FeMODBUS

Frandos Engineering LLC

User Manual

Windows software for Modbus TCP/IP, RTU communication.

Silviu Frandos
sfrandos@gmail.com

Contents

Introduction 2
1) Starting the program 2
2) Program configuration 3
3) Modbus TCP Client Configuration 4
4) Tags definition 7
5) Modbus TCP Server Configuration 8
6) Modbus RTU Server Configuration 10
7) Conclusions 11

FeMODBUS – User Manual

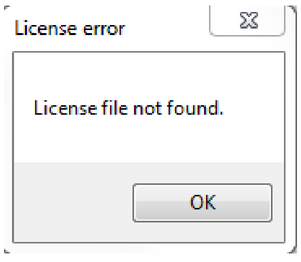
Introduction

FeMODBUS is a Windows based, DDE Server, TCP/IP and serial communication program designed to allow access to multiple remote Modbus servers (PLCs or other industrial controllers) and to be able to share the data with other DDE Clients. This manual will cover the following topics:

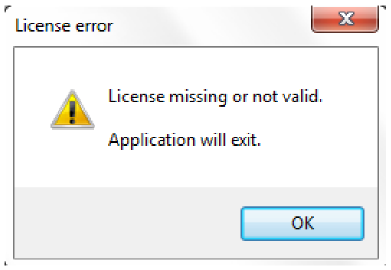
1. Starting the program
2. Program configuration
3. Modbus TCP Client Configuration
4. Tags definition
5. Modbus TCP Server Configuration
6. Modbus RTU Server Configuration
7. Conclusions

1) Starting the program

Create a new folder and copy the files *FeMODBUS.exe* and *FeMODBUS.lic* in it. If you do not have the file *FeMODBUS.lic*, run the program *FeMODBUS.exe*. You will get a message that the license file was not found and another message that the license is not valid. The program will end after it will create a new file *Hardware.lic*. This file should be sent to the software provider, who will generate a *FeMODBUS.lic* file to match your hardware.



License file not found.

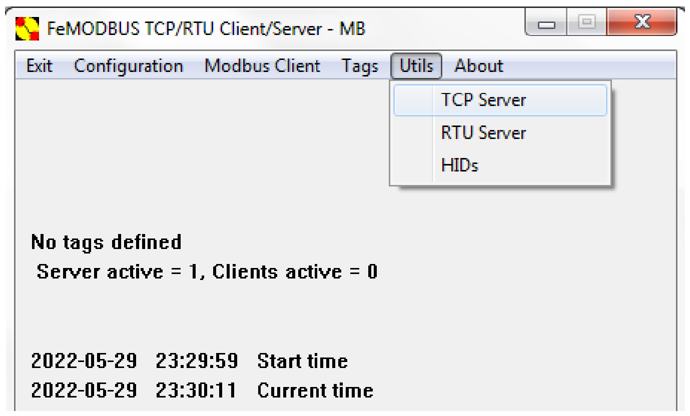


License missing or not valid.

Application will exit.

If you have the right license the program will start and it will create a default *MB.ini* configuration file.

The main window of the program is showing the menu: *Exit*, *Configuration*, *Modbus Client*, *Utils* and *About*.



No tags defined
 Server active = 1, Clients active = 0

2022-05-29   23:29:59   Start time
2022-05-29   23:30:11   Current time

It will also show if tags are defined, if the Modbus TCP server is active and how many clients are connected to it.

The start and current times are displayed at the bottom of the window.

The next step is to configure the program. For this select **Configuration** from the menu.

2) Program configuration

The configuration dialog is shown in the picture below. In the dialog window there are settings for:

TCP Server COM settings – a COM port used by the Modbus TCP Server to redirect a request from the TCP/IP network to a serial COM port. The answer is sent back on the TCP/IP network. It is a Modbus TCP to RTU bridge feature. Port *NONE* means that the feature is not used
TCP Server port – defines the port on which the Modbus TCP Server will listen
RTU Server COM settings – is a serial COM port used for the Modbus RTU Server feature
RTU Server address – defines the address for the Modbus RTU Server. The server will respond only to the requests with this address
Accept Broadcast Connections – the check box indicates if the Modbus TCP Server should accept requests from any IP address on the network or not. If unchecked, the list
No Broadcast - Accepted connections will be enabled and can be edited by double clicking on the fields of all the 16 lines. If **Type** is *Client* then the **IP address** can read but cannot write. Any *Master* can read and write. More than one *Master* can be defined
Tags File Name – defines the file from where the program will read the tags, if the check box **Auto load at start-up** is checked
Client File Name – defines the file from where the program will load the connections to be made and the requests to be sent, to read/write data from/to remote Modbus TCP servers

