

Modbus RTU made simple with detailed descriptions and examples

From this article, you will learn about the Modbus RTU protocol, which is widely used in the process control system.

[Detailed description of the Modbus TCP protocol with command examples](#)

Contents:

- [Modbus RTU protocol description](#)
- [What are Modbus RTU commands?](#)
- [How can I send a Modbus RTU command to read discrete output? Command 0x01](#)
- [How can I send a Modbus RTU command to read a digital input? Command 0x02](#)
- [How can I send a Modbus RTU command to read analog output? Command 0x03](#)
- [How can I send the Modbus RTU command to read the analog input? Command 0x04](#)
- [How can I send a Modbus RTU command to write discrete output? Command 0x05](#)
- [How can I send a Modbus RTU command to record analog output? Command 0x06](#)
- [How can I send a Modbus RTU command to write multiple discrete pins? Command 0x0F](#)
- [How can I send a Modbus RTU command to record multiple analog outputs? Command 0x10](#)
- [What are the errors of the Modbus request?](#)

- [Programs for working with Modbus RTU protocol](#)
- [Equipment with Modbus RTU support](#)

Modbus RTU protocol description

Modbus - communication protocol is based on the master-slave architecture. It uses RS-485, RS-422, and RS-232 interfaces, as well as Ethernet TCP / IP networks (Modbus TCP protocol) for data transfer.

The Modbus RTU message consists of the address of the SlaveID device, the function code, and the special data, depending on the function code and the CRC of the checksum.

SlaveID	Function code	Special data	CRC
---------	---------------	--------------	-----

If you discard the SlaveID address and the CRC checksum, you get the PDU, Protocol Data Unit.

SlaveID is the address of the device, it can take a value from 0 to 247, and addresses from 248 to 255 are reserved.

Data in the module is stored in 4 tables.

Two tables are read-only and two are read-write.

9999 values are placed in each table. read-write

REGISTER NUMBER	REGISTER ADDRESS HEX	TYPE	NAME	TYPE
1-9999	0000 to 270E	read-write	Discrete Output Coils	DO
10001-19999	0000 to 270E	read	Discrete Input Contacts	DI
30001-39999	0000 to 270E	read	Analog Input Registers	AI
40001-49999	0000 to 270E	read-write	Analog Output Holding Registers	AO

The Modbus message uses the register address.

For example, the **first** register of AO Holding Register has the **number** 40001, but its **address** is 0000.

The difference between these two quantities is “offset”.

Each table has its own offset, respectively: 1, 10001, 30001 and 40001.

The following is an example of a Modbus RTU request for obtaining the AI value of the holding registers from registers # 40108 to 40110 with the address of device 17.

11 03 006B 0003 7687

11	THE ADDRESS OF THE SLAVEID DEVICE (17 = 11 HEX)
03	Functional code Function Code
006B	The address of the first register (40108-40001 = 107 = 6B hex)
0003	The number of required registers (reading 3 registers from 40108 to 40110)
7687	CRC checksum

In response to the Modbus RTU Slave device we get:

11 03 06 AE41 5652 4340 49AD

Where: The value of the upper register bit

11	DEVICE ADDRESS (17 = 11 hex)	SlaveID
03	Function code	Function Code
06	The number of bytes further (6 bytes follow)	Byte Count
AE	The value of the upper register bit (AE hex)	Register value Hi (AO0)
41	The low-order bit of the register (41 hex)	Register value Lo (AO0)
56	The value of the upper register bit (56 hex)	Register value Hi (AO1)
52	The low-order bit of the register (52 hex)	Register value Lo (AO1)

43	The value of the upper register bit (43 hex)	Register value Hi (AO2)
40	The low-order bit of the register (40 hex)	Register value Lo (AO2)
49	Checksum	CRC value Hi
AD	Checksum	CRC value Lo

The analog output register AO0 has the value AE 41 HEX or 44609 in the decimal system.

The analog output register AO1 has a value of 56 52 HEX or 22098 in the decimal system.

The analog output register AO2 has a value of 43 40 HEX or 17216 in the decimal system.

The AE 41 HEX value is 16 bits 1010 1110 0100 0001, which can take a different value, depending on the type of representation.

The value of register 40108 when combined with register 40109 gives a 32-bit value.

An example of a representation.

