domat
control system

RcWare SoftPLC
Modbus server mapping editor

User manual

1 Contents

2 Why SoftPLC as a Modbus server

Modbus is an old, yet widely used industrial serial communication standard, created by Modicon in 1979, see e.g. http://www.modbus.org/specs.php.

Although SoftPLC runtimes usually communicate as Modbus masters, i.e. read data from the slave I/O modules, sometimes it might be useful to share the process data to a 3rd party PLC or SCADA using the Modbus protocol. The SoftPLC runtime then acts as a Modbus server (slave), waiting for requests from a client (master).

The standard used is Modbus RTU (binary data). Communication parameters are 1200...38400 bit/s (configurable), N, 8, 1.

Modbus functions used to access and write data:

Analog values

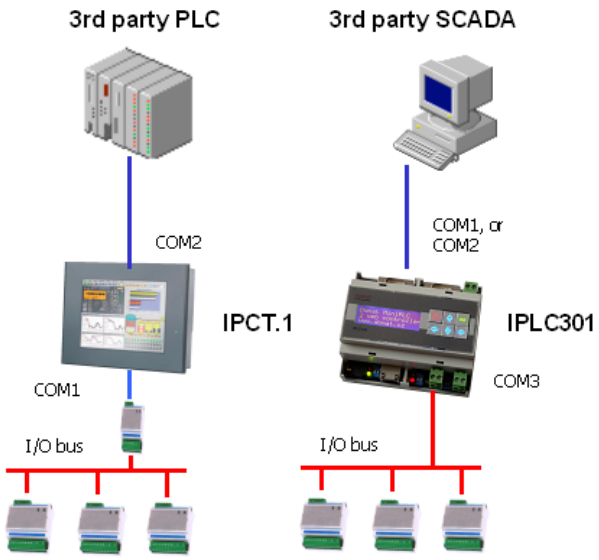
Read: F03 Read holding register or F04 Read input registers
Write: F06 Preset single register or F16 Preset multiple registers

Digital values

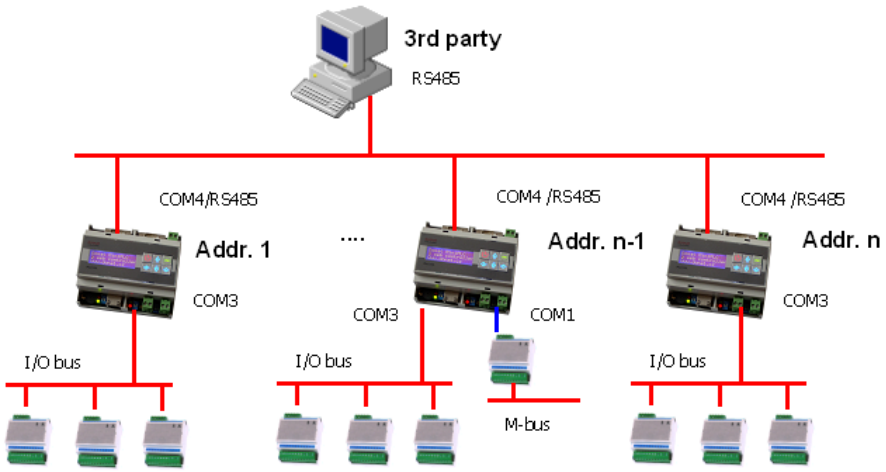
Read: F01 Read coil status or F02 Read input status
Write: F05 Force single coil or F15 Force multiple coil

The data are sent in Hi – Lo order, e.g. 0x1234: first byte sent is 0x12, second byte is 0x34.

Topology of the Modbus communication



Both IPCT (or another PC-based hardware) and MiniPLC can be used as a Modbus server. Check if you have correct port types enough to fit your application; e.g. more IPCT Modbus servers at one bus need a RS485 bus and therefore a M011 converter to each IPCT.



