# OEL15&30&60 Modbus 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.

## Sep. 2026 edition V1.0.0

Copyright © LILLIPUT Company. All rights reserved.

The LILLIPUT's products are under the protection of the patent rights, including ones which have already obtained the patent rights and those which are applying for. The information in this manual will replace all that in the materials published originally.

The information in this manual was correct at the time of printing. However, LILLIPUT will continue to improve products and reserves the rights to change specification at any time without notice.

OWON® is the registered trademark of the LILLIPUT Company.

**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

# Table of Contents

1. Overview ..... 1
1.1 Configure communication baud rate ..... 1
1.2 Select communication protocol ..... 2
1.3 Set communication address ..... 2
2. Modbus Introduction ..... 4
2.1 Frame format ..... 4
2.2 Function Code ..... 4
3. Address List ..... 11
3.1 Register Address ..... 11
3.2 Register Value Description ..... 15
4. Programming Reference ..... 17
4.1 Communication Model ..... 17
4.2 Number Conversion ..... 17
4.3 Touchscreen Communication Configuration ..... 18

## Statement

Copyright of this manual belongs to our company. The information contained in this manual is intended for customers' secondary development or building test systems.

The company shall not be held liable for any errors contained in this manual or damages resulting from the provision, performance, and use of this manual.

## Version Revision History

| Date | Version | Revised Chapters |
|---|---|---|
| 2026.09 | V1.0.0 | Complete this manual |

# 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, 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 instrument (Note: Supported in V1.6.0 and later versions).

## 1.1 Configure communication baud rate

The load supports communication interfaces such as USB (VCP) and RS232. The "Communication Interface" option on the front panel displays USB / RS232. When RS232 is selected, data is transmitted and received via UART4 of the instrument (shared physical serial port with RS485).

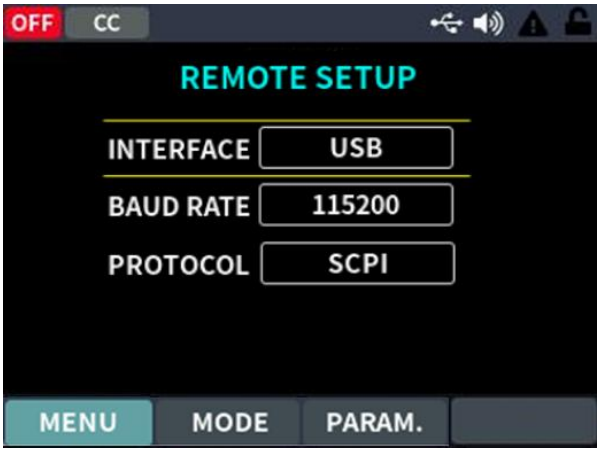
| Interface | Description |
|---|---|
| USB | USB to Virtual COM Port (VCP), SCPI protocol only |
| RS232 | Standard serial port (shared physical port with RS485), SCPI or Modbus selectable |

Fixed serial parameters: 8 data bits, no parity, 1 stop bit.

Available baud rates: 2400, 4800, 9600, 19200, 38400, 57600, 115200. Factory default: 115200.

The configuration method of communication parameters is as follows:

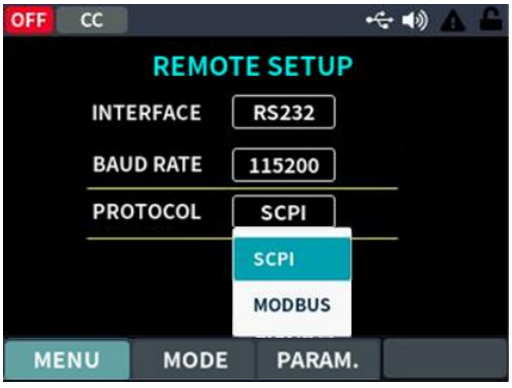
1. Enter the system menu on the front panel, select "Remote Settings" via the knob, and press the confirm key to enter the remote settings interface.
2. Rotate the knob to move the cursor to the "Baud Rate" item, then press the confirm key to enter the edit state.
3. Rotate the knob to select the corresponding parameter.
4. Press the confirm key to activate the communication parameters. The baud rate applies to both USB virtual COM port and RS232.




## 1.2 Select communication protocol

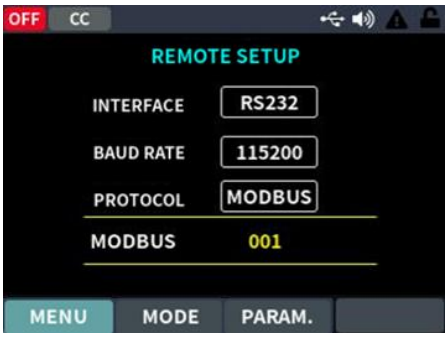
The load supports two communication protocols: SCPI and Modbus. The default protocol is SCPI. The USB interface only supports SCPI. To use Modbus, switch the communication interface to RS232, then follow the steps below:

1. Enter the system menu on the front panel, select "Remote Settings" via the knob, and press the confirm key to enter the remote settings interface.
2. Rotate the knob to move the cursor to the "Communication Protocol" item, then press the confirm key to enter the edit state.
3. Rotate the knob to select "Modbus".
4. Press the confirm key to activate the communication protocol.