View type	Value Range	Example in HEX	In decimal form
16-bit unsigned integer	0 to 65535	AE41	44,609
16-bit signed integer	-32768 to 32767	AE41	-20,927
two-character ASCII string	2 char	AE41	® A
discrete on/off value	0 and 1	0001	0001
32-bit unsigned integer	0 to 4,294,967,295	AE41 5652	2,923,517,522
32-bit signed integer	-2,147,483,648 to 2,147,483,647	AE41 5652	- 1,371,449,774
32-bit single precision IEEE floating point number	$1,2 \cdot 10^{-38}$ to $3,4 \times 10^{+38}$	AE41 5652	-4.395978 E- 11

four-character ASCII string	4 char	AE41 5652	® A V R
-----------------------------	--------	--------------	---------

[Back to contents](#)

What are Modbus RTU commands?

Here is a table with the codes for reading and writing the Modbus RTU registers.

FUNCTION CODE	WHAT THE FUNCTION DOES		VALUE TYPE	ACCESS TYPE
01 (0x01)	Read DO	Read Coil Status	Discrete	Read
02 (0x02)	Read DI	Read Input Status	Discrete	Read
03 (0x03)	Read AO	Read Holding Registers	16 bit	Read
04 (0x04)	Read AI	Read Input Registers	16 bit	Read
05 (0x05)	Write one DO	Force Single Coil	Discrete	Write
06 (0x06)	Write one AO	Preset Single Register	16 bit	Write
15 (0x0F)	Multiple DO recording	Force Multiple Coils	Discrete	Write
16 (0x10)	Multiple AO recording	Preset Multiple Registers	16 bit	Write

[Back to contents](#)

How can I send a Modbus RTU command to read discrete output? Command 0x01

This command is used to read the values of the DO digital outputs.

The PDU request specifies the start address of the first DO register and the subsequent number of required DO values. In the PDU, the DO values are addressed starting from zero.

The DO values in the response are in one byte and correspond to the value of the bits.

The bit values are defined as 1 = ON and 0 = OFF.

The low bit of the first data byte contains the DO value whose address was specified in the request. The remaining values of DO follow the increasing value to the highest value of the byte. Those. from right to left.

If less than eight DO values were requested, the remaining bits in the response will be filled with zeros (in the direction from the low to high byte). Byte Count **The number of bytes further** indicates the number of full bytes of data in the response.

Example of a DO query from 20 to 56 for the device's SlaveID address 17. The address of the first register will be 0013 hex = 19, because The account is maintained from 0 address (0014 hex = 20, -1 zero offset = we get 0013 hex = 19).

BYTE	REQUEST	BYTE	ANSWER
(Hex)	Field name	(Hex)	Field name
11	Device address	11	Device address
01	Functional code	01	Functional code
00	Address of the first register Hi bytes	05	Number of bytes more
13	Address of the first register Lo bytes	CD	Register value DO 27-20 (1100 1101)
00	Number of registers Hi bytes	6B	Register value DO 35-28 (0110 1011)
25	Number of registers Lo bytes	B2	Register value DO 43-36 (1011 0010)
0E	Checksum CRC	0E	Register value DO 51-44 (0000 1110)
84	Checksum CRC	1B	Register value DO 56-52 (0001 1011)

	45	Checksum CRC
	E6	Checksum CRC

The output states of DO 27-20 are shown as the values of the byte CD hex, or in the binary system 1100 1101.

In register DO 56-52, 5 bits on the right were requested, and the remaining bits are filled with zeros to the full byte (**0001 1011**).

Channels	-	-	-	DO 56	DO 55	DO 54	DO 53	DO 52
Bits	0	0	0	1	1	0	1	1
Hex	1B							

[Back to contents](#)

How can I send a Modbus RTU command to read a digital input? Command 0x02

This command is used to read the values of digital inputs DI.

Example of a DI request from the registers from # 10197 to 10218 for the device's SlaveID address 17. The address of the first register will be 00C4 hex = 196, because the Account is maintained from 0 address.

BYTE	REQUEST	BYTE	ANSWER
(Hex)	Field name	(Hex)	Field name
11	Device address	11	Device address
02	Functional code	02	Functional code
00	Address of the first register Hi bytes	03	Number of bytes more
C4	Address of the first register Lo bytes	AC	Register value DI 10204-10197 (1010 1100)

00	Number of registers Hi bytes	DB	Register value DI 10212-10205 (1101 1011)
16	Number of registers Lo bytes	35	Register value DI 10218-10213 (0011 0101)
BA	Checksum CRC	20	Checksum CRC
A9	Checksum CRC	18	Checksum CRC

[Back to contents](#)

How can I send a Modbus RTU command to read analog output? Command 0x03

This command is used to read the values of the analog outputs AO.

Example of an AO request from registers from # 40108 to 40110 for the SlaveID of the device address 17. The address of the first register will be 006B hex = 107, because Account is maintained from 0 address.

