MODBUS TESTER SOFTWARE

USER MANUAL

TABLE OF CONTENTS

1. General information

The program is intended for testing purposes when MODBUS protocol is applicable.

Since MODBUS has wide usage very often installers and engineers have to use it. Normally devices that have modbus protocol comes with manufacturer interface and software, but sometimes is needed an external verifying program. That is the purpose of this simple modbus terminal.

It can do the following:

- Communication protocol Modbus RTU and/or Modbus TCP
- read of one or many Input registers
- read of one or many Holding registers
- read of records (strings in general case)*
- automatic read, range 100ms - 10 000 ms
- write of single Holding register
- write of multiple Holding registers
- data can be represent like:
  - 16-bit hex
  - 16-bit integer (decimal, 1 register)
  - 32-bit integer (decimal, 2 registers)
  - 64-bit integer (decimal, 4 registers)
  - 32-bit float (2 registers, IEEE)
  - 8-bit char

- full log of the communication in hexadecimal style

*in next version

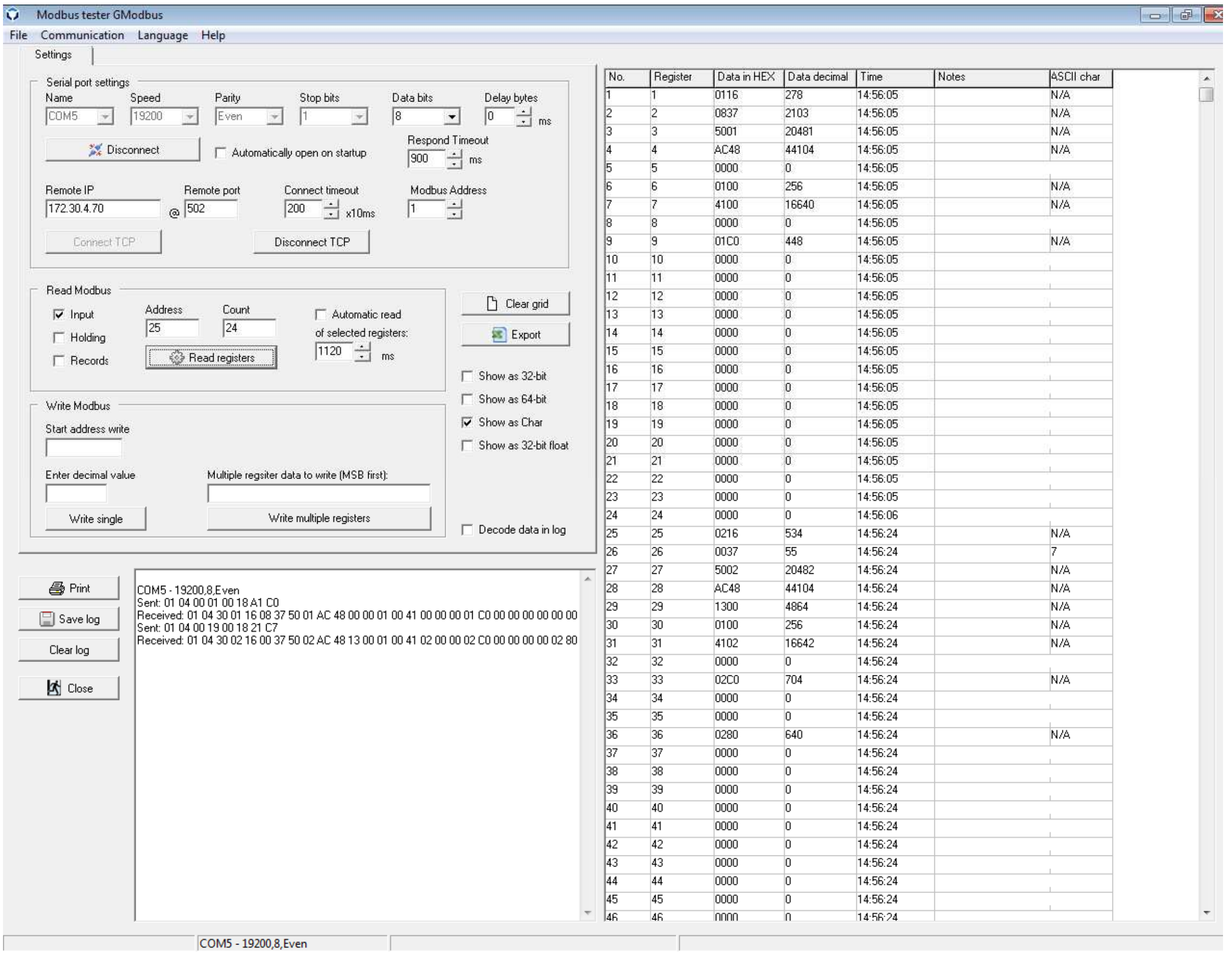
This software is intended for MS Windows operating system and will work on 2000/XP/Vista/Windows 7/Windows 8/Windows 10, no matter 32-bit or 64-bit platform.