## 1.3 Set communication address

Modbus slave address can be set from 1 to 247, and responds to broadcast address 255 (0xFF).




The front panel will be remotely locked during valid Modbus communication, and automatically unlock after approximately 3 seconds without new frames.

# 2. Modbus Introduction

The load supports standard Modbus RTU protocol.

## 2.1 Frame format

### 2.1.1 Modbus RTU

For serial communication (e.g. RS232), Modbus messages are transmitted in RTU mode. The format is as follows.

| Field | Byte | Description |
|---|---|---|
| Slave Address | 1 Byte | 1 for this instrument; 255 for broadcast |
| Function Code | 1 Byte | Operation requested |
| Data | N Byte | Starting address, quantity or register values, etc. |
| CRC Checksum | 2 Byte | Calculated over address, function code and data; least significant byte first |

- **Slave Address:** Modbus communication address, set to 1 for this instrument.
- **Function Code:** Request to perform a certain operation, such as reading or writing registers.
- **Data:** Describes detailed operation information, e.g. starting address and quantity of registers to read.
- **CRC Checksum:** CRC16 result calculated over Slave Address, Function Code and Data. Least significant byte transmitted first, most significant byte last.

## 2.2 Function Code

The Modbus protocol of instrument 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 | 1~125 |

**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**

Read holding registers, starting at address 1000, read 2 consecutive registers.

Request: 01 03 03 E8 00 02 44 7B

Response: 01 03 04 00 01 00 00 AB F3

| **Request** | | **Response** | |
|---|---|---|---|
| Field | 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 | 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 | 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 | 0x06 |
| 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**

Write holding register 1000 to 1 (Turn On).

Request: 01 06 03 E8 00 01 C8 7A

Response: 01 06 03 E8 00 01 C8 7A

| **Request** | | **Response** | |
|---|---|---|---|
| Field | 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) | C8 |
| 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 holding registers.

Floating-point parameters must be written to 2 registers at a time.

**Request**

| | | |
|---|---|---|
| Function Code | 1 Byte | 0x10 |
| Initial Address | 2 Bytes | 0x0000~0xFFFF |
| Register Count | 2 Bytes | N |
| Byte count | 1 Byte | 2xN |
| Register Value | 2xN Bytes | value |

**Response**

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

**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**

Write resistance value 1000.0 to holding register 1012 (IEEE754 format, low word first).

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** | |
|---|---|---|---|
| Field | 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 | 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 Value | 2 Bytes | 1~125 |

**Response**

| | | |
|---|---|---|
| Function Code | 1 Byte | 0x04 |
| Return Byte Count | 1 Byte | 2xN |
| Register Value | N x 2 Bytes | value |

**Error**

| **Function Code** | **Byte** | **Value** |
|---|---|---|
| Error Code | 1 Byte | 0x84 |
| 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**

Read input registers, starting at address 150, read 2 consecutive registers (Output voltage).

Request: 01 04 00 96 00 02 91 E7

Response: 01 04 04 AA C6 3D 1B 6A FA (The value varies with the measured voltage)

| **Request** | | **Response** | |
|---|---|---|---|
| Field | 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 | 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 | 1B |
| CRC Checksum (High Byte) | E7 | CRC Checksum (Low Byte) | 6A |
| | | CRC Checksum (High Byte) | FA |

# 3. Address List

## 3.1 Register Address

Load registers are shown in Table 3-1 Holding Registers and Table 3-2 Input Registers. Unlisted addresses are reserved, do not write to them.

Table 3-1 Holding Registers (Function Code 03 / 06 / 10, Start Address 1000)

| Address | Function | Data Type | Address Increment |
|---|---|---|---|
| 1000 | Output control (ON/OFF) | uint16 | 1 |
| 1001 | reserve | uint16 | 1 |
| 1002 | Output voltage setting | float | 2 |
| 1004 | Output current setting | float | 2 |
| 1006 | Current rise slope | float | 2 |
| 1008 | Current fall slope | float | 2 |
| 1010 | Output power setting | float | 2 |
| 1012 | Output resistance setting | float | 2 |
| 1014 | Voltage range selection | uint16 | 1 |
| 1015 | Current range selection | uint16 | 1 |
| 1016 | Voltage rise slope | uint16 | 1 |
| 1017 | reserve | uint16 | 1 |
| 1018 | Voltage fall slope | uint16 | 1 |
| 1019 | reserve | uint16 | 1 |
| 1020 | Output mode | int16 | 1 |
| 1021 | Short circuit mode | uint16 | 1 |
| 1022 | Load voltage | float | 2 |
| 1024 | reserve | uint16 | 2 |
| 1026 | Dynamic way | uint16 | 1 |
| 1027 | CC dynamic low current value | float | 2 |

