SystemCORP ENERGY

Modbus

eNode Configuration Manual

This PDF Document contains internal hyperlinks for ease of navigation.
For example, click on any item listed in the Table of Contents to go to that page.

Documentation Control

Author:	Nicholas Rixson
Revision:	1.03
Revision History:	1.00 - Initial release 1.01 - Interoperability added. NR 1.02 - Updated for Modbus eNode module v1.01.000. NR 1.03 - Add multi-address types (Double positions and 32-bit variables)
Creation Date:	7 June 2016
Last Revision Date:	18 October 2018
Product Reference:	197-0300
Document Status:	Released

Table of Contents

Table of Figures

1 Introduction

This document describes how to use the **Modbus eNode Designer Module** to configure SystemCORP's *Modbus ADH Application* within the eNode Designer.

1.1 Scope

This document is divided into 3 major sections:

- **General Description**;
- **Configuration Guide**; and
- **Modbus ADH Application Interoperability**

1.2 Document Reference

[1] Document Title: eNode Designer User Manual: 197-0100
Revision: Version 1.00 or higher

[2] Document Title: Modicon Modbus Protocol – Reference Guide, PI_MBUS-300
Revision: Rev. J – 06/1996

1.3 List of Abbreviations

ADH = Application Data Hub
IED = Intelligent Electronic Device
IP = Internet Protocol
TCP = Transmission Control Protocol

2 General Description

The Modbus eNode Module can be used to add a Modbus ADH Application as a master or slave. For naming consistency across eNode Designer, the Modbus master is called a client, and the Modbus slave is called a server.

The Modbus client can communicate with many Modbus servers, all whose data point details can be configured using this module.

2.1 Configuration Theory

The configuration properties always describe a *server* (slave). When configuring the Modbus server, you are configuring the properties of the server itself. When configuring the Modbus client, you are describing the properties of all the remote servers with which the client is communicating.

Configuring the protocol specific information (such as object addresses) is handled in the Modbus module. This is explained in this document.

Communication port properties (such as Baud Rate) are configured on the communication port itself. The Device module handles the communication port properties, so heavy details are outside the scope of this document. However, screenshots of the typical configuration method are shown in section 4. The relevant properties of the communication ports automatically apply to the application. For example, in a TCP Modbus server application, the IP Address the application binds to is taken from the parent Ethernet port.

2.2 General Screen Description

A small example configuration is shown below to help describe the layout of the screen.

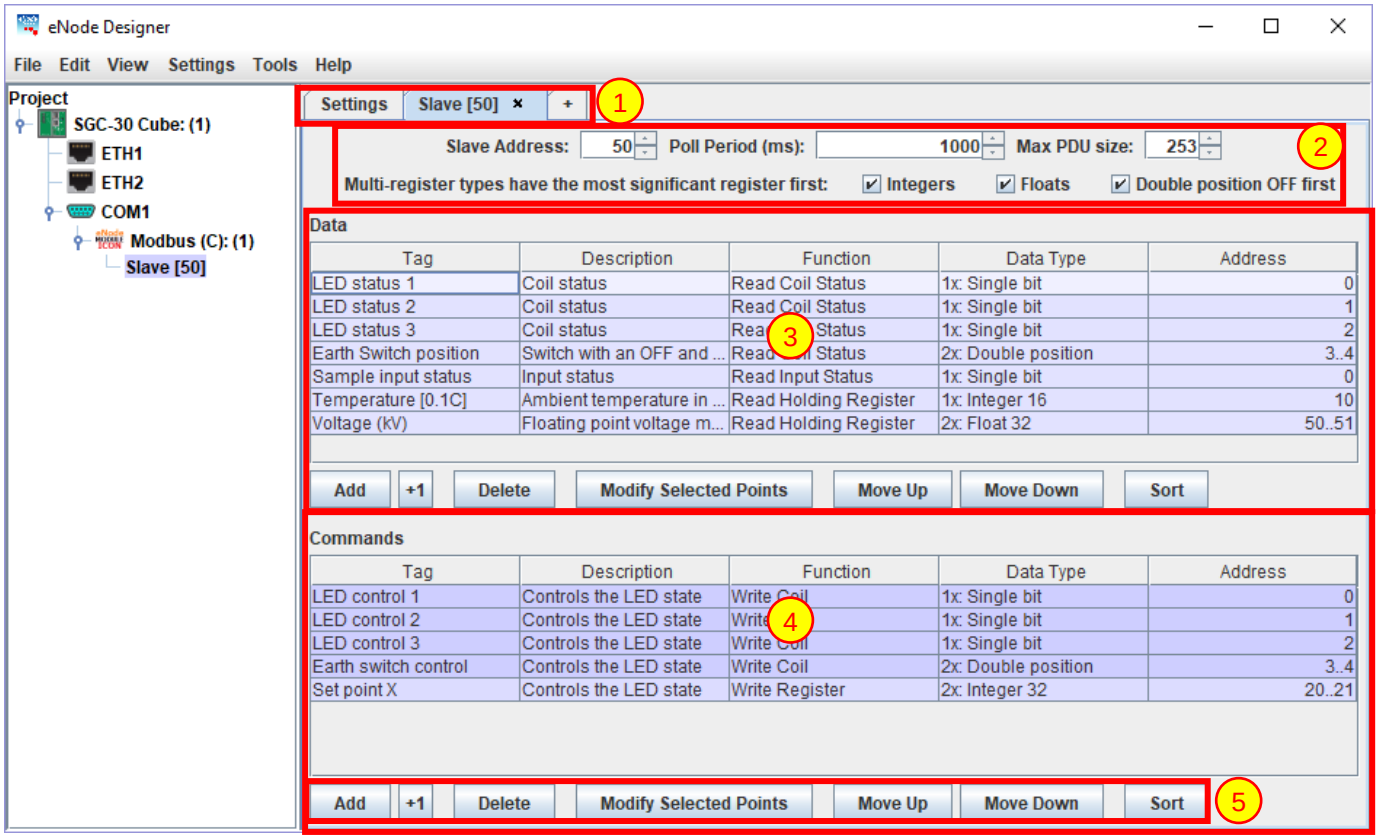


Figure 2-1 - Example Screen

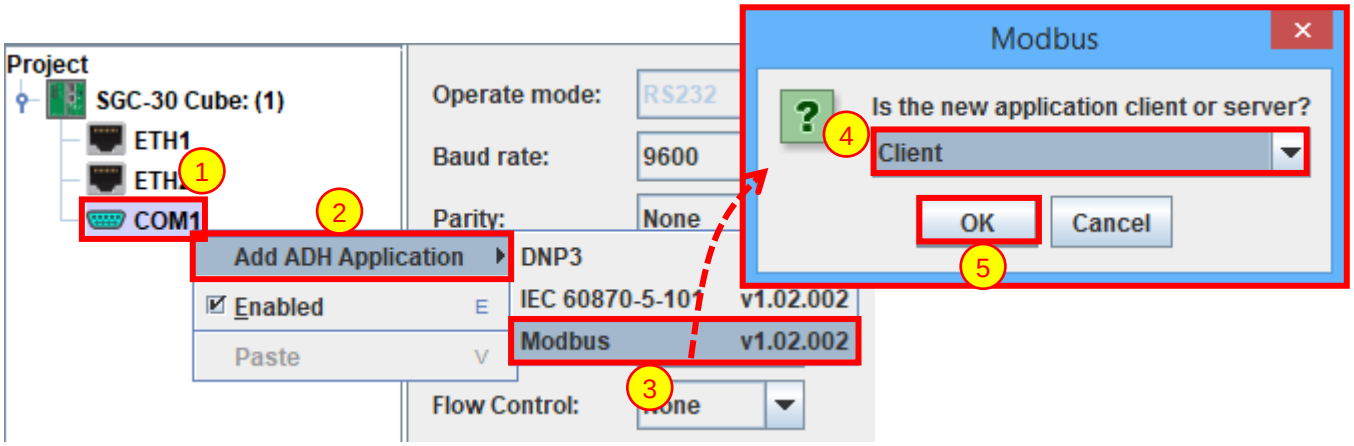
1. **Tabs** – The setting tab for general configuration, and then a set of “Slave” tabs. In the client, each slave tab represents the properties of a single *remote* Modbus server. In the server, there is a single tab describing the properties of the server itself.
2. **Server Properties** – Describes the protocol IED-specific properties of the server.
3. **Data Table** – Shows all (information) data associated with the IED.
4. **Commands Table** – Shows all commands associated with the IED.
5. **Buttons** – Used to add, remove and modify data points.