Another possible topology with more Modbus servers at one bus

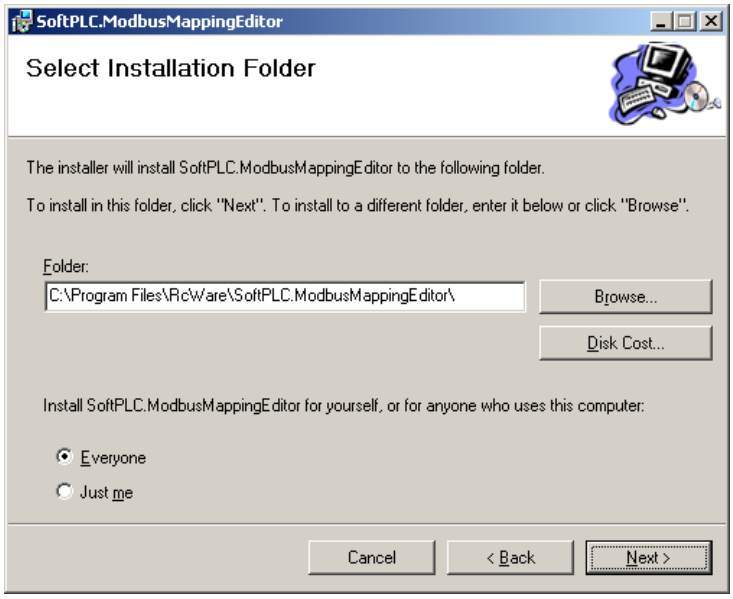
3 Installation and setup of the Modbus mapping editor

Install the Modbus mapping editor using the Common SoftPLC installer.

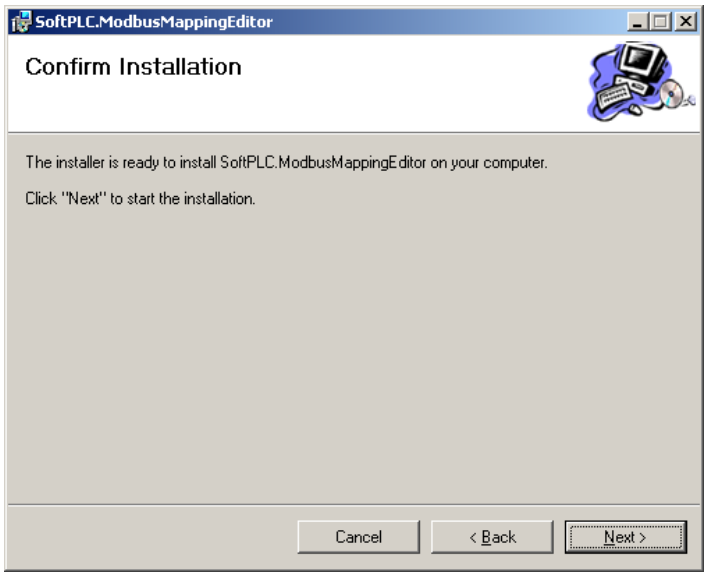
For older installations, the following procedure applies:

Run the *ESG.SoftPLC.ModbusMappingEditor.Install.msi* installer file.

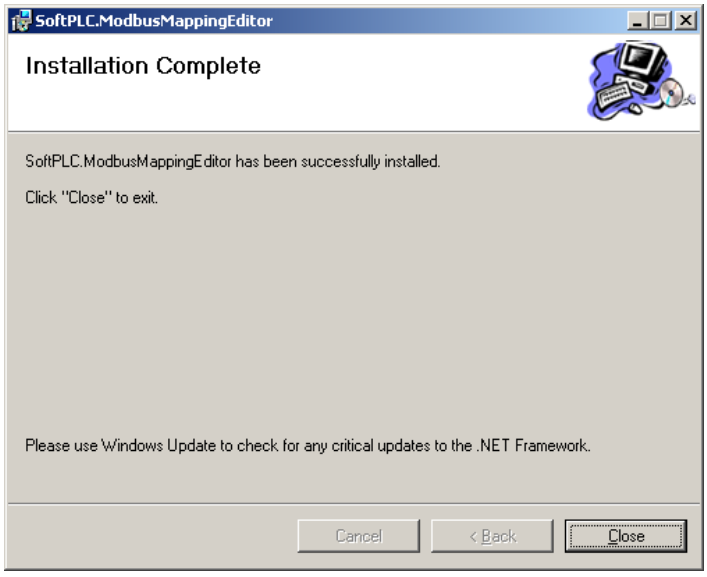
Click Next.



Specify another installation folder if desired or just click Next.



Click Next to start the installation.



