OWH80 Series Programmable Power Supply Communication Protocol Manual

For product support, visit:www.owon.com.hk/download

※：The illustrations, interface, icons and characters in the user manual may be slightly different from the actual product. Please refer to the actual product.

Jan. 2026 edition V1.0.0

Fujian LILLIPUT Optoelectronics Technology Co., Ltd.
No. 19, Heming Road
Lantian Industrial Zone, Zhangzhou 363005 P.R. China

Tel: +86-596-2130430 **Fax:** +86-596-2109272

Web: www.owon.com.cn **E-mail:** info@owon.com.cn

Statement

Version Revision History

Date	Version	Revised Chapters
2026.01	1.0.0	Complete this manual

Table of Contents

1. Overview

To meet users' needs for developing upper computers using configuration software or other software packages, our company has developed standard Modbus communication protocol based on the original design. Therefore, the instrument supports both SCPI and Modbus communication protocols. Users should select the appropriate protocol on the display panel before use.

This manual describes the Modbus communication protocol for OWH80.

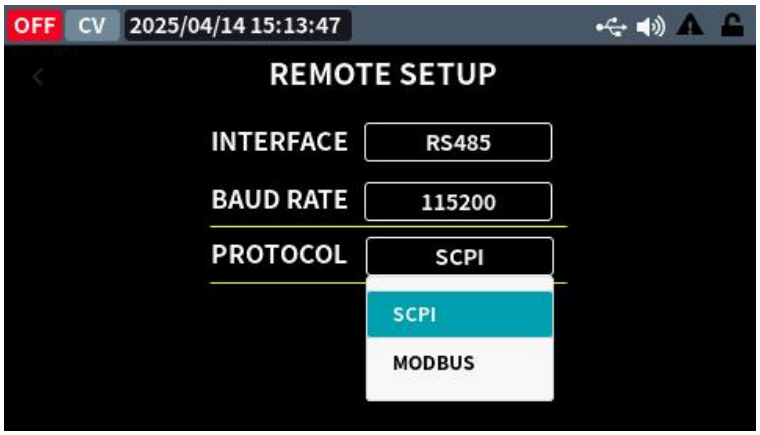
1.1 Configure communication baud rate

Instrument supports communication interfaces such as RS232 and RS485.

Interface	Description
RS232	Standard Serial Port
RS485	Standard Serial Port

The configuration method for communication parameters is as follows:

1. On the front panel, press **config** key, use the direction keys to move the cursor to "REMOTE SETTING" and press the **Enter** key to access the remote settings menu.



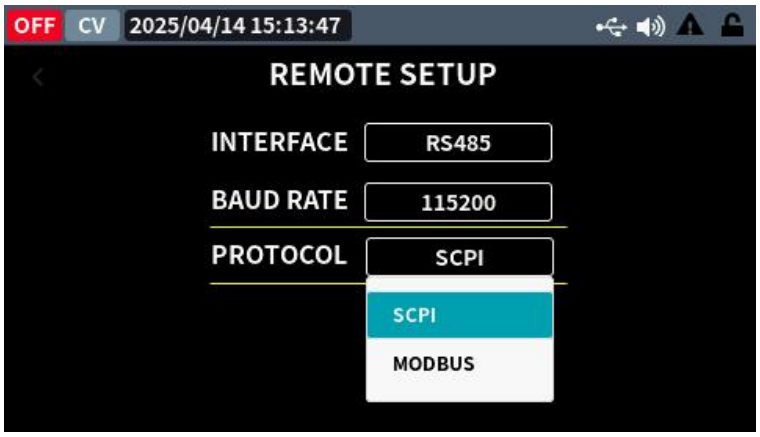
2. Use the direction keys to move the cursor to the "Baud Rate" option, and press the **Enter** key to enter edit mode.
3. Use the direction keys to select relevant parameters (Baud Rate: 2400 - 115200; USB, RS232, and RS485 share the same baud rate).

4. Press the **Enter** key to activate the communication parameters.

1.2 Select communication protocol

The digital power meter supports both SCPI and Modbus communication protocols. The system defaults to SCPI protocol. If you need to switch to Modbus protocol, please follow the steps below:

1. On the front panel, press **config** key, use the direction keys to move the cursor to "REMOTE SETTING" and press the **Enter** key to access the remote settings interface.



2. Use the direction keys to move the cursor to the "PROTOCOL" option, and press the **Enter** key to enter edit mode.
3. Use the direction keys to select "MODBUS".
4. Press the **Enter** key to confirm it.