3 Modbus Configuration Guide

3.1 Adding the Module in eNode Designer

The Modbus module can be added to *Ethernet* and *Serial* ports.

The application can be set up as a Client or a Server – the choice will be presented when adding it to the project.



1. Right click the desired **communication port**.
2. Open the **Add ADH Application** menu.
3. Select **Modbus**.
4. Select **Client** or **Server** from the drop-down menu.
5. Click **OK**.

3.2 Settings Tab

The settings tab contains channel and application-specific settings. Only options relevant to the communication port and client/server type are shown to the user.

	Ethernet	Serial
Client	Frame format: TCP Response Timeout (ms): 500 Connect timeout (ms): 5000 Maximum server transactions: 100	Frame format: RTU Response Timeout (ms): 500 Message retries: 3
Server	Frame format: TCP Maximum connections: 10	Frame format: RTU

Table 3-1 - Settings tab configuration options

3.2.1.1 Frame Format

Description	The Modbus frame format of the communication line. In serial communications, it can be RTU or ASCII. For Ethernet communications, it is automatically set to TCP.
Data Entry	Drop down menu
Choices	*RTU, ASCII or TCP*
Input Option	Mandatory

3.2.1.2 Response Timeout (ms)

Client only

Description	The amount of time (in milliseconds) a master will wait for a slave's response before attempting a retransmission.
Data Entry	Integer
Range	*1 to 65535*
Input Option	Mandatory

3.2.1.3 Connect Timeout (ms)

Ethernet client only

Description	The amount of time (in milliseconds) a master will allow for a connection to be established to the slave.
Data Entry	Integer
Range	*100 to 65535*
Input Option	Mandatory

3.2.1.4 Maximum server transactions

Ethernet client only

Description	The maximum number of requests that may be sent to the server to be queued, without response.
Data Entry	Integer
Range	*1 to 1000*
Input Option	Mandatory

3.2.1.5 Message retries (ms)

Serial client only

Description	The number of times to attempt a message when no reply is received within the response timeout. This value only applies to known online points. Offline points are always scanned only once.
Data Entry	Integer
Range	*1 to 10*
Input Option	Mandatory

3.2.1.6 Maximum Connections

Ethernet server only

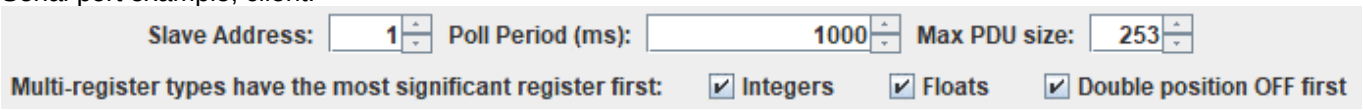
Description	The maximum number of clients which may connect to this server.

Data Entry	Integer
Range	*1 to 1000*
Input Option	Mandatory

3.3 Slave Properties

The slave properties are at the top of the slave configuration tab. The options available are limited to what is relevant for the communication port and client/server type.

Serial port example, client:



Ethernet port example, client:

Each property is described in detail below.

3.3.1.1 Slave Address

Description	The slave address of the server IED. For servers it describes its own slave address. For clients, it describes the slave address of the remote server.
Data Entry	Integer
Range	*1 to 255*
Input Option	Mandatory

3.3.1.2 Poll Period (ms)

Client only

Description	The period (in milliseconds) to poll all values on the slave. If the whole poll cycle takes longer than this period, the next poll cycle will begin immediately.
Data Entry	Integer
Range	*0 to 2147483647*
Input Option	Mandatory

3.3.1.3 Max PDU size

Client only

Description	The maximum number of bytes that the server can handle in a request or response PDU. Messages will be created sufficiently small to only request PDUs in which the request and response will be within this size requirement. Default: 253.
Data Entry	Integer
Range	*5 to 253*
Input Option	Mandatory

3.3.1.4 IP Address

Ethernet client only

Description	The IP Address of the remote server IED. This option is only available in clients, since in servers the IP Address is taken from the Ethernet port.
Data Entry	IP Address String
Range	*Valid IPv4 Addresses (0.0.0.0 to 255.255.255.255)*
Input Option	Mandatory

3.3.1.5 Port

Ethernet only

Description	The IP Port used by the Modbus server IED. Defaults to 502.
Data Entry	Integer
Range	*1 to 65535*
Input Option	Mandatory

3.3.1.6 Multi-register types have most significant register first: Integers

Description	Flag that multi-register integer types have the most significant register first. Each register in Modbus has the most significant byte first, but there is no rule for multi-register values. An example when the register containing “AB” has greater weight: Most significant register first: “AB CD” Least significant register first: “CD AB”
Data Entry	Checkbox
Input Option	Mandatory

3.3.1.7 Multi-register types have most significant register first: Floats

Description	Flag that multi-register Float 32 types (IEEE-754 single precision) have the most significant register first (that is, the part with sign and exponent). Each register in Modbus has the most significant byte first, but there is no rule for multi-register values. An example when the register containing “AB” has greater weight: Most significant register first: “AB CD” Least significant register first: “CD AB”
Data Entry	Checkbox
Input Option	Mandatory

3.3.1.8 Double position OFF first

Description	Flag that double position types have the “OFF” contact at the first address, followed by the “ON” contact at the higher address. If unchecked, the “ON” contact is at the first address, and the “OFF” contact is as the higher address.
Data Entry	Checkbox
Input Option	Mandatory

3.4 Client Configuration

Adding a Modbus client application will immediately show a view similar to the following figure. In this example, the application has been added to a serial port. If it were added to an Ethernet port, there would be some different server IED properties at the top of the screen, as discussed in section 3.3. Each slave tab shows a single Modbus server with which this client is communicating.

