Modbus RTU Analog Output 8CH (B)

From Waveshare Wiki
Jump to: navigation, search

Overview

Hardware Description

- AO1~AO8 is the analog output, AGND is the analog ground, and the output range is 0-10V.

Modbus RTU Analog Output 8CH (B)



(https://www.waveshare.com/modbus-rtu-analog-output-8ch.htm)

Voltage Analog Output Module, RS485



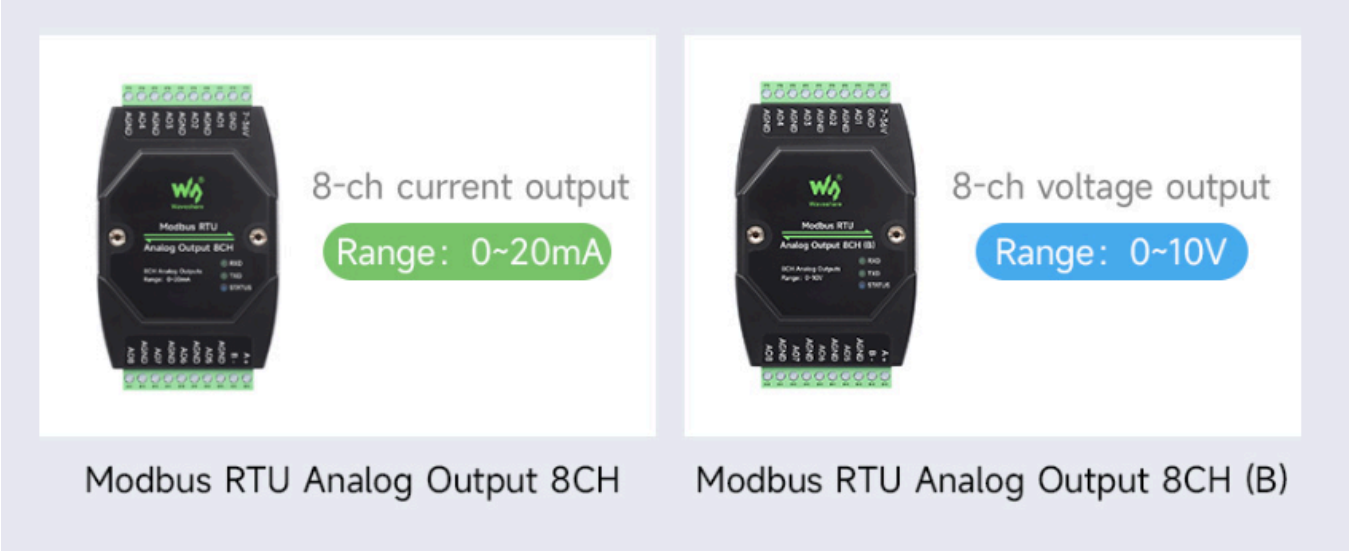
(/wiki/File:Modbus_RTU_Analog_Output_8CH-b-01.jpg)

Note: The input voltage must be greater than the output voltage. Insufficient input voltage may result in insufficient output voltage. For example, to output 10V, the input voltage must be at least 15V.

Version Comparision

Currently, the analog output has two versions: the current output and the voltage output.

- Modbus RTU Analog Output 8CH outputs the current with the range of 0-20mA.
- Modbus RTU Analog Output 8CH (B) outputs the voltage with the range of 0-10V.



(/wiki/File:Modbus_RTU_Analog_Output_8CH-02.png)

Hardware Connection

- Connect the USB TO 485 and the target board through the wires, connects A --> A and B --> B as shown below:



(/wiki/File:Modbus_RTU_Analog_Output_8CH-03.jpg)

Software Test

SSCOM

- Download SSCOM (https://files.waveshare.com/upload/b/b3/Sscom5.13.1.zip) and open it on the PC, open the corresponding port, set the baud rate of 9600, and select SendHEX and HEXshow.

Send the following commands, channel 1 will then output 1V.

```
01 04 00 00 00 08 F1 CC
```

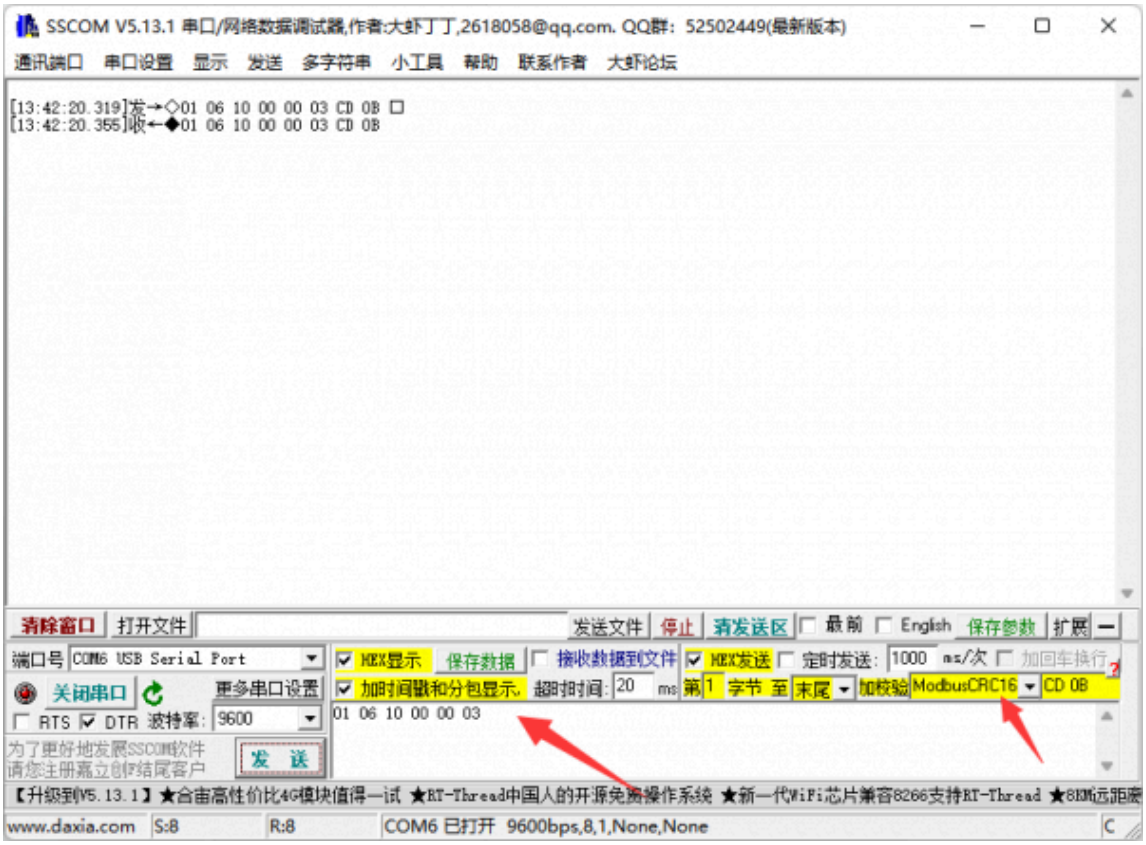


(/wiki/File:Modbus_RTU_Analog_Output_8CH-b-SSCOM.png)

- If you need to send other commands then select SendHEX, set Verify as ModbusCRC16, enter the first six bytes of the command, and click SEND then the CRC checksum will be added automatically.

