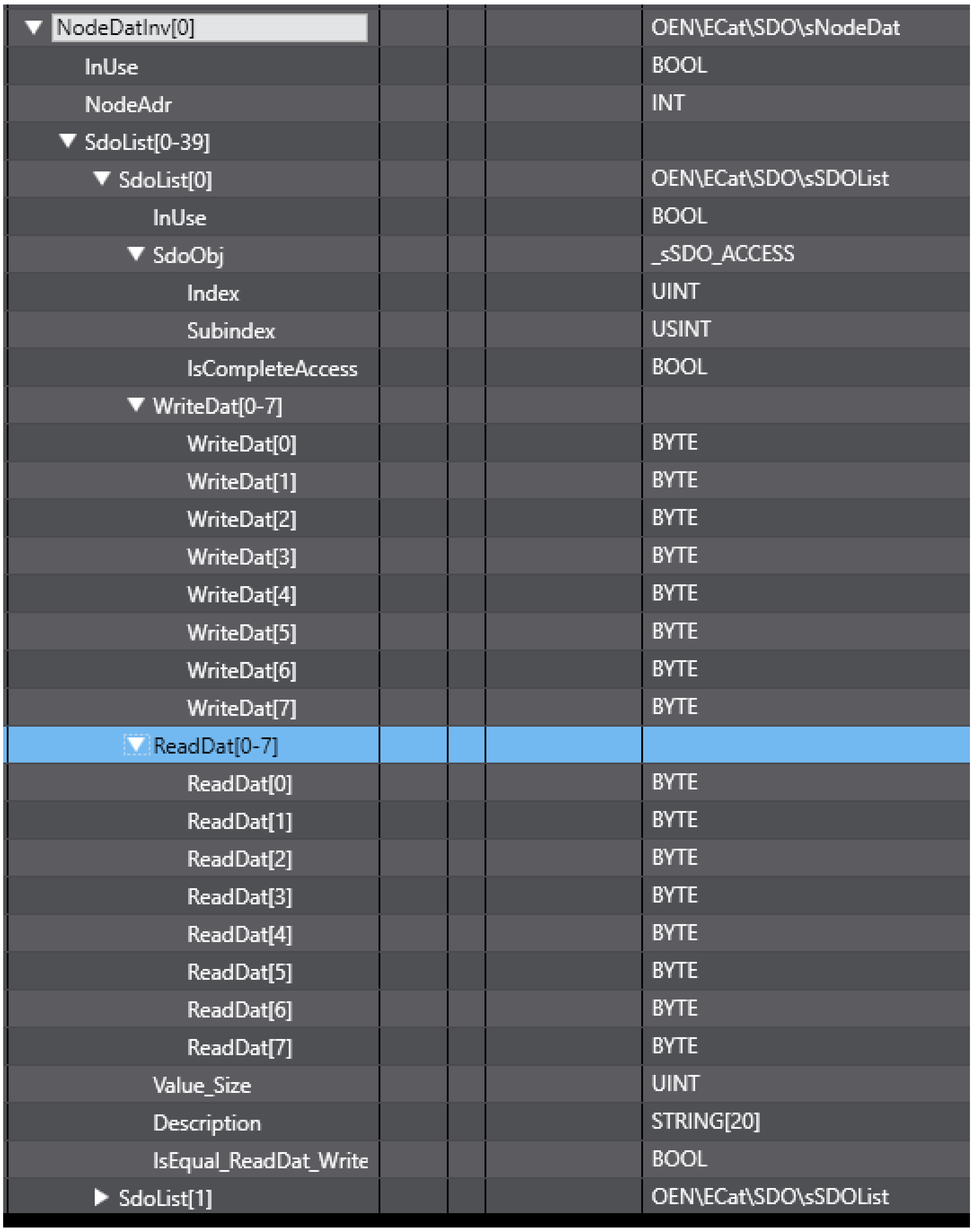
Revision	In Library	Correction
1.0.1	1.00.22	Replaced NodeDat with dynamic ARRAY. Removed Input NoOfNodes

6.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

6.7. Example

Picture of OEN\ECat\SDO\sNodeDat:



▼ NodeDatInv[0]				OEN\ECat\SDO\sNodeDat
InUse				BOOL
NodeAdr				INT
▼ SdoList[0-39]				
▼ SdoList[0]				OEN\ECat\SDO\sSDOList
InUse				BOOL
▼ SdoObj				_sSDO_ACCESS
Index				UINT
Subindex				USINT
IsCompleteAccess				BOOL
▼ WriteDat[0-7]				
WriteDat[0]				BYTE
WriteDat[1]				BYTE
WriteDat[2]				BYTE
WriteDat[3]				BYTE
WriteDat[4]				BYTE
WriteDat[5]				BYTE
WriteDat[6]				BYTE
WriteDat[7]				BYTE
▼ ReadDat[0-7]				
ReadDat[0]				BYTE
ReadDat[1]				BYTE
ReadDat[2]				BYTE
ReadDat[3]				BYTE
ReadDat[4]				BYTE
ReadDat[5]				BYTE
ReadDat[6]				BYTE
ReadDat[7]				BYTE
Value_Size				UINT
Description				STRING[20]
IsEqual_ReadDat_Write				BOOL
▶ SdoList[1]				OEN\ECat\SDO\sSDOList

7. *NodeDatCmp_Read_Write*

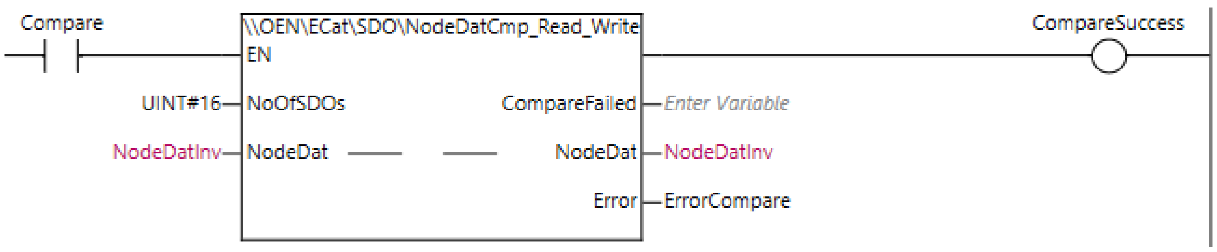
Compares all the ReadDat values with the WriteDat values for all nodes and all SDO's.

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:"

7.1. FB Layout



7.2. 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.

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
(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.

7.5. Revisions

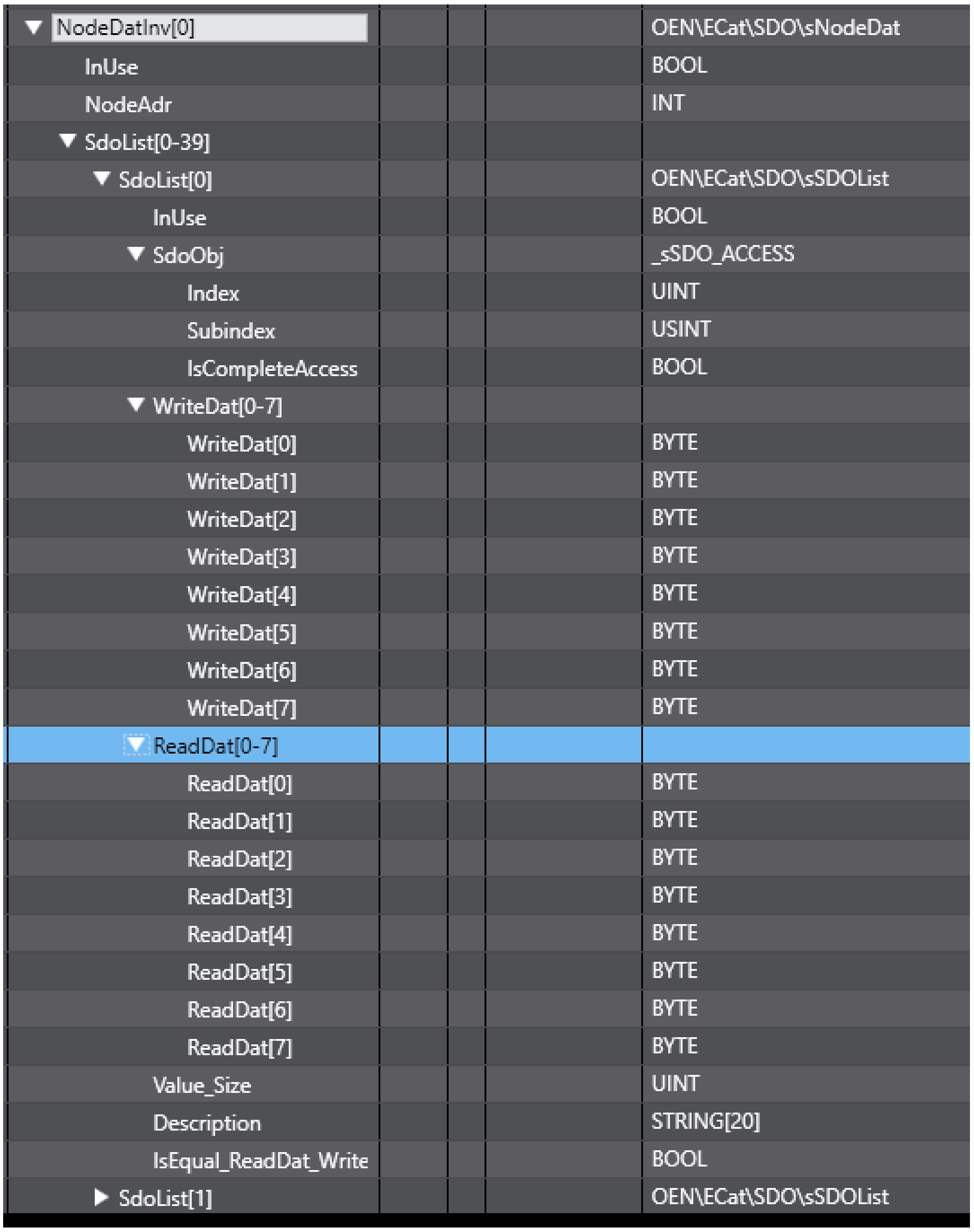
Revision	In Library	Correction
1.0.1	1.00.22	Replaced NodeDat with dynamic ARRAY. Removed Input NoOfNodes