2. Starting the program

The program is started with double click on the main executable file – "*Gmodbus.exe*". It is NOT writing to Your registry or anywhere else on Your computer. To use it just have to unzip/copy somewhere and start executable file.

3. Main window

The program consists of one TAB, which has several areas to work with. Main view of the program is shown on Fig. 1:



Serial port settings
Name: COM5
Speed: 19200
Parity: Even
Stop bits: 1
Data bits: 8
Delay bytes: 0 ms
☐ Automatically open on startup
Respond Timeout: 900 ms
Remote IP: 172.30.4.70 @ Remote port: 502
Connect timeout: 200 x10ms
Modbus Address: 1

Read Modbus
☑ Input
☐ Holding
☐ Records
Address: 25
Count: 24
☐ Automatic read of selected registers: 1120 ms

☐ Show as 32-bit
☐ Show as 64-bit
☑ Show as Char
☐ Show as 32-bit float

Write Modbus
Start address write:
Enter decimal value:
Multiple regsiter data to write (MSB first):
☐ Decode data in log

```
COM5 - 19200,8,Even
Sent: 01 04 00 01 00 18 A1 C0
Received: 01 04 30 01 16 08 37 50 01 AC 48 00 00 01 00 41 00 00 00 01 C0 00 00 00 00 00 00
Sent: 01 04 00 19 00 18 21 C7
Received: 01 04 30 02 16 00 37 50 02 AC 48 13 00 01 00 41 02 00 00 02 C0 00 00 00 00 02 80
```

No.	Register	Data in HEX	Data decimal	Time	Notes	ASCII char
1	1	0116	278	14:56:05		N/A
2	2	0837	2103	14:56:05		N/A
3	3	5001	20481	14:56:05		N/A
4	4	AC48	44104	14:56:05		N/A
5	5	0000	0	14:56:05		
6	6	0100	256	14:56:05		N/A
7	7	4100	16640	14:56:05		N/A
8	8	0000	0	14:56:05		
9	9	01C0	448	14:56:05		N/A
10	10	0000	0	14:56:05		
11	11	0000	0	14:56:05		
12	12	0000	0	14:56:05		
13	13	0000	0	14:56:05		
14	14	0000	0	14:56:05		
15	15	0000	0	14:56:05		
16	16	0000	0	14:56:05		
17	17	0000	0	14:56:05		
18	18	0000	0	14:56:05		
19	19	0000	0	14:56:05		
20	20	0000	0	14:56:05		
21	21	0000	0	14:56:05		
22	22	0000	0	14:56:05		
23	23	0000	0	14:56:05		
24	24	0000	0	14:56:06		
25	25	0216	534	14:56:24		N/A
26	26	0037	55	14:56:24		7
27	27	5002	20482	14:56:24		N/A
28	28	AC48	44104	14:56:24		N/A
29	29	1300	4864	14:56:24		N/A
30	30	0100	256	14:56:24		N/A
31	31	4102	16642	14:56:24		N/A
32	32	0000	0	14:56:24		
33	33	02C0	704	14:56:24		N/A
34	34	0000	0	14:56:24		
35	35	0000	0	14:56:24		
36	36	0280	640	14:56:24		N/A
37	37	0000	0	14:56:24		
38	38	0000	0	14:56:24		
39	39	0000	0	14:56:24		
40	40	0000	0	14:56:24		
41	41	0000	0	14:56:24		
42	42	0000	0	14:56:24		
43	43	0000	0	14:56:24		
44	44	0000	0	14:56:24		
45	45	0000	0	14:56:24		
46	46	0000	0	14:56:24		

COM5 - 19200,8,Even

Fig. 1. Main window (TAB 'Settings')

Program is set in TAB, because in the future its possible to update it with more features, if needed.