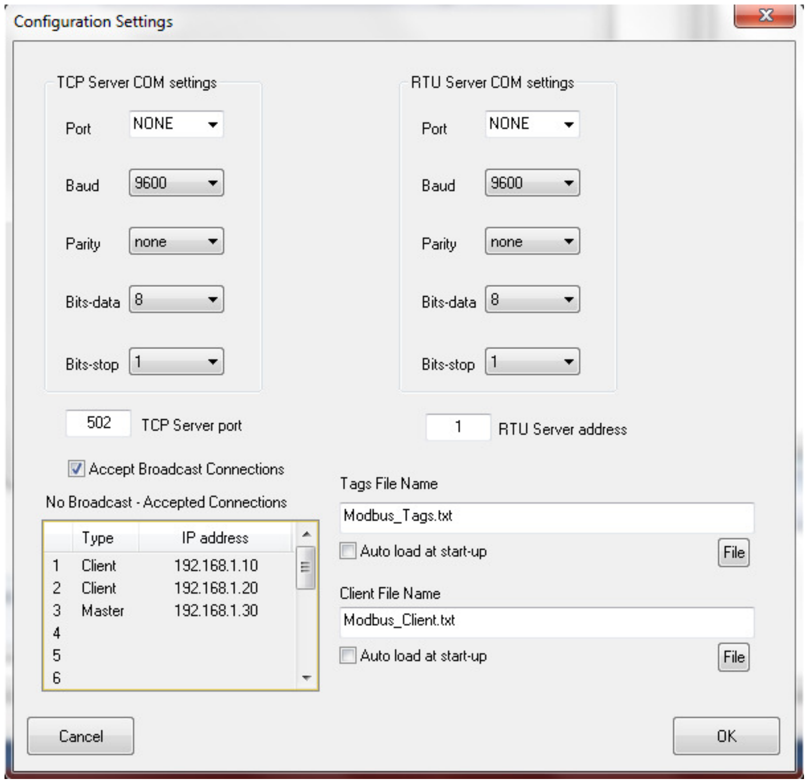
By default, the Modbus TCP port is 502 and the Modbus RTU server's address is 1. These values can be changed by the user. For example, if you run more instances of the program on the same computer then the server ports have to be different.

Baud, **Parity**, **Bits-data** and **Bits-stop** settings must match the serial communication network.

By default the Modbus TCP Server is accepting connections from any computer on the network, including itself, with the address 127.0.0.1. There are maximum 16 connections accepted at any given time.

The names *Modbus_Tags.txt* and *Modbus_Client.txt* can be changed to suit the needs of the project.

All the settings above are saved in the *MB.ini* configuration file, in the current folder of *FeMODBUS.exe*.


Configuration Settings

TCP Server COM settings
Port: NONE
Baud: 9600
Parity: none
Bits-data: 8
Bits-stop: 1

RTU Server COM settings
Port: NONE
Baud: 9600
Parity: none
Bits-data: 8
Bits-stop: 1

502 TCP Server port

1 RTU Server address

[✓] Accept Broadcast Connections

No Broadcast - Accepted Connections

	Type	IP address
1	Client	192.168.1.10
2	Client	192.168.1.20
3	Master	192.168.1.30
4		
5		
6		

Tags File Name: Modbus_Tags.txt
[ ] Auto load at start-up

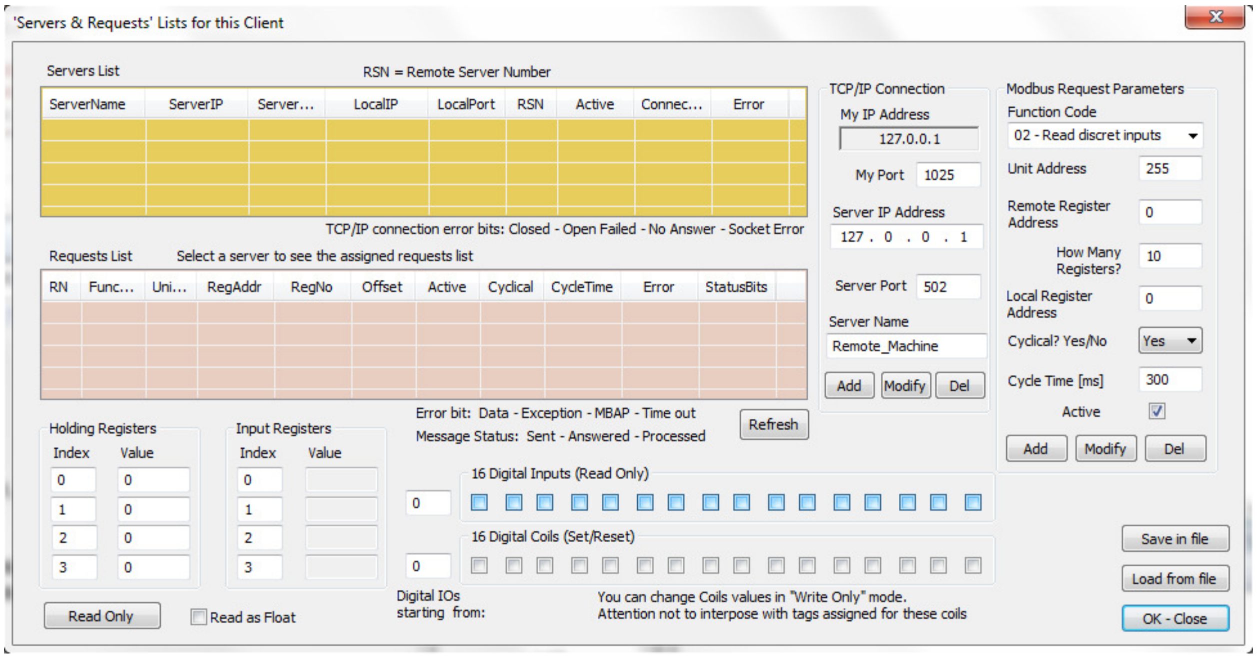
Client File Name: Modbus_Client.txt
[ ] Auto load at start-up

3) Modbus TCP Client Configuration

FeMODBUS program is first and foremost used as a Modbus TCP Client. A Modbus TCP client will initiate and connect to a Modbus TCP server and ask to read or to write data. *FeMODBUS* is able to be setup to connect to multiple Modbus TCP servers on the network.

To start the settings dialog, select **Modbus Client** from the menu (see picture below). The **Servers List** is on the top of the window. You can edit the **Server IP Address**, the **Server Port** number and the **Server Name** text. Click the **Add** button, under the **Server Name**, to add a new server to the list. You can add multiple servers. Every line in the **Servers List** represents a new TCP/IP connection.

The user can modify the settings for any Modbus server. First select the line, edit the changes in the **TCP/IP Connection** settings group, and press the **Modify** button. You can delete a selected line with the **Del** button from the **TCP/IP Connection** group. You have to **Save in file** before you **OK-Close** the dialog window.