For example, send the following command to set channel 1 to output 5V.

```
01 06 10 00 00 03
```

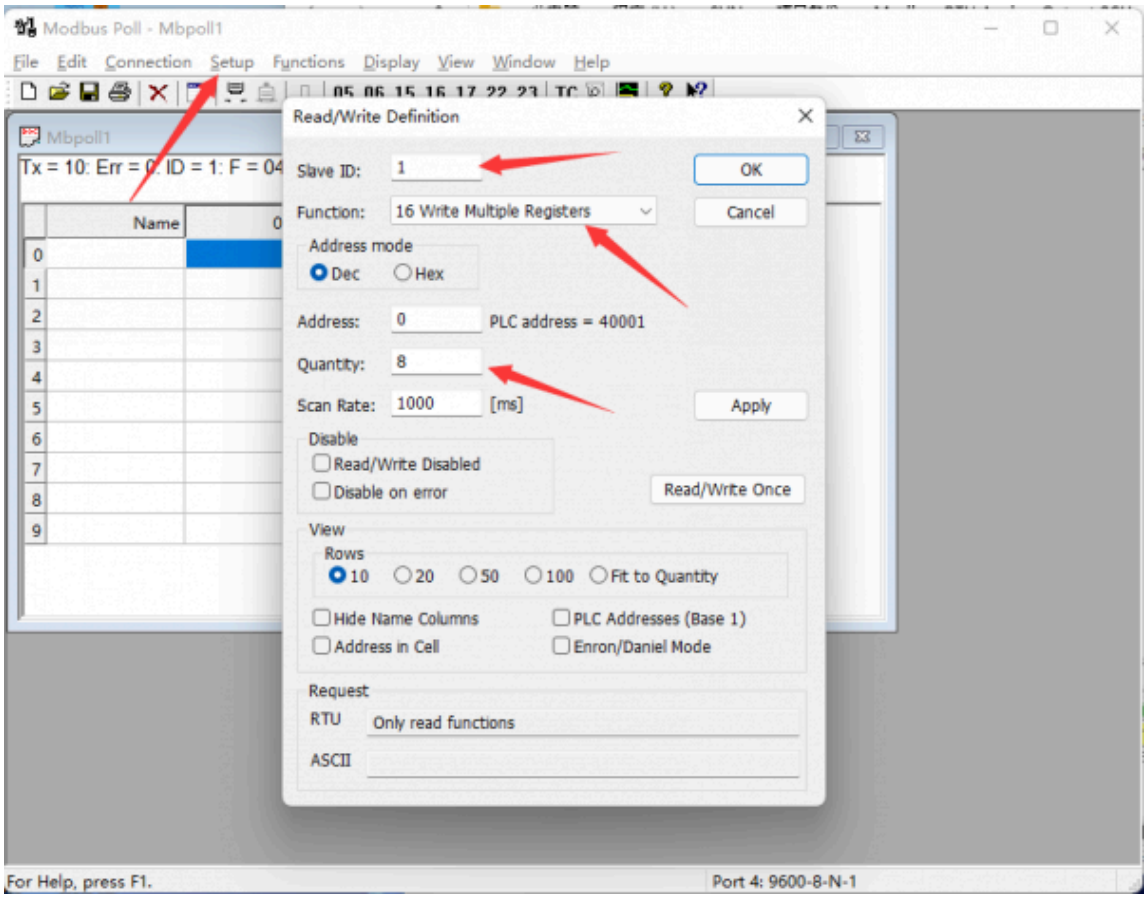


(/wiki/File:Modbus_RTU_Analog_Output_8CH-b-SSCOM02.png)

- For more details, you can see the development protocol.

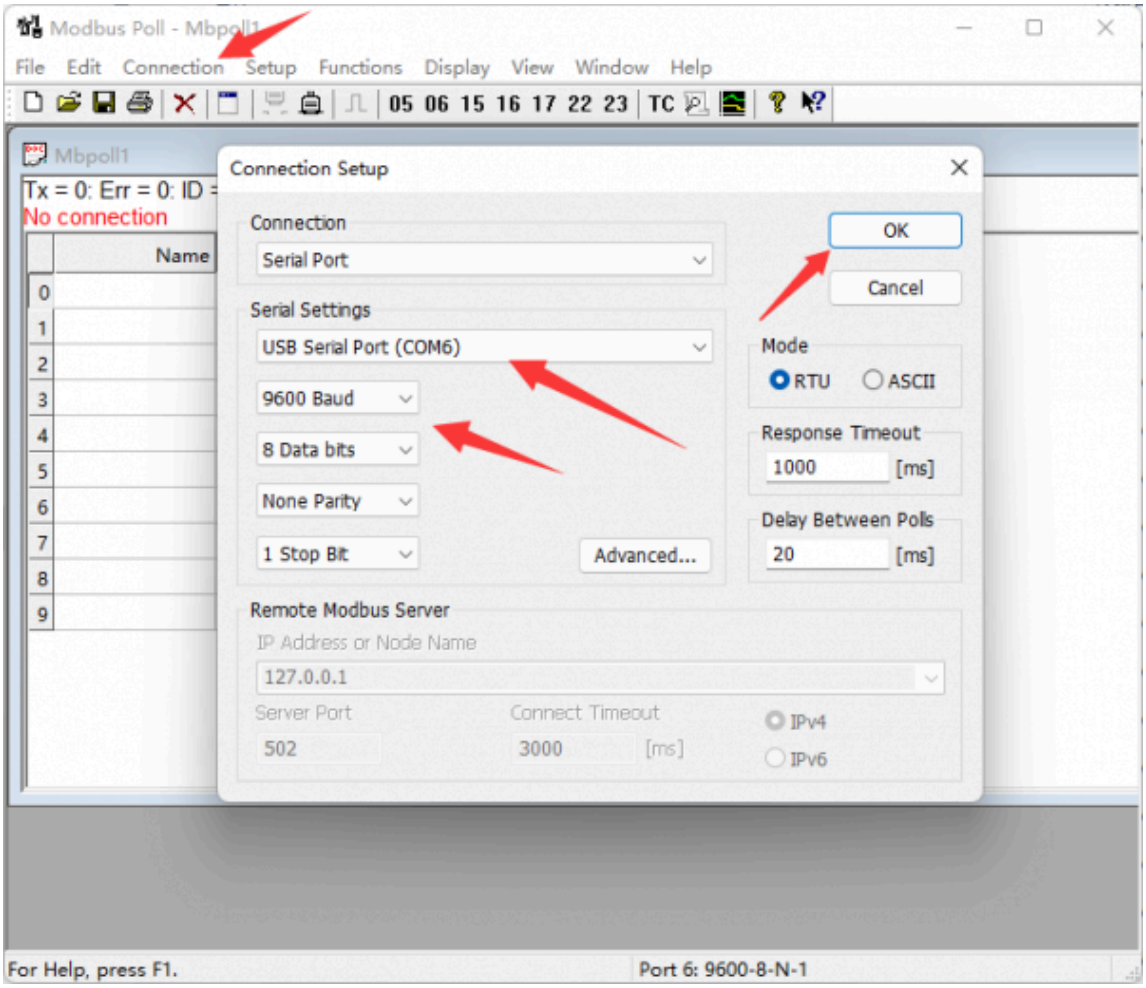
Modbus Poll Software

- It is not convenient to observe data through the SSCOM software. You can use Modbus Poll software to read data. Download and install Modbus Poll software (https://www.modbustools.com/download.html).
- Open the software, and select "Setup -> Read/Write Definition". Choose the actual device address for Slave ID, select function code 16, and change the Quantity to 8 channels. Click "OK" to confirm.