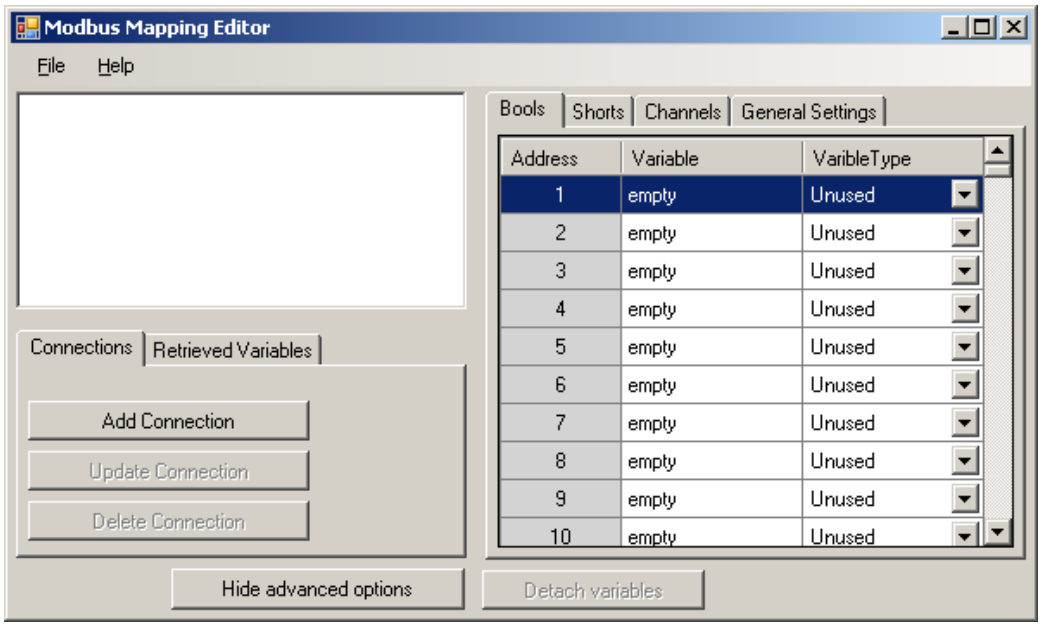
Close the installer.

4 Creating and editing the project

4.1 Import of the variables

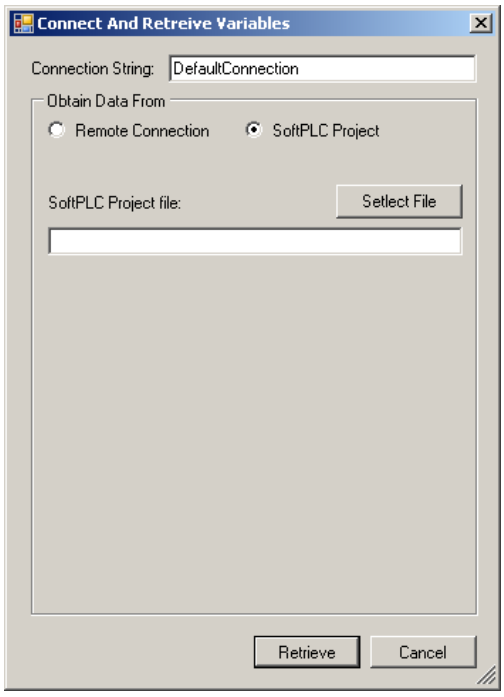
As a prerequisite, a SoftPLC project must be engineered. Its variables will be shared over the Modbus server which runs as part of the SoftPLC runtime.

Then run *Programs – RcWare – SoftPLC – Modbus Server Mapping Editor*. The program window opens.



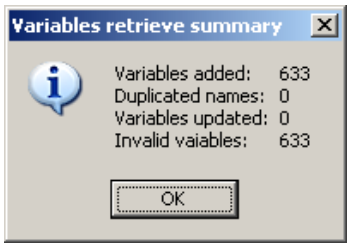
First, the SoftPLC project variables must be read:

Select *Connection – Add connection*, Obtain data from SoftPLC project: SoftPLC project.

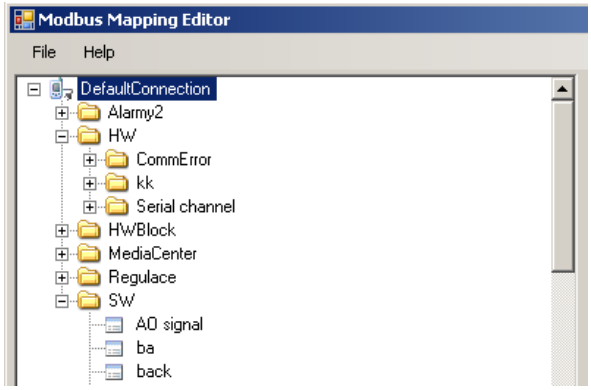


Click *Select file* and select your **.splcproj** file. Then click *Retrieve*.