7.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

7.7. Example

Picture of OEN\ECat\SDO\sNodeDat:



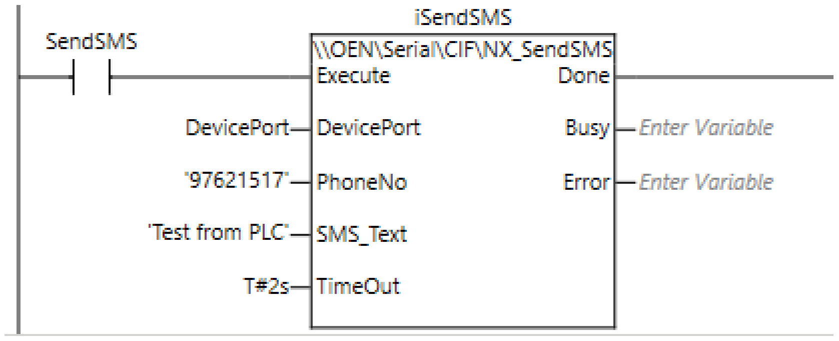
▼ NodeDatInv[0]			OEN\ECat\SDO\sNodeDat
InUse			BOOL
NodeAdr			INT
▼ SdoList[0-39]			
▼ SdoList[0]			OEN\ECat\SDO\sSDOList
InUse			BOOL
▼ SdoObj			_sSDO_ACCESS
Index			UINT
Subindex			USINT
IsCompleteAccess			BOOL
▼ WriteDat[0-7]			
WriteDat[0]			BYTE
WriteDat[1]			BYTE
WriteDat[2]			BYTE
WriteDat[3]			BYTE
WriteDat[4]			BYTE
WriteDat[5]			BYTE
WriteDat[6]			BYTE
WriteDat[7]			BYTE
▼ ReadDat[0-7]			
ReadDat[0]			BYTE
ReadDat[1]			BYTE
ReadDat[2]			BYTE
ReadDat[3]			BYTE
ReadDat[4]			BYTE
ReadDat[5]			BYTE
ReadDat[6]			BYTE
ReadDat[7]			BYTE
Value_Size			UINT
Description			STRING[20]
IsEqual_ReadDat_Write			BOOL
▶ SdoList[1]			OEN\ECat\SDO\sSDOList

8. *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.

8.1. FB Layout



8.2. 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.