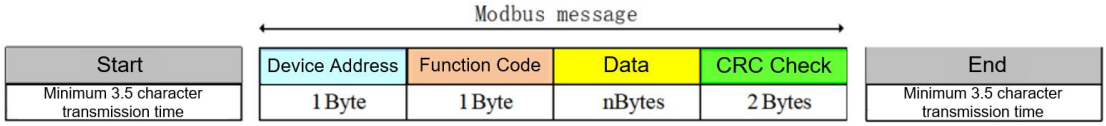
2. Modbus Introduction

The instrument supports standard Modbus RTU protocol.

2.1 Frame format

2.1.1 Modbus RTU

For serial communication (e.g., RS232), when using RTU mode to transmit Modbus messages, the format is shown in the following figure.



- **Device Address:** Modbus Communication Address, Range 1～9.
- **Function Code:** Requests a specific operation to be performed, such as reading registers, writing registers, reading block data, etc.
- **Data:** Describes specific operation information, such as the starting address of reading registers and the number of registers to read.
- **CRC Checksum:** Calculate the CRC checksum result for "Device Address", "Function Code", and "Data".

2.2 Function Code

The Modbus protocol only uses partial function codes from the 《Standard Modbus Protocol》, as detailed in the following table.

Function Code	Description	Note
0x03	Read Retention Register	Read in 16-bit bytes
0x04	Read Input Register	Read in 16-bit bytes
0x06	Write a holding register	Write in 16-bit bytes
0x10	Write multiple holding registers	Write in 16-bit bytes

2.2.1 03 (0x03) Read Register

This function code is used to read the contents of multiple consecutive registers.

Request

Function Code	1 Byte	0x03
Initial Address	2 Bytes	0x0000~0xFFFF
Register Count	2 Bytes	

Response

Function Code	1 Byte	0x03
Number of bytes returned	1 Byte	2xN
Register Value	N x 2 Bytes	value

Error

Error Code	1 Byte	0x83
Exception Code	1 Byte	01/02/03/04

Exception Code Description

Error Code	0x01	0x02	0x03	0x04
Description	Unsupported Function Code	Register address out of range	The number of registers exceeds the range	Read failure

For example

Request: 01 03 03 E8 00 02 44 7B

Response: 01 03 04 00 01 00 00 AB F3

Request		Response	
Filed	Hex	Field	Hex
Device Address	01	Device Address	01
Function Code	03	Function Code	03
Register Address(High Byte)	03	Byte count	04
Register Address (Low Byte)	E8	Register Value (High Byte)	00

Number of Registers(High Byte)	00	Register Value	01
Number of Registers (Low Byte)	02	Register Value	00
CRC Checksum (Low Byte)	44	Register Value (Low Byte)	00
CRC Checksum (High Byte)	7B	CRC Checksum (Low Byte)	AB
		CRC Checksum (High Byte)	F3

2.2.2 06(0x06) Write to a single register

This function code is used to write a single register.

Request

Function Code	1 Byte	0x06
Initial Address	2 Bytes	0x0000~0xFFFF
Register Value	2 Bytes	value

Response

Function Code	1 Byte	0x10
Register Address	2 Bytes	0x0000~0xFFFF
Register Value	2 Bytes	value

Error

Error Code	1 Byte	0x86
Exception Code	1 Byte	01/02/03/04

Exception Code Description

Error Code	0x01	0x02	0x03	0x04
Description	Unsupported Function Code	Register Address	The number of registers exceeds the range, or the number of bytes is incorrect, or the register value is incorrect.	Write failure

For example

Request: 01 06 03 E8 00 01 C8 7A

Response: 01 06 03 E8 00 01 C8 7A

Request		Response	
Filed	Hex	Field	Hex
Device Address	01	Device Address	01
Function Code	06	Function Code	06
Register Address(High Byte)	03	Register Address (High Byte)	03
Register Address(Low Byte)	E8	Register Address (Low Byte)	E8
Number of Registers (High Byte)	00	Number of Registers (High Byte)	00
Number of Registers (Low Byte)	01	Number of Registers (Low Byte)	01
CRC Checksum (Low Byte)	C8	CRC Checksum (Low Byte)	CB
CRC Checksum (High Byte)	7A	CRC Checksum (High Byte)	7A

2.2.3 16(0x10) Write Register

This function code is used to write multiple consecutive registers.