As seen from the figure, in this scenario, there are 5 main areas to work:

- set communication parameters and choose protocol
- read Input or Holding registers
- write Holding registers
- work with received data
- status window, where all communication is shown

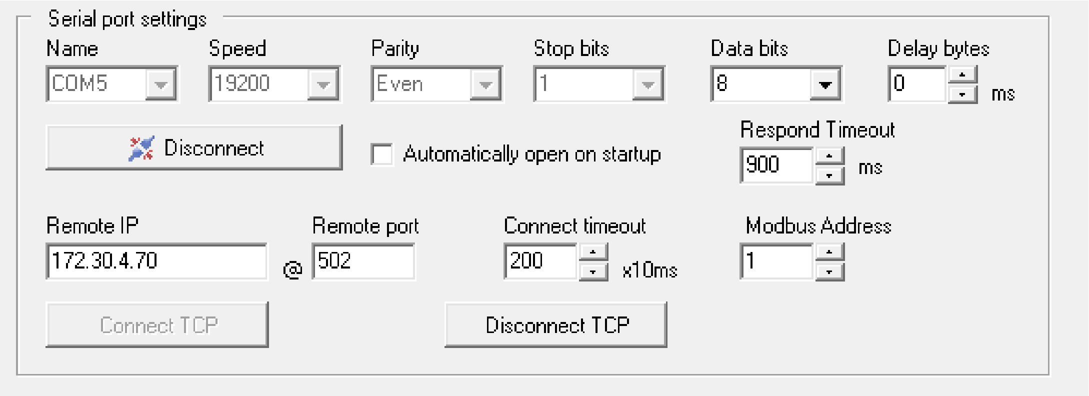
3.1. Setting communication

As known, there are two types of Modbus protocol - Modbus RTU and Modbus TCP. First is over serial RS-485/422 line and second is over 10/100 TCP. Both types are incorporated in Gmodbus, but before start communication, user should choose either serial port or Ethernet communication and do necessary settings.

About serial port (Modbus RTU) user can set following:

- serial port name/number (*COM1 - COM20*)

- serial port speed - all valid baud rates. Moreover, user can enter directly special baud rate and Gmodbus will work with this baud rate
- serial port parity - ***None***, *Even, Odd, Mark, Space*
- communication stop bits - ***1*** *or 2*
- communication data bits in a byte - *7 or* ***8***
- Delay between two consequential bytes, in ms - from **0** to 100 ms
- respond timeout - what time to wait external device to respond in modbus. This setting is also valid for Modbus TCP communication


Serial port settings

Name: COM5
Speed: 19200
Parity: Even
Stop bits: 1
Data bits: 8
Delay bytes: 0 ms
Respond Timeout: 900 ms
☐ Automatically open on startup
Remote IP: 172.30.4.70 @ Remote port: 502
Connect timeout: 200 x10ms
Modbus Address: 1

Fig. 2. Communication settings

About Ethernet port (Modbus TCP) user can set following:

- remote IP address - this is external modbus device IPv4 address
- remote port for trying to connect - modbus default **502**
- connection timeout - this is time, which Gmodbus will try to connect to external remote device. If no success - proper error will be shown in log window
- Modbus Address - mostly needed for Modbus RTU communication. **1** to 255.

Only one type of connection at a time is allowed. This means that if you are connected on serial port, Gmodbus will not allow you to connect also through TCP to another device. First you have to detach from serial port, then TCP connection will become available and vice-versa.

Worth mention again that you can write desired baud rate in *'Speed"* combo box. Then serial port will work on that speed for You.