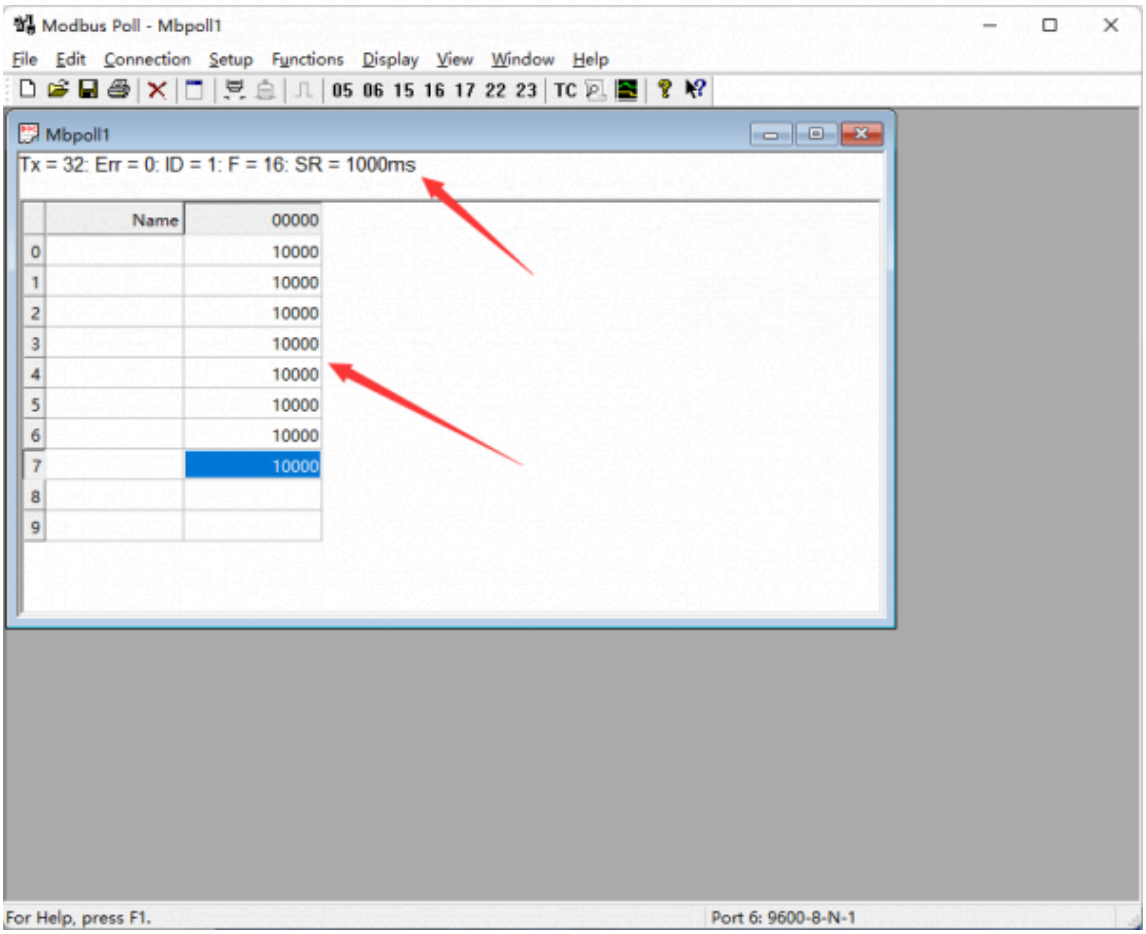
(/wiki/File:Modbus_RTU_Analog_Output_8CH-Poll01.png)

- Select Connection -> Connect..., and choose the corresponding serial ports, the baud rate as 9600, 8 Data bits, none parity mode. Click "OK" to connect.



(/wiki/File:Modbus_Poll-2.png)

- After the connection is normal, you can set 1-8 channels analog output voltage, unit in mV, with the range of 0-10000mV, or 0-10V.



(/wiki/File:Modbus_RTU_Analog_Output_8CH-Poll03.png)

Demo Test

Note: RS485 cannot be directly connected to the Raspberry Pi's serial port as it may damage the devices. It requires a 485 level converter. It's advisable to use the RS485 CAN HAT with the Raspberry Pi. For NUCLEO-F103RB and Arduino, it's recommended to use the RS485 CAN Shield module.

Raspberry Pi

Enter the Raspberry Pi terminal and input the following commands to enter the configuration interface:

```
sudo raspi-config
Select Interfacing Options -> Serial Port, then select Yes to open the hardware serial po
rt
```

And then reboot the Raspberry Pi:

```
sudo reboot
```

Insert the RS485 CAN HAT module into the Raspberry Pi, and connect the Modbus RTU Relay module to the RS485 CAN HAT module corresponding to the A and B connections.
If you are using other 485 devices, make sure the connection is A-A, B-B.
Run the following command to run the demo:

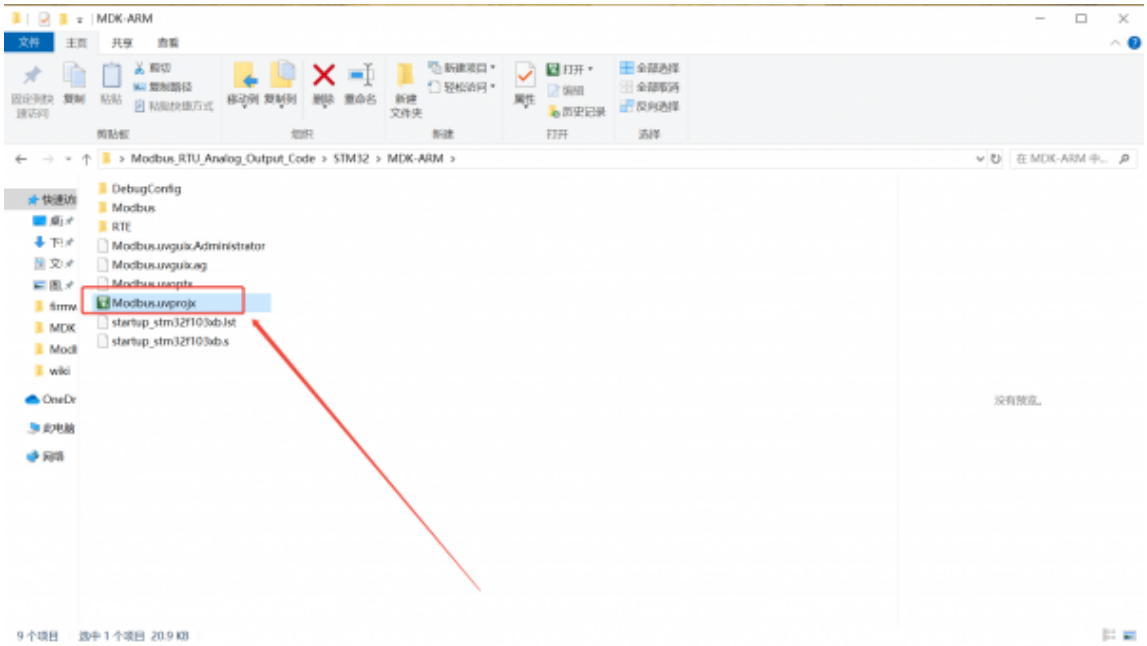
```
sudo apt-get install unzip
sudo apt-get install python3-pip
pip install modbus_tk
wget https://files.waveshare.com/wiki/Modbus-RTU-Analog-Output-8CH/Modbus_RTU_Analog_Outp
ut_Code.zip
unzip Modbus_RTU_Analog_Output_Code.zip
cd Modbus_RTU_Analog_Output_Code/Python3
sudo python3 modbus.py
```

All channels will output 5V after the demo runs normally.