'Servers & Requests' Lists for this Client
Servers List
RSN = Remote Server Number
ServerName
ServerIP
Server...
LocalIP
LocalPort
RSN
Active
Connec...
Error
TCP/IP connection error bits: Closed - Open Failed - No Answer - Socket Error
Requests List
Select a server to see the assigned requests list
RN
Func...
Uni...
RegAddr
RegNo
Offset
Active
Cyclical
CycleTime
Error
StatusBits
Error bit: Data - Exception - MBAP - Time out
Message Status: Sent - Answered - Processed
Refresh
TCP/IP Connection
Add
Modify
Del
Modbus Request Parameters
Add
Modify
Del
Holding Registers
Index
Value
Input Registers
Index
Value
16 Digital Inputs (Read Only)
16 Digital Coils (Set/Reset)
Digital IOs starting from:
You can change Coils values in "Write Only" mode.
Attention not to interpose with tags assigned for these coils
Read Only
Read as Float
Save in file
Load from file
OK - Close

My IP Address: 127.0.0.1
My Port: 1025
Server IP Address: 127 . 0 . 0 . 1
Server Port: 502
Server Name: Remote_Machine

Function Code: 02 - Read discret inputs
Unit Address: 255
Remote Register Address: 0
How Many Registers?: 10
Local Register Address: 0
Cyclical? Yes/No: Yes
Cycle Time [ms]: 300
Active: ✓

Holding Registers Index	Value	Input Registers Index	Value
0	0	0	
1	0	1	
2	0	2	
3	0	3	

Digital IOs starting from: 0 (Digital Inputs), 0 (Digital Coils)

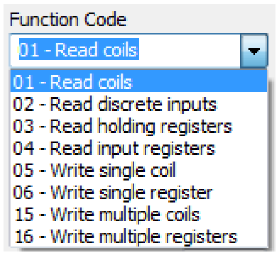
For every line in the Servers List the user can add multiple requests. A request is a command to:

- Read – one or multiple coils, discrete inputs, input registers or holding registers,
- Write – one or multiple coils or holding registers.

To add a request, select with the mouse a server from the **Servers List**, select the settings you want in the **Modbus Request Parameters** group, and press the **Add** button under the **Cycle Time [ms]**. The new request will be inserted in the **Requests List**.

You can change a request by clicking on it, editing the settings, and pressing **Modify** button. You can delete it by selecting the request and pressing **Del** button.

A request has 8 (eight) settings:



Function Code

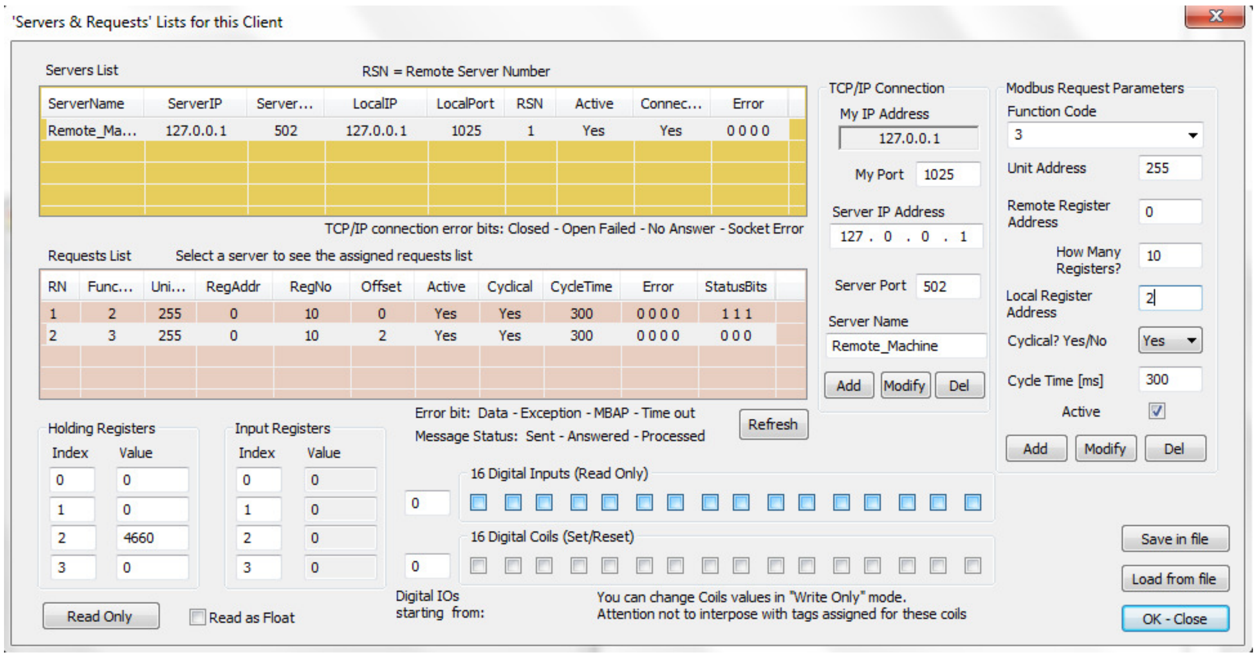
01 - Read coils

01 - Read coils
02 - Read discrete inputs
03 - Read holding registers
04 - Read input registers
05 - Write single coil
06 - Write single register
15 - Write multiple coils
16 - Write multiple registers

- **Function code** – the Modbus function code for the message to be sent.
- **Unit address** – the RTU address of the remote server. In Modbus TCP this number is always 255. In the case you are using the Modbus bridge to read/write data from/to an RTU unit, this address has to be less than 255 (from 1 to 254).
- **Remote Register Address** – is the address in the remote server from which to read/write.
- **How Many Registers**? – is the number of registers to be read/written.
- **Local Register Address** – is the address from the *FeMODBUS* program where to start copying in, if there is a reading, or to start copying from, if there is a writing. For every server in the **Servers List** there are created and reserved, in the program memory, 800 registers: 200 coils, 200 discrete inputs, 200 input registers, and 200 holding registers.
- **Cyclical? Yes/No** – the way to perform the function code. Usually is *Yes*.
- **Cycle Time [ms]** – the cycle interval in milliseconds for the function code to be retransmitted.
- **Active** – if the check box is checked the function code is executed (is active).