Request

Function Code	1 Byte	0x10
Initial Address	2 Bytes	0x0000~0xFFFF
Register Count	2 Bytes	2
Byte count	1 Byte	4
Register Value	4 Bytes	value

Response

Function Code	1 Byte	0x10
Initial Address	2 Bytes	0x0000~0xFFFF
Register Value	2 Bytes	2

Error

Error Code	1 Byte	0x90
Exception Code	1 Byte	01/02/03/04

Exception Code Description

Error Code	0x01	0x02	0x03	0x04
Description	Unsupported Function Code	Register Address	The number of registers exceeds the range, or the number of bytes is incorrect, or the register value is incorrect.	Write failure

For example

Request: 01 10 03 F4 00 02 04 00 00 44 7A 5B 0B

Response: 01 10 03 F4 00 02 00 7E

Request		Response	
Filed	Hex	Field	Hex
Device Address	01	Device Address	01
Function Code	10	Function Code	10
Register Address(High Byte)	03	Register Address (High Byte)	03
Register Address (Low Byte)	F4	Register Address (Low Byte)	F4
Number of Registers (High Byte)	00	Number of Registers (High Byte)	00
Number of Registers (Low Byte)	02	Number of Registers (Low Byte)	02
Byte count	04	CRC Checksum (Low Byte)	00
Register Value (High Byte)	00	CRC Checksum (High Byte)	7E
Register Value	00		
Register Value	44		
Register Value (Low Byte)	7A		
CRC Checksum (Low Byte)	5B		
CRC Checksum (High Byte)	0B		

2.2.4 04(0x04) Read Input Register

This function code is used to read the contents of multiple consecutive holding registers.

Request

Function Code	1 Byte	0x04
Initial Address	2 Bytes	0x0000~0xFFFF
Register Count	2 Bytes	2

Response

Function Code	1 Byte	0x04
Initial Address	1 Byte	2xN
Register Value	N x 2 Bytes	value

Error

Error Code	1 Byte	0x83
Exception Code	1 Byte	01/02/03/04

Exception Code Description

Error Code	0x01	0x02	0x03	0x04
Description	Unsupported Function Code	Register Address	The number of registers exceeds the range.	Read failure

For example

Request: 01 04 00 96 00 02 91 E7

Response: 01 04 04 AA C6 3D 1B 6A FA

Request		Response	
Filed	Hex	Field	Hex
Device Address	01	Device Address	01
Function Code	04	Function Code	04
Register Address (High Byte)	00	Byte count	04
Register Address (Low Byte)	96	Register Value (High Byte)	AA
Number of Registers(High Byte)	00	Register Value	C6
Number of Registers (Low Byte)	02	Register Value	3D
CRC Checksum (Low Byte)	91	Register Value (Low Byte)	1B
CRC Checksum (High Byte)	E7	CRC Checksum (Low Byte)	6A
		CRC Checksum (High Byte)	FA

3. Address List

3.1 Register Address

The instrument related registers are shown in below table.

Table 3-1 Holding Registers

Add.	Function	Data Type	Add. Incr.
1000	Output control (ON/OFF)	uint16_t	1
1002	Output voltage setting	float	2
1004	Output current setting	float	2
1006	Voltage slope	float	2
1008	Current slope	float	2
1021	CC/CV mode	uint16_t	1
1033	Output mode	uint16_t	1
1204	OVP	float	2
1206	OCP	float	2
1208	OPP	float	2