STM32

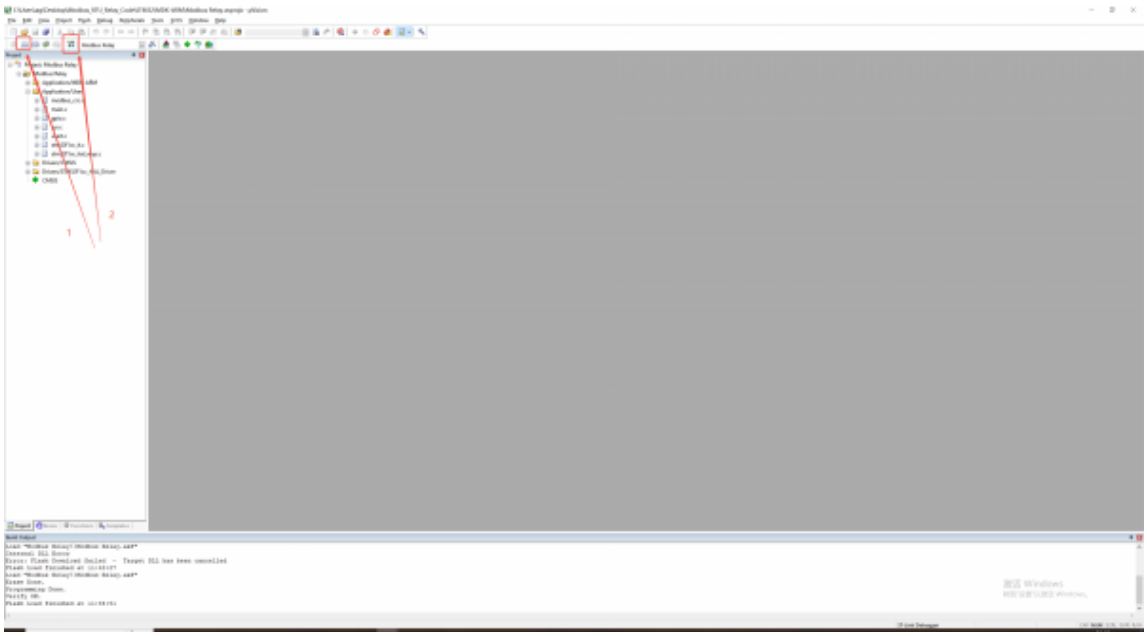
Note: The STM32 demo is based on the NUCLEO-F103RB and RS485 CAN Shield module.

1. Download Demo (https://files.waveshare.com/wiki/Modbus-RTU-Analog-Output-8CH/Modbus_RTU_Analog_Output_Code.zip), find the STM32 project file Modbus.uvprojx in the path Modbus_RTU_Analog_Output_Code\STM32\MDK-ARM, and double-click to open the STM32 project file. Note that you should ensure Keil5 software is installed on your computer before using it.



(/wiki/File:600px-Modbus-RTU-Relay-stm32-5.png)

2. Connect the STM32 to a computer via the STM32 download and debug probe. Compile and download the program to the development board.



(/wiki/File:600px-Modbus-RTU-Relay-stm32-2.png)

3. Install the RS485 CAN Shield module on the STM32. Connect the RS485_A on the RS485 CAN Shield module to the RS485_A on the Modbus RTU Analog Output 8CH (B) via a wire, and connect the RS485_B on the RS485 CAN Shield module to the RS485_B on the Modbus RTU Analog Output 8CH (B) via a wire. Then power on the Modbus RTU Analog Output 8CH (B) and the STM32 sequentially.

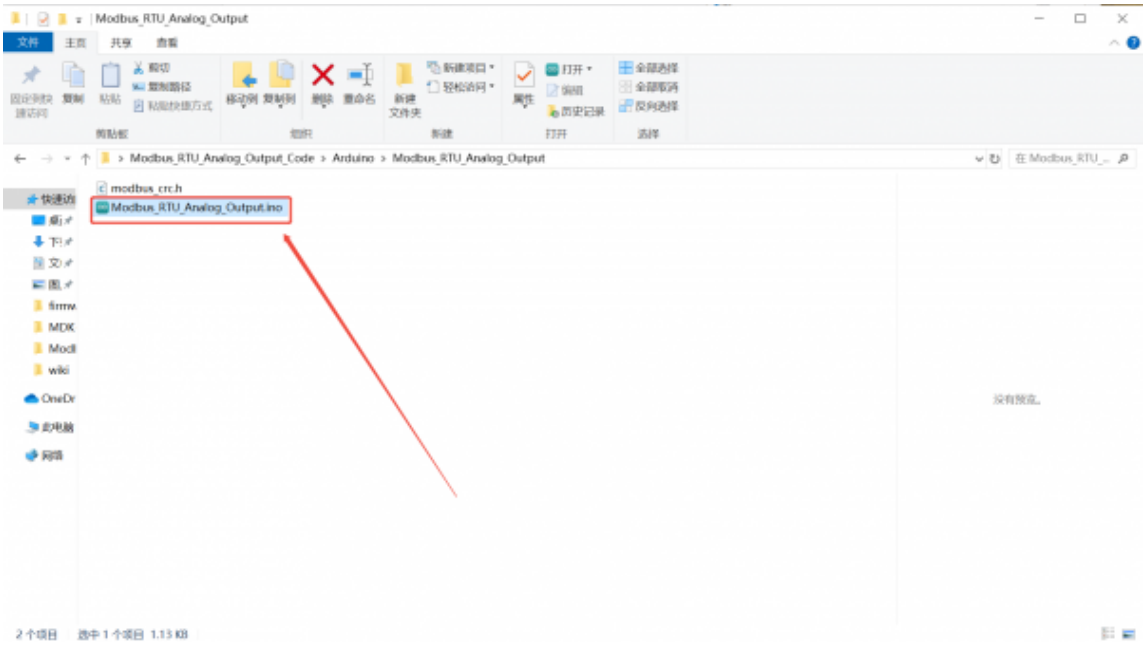
4. After the demo runs normally, all channels will output 5V voltage.

Arduino

Note: The Arduino demo is based on the UNO PLUS and RS485 CAN Shield module.

1. Download Demo (https://files.waveshare.com/wiki/Modbus-RTU-Analog-Output-8CH/Modbus_RTU_Analog_Output_Code.zip), find the Arduino project file Modbus_RTU_Analog_Output.ino in the path Modbus_RTU_Analog_Output_Code\Arduino\Modbus_RTU_Relay, and double-click to open the Arduino project file. Note that you should ensure Arduino IDE software is installed on