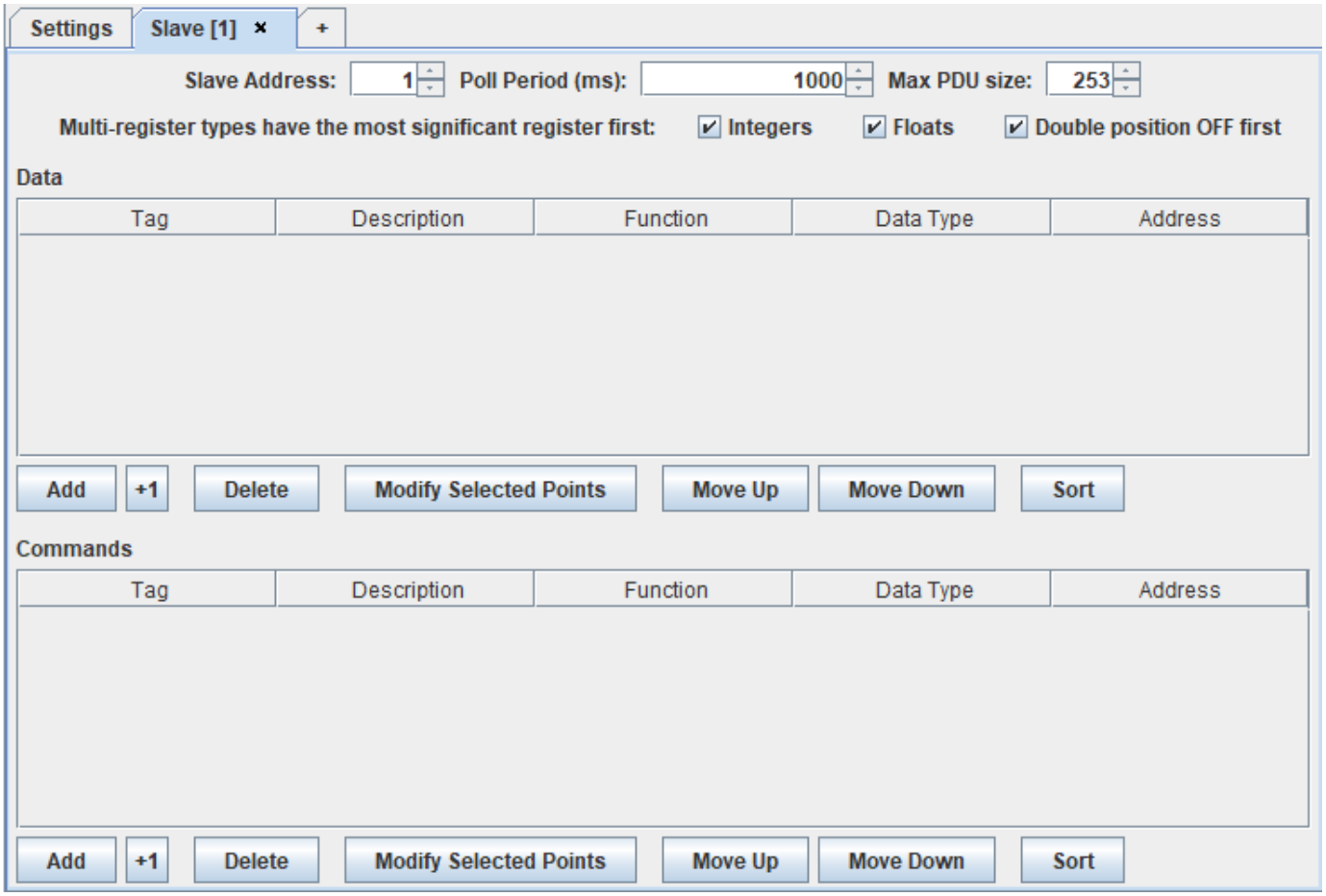


Figure 3-1 - Client has been added.

Here the “Add” and “+1” button can be used to add data points. Adding data points is explained fully in the next section, and the other buttons are described in section *7: Reference Guide*.

3.4.1 Adding Data Points

To add data points, left click the “Add” button beneath the one of the tables in the main view. Doing so will bring up the following window.

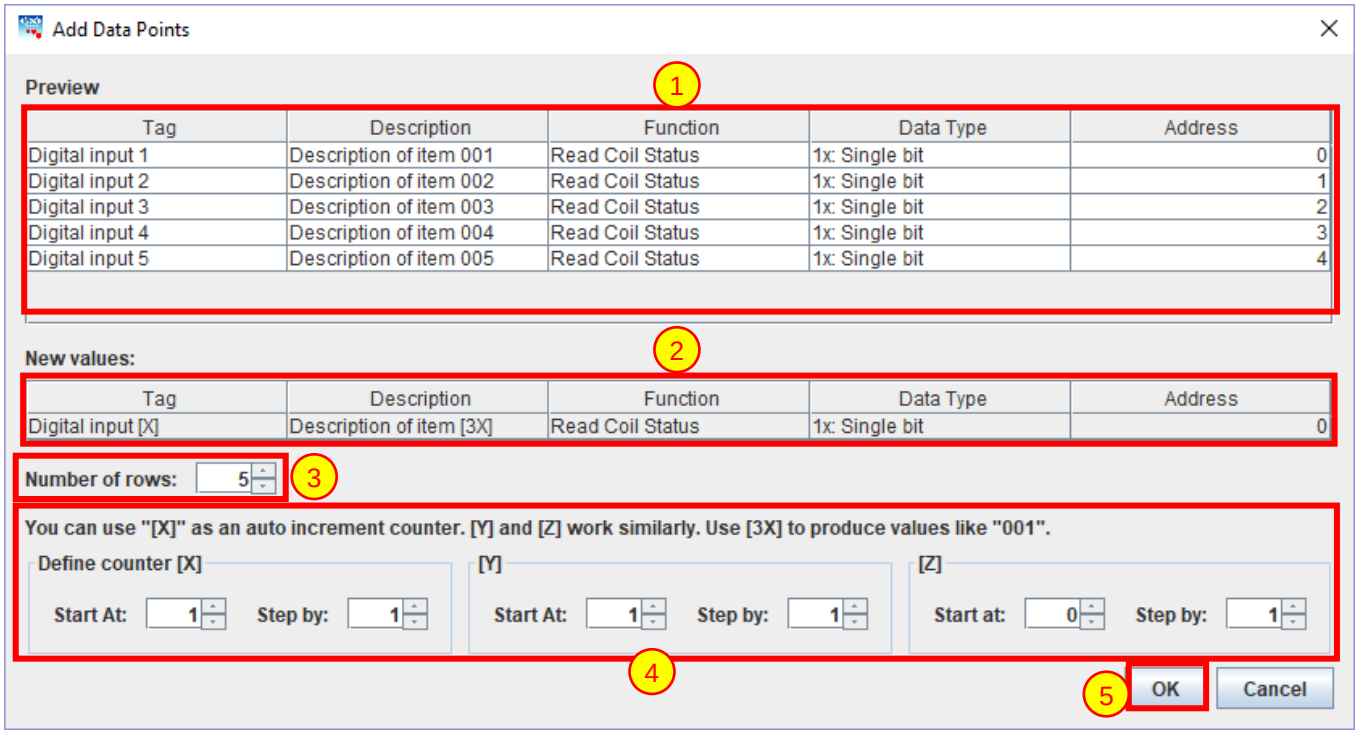


Figure 3-2 - Add data points window.

Here type in the new properties of the data points. Many data points can be added at using the value in the “Number of rows” spinner. For details on the meaning of each column, see section 7.2.

1 **Preview Area** – Shows the preview of the data points which will be added.

2 **New values** – The area used enter values. Tag, description, address and poll time use manual data entry (click the box and type new values), and the function uses a drop-down menu. Entering an integer into the address column will start at that number and automatically increment in each successive point.

3 **Number of rows –** A counter which can be used to add many data points at once.

4 **Automatic Counters** – Counters can be used in the input areas; their values will be substituted in the result and preview area. The starting values and step values can be changed in this area. This is explained further in section 5.

5 **OK button** – to accept the new data points.

3.4.2 Connected Servers (Remote IEDs)

Each connected slave IED is represented by a single tab, with the title “IED [{X}]” where {X} is the slave address of the slave. Each IED will also appear in the eNode Designer project tree.

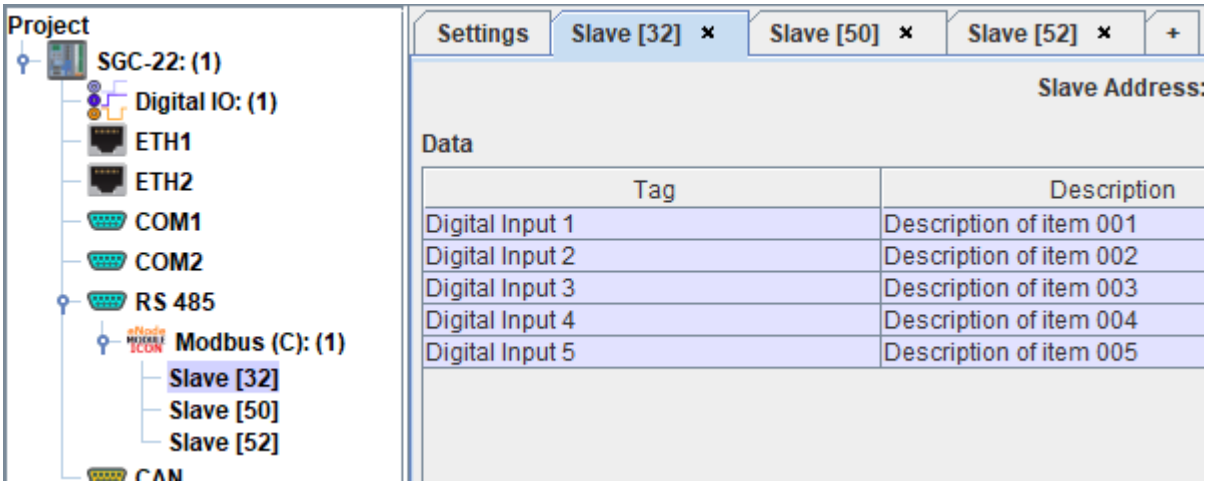


Figure 3-3 - Multiple connected servers example.