| | | | |
|---|---|---|---|
| 1029 | CC dynamic low current time value | float | 2 |
| 1031 | CC dynamic high current value | float | 2 |
| 1033 | CC dynamic high current time value | float | 2 |
| 1035 | CC dynamic current rise slope | float | 2 |
| 1037 | CC dynamic current drop slope | float | 2 |
| 1039 | Remote compensation settings | uint16 | 1 |
| 1040 | reserve | uint16 | 1 |
| 1041 | Trigger signal | uint16 | 1 |
| 1042 | Battery Test - Cut-off discharge voltage | float | 2 |
| 1044 | Battery Test - discharge capacity | float | 2 |
| 1046 | Battery Test - discharge time | float | 2 |
| 1048 | Battery Test - test condition mask | uint16 | 1 |
| 1049 | Battery Test - Start-Stop control test | uint16 | 1 |
| 1050 | Serial/Parallel mode selection (reserve) | uint16 | 1 |
| 1051 | Number of series/parallel units (reserve) | uint16 | 1 |
| 1052 | Master-Slave settings (reserve) | uint16 | 1 |
| 1053 | Local address (reserve) | uint16 | 1 |
| 1054 | Parallel control (reserve) | uint16 | 1 |
| 1055 | reserve | uint16 | 1 |
| 1056 | CR rise slope | uint16 | 1 |

| | | | |
|---|---|---|---|
| 1057 | CR fall slope | uint16 | 1 |
| 1058 | CP rise slope | uint16 | 1 |
| 1059 | CP fall slope | uint16 | 1 |
| 1060 | Battery Test - Current settings | float | 2 |
| 1062~1099 | reserve | — | — |
| 1100 | OCP starting voltage | float | 2 |
| 1102 | OCP initial current | float | 2 |
| 1104 | OCP step Current | float | 2 |
| 1106 | OCP step delay | float | 2 |
| 1108 | OCP protection voltage | float | 2 |
| 1110 | OCP test maximum current | float | 2 |
| 1112 | reserve | float | 2 |
| 1114 | reserve | float | 2 |
| 1116 | OPP starting voltage | float | 2 |
| 1118 | OPP initial power | float | 2 |
| 1120 | OPP step power | float | 2 |
| 1122 | OPP step delay | float | 2 |
| 1124 | OPP protection voltage | float | 2 |
| 1126 | OPP maximum power | float | 2 |
| 1128 | OPP minimum power | float | 2 |
| 1130~1203 | reserve | — | — |
| 1204 | OVP | float | 2 |
| 1206 | OCP | float | 2 |
| 1208 | OPP | float | 2 |
| 1210 | Trigger Mode Selection | uint16 | 1 |
| 1211 | reserve | uint16 | 1 |
| 1212 | reserve | float | 2 |
| 1214 | reserve | float | 2 |
| 1216 | reserve | float | 2 |

Table 3-2 Input Registers (Function Code 04, Start Address 150, Read-only)

| Address | Function | Data Type | Address Increment |
|---|---|---|---|
| 150 | Output voltage (Main host) | float | 2 |
| 152 | Output current (Main host) | float | 2 |
| 154 | Output power (Main host) | float | 2 |
| 156 | Discharge Capacity Ah | float | 2 |
| 158 | Discharge Energy Wh | float | 2 |
| 160 | Discharge Time | float | 2 |
| 162 | Battery Test Status | uint16 | 1 |
| 163 | reserve | uint16 | 1 |
| 164 | OCP test value | float | 2 |
| 166 | OCP test status | uint16 | 1 |
| 167 | reserve | uint16 | 1 |
| 168 | OPP test value | float | 2 |
| 170 | OPP test status | uint16 | 1 |
| 171~250 | reserve | — | — |
| 251 | System Operation Flag | uint16 | 1 |
| 252 | reserve | uint16 | 1 |
| 253 | reserve | uint16 | 1 |
| 254 | reserve | uint16 | 1 |
| 255 | Internal Fault Word 2 | uint16 | 1 |
| 256 | Output Fault Word | uint16 | 1 |
| 257 | Parallel Fault Word | uint16 | 1 |
| 258 | Alarm Word | uint16 | 1 |
| 259 | reserve | uint16 | 1 |
| 260 | reserve | uint16 | 1 |
| 261 | reserve | uint16 | 1 |
| 262 | reserve | uint16 | 1 |
| 263 | Mains Fault Word (Latched) | uint16 | 1 |
| 264 | reserve | uint16 | 1 |

| | | | |
|---|---|---|---|
| 265 | Internal Fault Word 2 (Latched) | uint16 | 1 |
| 266 | Output Fault Word (Latched) | uint16 | 1 |
| 267 | Parallel Fault Word (Latched) | uint16 | 1 |
| 268~319 | reserve | — | — |