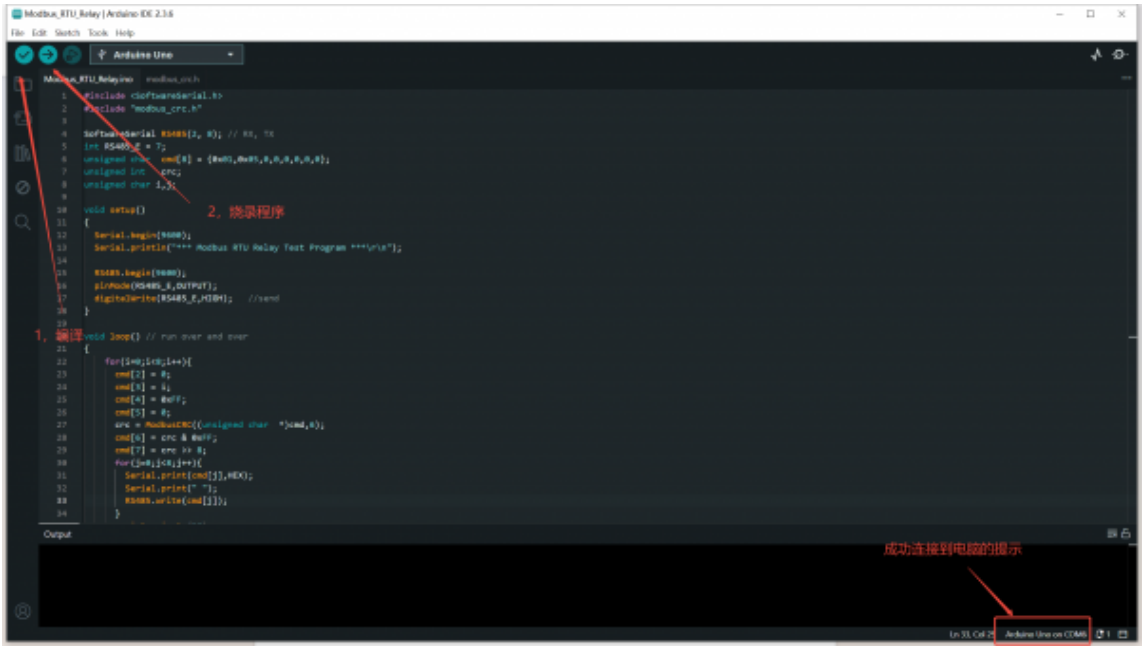
your computer before using it.



(/wiki/File:600px-Modbus-RTU-Relay-arduino-5.png)

2. Connect the Arduino to the computer via a USB cable. In the Arduino IDE software, select the Arduino board model under Tools->Board. Choose the COM port that the Arduino is connected to under Tools->Port.

3. After seeing the prompt to connect to the computer in the lower right corner, click to compile and flash the program, and wait for the flashing to complete.



(/wiki/File:600px-Modbus-RTU-Relay-arduino-2.png)

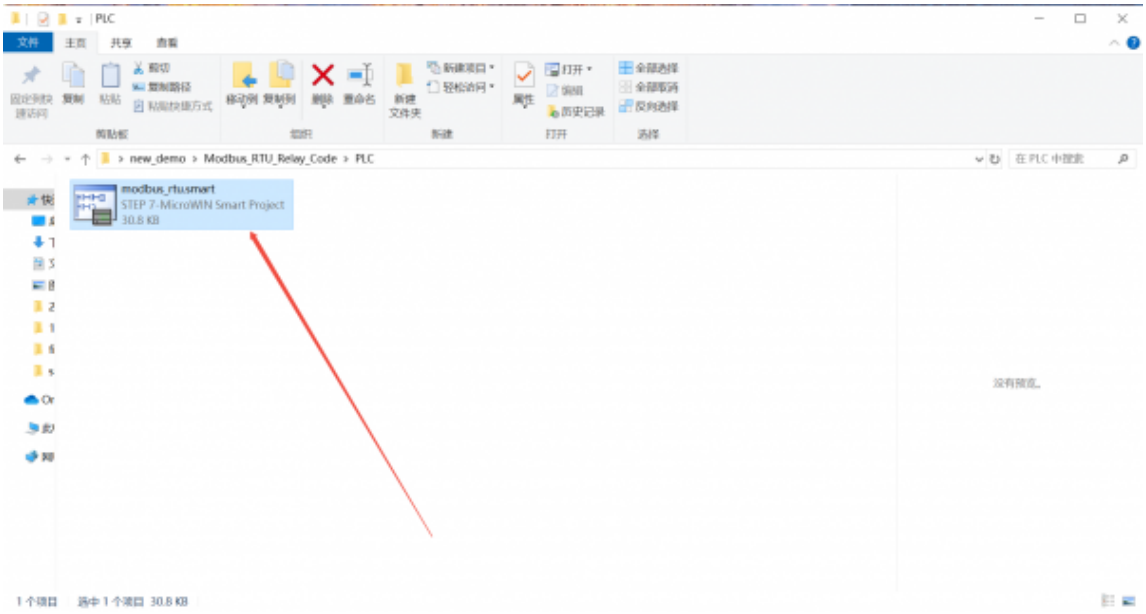
4. Install the RS485 CAN Shield module on the Arduino. Connect the RS485_A on the RS485 CAN Shield module to the RS485_A on the Modbus RTU Analog Output 8CH (B) via a wire, and connect the RS485_B on the RS485 CAN Shield module to the RS485_B on the Modbus RTU Analog Output 8CH (B) via a wire. Then power on the Modbus RTU Analog Output 8CH (B) and the Arduino sequentially.

5. After the demo runs normally, all channels will output 5V voltage.

PLC

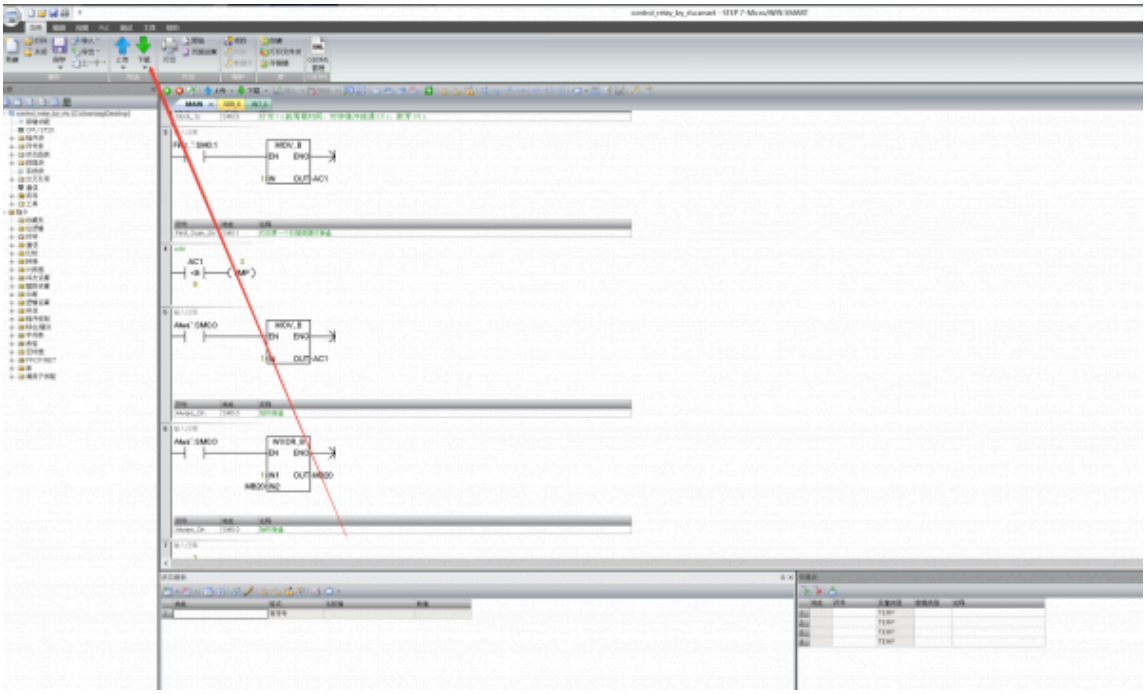
Note: The PLC demo is based on SIMATIC S7-200 SMART.