A window indicates how many variables have been added / updated to the project, similar to the dialogue in Touchscreen Editor.



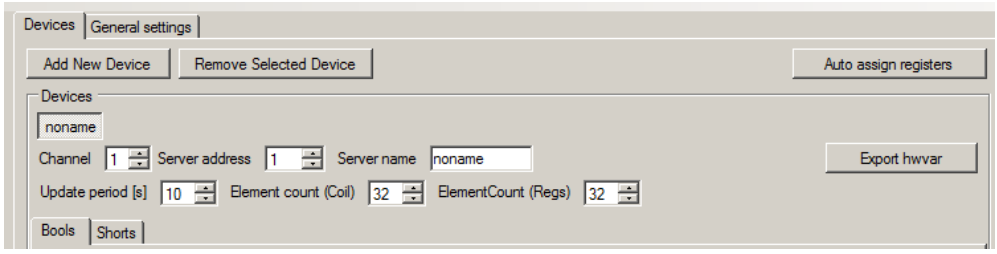
The variables also appear in the variable tree at the left pane of the program window:



Rename the DefaultConnection to any suitable name if necessary by right-clicking on the connection name and selecting *Rename Connection* in the context menu.

4.2 Settings

4.2.1 Devices



Channel: See the General settings tab where channels are defined. Select a channel (defining COM port parameters) where this device, or Modbus node, should be active.

Server address: set the Modbus server (slave) address here in range of 1 to 256. Typically it is 1, if there are more Modbus servers at one bus, each of them must have different server address.

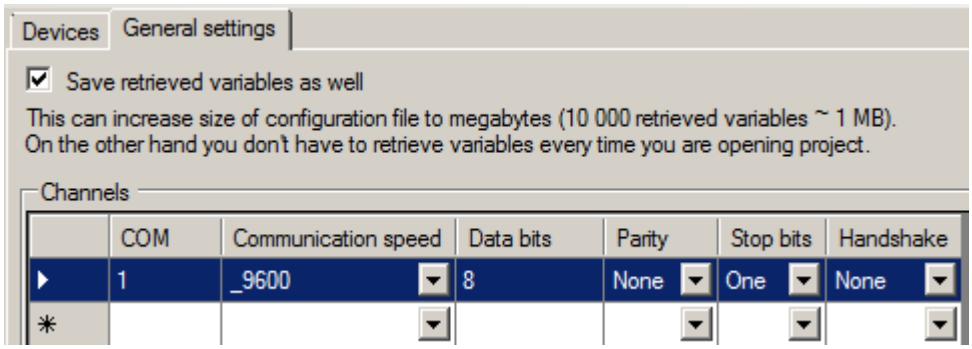
Server name: A readable name, used for your reference. Not mandatory.

Update period: How often the Modbus server should refresh valuesfrom the SoftPLC runtime.

Element count: How many registers are processed at once. Leave default values here.

4.2.2 Channels

See the General settings tab.



Set the **COM port** of the target device which should act as a Modbus server. It must be different from any of the ports used for I/O communication, M-Bus integration etc. For IPCT, it will typically be COM2; for IPLC301, it will be COM1 (RS232) or COM2 (RS232/RS485).

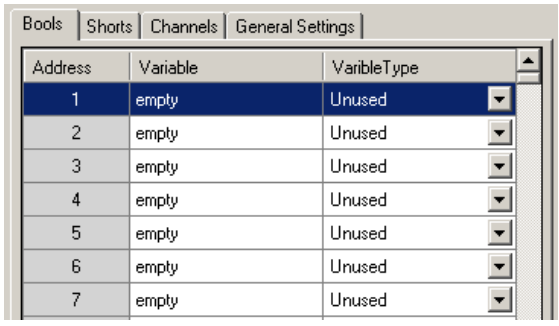
Set the **communication speed** (1200...38400 bps) as agreed with the 3rd party device supplier. The **other communication parameters** are No parity, 8 bits, 1 stop bit by default.

Save retrieved variables as well: if checked, the imported variables are also stored as part of the .mmc file. This increases the size of the file, but the variables do not have to be imported each time the project is open. Uncheck for larger MiniPLC projects as MiniPLC has limited storage space.