BYTE	REQUEST	BYTE	ANSWER
(Hex)	Field name	(Hex)	Field name
11	Device address	11	Device address
03	Functional code	03	Functional code
00	Address of the first register Hi bytes	06	Number of bytes more
6B	Address of the first register Lo bytes	AE	Register value Hi #40108
00	Number of registers Hi bytes	41	Register value Lo #40108
03	Number of registers Lo bytes	56	Register value Hi #40109
76	Checksum CRC	52	Register value Lo #40109
87	Checksum CRC	43	Register value Hi #40110
		40	Register value Lo #40110

	49	Checksum CRC
	AD	Checksum CRC

[Back to contents](#)

How can I send the Modbus RTU command to read the analog input? Command 0x04

This command is used to read the values of analog inputs AI.

Example of an AI request from the register # 30009 for the SlaveID of the device address 17. The address of the first register is 0008 hex = 8, because Account is maintained from 0 address.

BYTE	REQUEST	BYTE	ANSWER
(Hex)	Field name	(Hex)	Field name
11	Device address	11	Device address
04	Functional code	04	Functional code
00	Address of the first register Hi bytes	02	Number of bytes more
08	Address of the first register Lo bytes	00	Register value Hi #30009
00	Number of registers Hi bytes	0A	Register value Lo #30009
01	Number of registers Lo bytes	F8	Checksum CRC
B2	Checksum CRC	F4	Checksum CRC
98	Checksum CRC		

[Back to contents](#)

How can I send a Modbus RTU command to write discrete output? Command 0x05

This command is used to record one value of the DO digital output.

The value of FF 00 hex sets the output to ON.

The value 00 00 hex sets the output to OFF.

All other values are invalid and will not be affected by the output value.

The normal response to such a request is an echo (a repeat request in the response), is returned after the DO state has been changed.

An example of a DO record with register # 173 for the SlaveID address of the device 17. The register address will be 00AC hex = 172, because Account is maintained from 0 address.

BYTE	REQUEST	BYTE	ANSWER
(Hex)	Field name	(Hex)	Field name
11	Device address	11	Device address
05	Functional code	05	Functional code
00	Address of the first register Hi bytes	00	Address of the first register Hi bytes
AC	Address of the first register Lo bytes	AC	Address of the first register Lo bytes
FF	Value of Hi bytes	FF	Value of Hi bytes
00	Value of Lo bytes	00	Value of Lo bytes
4E	Checksum CRC	4E	Checksum CRC
8B	Checksum CRC	8B	Checksum CRC

The DO173 output state has changed from OFF to ON.

[Back to contents](#)

How can I send a Modbus RTU command to record analog output? Command 0x06

This command is used to record one value of the analog output AO.

Example of recording in AO with register # 40002 for SlaveID address of the device 17. The address of the first register will be 0001 hex = 1, because Account is maintained from 0 address.

BYTE	REQUEST	BYTE	ANSWER
(Hex)	Field name	(Hex)	Field name
11	Device address	11	Device address
06	Functional code	06	Functional code
00	Address of the first register Hi bytes	00	Address of the first register Hi bytes
01	Address of the first register Lo bytes	01	Address of the first register Lo bytes
00	Value of Hi bytes	00	Value of Hi bytes
03	Value of Lo bytes	03	Value of Lo bytes
9A	Checksum CRC	9A	Checksum CRC
9B	Checksum CRC	9B	Checksum CRC

[Back to contents](#)

How can I send a Modbus RTU command to write multiple discrete pins? Command 0x0F

This command is used to record multiple values of DO's digital output.

An example of writing in several DOs with registers from # 20 to # 29 for the SlaveID address of the device 17. The register address will be 0013 hex = 19, since Account is maintained from 0 address.

BYTE	REQUEST	BYTE	ANSWER
(Hex)	Field name	(Hex)	Field name
11	Device address	11	Device address
0F	Functional code	0F	Functional code
00	Address of the first register Hi bytes	00	Address of the first register Hi bytes
13	Address of the first register Lo bytes	13	Address of the first register Lo bytes
00	Number of registers Hi bytes	00	Number of recorded registers Hi bytes
0A	Number of registers Lo bytes	0A	Number of recorded registers Lo bytes
02	Number of bytes more	26	Checksum CRC
CD	Byte Value DO 27-20 (1100 1101)	99	Checksum CRC
01	Byte Value DO 29-28 (0000 0001)		
BF	Checksum CRC		
0B	Checksum CRC		

The answer returns the number of registers recorded.

[Back to contents](#)

How can I send a Modbus RTU command to record multiple analog outputs? Command 0x10

⌘ This command is used to record multiple values of the analog output AO.

An example of recording in several AO with registers # 40002 and # 40003 for the SlaveID address of the device 17. The address of the first register will be 0001 hex = 1, because Account is maintained from 0 address.