In the picture below one server connection was defined, at 127.0.0.1 (TCP/IP connection with itself). For this server, two requests were setup, with a 300ms cycle time:

- Read the first 10 *discrete inputs* (from remote address 0) and map them to the client's RSN 1 memory allocation area, starting at the local address 0 (zero). (RSN = Remote Server Number);
- Read the first 10 *holding registers* (from remote address 0) and copy them in the client's RSN 1 memory allocation area for the holding registers, starting from the address 2 (two).



'Servers & Requests' Lists for this Client

Servers List RSN = Remote Server Number

ServerName	ServerIP	Server...	LocalIP	LocalPort	RSN	Active	Connec...	Error
Remote_Ma...	127.0.0.1	502	127.0.0.1	1025	1	Yes	Yes	0 0 0 0

TCP/IP connection error bits: Closed - Open Failed - No Answer - Socket Error

Requests List Select a server to see the assigned requests list

RN	Func...	Uni...	RegAddr	RegNo	Offset	Active	Cyclical	CycleTime	Error	StatusBits
1	2	255	0	10	0	Yes	Yes	300	0 0 0 0	1 1 1
2	3	255	0	10	2	Yes	Yes	300	0 0 0 0	0 0 0

Error bit: Data - Exception - MBAP - Time out
Message Status: Sent - Answered - Processed

TCP/IP Connection
My IP Address: 127.0.0.1
My Port: 1025
Server IP Address: 127 . 0 . 0 . 1
Server Port: 502
Server Name: Remote_Machine

Modbus Request Parameters
Function Code: 3
Unit Address: 255
Remote Register Address: 0
How Many Registers?: 10
Local Register Address: 2
Cyclical? Yes/No: Yes
Cycle Time [ms]: 300
Active: ✓

Holding Registers

Index	Value
0	0
1	0
2	4660
3	0

Input Registers

Index	Value
0	0
1	0
2	0
3	0

16 Digital Inputs (Read Only): 0
16 Digital Coils (Set/Reset): 0
Digital IOs starting from:

You can change Coils values in "Write Only" mode.
Attention not to interpose with tags assigned for these coils

The settings from the bottom of the dialog window are used to troubleshoot and visualize on-line data from the servers' list. By default the user is in **Read Only** mode. In this mode, the local data of the holding registers and input registers is updated based on the **Index** number, which the user can change. The same is done with the digital inputs and coils. The first 16 of them are displayed, starting with the addresses setup in **Digital IOs starting from:** edit boxes. If the user is pressing the **Read Only** button, the mode will toggle to **Write Only**. In this mode, any data that the user is entering in *Value* field will be written in the holding register of the RSN memory allocation, at the register number given by **Index**. Also, in **Write Only** mode, the user can click on coils to turn them on or off.

If the user is checking the option **Read as Float**, then 2 consecutive registers of 16 bits will be converted to a floating number of 32 bits and displayed as a real number in the **Holding Registers** and **Input Registers** value fields. For example, if the **Index** is 4, then the registers 4 and 5 will be used to represent the real number. Register 4 is the least significant word and register 5 is the most significant word.

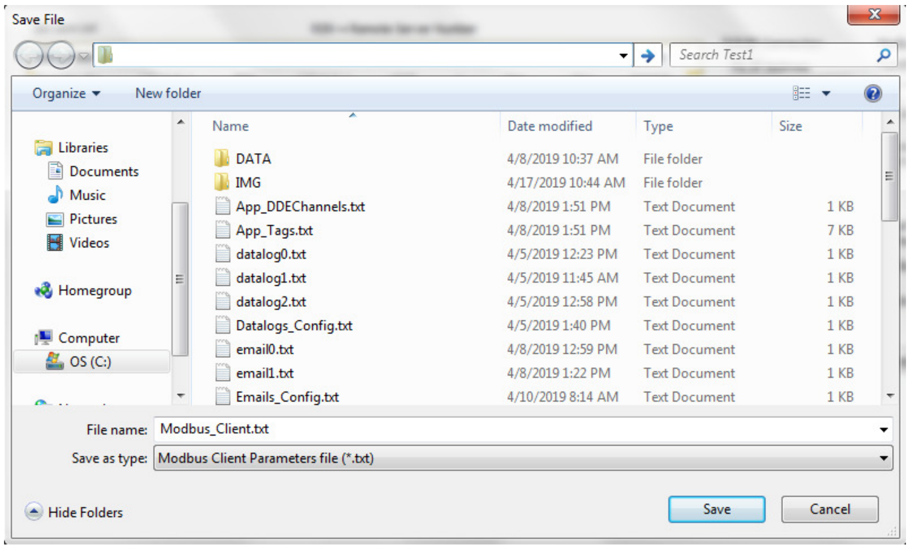
The user can troubleshoot the TCP/IP connections in the **Servers List** by watching the **Error** column. There are four digits that can be 0 – no error or 1 – error. From left to right they mean:

1	2	3	4
Connection Closed by Remote Peer	Connection Open Failed	No Answer	Socket Error

Once a connection is established, the user can troubleshoot the status of any Modbus request in the **Requests List**, by watching the column **Error**. The **Error** column has four digits that can be 0 – no error or 1 – error. These are errors from Modbus communication protocol. From left to right they mean:

1	2	3	4
Data Error	Exception Response	MBAP error	Time Out

Once the remote servers and the requests for each of them are defined, save the settings in the file *Modbus_Client.txt*. This file will be loaded whenever the program starts, if the check box **Auto load at start-up** is checked in the **Configuration** window. For this click the button **Save in file**, edit the file name and select the folder where the settings have to be saved. By default, *FeMODBUS* is checking for the file name in the folder where it is installed, but with a full file name it is possible to open a file from another folder. If you want to use a full file name from another folder than the *FeMODBUS* program, make sure the same full file name is typed in the **Configuration** window.