1. Download Demo (https://files.waveshare.com/wiki/Modbus-RTU-Analog-Output-8CH/Modbus_RTU_Analog_Output_Code.zip), find the PLC project file modbus_rtu.smart in the path Modbus_RTU_Analog_Output_Code\PLC, and double-click to open the PLC project file. Note that you should ensure STEP 7-MicroWIN SMART software is installed on your computer before using it.



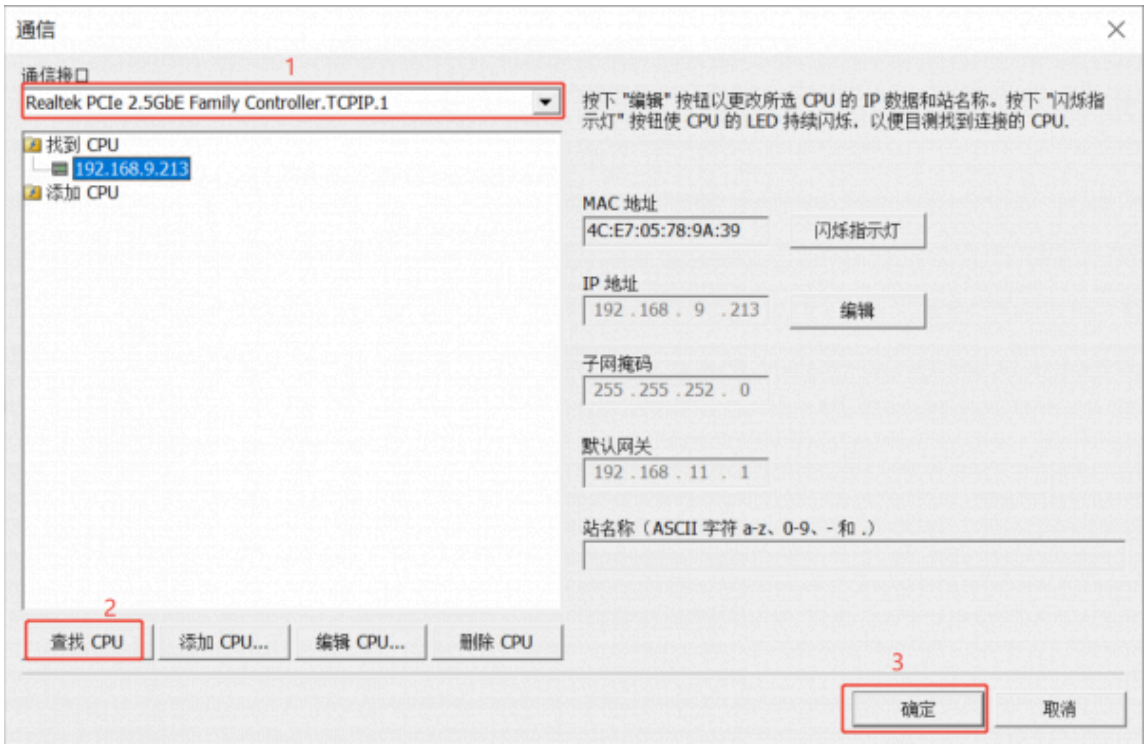
(/wiki/File:600px-Modbus-RTU-Relay-plc-1.png)

2. Connect the PLC to the computer via a network cable. Click to download



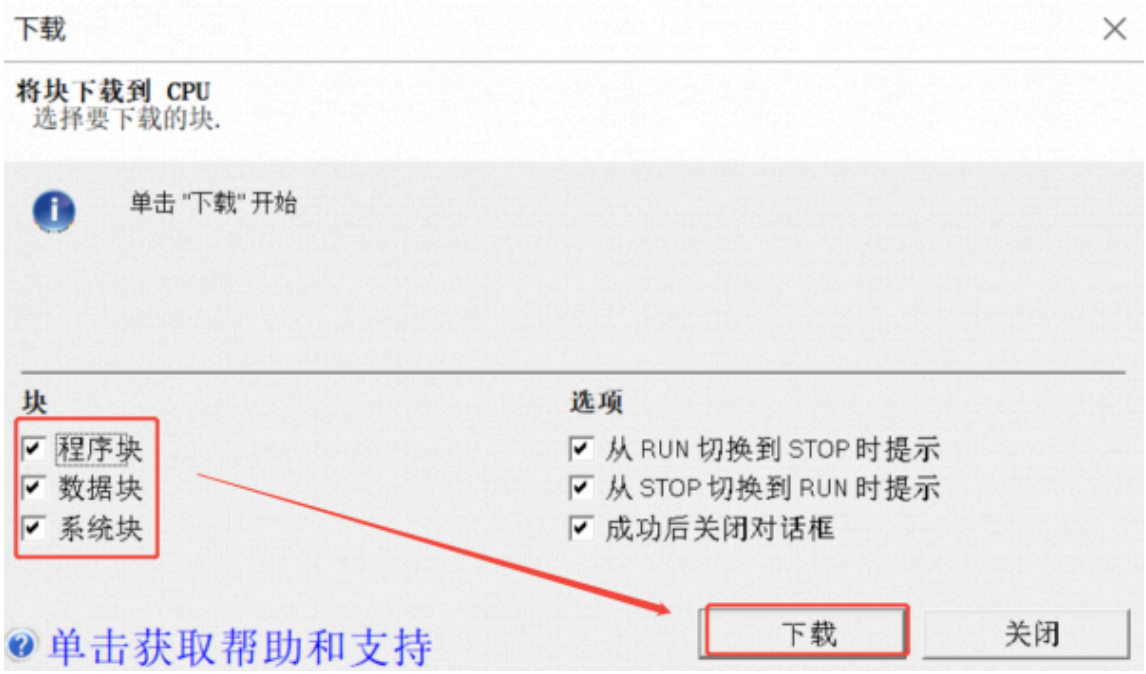
(/wiki/File:600px-Modbus-RTU-Relay-plc-2.png)

3. Select the communication interface in the communication popup, find your device, and click Confirm



(/wiki/File:600px-Modbus-RTU-Relay-plc-3.png)