Modifying the connected IEDs list is demonstrated below:

1. To **add** a new remote IED, click the “+” tab at the end of the list of existing remote servers.

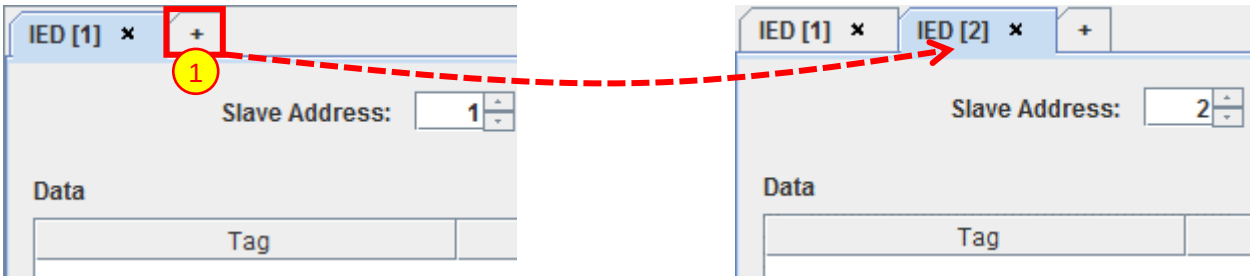


Figure 3-4 - Add a connected server.

To remove a remote IED, it must have no data points specified. If there are data points in the table and you still wish to remove the IED, you will have to remove those data points first.

1. To **remove** a remote IED, click the cross on the right side of the tab of the IED you wish to remove.

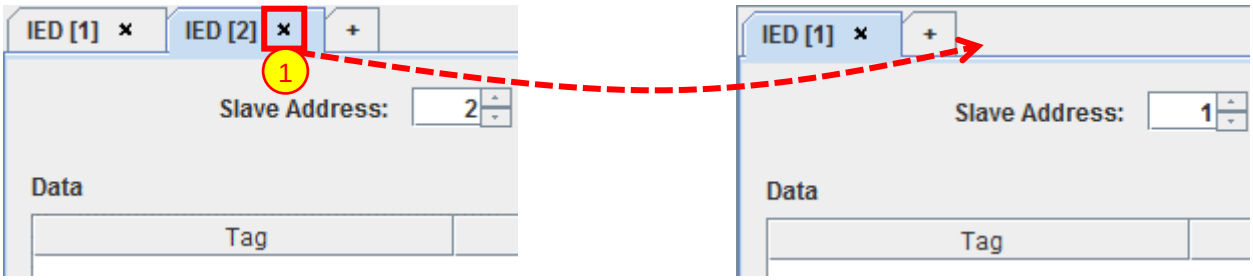


Figure 3-5 - Removed a connected server.

3.4.3 Set Default Response Timeout

Each time a new Modbus Client is added, the response timeout is set to the default value. This default value can be changed by using the “Tools” menu.

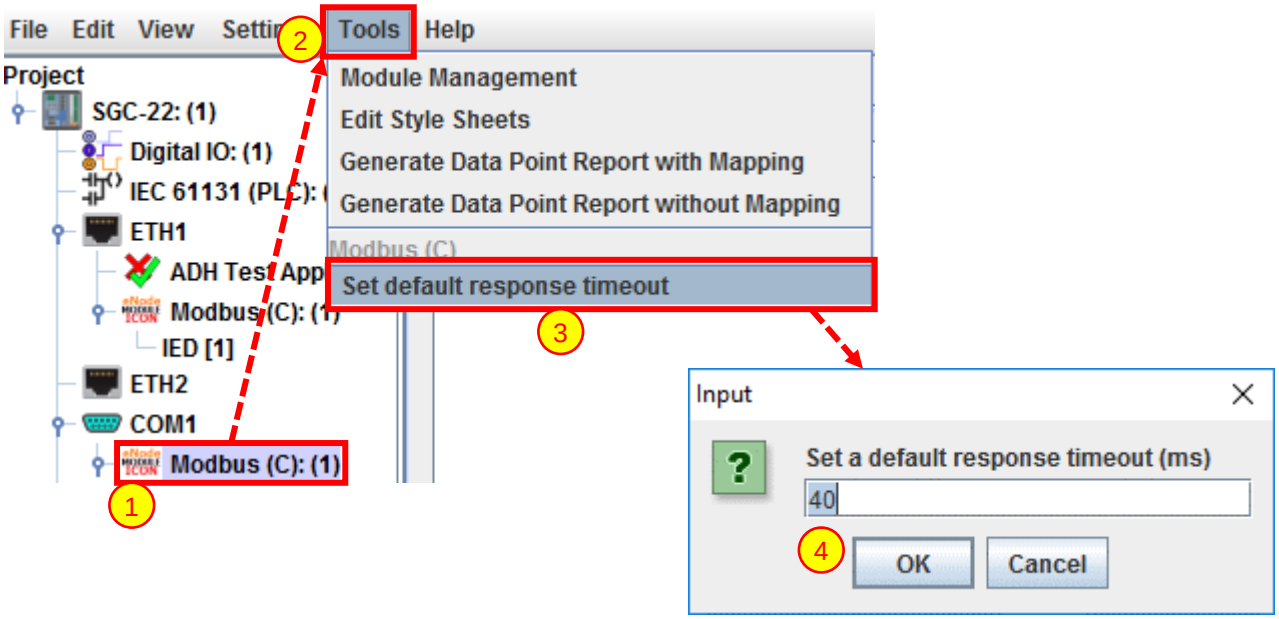


Figure 3-6 - Set default response timeout.

1. Select the **Modbus** tree node – This adds the modbus menu items to the eNode Designer menu bar.
2. Open the **“Tools”** menu
3. Select **“Set default response timeout”** – a window will appear.
4. Set a **new default value**, in milliseconds. All new Modbus clients will use this value by default.

3.5 Server Configuration

A Modbus server application outputs data from the ADH database and receives commands and passes them into the ADH system to command another application to perform the operation. Thus, all server operations involve using data point *references* to already existing data points (created by other application clients or server-clients).

The options for configuring the server are very similar to the client, only that the options are now being used to describe the local server itself. The following view will be shown when an eNode Modbus server application is added.