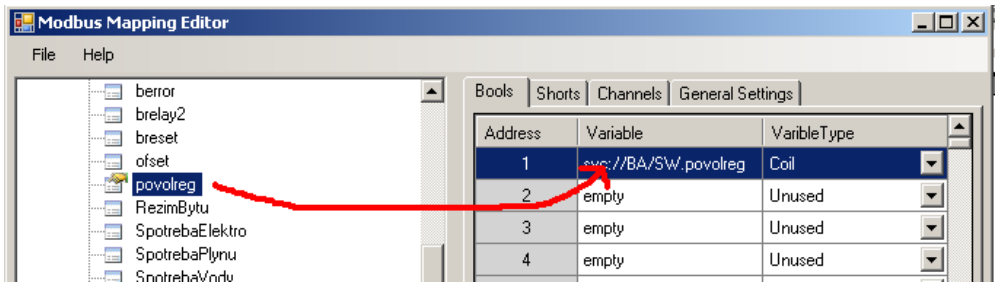
4.3 Defining variables

4.3.1 Digital variables

The first tab is a table for editing digital (boolean) variables.



Select a boolean variable in the left pane and click it. The variable is marked by a hand icon.



Then drag and drop the variable to the desired line in the Bools definition table. Its address appears in the Variable column. (NB. the connection has been renamed from *DefaultConnection* to *BA*.)

Select the requested variable type:

Coil or
Input.

In this version there is no difference between a value defined as „coil“ or as „input“, both of them are accessible with the same functions (F01, F02, F05, F15). The choice is for future extensions. It is recommend to select **Coil** here as coils can be written, too.

The Address column gives the Modbus register address to read from. Together with the variable name and type, it will be part of the documentation you will provide to the 3rd party developer. There are up to 65535 addresses (2 bytes) in the Modbus definition, however, only 1 to 2048 address fields can be mapped in the Modbus Mapping Editor.

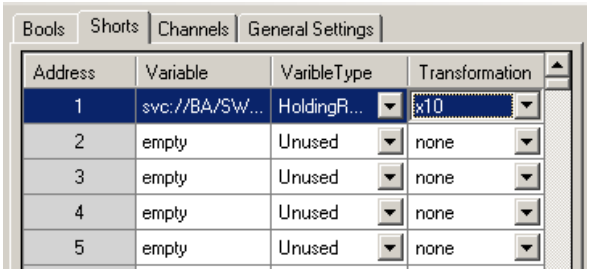
Example of the Modbus table:

Digital variables:

Modbus register	**Read / Write**	**Description**	**Note**
1	R	Unit status (On/Off)	0 = Off, 1 = On
2	R	Alarm status	0 = OK, 1 = Alarm
3	R/W	Enable unit	0 = Off, 1 = On
4	R/W	Night Mode	0 = Day, 1 = Night
....			

4.3.2 Analogue variables

The analogue variables are communicated as short integers (16 bit). Double type variables are converted into integers automatically.



Click a variable you want to map to the Modbus table. The variable is marked by a hand icon. Then drag and drop it to the desired Modbus address row.

Select variable type:

Input register
Holding register.

In this version there is no difference between a value defined as „holding register“ or as „input register“, both of them are accessible with the same functions (F03, F04, F06, F16). The choice is for future extensions. It is recommend to select **Holding register** here as holding registers can be written, too.

To increase resolution for analogue variables, they may be transformed before conversion. For example temperatures are usually multiplied by ten, 21.5°C giving an

integer of 215 (also called *HVAC integer*). Select the appropriate transformation factor in the *Transformation* column if necessary.

If the Transformation factor is set to None, the decimal part of the double type variable is rounded (163.5 -> 164).

Example of the Modbus table:

Analogue variables:

Modbus register	Read / Write	Description	Note
1	R	Unit mode	0 = Off, 1 = St1, 2 = St2, 3 = Night
2	R/W	Temp. setpoint	x10, e.g. 216 is 21.6°C
3	R	Room temp.	x10, e.g. 216 is 21.6°C
4	R	Run hours	Total unit run hours
....			

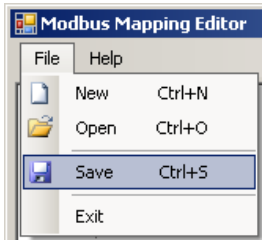
4.3.3 Changing and deleting variables

To change the variable in the table, simply drag and drop another variable to the Modbus table row.

To delete the variable from the Modbus table, focus the row and click the Detach variables button.

4.4 Saving the project

To save the project to the .mmc file, select File – Save or Ctrl-S and enter the name of your project.



4.5 Opening a saved project

A saved project can be opened for further processing in the *File – Open* menu or by pressing the Ctrl-O keys. Remember that if the *Save retrieved values...* option in the *General settings* menu has been unchecked before saving, it is necessary to re-import the variables to be able to add some.