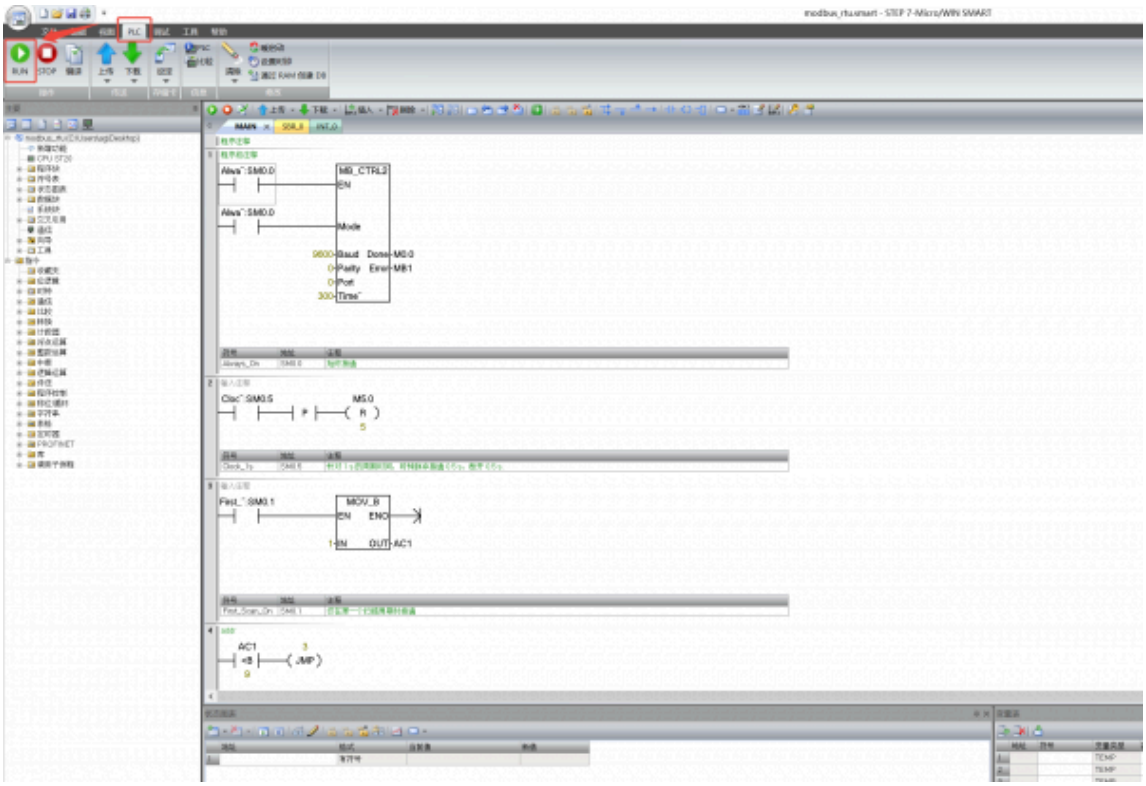
4. In the download pop-up window, check the boxes for program blocks, data blocks, and system blocks, and then click Download.



(/wiki/File:600px-Modbus-RTU-Relay-plc-4.png)

5. Connect the 485_A on the PLC module to RS485_A on Modbus_RTU_Analog_Output with a wire, and connect the 485_B on the PLC module to RS485_B on Modbus_RTU_Analog_Output with a wire. Then power on the Modbus_RTU_Analog_Output.

6. After powering on the Modbus_RTU_Analog_Output, in the STEP 7-MicroWIN SMART software, find the "PLC" tab in the upper menu bar, switch to this tab, and click the green "RUN" button in the lower "Operation" area to perform the operation to make the PLC enter the running state



(/wiki/File:600px-Modbus-RTU-Relay-plc-5.png)

7. After the demo runs normally, all channels will output 5mA current.

Development Protocol V2

Function Code Introduction

Function Code	Description
03	Read holding register
06	Write single holding register
10	Write multiple holding register

Register Address Introduction

Address (HEX)	Address storage content	Register value	Permission	Modbus Function Code
4x0000 ..... 4x0007	Channels 1~8 output data	The values are unsigned hexadecimal	Read/write	0x03,0x06,0x10
4x2000	UART Parameter	The high eight bits indicate the parity mode: 0x00~0x02 The low eight bits indicate the baud rate mode: 0x00~0x07	Read/Write	0x03, 0x06
4x4000	Device Address	Directly store Modbus address Device address: 0x0001-0x00FF	Read/Write	0x03, 0x06

4x8000	Software Version	Converting to decimal and then shifting the decimal point two places to the left will represent the software version 0x0064 = 100 = V1.00	Read	0x03

Operation Command Introduction

Set Single-channel Output Command

Send code: 01 06 00 00 03 E8 89 74

Field	Description	Note
01	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
06	06 Command	Write single register
00 00	Register Address	0x0000 - 0x0007 correspond to output data for channels 1~8
03 E8	Set Values	Set the value to hexadecimal in mV, with the high bits in front and the low bits in the back 0x03E8 = 1000mV, i.e., 1V
89 74	CRC16	The CRC16 checksum of the first 6 bytes of data

Return code: 01 06 00 00 03 E8 89 74

Field	Description	Note
01	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
06	06 Command	Write single register
00 00	Register Address	0x0000 - 0x0007 correspond to data types of 1~8 output channels
03 E8	Set Values	Set the value to hexadecimal in mV, with the high bits in front and the low bits in the back 0x03E8 = 1000mV, i.e., 1V
89 74	CRC16	The CRC16 checksum of the first 6 bytes of data

For example: [Address 1 device]

```
Set channel 1 output 1V : 01 06 00 00 03 E8 89 74
Set channel 2 output 5V : 01 06 00 01 13 88 D5 5C
```

Set Multi-channel Output Command

Send code: 01 10 00 00 00 08 10 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 3C 05

Field	Description	Note
01	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
10	10 Command	Write multiple registers
00 00	Register Start Address	0x0000 - 0x0007 correspond to 1~8 output channels
00 08	Register Number	Set register number, which must not exceed the maximum number of the channels
10	Byte Number	Set the number of bytes to be output
03 E8 ..... 03 E8	Command	Start channel analog output ..... Final channel analog output Set the value to hexadecimal in mV, with the high bits in front and the low bits in the back 0x03E8 = 1000mV, i.e., 1V
3C 05	CRC16	The CRC16 checksum of the first 6 bytes of data

Return code: 01 10 00 00 00 08 C1 CF

Field	Description	Note
01	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
10	10 Command	Write multiple registers
00 00	Register Start Address	0x0000 - 0x0007 correspond to 1~8 output channels
00 08	Register Number	Set register number, which must not exceed the maximum number of the channels
C1 CF	CRC16	The CRC16 checksum of the first 6 bytes of data

For example: [Address 1 device]

```
Set channels 1-8 output 1V: 01 10 00 00 00 08 10 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E
8 03 E8 3C 05
Set channels 3-5 output 2V: 01 10 00 02 00 03 06 07 D0 07 D0 07 D0 84 0E
```

Read Channel Output Command

Send code: 01 03 00 00 00 08 44 0C

Field	Description	Note
01	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
03	03 Command	Read holding register
00 00	Register Start Address	0x0000 - 0x0007 correspond to 1~8 input channels
00 08	Register Number	The number of the registers to be read, which must not exceed the maximum number of the channels
44 0C	CRC16	The CRC16 checksum of the first 6 bytes of data

Return code: 01 03 10 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 C1 91

Field	Description	Note
01	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
03	03 Command	Read holding register
10	Byte Number	The number of all bytes of the returned status information
03 E8 ..... 03 E8	Output value	Start channel analog output ..... Final channel analog output Set the value to hexadecimal in mV, with the high bits in front and the low bits in the back 0x03E8 = 1000mV, i.e., 1V
C1 91	CRC16	The CRC16 checksum of the first 6 bytes of data

For example: [Address 1 device]

```
Read data types for channels 1-8 : 01 03 00 00 00 08 44 0C
Read data type for channel 1 : 01 03 00 00 00 01 84 0A
Read data type for channel 2 : 01 03 00 01 00 01 D5 CA
Read data type for channels 3-5 : 01 03 00 02 00 03 A4 0B
```

Set Baudrate Command

Send code: 00 06 20 00 00 05 43 D8

Field	Description	Note
00	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
06	06 command	Set the baud rate and device address
20 00	Command Register	0x2000: set the baud rate; 0x4000: set the device address
00	Parity Method	0x00: no parity, 0x01: even parity; 0x02: odd parity
05	Baud Rate Value	Correspondence of baud rate values 0x00: 4800 0x01: 9600 0x02: 19200 0x03: 38400 0x04: 57600 0x05: 115200 0x06: 128000 0x07: 256000
43 D8	CRC16	The CRC16 checksum of the first 6 bytes of data