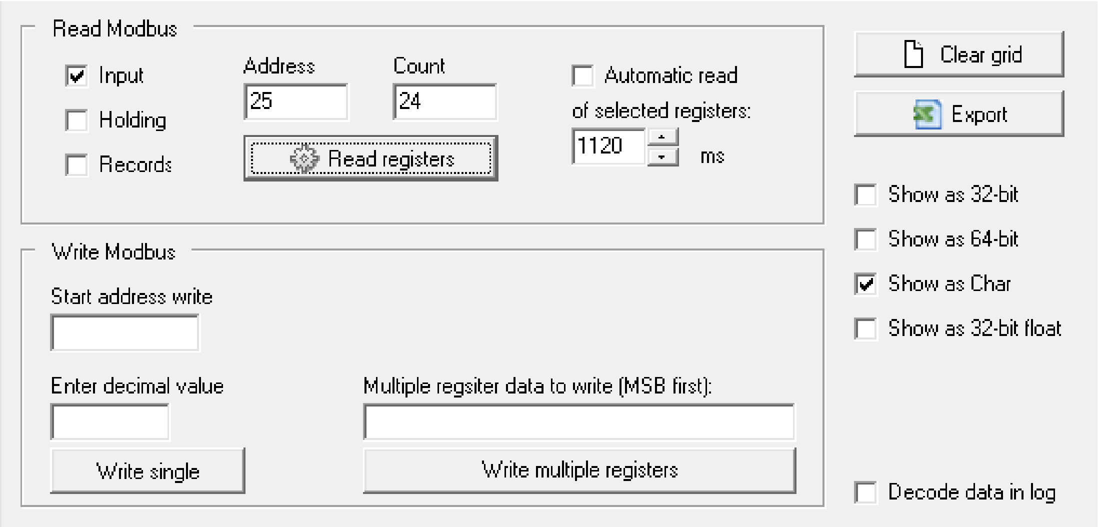
Serial port of PC can be set for automatic opening when program starts. When settings of serial port are gray this notify that the port is open. Program works in mode Master-Slave. This means that program sends command to ModBus interface (reading or writing registers) and then waits for device response. How much time will wait the response can be set in *"Respond Timeout, ms"* property, right to serial port settings. Since ModBus has no special stop byte in the protocol and you can not tell automatically how many registers will be returned user should set this timeout according to speed and reading registers. Something between

600ms and **1000ms** should be enough for most of the commands, but this generally depends on what amount of bytes one is reading/writing and on device.

3.2. Read and write registers

There are five operations you can do:

- read Input registers
- read Holding registers
- read Records
- write single Holding register
- write multiple Holding registers



Read Modbus

☑ Input
□ Holding
□ Records

Address: 25
Count: 24

□ Automatic read of selected registers: 1120 ms

Write Modbus

Start address write:
Enter decimal value:
Multiple regsiter data to write (MSB first):

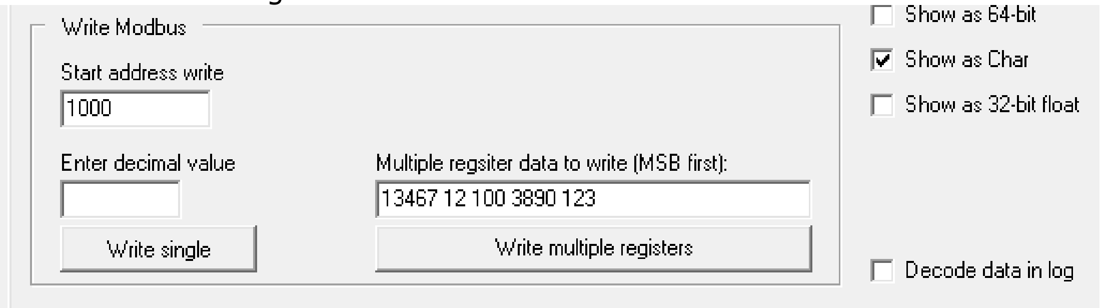
□ Show as 32-bit
□ Show as 64-bit
☑ Show as Char
□ Show as 32-bit float
□ Decode data in log

Fig.3. Read/Write modbus registers

Window is pretty self-explainable. Will do a note only on multiple write register option.

In order to make multiple register write, user should enter values he wants in **DECIMAL** values, separated by ' ' (SPACE).