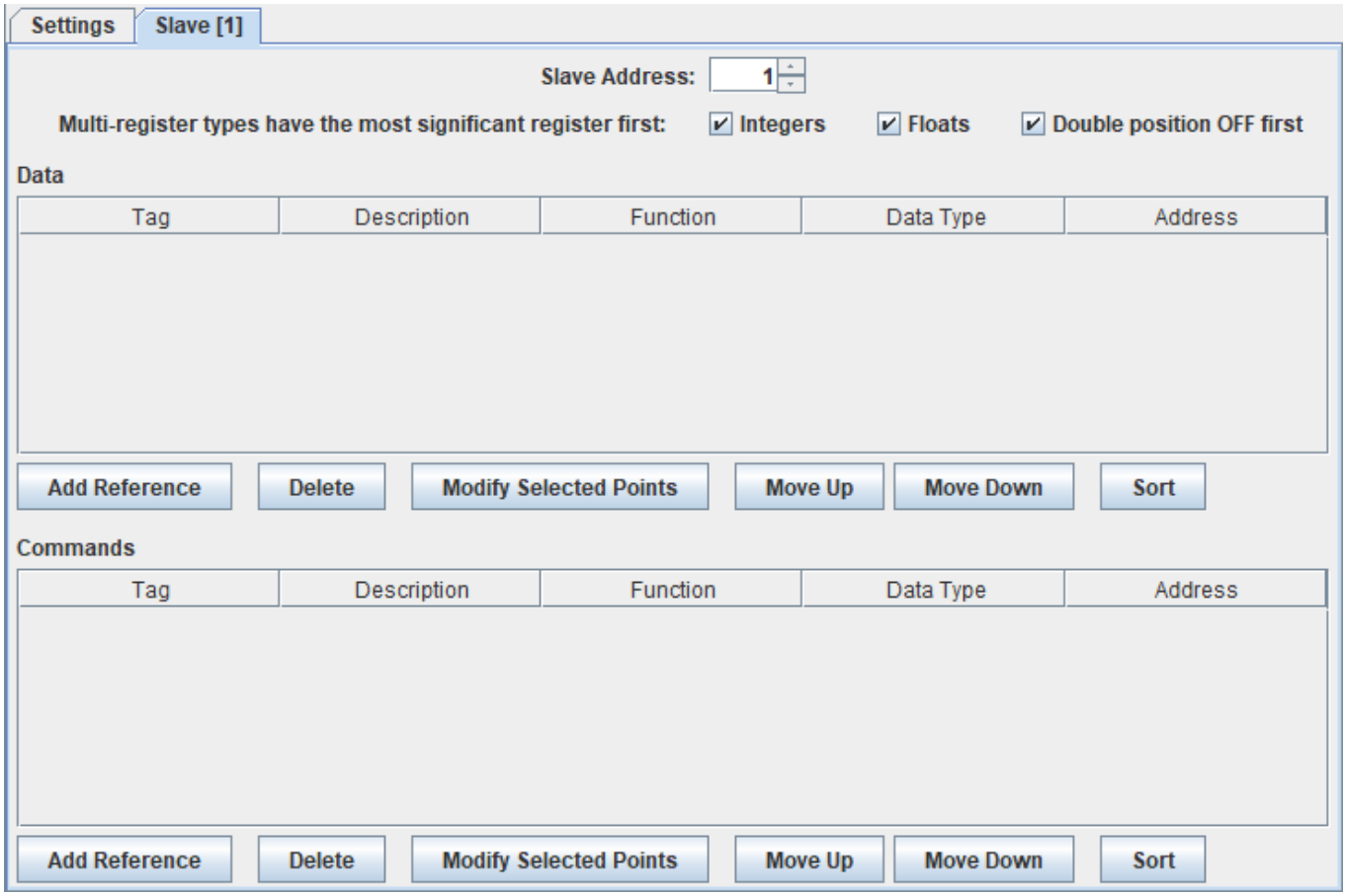


Figure 3-7 – Server has been added.

Here the “Add Reference” button can be used to add data point references. This is explained fully in the next section, and the other buttons are described in section 7: Reference Guide.

3.5.1 Adding Data Point References

To add new data point references, left click the “Add Reference” button underneath the tables in the main view. This will bring up the Add References window defined by the eNode Designer main application. It should appear similar to the following:

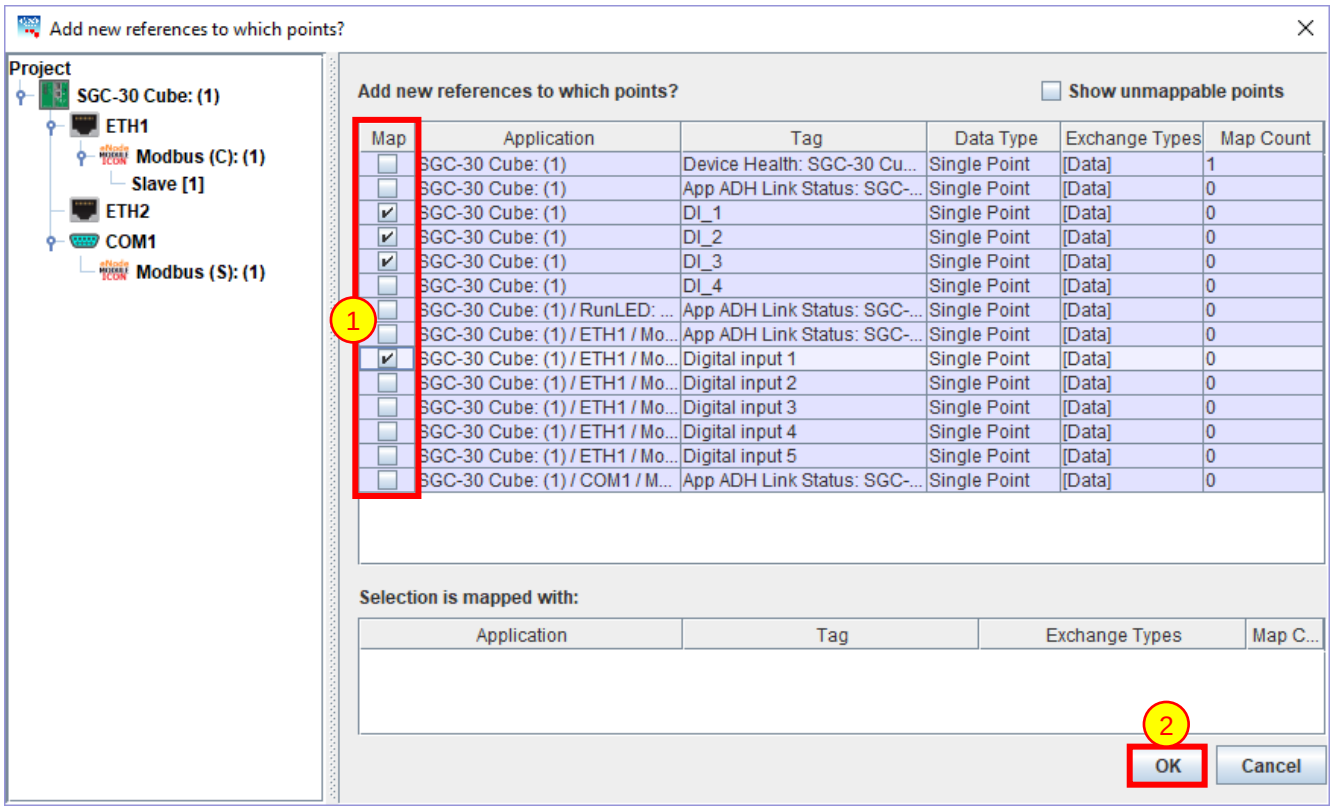
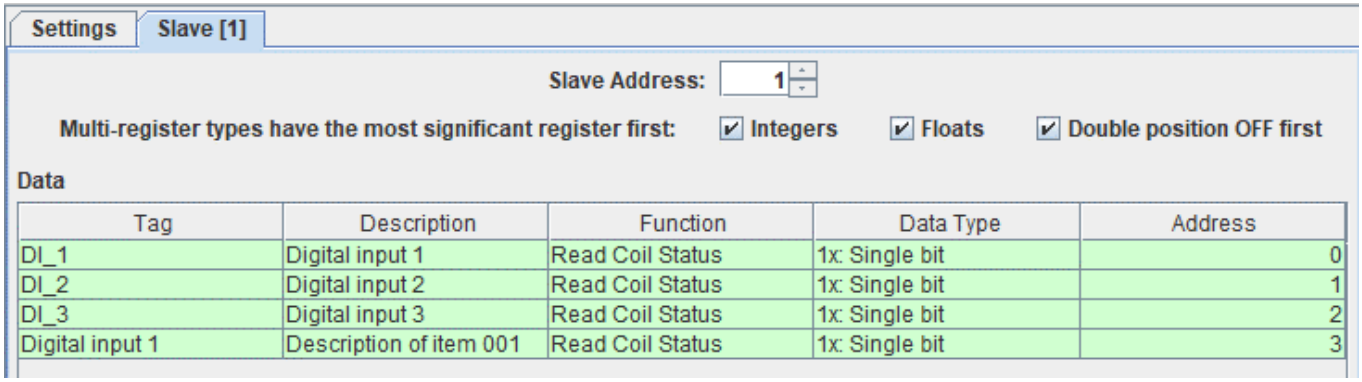


Figure 3-8 - Add new references window.