8.3. In-Out Variables

Name	Data type	Description

8.4. 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

8.5. Revisions

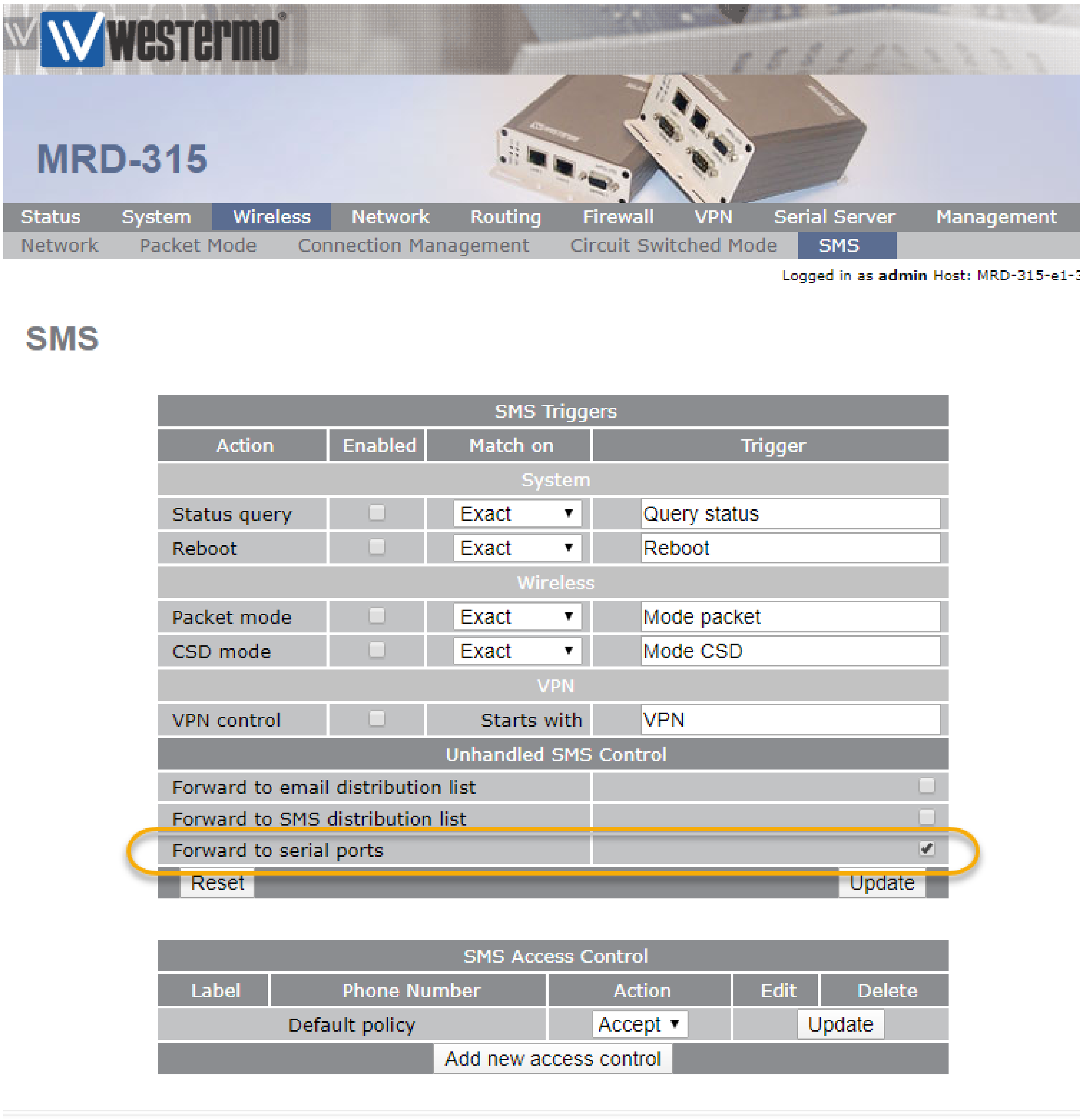
Revision	In Library	Correction
1.0.21	1.00.22	

8.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

8.7. Example

Settings used in Westermo MRD-315:

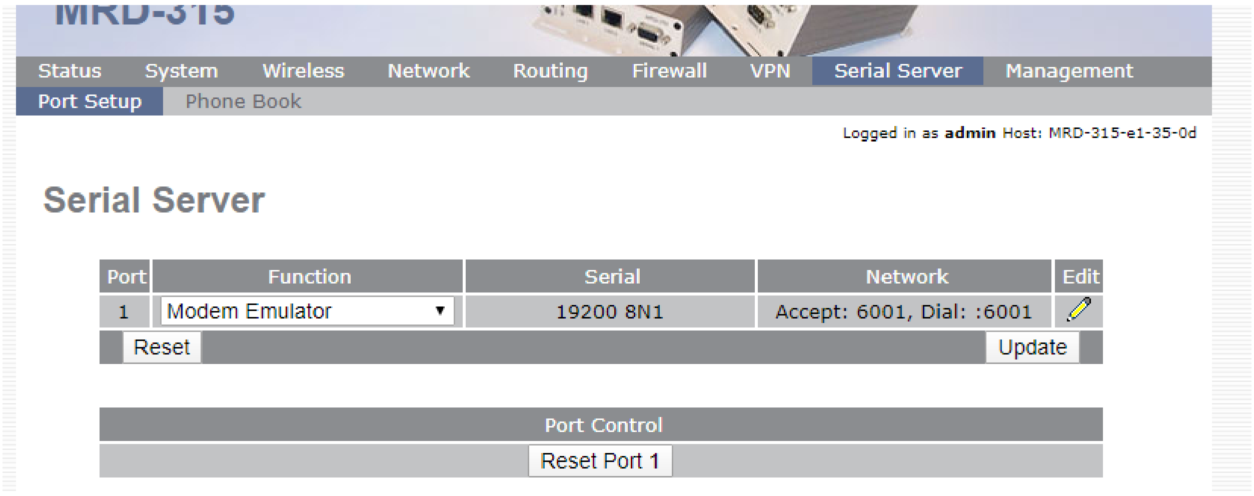


Logged in as **admin** Host: MRD-315-e1-3

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				



Logged in as **admin** Host: MRD-315-e1-35-0d

Serial Server

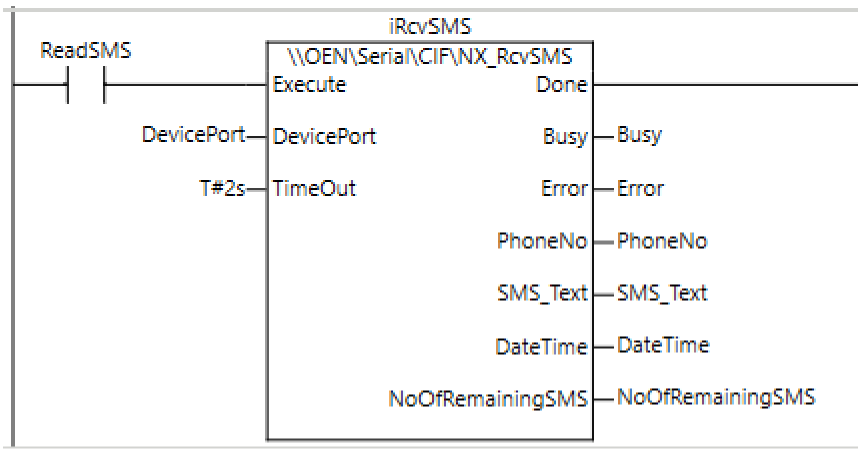
Port	Function	Serial	Network	Edit
1	Modem Emulator	19200 8N1	Accept: 6001, Dial: :6001	✎

9. *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.

9.1. FB Layout



9.2. 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.

9.3. In-Out Variables

Name	Data type	Description

9.4. 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.

9.5. Revisions

Revision	In Library	Correction
1.0.21	1.00.22	Complete redesign

9.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

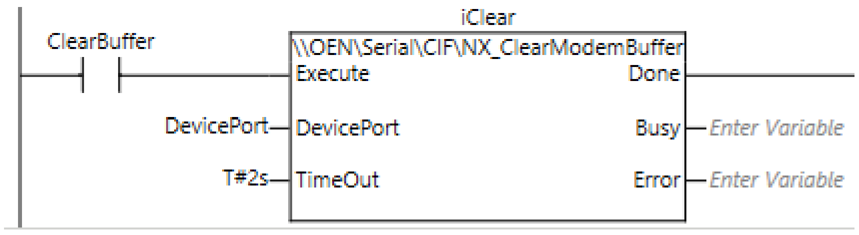
9.7. Example

10. *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.

10.1. FB Layout



10.2. 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.

10.3. In-Out Variables

Name	Data type	Description

10.4. 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

10.5. Revisions

Revision	In Library	Correction
1.0.21	1.00.22	

10.6. Credits

	Name
Omron - Norway	Bjarte Myklebust

10.7. Example

11. *Template*

text

11.1. FN Layout

11.2. Input Variables

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

11.3. In-Out Variables

Name	Data type	Description

11.4. Output Variables

Name	Data Type	Description
	BOOL	

11.5. Revisions

Revision	In Library	Correction
1.0.0	1.00.0	

11.6. Credits

	Name
Omron - Norway	

11.7. Example