This is shown on figure below:



Write Modbus

Start address write: 1000
Enter decimal value:
Multiple regsiter data to write (MSB first): 13467 12 100 3890 123

□ Show as 64-bit
☑ Show as Char
□ Show as 32-bit float
□ Decode data in log

Fig. 4. Multiple register write

On this figure we see the following:

- data to write will be *13467, 12, 100, 3890, 123*
- these value will go in Holding registers from 1000 to 1004, including

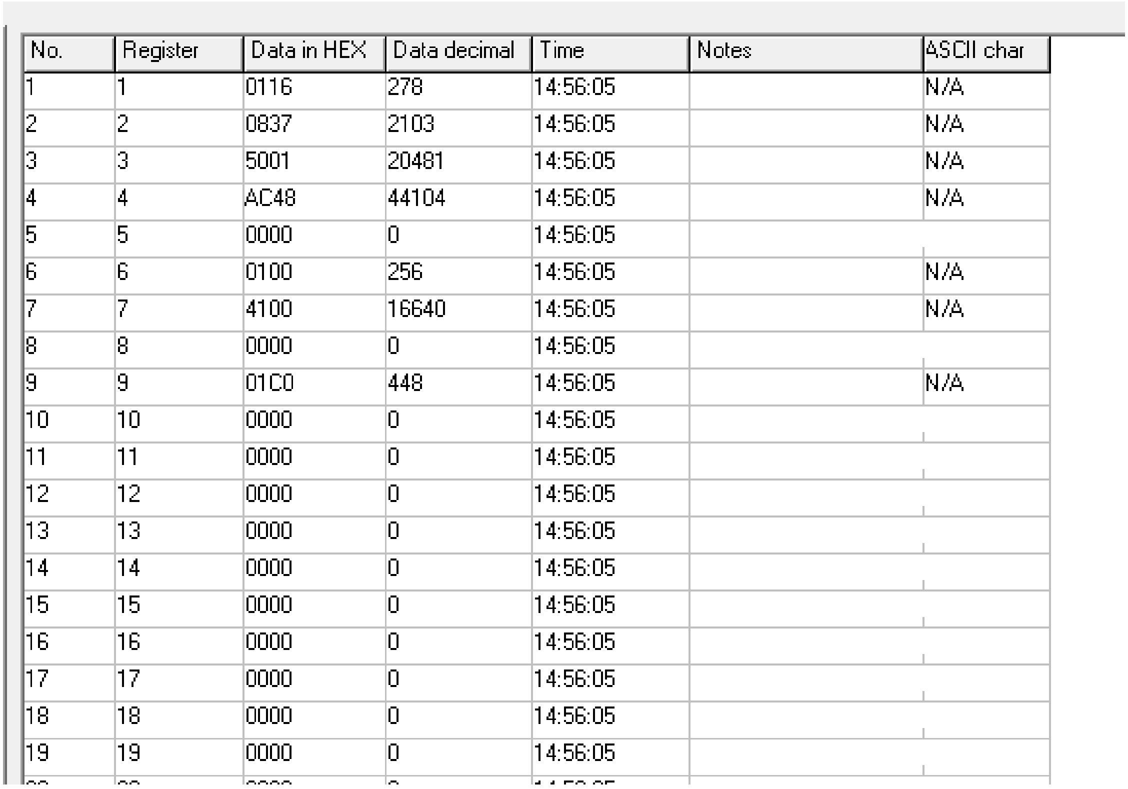
Since each modbus register is 16-bit, user is not allowed to write a number above **65535**. If one try to do this, then nothing will happen and proper error message will be shown in log window. Same will happen, if you try to send a negative number.

This is valid also when user tries to write single register through Modbus.

3.3. Decoding received data

When command is send, usually external device responds with data requested (or error telegram). The data in Gmodbus is decoded and filled in result table. For each received register data is shown:

- consequential number
- register real number
- data in this register in HEX value
- data in this register in decimal value
- notes (usually program hints)



No.	Register	Data in HEX	Data decimal	Time	Notes	ASCII char
1	1	0116	278	14:56:05		N/A
2	2	0837	2103	14:56:05		N/A
3	3	5001	20481	14:56:05		N/A
4	4	AC48	44104	14:56:05		N/A
5	5	0000	0	14:56:05		
6	6	0100	256	14:56:05		N/A
7	7	4100	16640	14:56:05		N/A
8	8	0000	0	14:56:05		
9	9	01C0	448	14:56:05		N/A
10	10	0000	0	14:56:05		
11	11	0000	0	14:56:05		
12	12	0000	0	14:56:05		
13	13	0000	0	14:56:05		
14	14	0000	0	14:56:05		
15	15	0000	0	14:56:05		
16	16	0000	0	14:56:05		
17	17	0000	0	14:56:05		
18	18	0000	0	14:56:05		
19	19	0000	0	14:56:05		

Fig. 5. Results

Additionally, user can combine the data and view it in the following types:

- 32-bit decimal value - combines TWO modbus registers, MSB first
- 64-bit decimal value - combines FOUR modbus register, MSB first

- 32-bit float value - combines TWO modbus registers, but converts the result to IEEE 32-bit float value
- char - shows register as ASCII char. This works only to 255, because program uses standard ASCII table.