1. **Select Data Points** – Adding a reference to a point creates a “mapping” to that point. Select which data points the Modbus server application is interested in using.
2. Left Click **OK** when done to accept the new references.

The only data points that will appear in the list, and be mappable, are those whose data point type is compatible with the Modbus application. For the table matching Modbus function types to ADH data types, see 7.3 Modbus Relation to ADH Types.



Tag	Description	Function	Data Type	Address
DI_1	Digital input 1	Read Coil Status	1x: Single bit	0
DI_2	Digital input 2	Read Coil Status	1x: Single bit	1
DI_3	Digital input 3	Read Coil Status	1x: Single bit	2
Digital input 1	Description of item 001	Read Coil Status	1x: Single bit	3

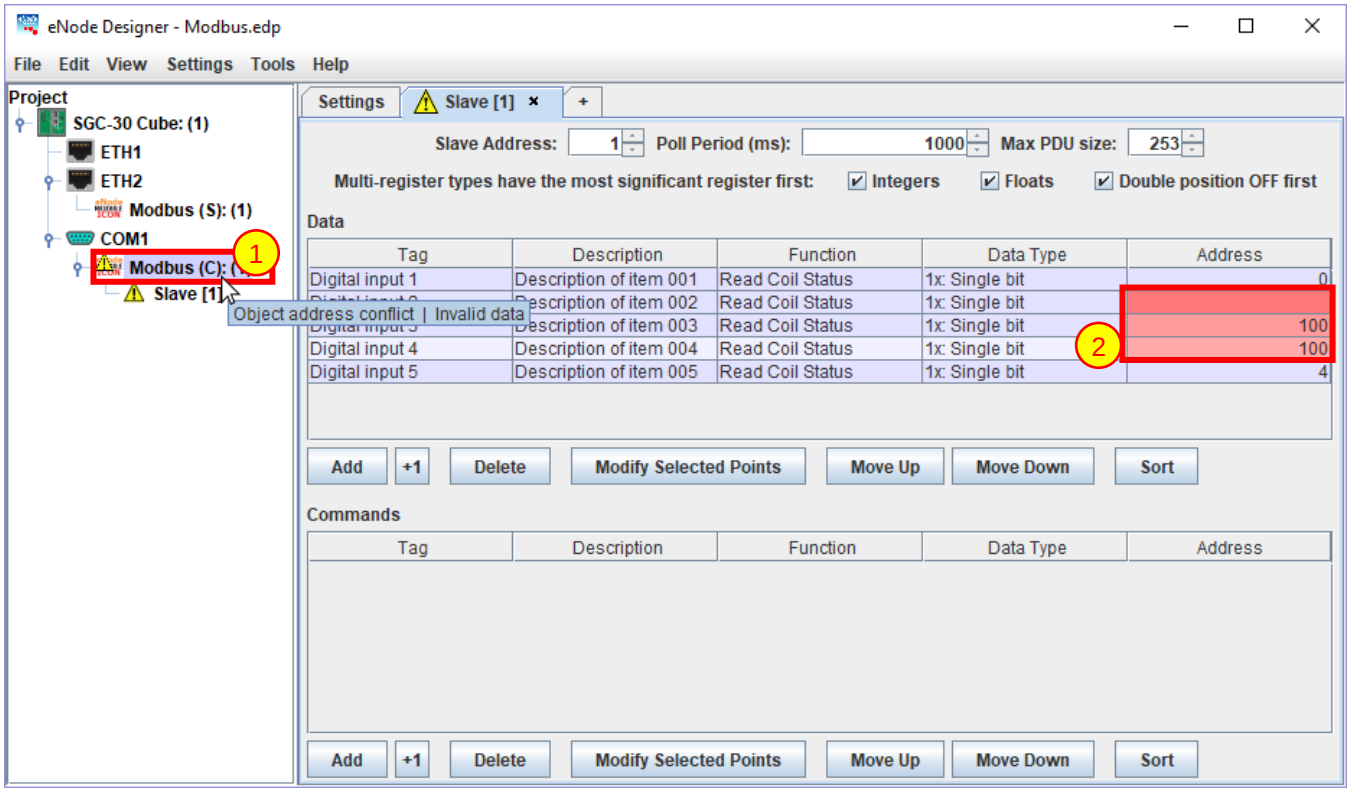
Figure 3-9 - Data point references added.

When the data points have been added, the Modbus Address field will be automatically generated starting from 0 and avoid conflicting addresses with all existing data points. This may need to be modified if the user wants different addresses.

3.6 Miscellaneous Common

3.6.1 Incomplete or Conflicting Information

Incomplete or conflicting information in the data points are shown in the table with a red cell background, and show the warning symbol on the tab (in clients) and in the project tree. Hovering over the warning icons will show further details about what is causing the warning. This allows the user to find invalid information quickly so it can be fixed.



1. **Mouse-over a warning** to show a tooltip explaining the warning.
2. **Invalid Cells** show in red. The darker red means there is no value entered, and the lighter red means there is an address conflict. These cells can also be mouse-overed to show the error details.

3.6.2 Modify Selected Points Window

The “Modify Selected Points” window is used to change many row properties in one step.

Select the data points you want to change, and then press the “Modify Selected Points” button beneath the tables. It will generate the following window.