Name	Date modified	Type	Size
DATA	4/8/2019 10:37 AM	File folder	
IMG	4/17/2019 10:44 AM	File folder	
App_DDEChannels.txt	4/8/2019 1:51 PM	Text Document	1 KB
App_Tags.txt	4/8/2019 1:51 PM	Text Document	7 KB
datalog0.txt	4/5/2019 12:23 PM	Text Document	1 KB
datalog1.txt	4/5/2019 11:45 AM	Text Document	1 KB
datalog2.txt	4/5/2019 12:58 PM	Text Document	1 KB
Datalogs_Config.txt	4/5/2019 1:40 PM	Text Document	1 KB
email0.txt	4/8/2019 12:59 PM	Text Document	1 KB
email1.txt	4/8/2019 1:22 PM	Text Document	1 KB
Emails_Config.txt	4/10/2019 8:14 AM	Text Document	1 KB

File name: Modbus_Client.txt

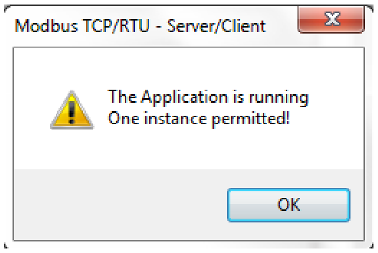
Save as type: Modbus Client Parameters file (*.txt)

4) Tags definition

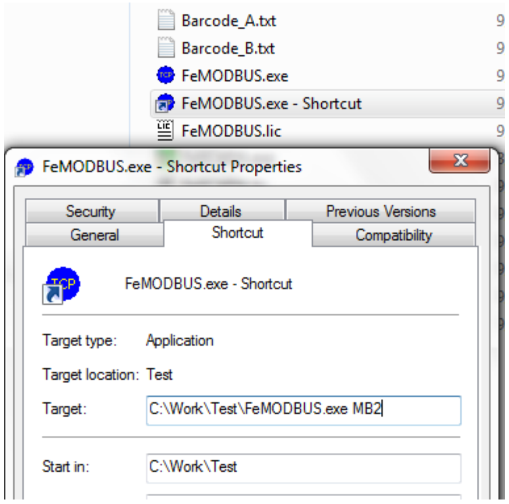
The tags in the *FeMODBUS* program are used for DDE communication. *FeMODBUS* is a DDE Server with the default settings:

DDE Application	DDE Topic	DDE Item
MB	TAGS	TagName

where "TagName" is the name of any tag defined by the user.


Modbus TCP/RTU - Server/Client

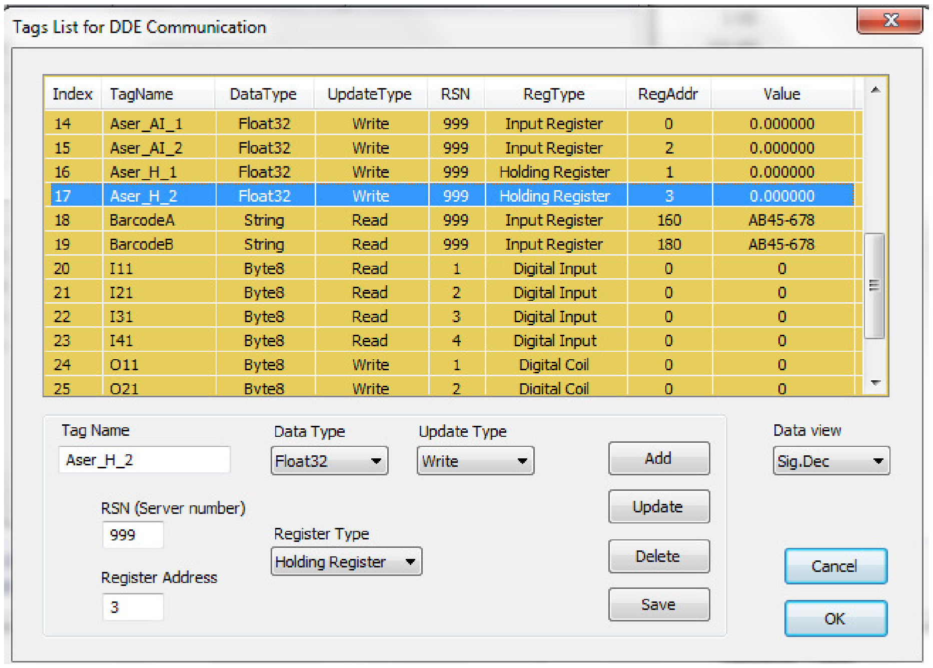
The Application is running
One instance permitted!



Target type: Application
Target location: Test
Target: C:\Work\Test\FeMODBUS.exe MB2
Start in: C:\Work\Test

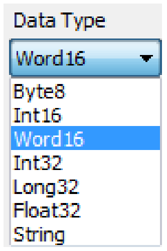
If *FeMODBUS* is running and the user wants to start another instance of it, a message will inform that the program is already running. The new instance will not start. This is because two DDE Servers with the same name cannot run correctly in the same time. The DDE Application ("MB") has to be a unique identifier. Another instance of *FeMODBUS* can be started if a shortcut link is made to *FeMODBUS* and the start command line is like: "FeMODBUS.exe MB2". In this case the DDE Application name will be "MB2" and it is possible to have multiple instances of the *FeMODBUS* program running in the same time.