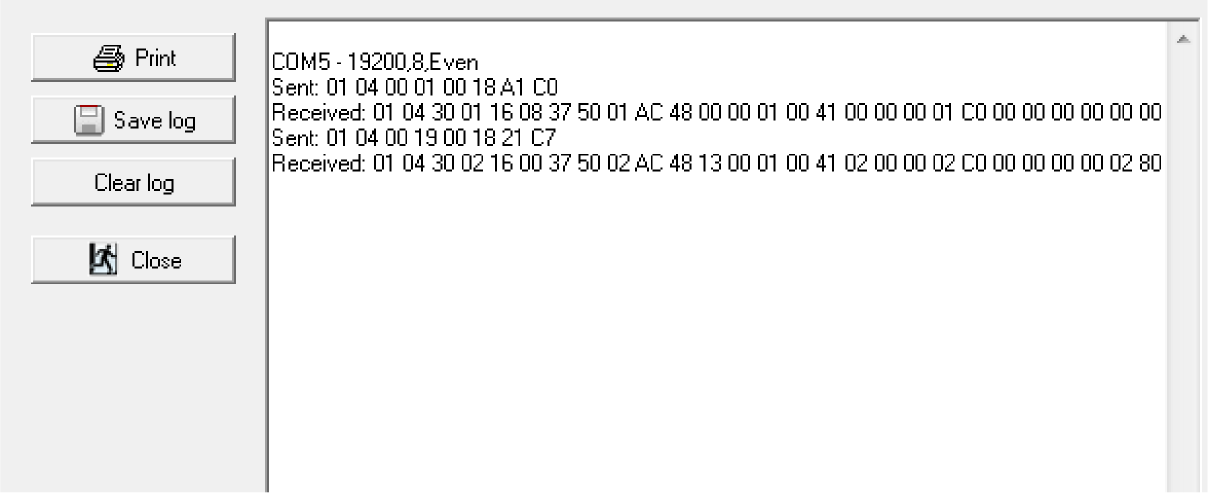
Table can be exported in CSV, in order to be open by MSExcel. Or cleared.

When user wants to read in automatic mode same registers, then table is updated, but not expanded.

3.4. LOG window

In log window user can see the following:

- full telegrams from communication
- messages from the program for incorrect input data or actions
- exceptions of any kind (which should not happen at all)



```
COM5 - 19200,8,Even
Sent: 01 04 00 01 00 18 A1 C0
Received: 01 04 30 01 16 08 37 50 01 AC 48 00 00 01 00 41 00 00 00 01 C0 00 00 00 00 00 00
Sent: 01 04 00 19 00 18 21 C7
Received: 01 04 30 02 16 00 37 50 02 AC 48 13 00 01 00 41 02 00 00 02 C0 00 00 00 00 02 80
```

Fig. 6. Status window

Log information can be saved to text file, printed or cleared.

4. Help

This menu contains two options:

About – shows information about the program

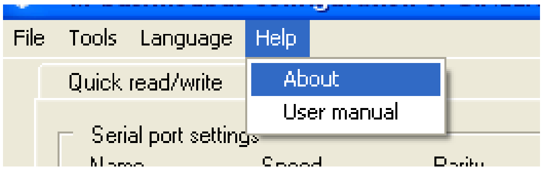


Fig. 10. Help

User manual – starts this user manual

5. Contacts

"GINEERS" Ltd. - Electronics, automation and software

1528, Sofia, Bulgaria

7 “Iskarsko shausse” blvd, building 7

phone/fax: +359 2 975 81 05

office@gineers.com

If any questions, suggestions or troubles using this program, please write to support@gineers.com or ivan@gineers.com