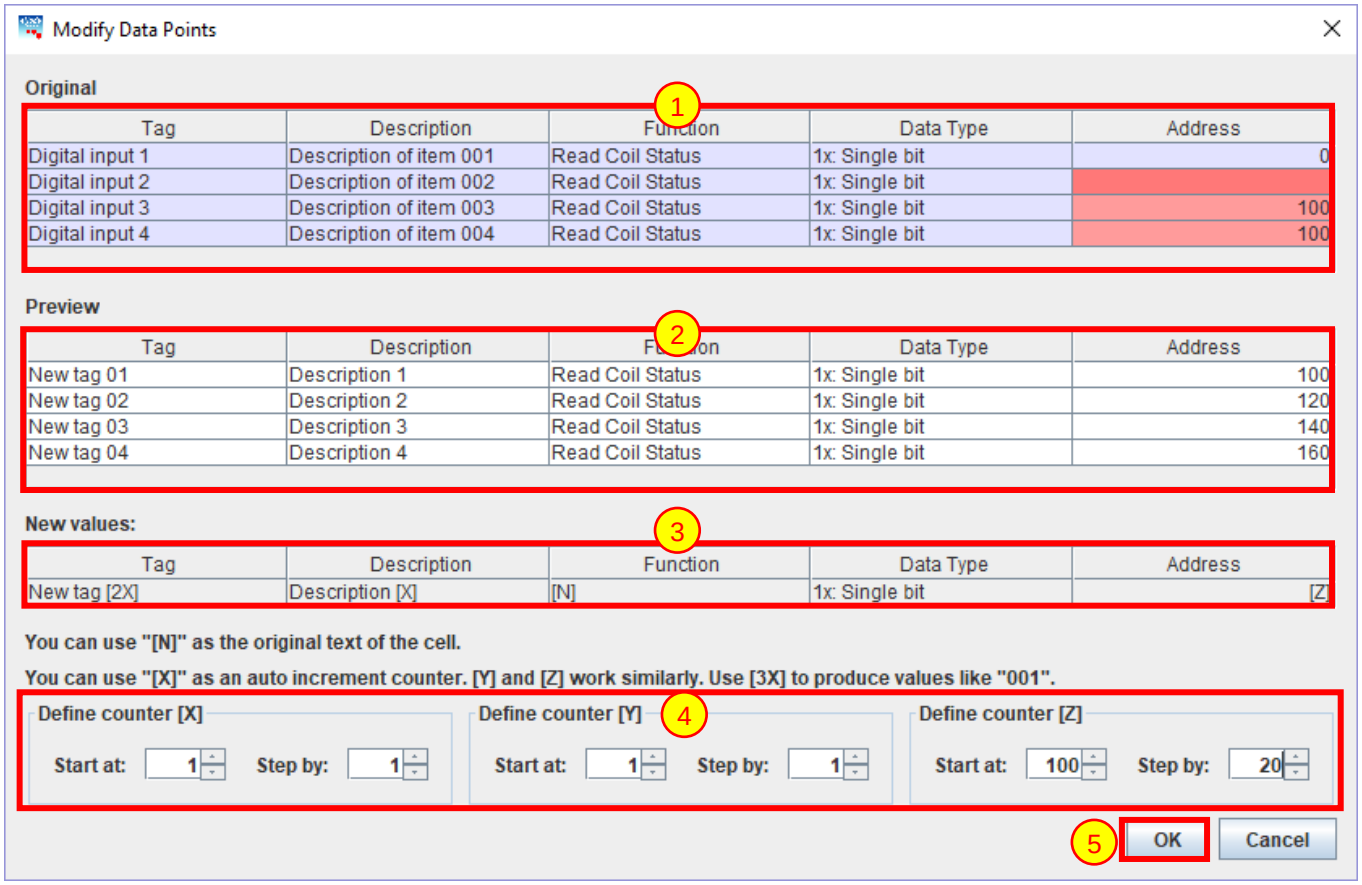


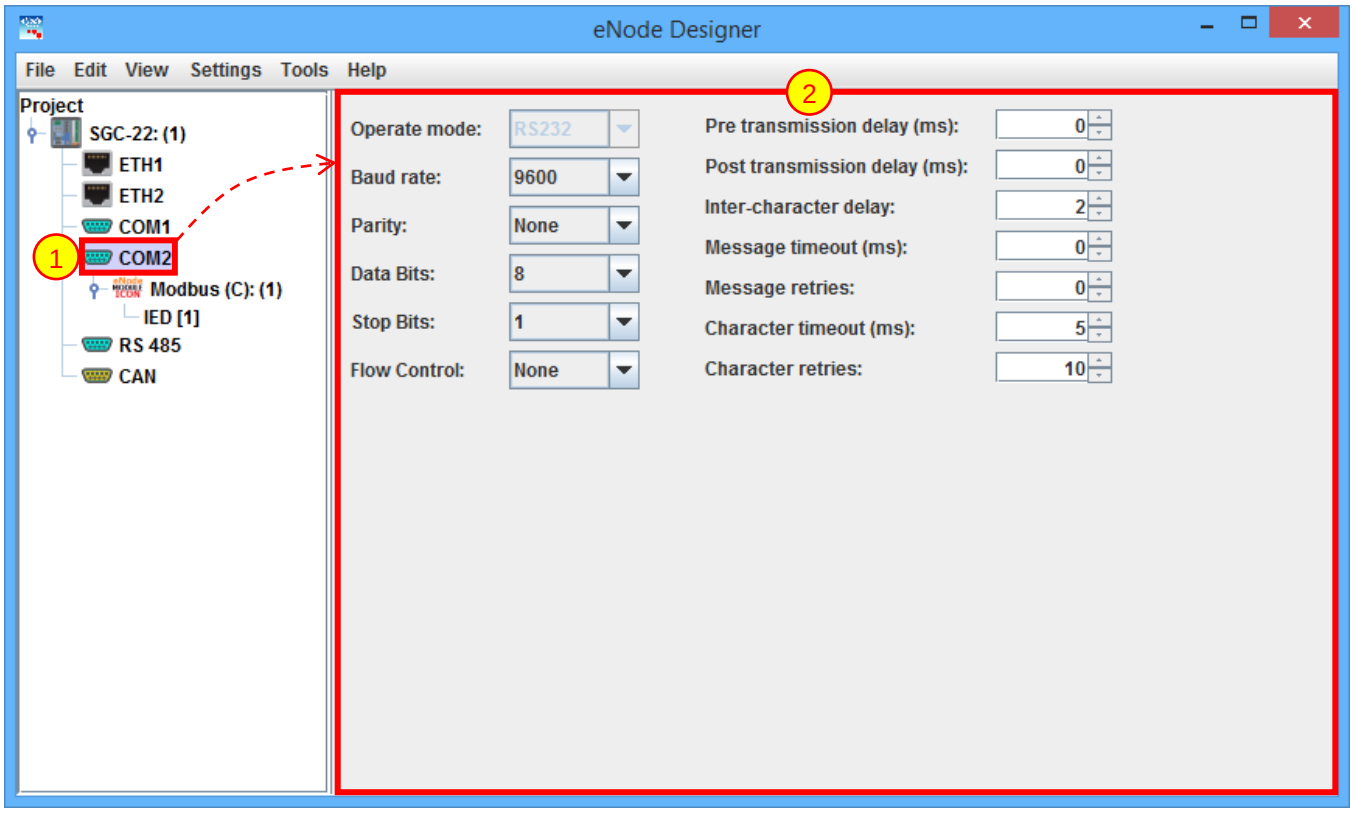
Figure 3-10 - Modify data points window example.

1. **Original table data** – Shows the original table data.
2. **Preview** – Shows the new table data that will be used if the modifications are accepted. These fields update according to the contents of (3).
3. **New values –** The new values for the table cells. “[N]” can be used to maintain the original value of the cell, and the auto-incrementing counters [X], [Y] and [Z] can be used to add numbers. See Using Auto-increment.
4. **Counter properties** – Sets the initial values and step amounts of the counters [X], [Y] and [Z].
5. **OK button** – to accept the modifications.

Data point references always use the *tag* and *description* of the “real” point. Consequently, these values will not be changed in a server application.

4 Communication Port Properties

The device module handles how the communication port properties are displayed; however, the typical method is described briefly below.



1. **Select the communication port in the project tree** – This will typically cause the central panel to show the port's properties.
2. **Properties** – The communication port's properties can be set.

5 Using Auto-increment Counters

The following is a full example showing how auto-increment works. The example given shows the IEC 60870-5-104 window, however the Modbus auto-increment works in the same way.

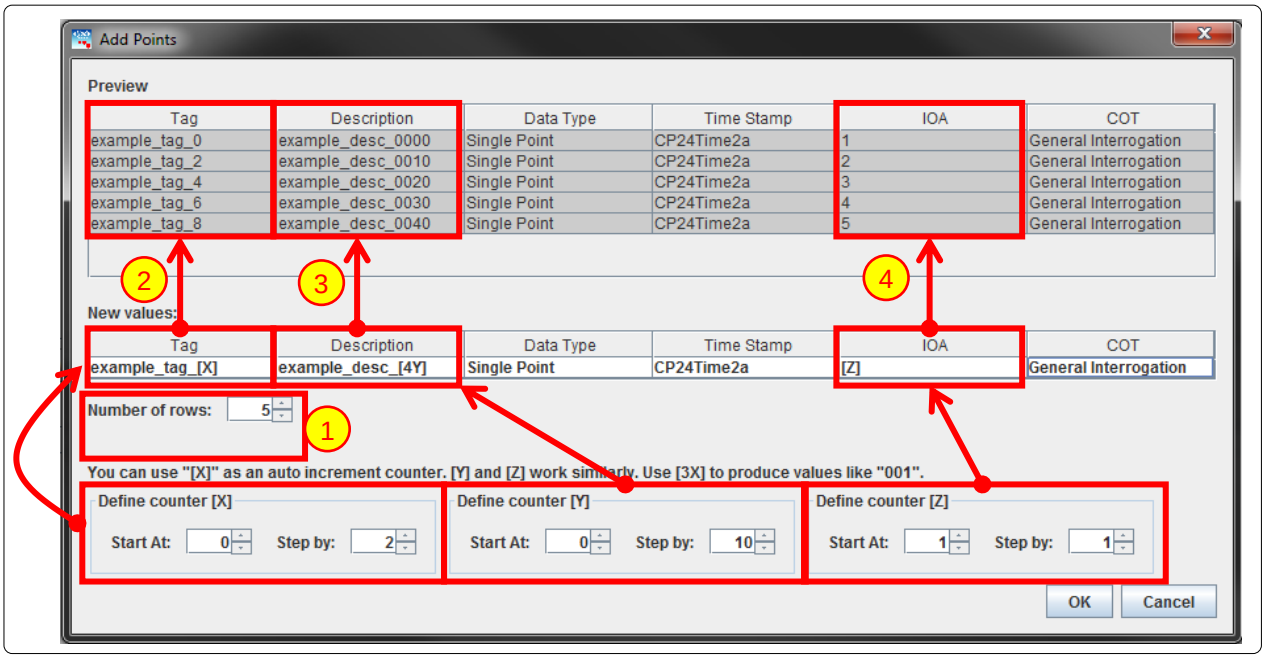


Figure 5-1 – Using Auto Increment when adding Data Points or Commands.