Table 3-2 Input Registers

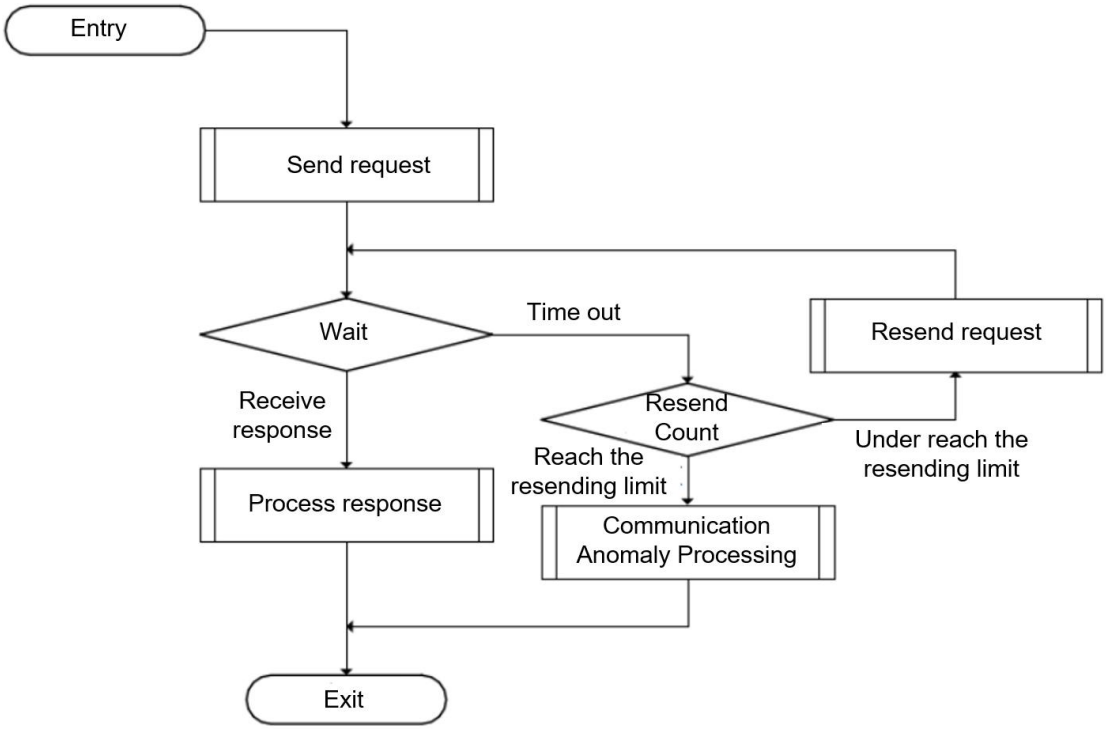
Add.	Function	Data Type	Add. Incr.
150	Output voltage (local machine)	float	2
152	Output current (local machine)	float	2
154	Output power (local machine)	float	2
251	System run icon	uint16_t	1
254	Internal fault code 1	uint16_t	1

255	Internal fault code 2	uint16_t	1
256	Output fault code	uint16_t	1
261	System run icon	uint16_t	1
264	Internal fault code 1 (latch)	uint16_t	1
265	Internal fault code 2 (latch)	uint16_t	1
266	Output fault code (latch)	uint16_t	1

4. Programming Reference

4.1 Communication Model

The timeout period is recommended to be set between 50～500ms, depending on the length of the Response frame. The longer the Response frame, the longer the timeout period should be. The number of retries is recommended to be set between 3～5 times.



4.2 Number Conversion

Modbus data transmission uses big-endian format. If the control device (PC, PLC, configuration software, user APP, etc.) uses little-endian format, then the data needs to be converted between big-endian and little-endian formats before use.

```
typedef union
{
    int reg_int;                    //32-bit signed integer
```

```
    float     reg_float;              //Single-precision floating-point
    UCHAR     reg_bytes[4];     //byte sequence
}unReg32Type;
UCHAR buff_modbus[256];       //Modbus Data Buffer
unReg32Type    data_reg;         //Composite Register Data
```

On little-endian devices

```
data_reg.reg_bytes[3] = buff_modbus[0];
data_reg.reg_bytes[2] = buff_modbus[1];
data_reg.reg_bytes[1] = buff_modbus[2];
data_reg.reg_bytes[0] = buff_modbus[3];
```

On big-endian devices

```
data_reg.reg_bytes[0] = buff_modbus[0];
data_reg.reg_bytes[1] = buff_modbus[1];
data_reg.reg_bytes[2] = buff_modbus[2];
data_reg.reg_bytes[3] = buff_modbus[3];
```

The above code is for reference only. Users should properly handle data according to the actual situation of the controlled device in communication applications.

4.3 Touchscreen Communication Configuration

Device Property Name	Device attribute value
Collection Optimization	1 ~ Optimization
Device name	Device 0
Device notes	ModbusRTU_serial port
Initial working status	1 ~ start
Minimum acquisition period(ms)	100
Device address	1
Response time	200
16-bit integer byte order	0 ~ 22

32-bit integer byte order	2 ~ 3412
32-bit floating point byte order	2 ~ 3412
Byte Order	0 ~ 21
Character Encoding Format	0 ~ ASCII
Verify Data Byte Order	0 ~ LH(Low byte,High byte)
Function Code Verification	0 ~ Check Function Code
Block Acquisition Method	0 ~ Chunk by maximum length
Write Function Code to Zone 0	2 ~ default value
Write Function Code to Zone 4	2 ~ default value
Maximum block length in Zone 1	120
Maximum block length in Zone 0	120