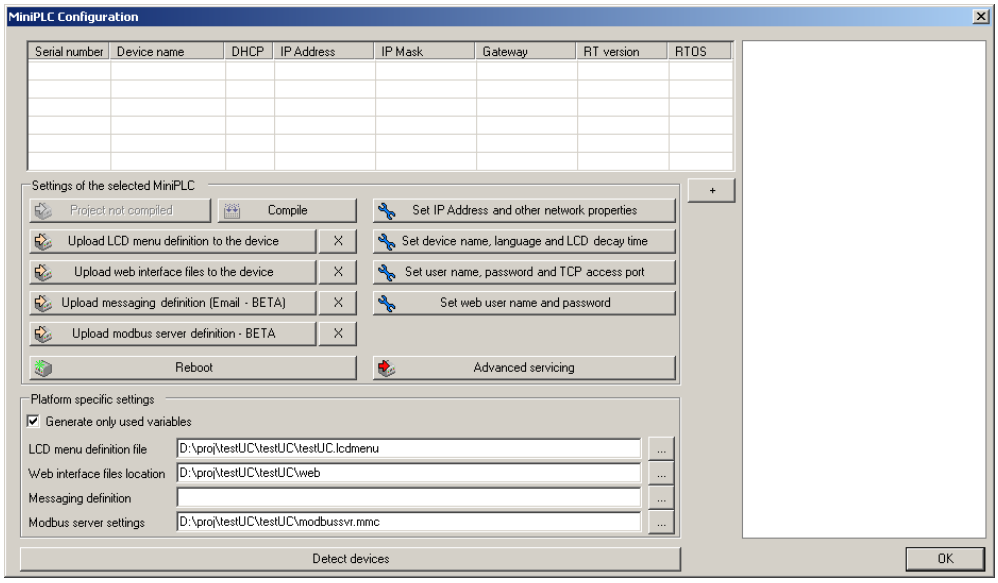
4.6 Loading the configuration file into the runtime

4.6.1 SoftPLC – Windows runtime

Copy the .mmc file to the folder where Modbus server is installed.

4.6.2 MiniPLC

In the *Platform config* dialogue define the path and file name in the *Modbus server settings* item:



Then focus the MiniPLC concerned and click *Upload modbus server definition*. Remember to reboot the device to start the Modbus server.

5 Testing and troubleshooting

For testing, use a Modbus client you are familiar with on the other end of the line, e.g. Modbus tester at www.modbus.pl, or RcWare Vision.

No data are send or received

- Check cabling and port health (e.g. using two Hyperterminals against each other).

- Check if comm speed and other physical parameters are set at both ends.

Some characters are received, but they are not interpreted as valid Modbus responses

- Check if comm speed and other physical parameters are set at both ends

- Check Modbus client settings: define more datapoint in the RcWare Vision with different Hi-Lo parameters, data types, answer lengths etc. One of them should fit then.

Data are interpreted as valid Modbus telegrams, but values are not correct

- Check the Modbus listing against the Modbus protocol description.

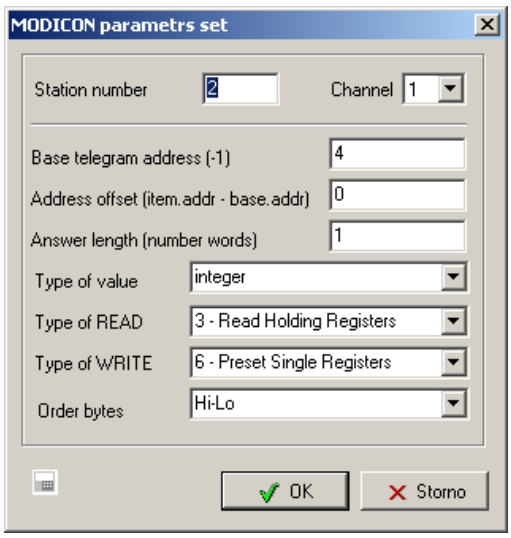
- Check Modbus client settings: define more datapoint in the RcWare Vision with different Hi-Lo parameters, data types, answer lengths etc. One of them should fit then.

Example:

Modbus Mapping Editor:

Analogue value, address 3, Holding register, Transformation: None

RcWare Vision Modbus parameter settings:



NB. the Base telegram address parameter is 4 (and not 3), as the RcWare Vision decreases the address by one.