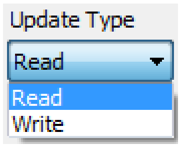
To start the setup dialog for the tags definition, select **Tags** from the menu (see picture below). To add a tag, type the tag name in **Tag Name** edit box, type the remote server number (**RSN**) to associate the tag with, type the register number in **Register Address**, select the **Register Type** (coil, discrete input, input register or holding register), select the **Data Type** (*Byte8*, *Int16*, *Word16*, *Int32*, *Long32*, *Float32*, *String*), select the **Update Type** (Read or Write) and press the **Add** button.

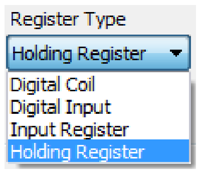

Tags List for DDE Communication

Index	TagName	DataType	UpdateType	RSN	RegType	RegAddr	Value
14	Aser_AI_1	Float32	Write	999	Input Register	0	0.000000
15	Aser_AI_2	Float32	Write	999	Input Register	2	0.000000
16	Aser_H_1	Float32	Write	999	Holding Register	1	0.000000
17	Aser_H_2	Float32	Write	999	Holding Register	3	0.000000
18	BarcodeA	String	Read	999	Input Register	160	AB45-678
19	BarcodeB	String	Read	999	Input Register	180	AB45-678
20	I11	Byte8	Read	1	Digital Input	0	0
21	I21	Byte8	Read	2	Digital Input	0	0
22	I31	Byte8	Read	3	Digital Input	0	0
23	I41	Byte8	Read	4	Digital Input	0	0
24	O11	Byte8	Write	1	Digital Coil	0	0
25	O21	Byte8	Write	2	Digital Coil	0	0

Tag Name Aser_H_2
Data Type Float32
Update Type Write
Data view Sig.Dec
RSN (Server number) 999
Register Type Holding Register
Register Address 3







Byte8 is an 8 bit integer with no sign. *Int16* is a 16 bit integer with sign. *Word16* is a 16 bit integer with no sign. The *Int32* is defined as 32 bit integer with sign. The *Long32* is a 32 bit integer with no sign. The *String* data type is maximum 100 bytes, zero value terminated.

For the data types *Int32*, *Long32*, and *Float32* the program is reading 2 consecutive 16 bits registers. The program is using the little-endian system: first word is the least significant, second word is the most significant.

To change the settings of a tag, click on the tag line from the tags list, change the settings and press **Update** button to memorize in the list. You can double click some of the list fields to edit in-place. To save the tags in file press **Save** button. You have to restart the program for the changes to take effect.

RSN = 0	a tag with **RSN** set to 0 (zero) is mapped to **Modbus RTU Server** registers
RSN = 999	a tag with **RSN** set to 999 is mapped to **Modbus TCP Server** registers

A tag that has the **Update Type** = *Read* will take the value of the register from the **RSN** number to which it is assigned. Other DDE Clients can read the value of the tag but cannot write to this tag.

A tag that has the **Update Type** = *Write* will copy the value from the tag's memory to the register of the **RSN** number to which it is assigned. Other DDE Clients can read and/or change the value of this tag.

5) Modbus TCP Server Configuration

The Modbus TCP Server feature is enabled as soon as the program starts. It can connect with maximum 16 clients at any time. The TCP Server has 200 coils, 200 discrete inputs, 200 input registers and 200 holding registers that the user can map with any tag from the tags list, if RSN is set to 999.

The discrete input number 0 (zero) is setup from the program to toggle its value with 2 Hz. It can be used for communication testing, as a heartbeat signal. The holding register number 0 (zero) is setup from the program to increment its value 10 times per second, from 0 to 65535, and starting again from 0. All other registers, with the exceptions described below, are free for being assigned to any tags and to any values that a project would require.

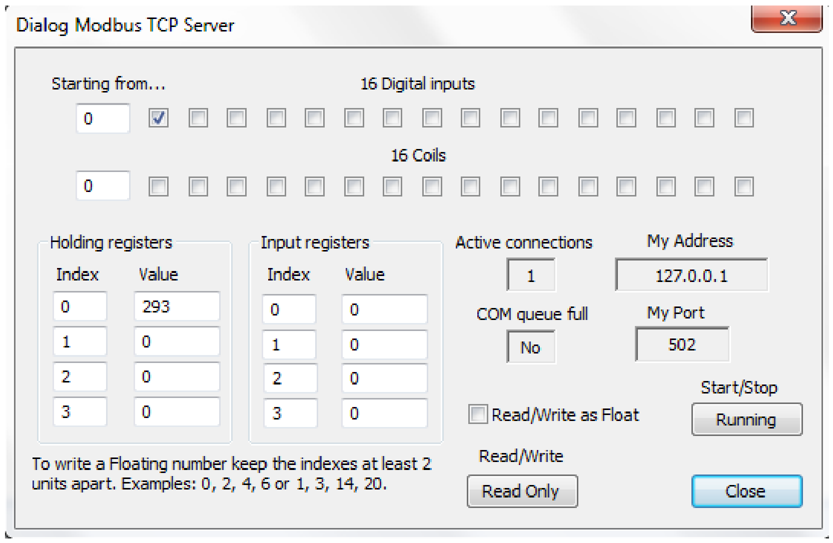
To open the dialog window with details about the Modbus TCP Server, select menu **Utils -> TCP Server**. On top, 16 consecutive discrete inputs and coils are displayed as check box controls, starting from the addresses in the left. The user can change the numbers. Also, the user can change the values of the indexes for input registers and holding registers, by selecting an **Index** number from 0 to 199.