1 *Number of Rows* can be altered to set the number of data points or commands created from the *New values* section. As shown in the example above, five data points/commands are created and shown in the preview section as the *Number of Rows* is set to 5.

When using the auto increment counters. Each auto-increment counter can be defined to “start at” and “step by” any integer amount – there is one section per counter where there values can be changed. Adjusting *Start At* will change the value that the first data point receives. Adjusting *Step By* adjusts the value that the second and subsequent values will be incremented by.

2 In this example, the *[X]* counter has been used. The *Start At* value has been set to 0 and the *Step By* value has been set to 2. This results in the values seen in the preview section.

It is also possible to include a number within the square brackets and before the X, Y or Z while using auto increment. This will produce values that contain the entered number of digits. Any digits that are not taken up by the value determined by the *Start At* and *Step By* values will be shown as 0s.

3 In this example, the *[Y]* counter has been used with the integer 4 to indicate the number structure. This results in the values shown in the preview section.

4 In this example, the *[Z]* counter has been used. The *Start At* and *Step By* values have been left at default, this results in the values shown

If no auto increment value is entered in any field, each data point/command field value will be created the same with the exception of *Tag* and the object address field (in this case the *IOA*). The first new data point/command’s *Tag* value will represent what was entered in the *New value* section however the subsequent data points/commands will contain the initial *Tag* value followed by an underscore and a number incrementing by one from 1 onwards. (Example: tag, tag_1, tag_2 etc.). This is an artefact of eNode Designer ensuring all data point tag names are unique.

5.1 Automatic Increments in Constant Values

In the Modbus eNode module, the following fields will be automatically increased by one for each row, even if a constant value is entered in the “New value” field.

- Address

6 Modbus Interoperability

Supports both clients (master) and servers (slave).
Supported slave addresses: 1 to 255.
Modbus data object addresses: 0 to 65535 on every data type.
Configurable maximum response timeout in milliseconds: 1 to 65535 ms.
Configurable poll time per slave: 0 to 65535 ms.

6.1 Supported Frame Formats

- Serial RTU
- Serial ASCII
- Ethernet (RTU as per Modbus standard) over TCP

6.2 Supported Function Codes

Data:

- Read Coil Status (Single Point)
- Read Input Status (Single Point)
- Read Holding Register (16-bits)
- Read Input Register (16-bits)

Commands:

- Force Single Coil (Single Point)
- Preset Single Register (16-bit)

All commands in Modbus are single-stage commands, as a restriction of the protocol itself.

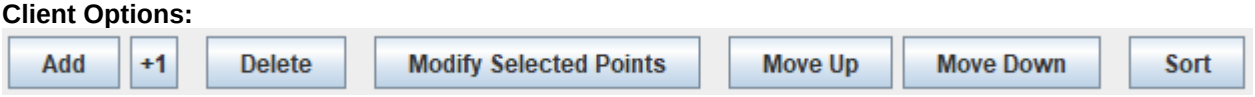
Description	Function Code	Sub-function	Client (Y/N)	Server (Y/N)
Read coils	01		Y	Y
Read discrete inputs	02		Y	Y
Read holding registers	03		Y	Y
Read input register	04		Y	Y
Write single coil	05		Y	Y
Write single register	06		Y	Y
Read exception status	07		N	Y always 0
Diagnostic	08	00-20	N	Y
Get com event counter	11		N	Y
Get com event log	12		N	N
Write multiple coils	15		Y (Dbpos)	Y
Write multiple registers	16		Y (32-bit types)	Y
Report server ID	17		N	Y
Read file record	20		N	N
Write file record	21		N	N
Mask write register	22		N	Y
Read/Write multiple registers	23		N	Y
Read FIFO queue	24		N	N
Read device identification	43	14	N	Y
Encapsulated interface transport	43	13	N	N

Table 6-1 - Supported function codes

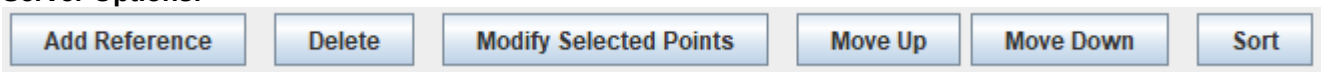
7 Reference Guide

7.1 Table Buttons

Client Options:



Server Options:



Add	Adds new data points in the client. See section 3.4.1.
+1	Add a single data point, copying values from the current selection and auto-incrementing the address.
Add Reference	Adds a new data point reference in the server. See section 3.5.1
Delete	Deletes the selected data points
Modify Selected Points	Modify the properties of the selected data points. See section 3.6.2.
Move Up	Moves the selected data points up one row in the table
Move Down	Moves the selected data points down one row in the table
Sort	Sorts the table by function code, then by object address.

7.2 Table Columns

7.2.1.1 Tag

Description	A unique Tag name for each data point
Data Entry	String
Min Length	1
Max Length	N/A
Input Option	Mandatory

7.2.1.2 Description

Description	User defined description for each data point
Data Entry	String
Min Length	1
Max Length	N/A
Input Option	Mandatory

7.2.1.3 Function

Description	The Modbus function type
Data Entry	Drop Down Menu
Types	*Read Coil Status, Read Input Status, Read Holding Register, Read Input Register, Force Single Coil, Preset Single Register*
Input Option	Mandatory

7.2.1.4 Data Type

Description	The ADH data type to used, which implies the signed-ness of the value and the number of coils or registers it uses. Data types marked with "2x" use 2 addresses (2 coils, 2 status addresses or 2 registers) for a single ADH data point.
Data Entry	Drop Down Menu, restricted based on function type.
Types	Single bit types (coil and input status) *1x: Single bit* *2x: Double position* Register types (holding register and input register) *1x: Integer 16* *2x: Integer 32* *1x: Unsigned 16* *2x: Unsigned 32* *2x: Float 32*
Input Option	Mandatory

7.2.1.5 Address

Description	The Modbus object address as sent in the Modbus telegram, starting at zero. Some Modbus object addresses are specified as from a count starting at one; in this case 1 must be subtracted from the value to obtain the ADH Modbus object address.
Data Entry	Integer
Range	*0 to 65535*
Input Option	Mandatory

7.3 Modbus Relation to ADH Types

The Modbus function types correspond to the ADH types given in the table below.

Modbus Function Type	Mappable ADH Data Types	ADH Exchange Type
Read Coil Status	Single Point / Double Point	Data
Read Input Status	Single Point / Double Point	Data
Read Holding Register	Integer 32 / Unsigned 32 / Float 32	Data
Read Input Register	Integer 32 / Unsigned 32 / Float 32	Data
Write Coil	Single Point / Double Point	Command (Single Stage)
Write Register	Integer 32 / Unsigned 32 / Float 32	Command (Single Stage)

Table 7-1 - Modbus relation to ADH data point types.

----- End of Document -----