## 3.2 Register Value Description

| Address | Description |
|---|---|
| 1000 | 0: power off (OFF); 1: power on (ON) |
| 1014 / 1015 | 0: low range; 1: high range |
| 1016 / 1018 / 1056~1059 | 0: Fast; 1: Medium; 2:S low |
| 1020 | See the table below "Output Mode Code" |
| 1021 | 0: Short circuit disabled; 1: Short circuit enabled |
| 1026 | 0: Continuous dynamic; 1: Single-shot dynamic |
| 1039 | 0: Remote compensation disabled; 1: Remote compensation enabled |
| 1041 | 0: no trigger; 1: trigger |
| 1048 bit0 | 1: Enable cut-off voltage condition |
| 1048 bit1 | 1: Enable discharge capacity condition |
| 1048 bit2 | 1: Enable discharge time condition |
| 1049 | Same as address 1000, used to control start and stop of battery test |
| 1210 | 0: Manual trigger; 1: External trigger |
| 1212 / 1214 / 1216 | Combined Mode Current Limit. There is no combined mode interface on the front panel of the instrument; the register only stores the value. |

**Output Mode Code (Address 1020)**

| Value | Mode | Description |
|---|---|---|
| 0 | CR | Constant resistance |
| 1 | CC | Constant current |
| 2 | CV | Constant voltage |

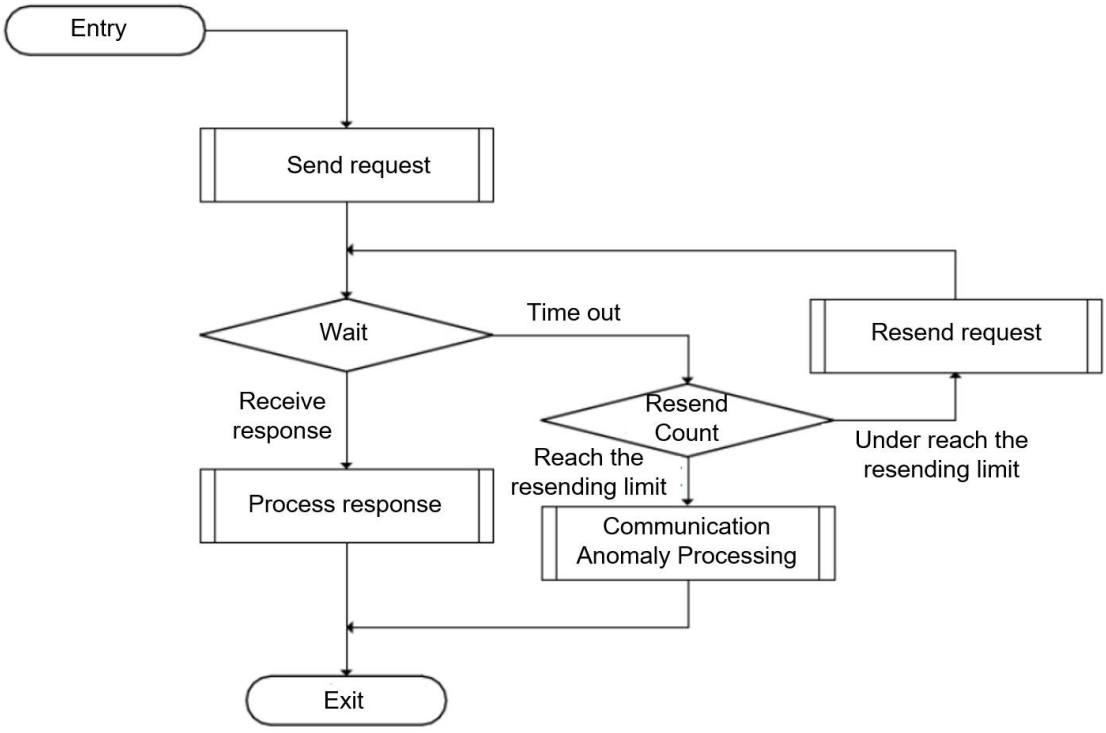
| | | |
|---|---|---|
| 3 | CP | Constant power |
| 4 | CCD | Dynamic current |
| 5 | Battery | Battery test |
| 6 | OCP | Overcurrent protection |
| 7 | OPP | Overpower protection |

When writing to address 1020, valid values range from 0 to 7. Values 8~10 are reserved combined mode codes. The current firmware will not switch the interface after writing these codes.

# 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 |