In the right side there are details about the TCP Server status. **Active connections** shows how many clients are connected to the server. The IP address of the server is displayed in **My Address**. In **My Port** is the TCP port on which the server is listening. **COM queue full** is an indicator about the serial COM port assigned to work with the Modbus TCP Server, from **Configuration** window.

Any request to the Modbus TCP Server in which the **Unit Address** is less than 255 will be redirected to the COM serial port defined in the **Configuration** settings. The COM port has a buffer of 10 requests. If the buffer is full the **COM queue full** will show **Yes**, and the new requests will be denied with a Modbus Exception Error response. This is necessary to be able to keep the speed of the serial communication in safe limits. All the answers from the serial COM port will be sent back, over the TCP network, to the client that sent the request.

The user can stop/start the TCP Server by toggling on the **Start/Stop** button. By default the TCP Server will run when the program is started. The **Read/Write** button is used to toggle the dialog from display only (**Read Only**) to data input/change (**Write Only**). In the **Write Only** mode the user can set/reset the digital inputs and the coils, and can change the **Input registers** and **Holding registers** values.

If **Read/Write as float** is checked the values of the registers are interpreted as real numbers of 32 bits, two consecutive words. If **Write Only** mode is selected, the user can input real numbers. In this case the **Index** numbers have to be at least 2 units apart because writing a floating number at the address **X** will change the value of the registers at the addresses **X** and **X+1**. Two consecutive registers will corrupt the data to each other.



Starting from... 0 (16 Digital inputs); 0 (16 Coils)

Holding registers Index	Value	Input registers Index	Value
0	293	0	0
1	0	1	0
2	0	2	0
3	0	3	0

Active connections: 1
My Address: 127.0.0.1
COM queue full: No
My Port: 502

To write a Floating number keep the indexes at least 2 units apart. Examples: 0, 2, 4, 6 or 1, 3, 14, 20.

Special features of the Modbus TCP server

A joystick (or gamepad) and 2 barcode readers can be connected to the computer through USB ports. They are HID peripherals that will send WM_INPUT messages to *FeMODBUS* program. HID is acronym for Human Interface Device. These messages are processed and the results are transferred in some of the server input registers. In the table below the registers are described.

Setup for Joystick and Barcode readers

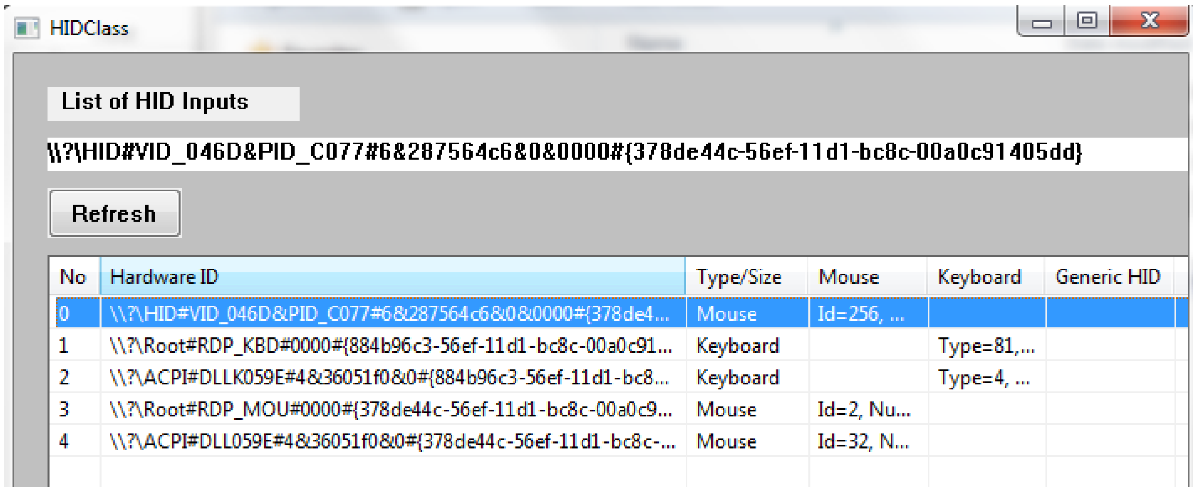
HID	HID message type	File name	String in file (Hardware ID - example)	Input registers
Joystick (gamepad)	RIM_TYPEHID (2)	Joystick.txt	\\?\HID#VID_046D&PID_C215#6&113e3c4&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}	150 – Byte 0 151 – Byte 1 152 – Byte 2 …. 157 – Byte 7
Barcode A	RIM_TYPEKEYBOARD (1)	Barcode1.txt	\\?\HID#VID_0416&PID_C1	Start at 160 Char values, zero terminated string
Barcode B	RIM_TYPEKEYBOARD (1)	Barcode2.txt	\\?\HID#VID_0416&PID_C1	Start at 180 Char values, zero terminated string

The files: “*Joystick.txt*”, “*Barcode1.txt*”” and “*Barcode2.txt*”, have to be created by the user and located in the same folder with FeMODBUS.exe.
The strings from the text files will be compared with the **Hardware ID** from the real devices. If they match then the data from that device will be processed and the Modbus TCP server input registers will be updated.

The **Hardware ID** string can be used with full description or shorter (the first part of the string). If it is shorter any device of that type (that match the string) can be recognized as valid and messages from it will be processed for input data. For example, the string “\\?\HID#VID” is common for any barcode reader or joystick, of different brands and models.