Return code: 00 06 20 00 00 05 43 D8

Field	Description	Note
00	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
06	06 command	Set the baud rate and device address
20 00	Command Register	0x2000: set the baud rate; 0x4000: set the device address
00	Parity Method	0x00: no parity, 0x01: odd parity; 0x02: even parity
05	Baud Rate	Correspondence of baud rate values 0x00: 4800 0x01: 9600 0x02: 19200 0x03: 38400 0x04: 57600 0x05: 115200 0x06: 128000 0x07: 256000
43 D8	CRC16	The CRC16 checksum of the first 6 bytes of data

For example: [Address 1 device]

```
Set the baud rate as 4800: 00 06 20 00 00 00 83 DB
Set the baud rate as 9600: 00 06 20 00 00 01 42 1B
Set the baud rate as 115200: 00 06 20 00 00 05 43 D8
```

Set Device Address Command

Send code: 00 06 40 00 00 01 5C 1B

Field	Description	Note
00	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
06	06 command	Set the baud rate and device address
40 00	Command Register	0x2000: set the baud rate; 0x4000: set the device address
00 01	Device Address	Set the device address, 0x0001-0x00FF
5C 1B	CRC16	The CRC16 checksum of the first 6 bytes of data

Return code: 00 06 40 00 00 01 5C 1B

Field	Description	Note
00	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
06	06 command	Set the baud rate and device address
40 00	Command Register	0x2000: set the baud rate; 0x4000: set the device address
00 01	Device Address	Set the device address, 0x0001-0x00FF
5C 1B	CRC16	The CRC16 checksum of the first 6 bytes of data

For example: [Address 1 device]

```
Set the device address as 0x01: 00 06 40 00 00 01 5C 1B
Set the device address as 0x02: 00 06 40 00 00 02 1C 1A
Set the device address as 0x03: 00 06 40 00 00 03 DD DA
```

Read Device Address Command

Send code: 00 03 40 00 00 01 90 1B

Field	Description	Note
00	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
03	03 Command	Read the device address
40 00	Command register	0x4000: read the device address, 0x8000: read software version
00 01	Byte Number	Fixed 0x0001
90 1B	CRC16	The CRC16 checksum of the first 6 bytes of data

Return code: 01 03 02 00 01 79 84

Field	Description	Note
01	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
03	03 Command	Read the software version and device address
02	Byte Number	The number of bytes returned
00 01	Device Address	Set the device address, 0x0001-0x00FF
79 84	CRC16	The CRC16 checksum of the first 6 bytes of data

For example: [Address 2 device]

```
Send: 00 03 40 00 00 01 90 1B
Return: 02 03 02 00 02 7D 85     // Address 0x02
```

Read Software Version Command

Send code: 00 03 80 00 00 01 AC 1B

Field	Description	Note
00	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
03	03 Command	Read the software version and device address
80 00	Command register	0x4000: read the device address, 0x8000: read software version
00 01	Byte Number	Fixed 0x0001
AC 1B	CRC16	The CRC16 checksum of the first 6 bytes of data

Return code: 01 03 02 00 64 B9 AF

Field	Description	Note
01	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
03	03 Command	Read the software version and device address
02	Byte Number	The number of bytes returned
00 64	Software Version	Converting to decimal and then shifting the decimal point two places to the left will represent the software version 0x0064 = 100 = V1.00
B9 AF	CRC16	The CRC16 checksum of the first 6 bytes of data

For example: [Address 1 device]

```
Send: 00 03 80 00 00 01 AC 1B
Return: 01 03 02 00 64 B9 AF       //0x0064 = 100 =V1.00
```

Exception Function Code

When the received command is incorrect or the device is abnormal, an exception response will be returned in the following format:

Return: 01 85 03 02 91

Field	Description	Note
01	Device Address	0x00 indicates the broadcast address, 0x01-0xFF indicates the device address
85	Exception Function Code	Exception function code = Request function code + 0x80
03	Byte Number	Exception Code
02 91	CRC16	The CRC16 checksum of the first 6 bytes of data

An exception code is a single-byte value that indicates the type of error. Several commonly used exception codes defined by the Modbus protocol:

Exception Code	Name	Description
0x01	Illegal Function	The requested function code is not supported
0x02	Illegal Data Address	The requested data address is incorrect
0x03	Illegal Data Value	The requested data value or operation cannot be executed
0x04	Server Failure	Server equipment failure
0x05	Response	The request has been received and is being processed
0x06	Device Busy	The device is currently busy and cannot perform the requested operation

Resources

Demos

- Demo (https://files.waveshare.com/wiki/Modbus-RTU-Analog-Output-8CH/Modbus_RTU_Analog_Output_Code.zip)

Software

- Sscom serial port debugging assistant (https://files.waveshare.com/wiki/common/Sscom5.13.1.zip)
- Modbus Poll software (https://www.modbustools.com/download.html)

- SecureCRT software (https://files.waveshare.com/wiki/common/SecureCRT.7z)

Related Resources

- Modbus Protocol Specification (https://www.waveshare.com/wiki/Modbus_Protocol_Specification)
- Modbus Series BootLoader Description (https://www.waveshare.com/wiki/Modbus_Series_BootLoader_Description)

FAQ

Question: What is the maximum output current of a single channel of the device in mA?

Answer:

Maximum output current per channel: 100mA; Actual output current: varies with the device's current load and voltage

Question: Where can I download the driver for my device?

Answer:

The device does not require an additional driver download. You can control the device relay's on/off or query the relay status by sending Modbus commands through the device's 485 interface.

Question: Can the slave address be set by a DIP switch?

Answer:

There is no DIP switch, it can only be set through commands

Question: What should I do if I forget the communication parameters of the 485?

Answer:

Please refer to Modbus Series BootLoader Description (https://www.waveshare.com/wiki/Modbus_Series_BootLoader_Description) to enter the boot mode, then you can select option 3 to read communication parameters or select option 2 to reset communication parameters

Question: What should I do if SecureCRT is garbled when entering boot mode?

Answer:

Check if the 485 bus is reversed. If not, connect a 120 Ω terminal resistor across the A and B cables

Question: What is the power consumption of the device?

Answer:

Under 5V power supply, the reference power consumption when in standby mode: 0.39W. Please refer to actual conditions for specific power consumption

Question: Why does the RX indicator blink after sending a command to the device, but there is no output, while there is an output when controlling via DI input?

Answer:

The device is in linkage mode at this time, just switch to other modes

Question: What is the Load Drive output Current value on each Analog output channels of Modbus_RTU_Analog_Output_8CH_(B) modules?

Answer:

The Modbus RTU Analog Output 8CH (B) can output currents from 0-100mA, depending on the load and voltage.

Question: I sent a command to control the relay, but it didn’t respond. What could be the issue??

Answer:

The command must be sent in hex format with a CRC checksum.

1. If the module does not respond to the command, verify that the baud rate and device ID are correct. You may also try Restoring the factory settings (https://www.waveshare.com/wiki/Template:Modbus_Series_Products:_Factory_Reset_and_Firmware_Upgrade_Guide).
2. If the above steps do not resolve the issue, please submit a ticket (https://service.waveshare.com/) to contact the Waveshare technical support team.

Support

Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.
Working Time: 9 AM - 6 PM GMT+8 (Monday to Friday)

Submit Now (https://service.waveshare.com/)

Retrieved from "https://www.waveshare.com/w/index.php?title=Modbus_RTU_Analog_Output_8CH_(B)&oldid=107691 (https://www.waveshare.com/w/index.php?title=Modbus_RTU_Analog_Output_8CH_(B)&oldid=107691)"