BYTE	REQUEST	BYTE	ANSWER
(Hex)	Field name	(Hex)	Field name
11	Device address	11	Device address
10	Functional code	10	Functional code
00	Address of the first register Hi bytes	00	Address of the first register Hi bytes
01	Address of the first register Lo bytes	01	Address of the first register Lo bytes
00	Number of registers Hi bytes	00	Number of recorded registers Hi bytes
02	Number of registers Lo bytes	02	Number of recorded registers Lo bytes
04	Number of bytes more	12	Checksum CRC
00	Value Hi 40002	98	Checksum CRC
0A	Value Lo 40002		
01	Value Hi 40003		
02	Value Lo 40003		
C6	Checksum CRC		
F0	Checksum CRC		

[Back to contents](#)

What are the errors of the Modbus request?

If the device receives a request, but the request can not be processed, the device will respond with an error code.

The response will contain the modified Function code, the high-order bit will be 1.

Example:

IT WAS	IT BECOME
FUNCTIONAL CODE IN REQUEST	Functional error code in response
01 (01 hex) 0000 0001	129 (81 hex) 1000 0001
02 (02 hex) 0000 0010	130 (82 hex) 1000 0010
03 (03 hex) 0000 0011	131 (83 hex) 1000 0011
04 (04 hex) 0000 0100	132 (84 hex) 1000 0100
05 (05 hex) 0000 0101	133 (85 hex) 1000 0101
06 (06 hex) 0000 0110	134 (86 hex) 1000 0110
15 (0F hex) 0000 1111	143 (8F hex) 1000 1111
16 (10 hex) 0001 0000	144 (90 hex) 1001 0000

Sample request and response with error:

BYTE	REQUEST	BYTE	ANSWER
(Hex)	Field name	(Hex)	Field name
0A	Device address	0A	Device address
01	Functional code	81	Functional code with changed bit
04	Address of the first register Hi bytes	02	Error code
A1	Address of the first register Lo bytes	B0	Checksum CRC
00	Number of registers Hi bytes	53	Checksum CRC
01	Number of registers Lo bytes		
AC	Checksum CRC		

63	Checksum CRC	
----	--------------	--

Explanation of error codes

01	FUNCTION CODE ACCEPTED CAN NOT BE PROCESSED.
02	The data address specified in the request is not available.
03	The value contained in the query data field is an invalid value.
04	An unrecoverable error occurred while the slave attempted to perform the requested action.
05	The slave has accepted the request and processes it, but it takes a long time. This response prevents the host from generating a timeout error.
06	The slave is busy processing the command. The master must repeat the message later when the slave is freed.
07	The slave can not execute the program function specified in the request. This code is returned for an unsuccessful program request using functions with numbers 13 or 14. The master must request diagnostic information or error information from the slave.
08	The slave detected a parity error when reading the extended memory. The master can repeat the request, but usually in such cases, repairs are required.

[Back to contents](#)

Programs for working with Modbus RTU protocol

The following are the programs that make it easier to work with Modbus.

DCON Utility Pro with support for Modbus RTU, ASCII, DCON. [Download](#)

Modbus Master Tool with support for Modbus RTU, ASCII, TCP. [Download](#)

Modbus TCP client with Modbus TCP support. [Download](#)

[Back to contents](#)

Equipment with Modbus RTU support

Gateway

Industrial protocol gateways provide equipment compatibility of different vendors in one automation solution. Gateways support MODBUS, CAN, HART and other protocols.

RS-485 Modules tM

The tM series is a compact version of the M-7000 series. Except the standard DCON and Modbus RTU protocols the Modbus ASCII protocol is available.

RS-485 Modules ADAM-4000

Modules of analog and discrete input-output of the ADAM-4000 series with the RS-485 interface and the ASCII and Modbus RTU protocols.

RS-485 / RS-232 Modules SCM9B

Modules, remote analog and discrete I / o series SCM9B interfaces RS-232C or RS-485 and ASCII Protocol.

RS-485 Control Modules Lighting LC

LC series modules are designed for economic control of lighting devices via the controller or the computer.

RS-485 Infrared IR Modules

IR series modules are constructed for control of various devices (TVs, air conditioners, etc.) via the infrared channel.

Compact Terminal TPD and VPD

Compact terminals of input-output of the TPD and VPD series with the color screen, touch management and Ethernet and COM communication interfaces.

Temperature and humidity meters

DL devices are constructed for temperature, humidity and concentration of CO2 measurements. They can archive data and send it via DCON, Modbus TCP or Modbus RTU protocols.

[Back to contents](#)

[Detailed description of the Modbus TCP protocol with command examples](#)