A full string for **Hardware ID** will provide security that the accepted data will come only from a certain device type, which is plugged in a certain USB port of the computer. For example, the string “\\?\HID#VID_0416&PID_C141#6&1a167eed&0&0000#{884b96c3-56ef-11d1-bc8c-00a0c91405dd}” is the **Hardware ID** of the barcode reader NADAMOO, model Bur3094, in the 3rd USB of my laptop.

The **Hardware ID** strings can be discovered with *FeMODBUS* program. Go to menu **Utils->HIDs**. The window presented below will open. It will show all HID devices. Select your device and copy the **Hardware ID** text of your connected joystick or barcode reader in the files Joystick.txt, Barcode1.txt or Barcode2.txt.



\\?\HID#VID_046D&PID_C077#6&287564c6&0&0000#{378de44c-56ef-11d1-bc8c-00a0c91405dd}

No	Hardware ID	Type/Size	Mouse	Keyboard	Generic HID
0	\\?\HID#VID_046D&PID_C077#6&287564c6&0&0000#{378de4...	Mouse	Id=256, ...		
1	\\?\Root#RDP_KBD#0000#{884b96c3-56ef-11d1-bc8c-00a0c91...	Keyboard		Type=81,...	
2	\\?\ACPI#DLLK059E#4&36051f0&0#{884b96c3-56ef-11d1-bc8...	Keyboard		Type=4, ...	
3	\\?\Root#RDP_MOU#0000#{378de44c-56ef-11d1-bc8c-00a0c9...	Mouse	Id=2, Nu...		
4	\\?\ACPI#DLL059E#4&36051f0&0#{378de44c-56ef-11d1-bc8c-...	Mouse	Id=32, N...		

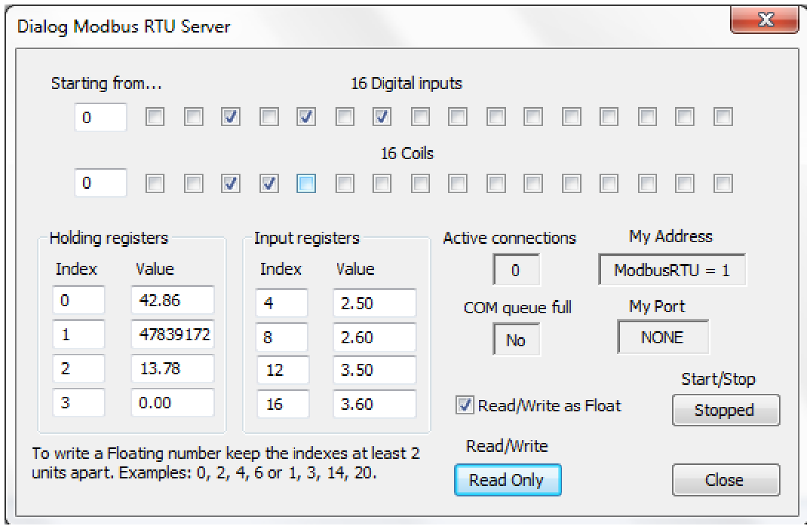
6) Modbus RTU Server Configuration

The RTU Server, when active, will open a serial COM port and will start listening for Modbus commands. There are 800 registers assigned for the RTU Server: 200 coils, 200 discrete inputs, 200 input registers, and 200 holding registers. The tags assigned for the RTU Server have to be defined with RSN = 0 (zero).

To open the dialog window with details about Modbus RTU Server, select menu **Utils -> RTU Server**.

As for the TCP Server, the user can toggle to **Start/Stop** the server and between **Read Only** and **Write Only** to display or modify registers. The input registers and holding registers can be read/write as floating numbers if the **Read/Write as Float** option is checked.

Active connections will be 1 if the RTU Server is running (the COM port is opened). By default, as long as the COM serial port is valid and not opened by another application, the RTU server will start running when the program is started. **My Address** is showing the RTU address of the server and **My Port** is showing the COM port number.



Dialog Modbus RTU Server

Starting from...

16 Digital inputs: 0 ☐ ☐ ☑ ☐ ☑ ☐ ☑ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

16 Coils: 0 ☐ ☐ ☑ ☑ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

Holding registers

Index	Value
0	42.86
1	47839172
2	13.78
3	0.00

Input registers

Index	Value
4	2.50
8	2.60
12	3.50
16	3.60

Active connections: 0

My Address: ModbusRTU = 1

COM queue full: No

My Port: NONE

☑ Read/Write as Float

Start/Stop: Stopped

Read/Write: Read Only

Close

To write a Floating number keep the indexes at least 2 units apart. Examples: 0, 2, 4, 6 or 1, 3, 14, 20.

7) Conclusions

FeMODBUS program is a versatile tool for communicating with other Modbus TCP Servers.

The program is a DDE Server for any DDE Client that is running on the same computer. The DDE settings are: DDE Application = “MB”, DDE Topic = “TAGS”, and DDE Item = “TagName”, where “TagName” is any tag defined by the user. More instances of *FeMODBUS* can be run in the same time if the start command line is setup like this: “FeMODBUS.exe NewDDEAppName”, where “NewDDEAppName” will be the new DDE Application name for that instance of the program. If “NewDDEAppName” already exists and is running, the instance will not start.

FeMODBUS is a Modbus TCP Client. For every server there are multiple requests that can be setup and sent to the server in a cyclic mode. There is no software limit for the number of TCP/IP connections that the program can open.

FeMODBUS is a Modbus TCP Server itself. More than that, it is a Modbus protocol bridge from the TCP/IP network to a serial line. It has a serial COM port that can be assigned to redirect the incoming Modbus commands, when the RTU address is less than 255. The answers from the serial COM port will be sent back on the TCP/IP network.

FeMODBUS is a Modbus RTU Server. A serial COM port has to be configured for this feature. All the tags assigned to the RTU Server will have to be defined with RSN = 0 (zero).