WUXI SUPER LASER TECHNOLOGY CO.,LTD

Handheld laser welding system communication protocol

V1.5

Modify record：

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Major changes | Date of creation/revision | Remarks |
| V1.0 | First release | 2021/7/7 |  |
| V1.1 | Add some examples | 2021/7/10 |  |
| V1.2 | Supplementary part description | 2021/11/01 |  |
| V1.3 | Added hardware interface definitions and updated document formats | 2021/11/27 |  |
| V1.4 | (1) Added function code 0x06 for writing a single register;  (2) Change CRC to low byte before high byte after;  (3) Add the definition of register addresses related to the wire feeder. | 2022/9/1 | This software protocol supports software version 557 and above. |
| V1.5 | Added the definition of hardware interfaces about the V6.1 main control board; | 2023/10/23 | This software protocol supports software version 557 and above. |

目录

[1. Hardware interface definition 1](#_Toc4738)

[2. Communication format 2](#_Toc10977)

[2.1 Basic parameters of communication module 2](#_Toc1822)

[2.2 Register address Definition 3](#_Toc27856)

[2.3 Data Frame Format 5](#_Toc19760)

[2.4 Read Register instruction - Function code 0x03 7](#_Toc13722)

[2.4.1 Function Code 0x03 Instruction structure 7](#_Toc26088)

[2.4.2 Example of Function Code 0x03 Instruction 8](#_Toc17866)

[2.5 Write Register Instruction - Function Code 0x06 and Function Code 0x10 9](#_Toc2934)

[2.5.1 Introduction to Function Code 0x06 Instruction Structure 9](#_Toc9069)

[2.5.2 Example of Function Code 0x06 Instruction 10](#_Toc5572)

[2.5.3 Function code 0x10 instruction structure 11](#_Toc28790)

[2.5.4 Function code 0x10 instruction example 11](#_Toc27113)

[2.6 Usage Instructions 12](#_Toc17641)

[2.7 Examples of Common commands 12](#_Toc29203)

[Appendix: CRC algorithm 14](#_Toc13190)

# Hardware interface definition

The external communication interface of our handheld laser welding product is RS485 interface, as shown in Figure 1.1 and Figure 1.2, which correspond to the hardware versions V6.1 and V6.2. Table 1.1, Table 1.2, and Table 1.3 respectively list the interface definitions of different hardware versions of the control box:

Table 1.1 Interface definitions for hardware version V5.81 and earlier

|  |  |  |
| --- | --- | --- |
| version V5.81 and earlier Interface: DB9 female socket | | |
| Pin number | Signal name | Function description |
| 1 | RS485\_B | Signal B of the RS485 |
| 2 | RS232\_TXD | For internal use, it is recommended not to connect |
| 3 | RS232\_RXD |
| 4 | NC | Not used |
| 5 | GND | GND |
| 6 | RS485\_A | Signal A of the RS485 |
| 7 | NC | Not used |
| 8 | NC | Not used |
| 9 | +5V | Not used |



Figure 1.1 Hardware Version V6.1 External Communication Interface Diagram

Table 1.2 Hardware Version V6.1 Interface Definition

|  |  |  |
| --- | --- | --- |
| V6.1 version Interface: 15EDGV-3.81mm-6Pin straight pin socket | | |
| Pin number | Signal name | Function description |
| 1 | TX | For internal use, it is recommended not to connect |
| 2 | RX |
| 3 | GND | GND |
| 4 | RS485\_B | Signal B of the RS485 |
| 5 | RS485\_A | Signal A of the RS485 |
| 6 | 5V output | Reserved,it is recommended not to connect |

Note: Please connect the wires according to the corresponding interface definition table based on the hardware version.

# Communication format

This protocol is compatible with the Modbus RTU specification.

## Basic parameters of communication module

Basic parameters of the communication module are shown in Table 2.1:

Table 2.1 Basic parameters of communication module

|  |  |
| --- | --- |
| Encoding | 8-bit binary |
| Data bit | 8-bit |
| Parity check bit | Without |
| Stop bit | 1-bit |
| Baud rate | 19200 bit/s |

## Register address Definition

The definition of register address is shown in Table 2.2:

Table 2.2 Register address definition table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| NO. | Function name | Data length  (Bytes) | Data type | Range | Register address | R/W  attribute |
| Handheld laser welding system related parameters | | | | | | |
| 1 | Scan speed | 2 | Unsigned,  Expand transmission by 10x, i.e. retain 1 decimal place | 2~6000 mm/s | 0x0000 | R/W |
| 2 | Scan width | 2 | Unsigned,  Expand transmission by 10x. | 0~6 mm | 0x0001 | R/W |
| 3 | Peak power | 2 | Unsigned | 1~3000W(Actual cannot exceed laser power) | 0x0002 | R/W |
| 4 | Duty cycle | 2 | Unsigned | 0~100 % | 0x0003 | R/W |
| 5 | Frequency  Bit 31~16 | 2 | Unsigned | 0~100000 Hz | 0x0004 | R/W |
| 6 | Frequency  Bit 15~0 | 2 | Unsigned | 0x0005 | R/W |
| 7 | Save current handheld welding data | 2 | Unsigned | 0 or 1 | 0x003F | W |
| Wire feeder related parameters | | | | | | |
| 1 | Automatic wire feeding speed | 2 | Unsigned | 15~600 cm/min | 0x0100 | R/W |
| 2 | Manual feed speed | 2 | Unsigned | 15~600 cm/min | 0x0101 | R/W |
| 3 | Manual Withdrawal speed | 2 | Unsigned | 15~600 cm/min | 0x0102 | R/W |
| 4 | Start-up delay | 2 | Unsigned | 0~2000 ms | 0x0103 | R/W |
| 5 | Withdrawal length | 2 | Unsigned | 0~100 mm | 0x0104 | R/W |
| 6 | Supplement length | 2 | Unsigned | 0~100 mm | 0x0105 | R/W |
| 7 | Supplement delay | 2 | Unsigned | 0~2000 ms | 0x0106 | R/W |
| 8 | Pulse period | 2 | Unsigned | 100~1000 ms | 0x0107 | R/W |
| 9 | Smoothness | 2 | Unsigned | 25 %~80 % | 0x0108 | R/W |
| 10 | Average speed | 2 | Unsigned | 15~150 cm/min | 0x0109 | R/W |
| 11 | Language selection | 2 | Unsigned | 0～8 | 0x010A | R/W |
| 12 | reserve | 2 | Unsigned |  | 0x010B | R/W |
| 13 | Mode setting | 2 | Unsigned | 0：Continuous mode  1：Pulse mode | 0x010C | R/W |
| 14 | reserve | 2 | / |  | 0x010D | R/W |
| 15 | reserve | 2 | / |  | 0x010E | R/W |
| 16 | reserve | 2 | / |  | 0x010F | R/W |
| 17 | reserve | 2 | / |  | 0x0110 | R/W |
| 18 | reserve | 2 | / |  | 0x0111 | R/W |
| 19 | Over current alarm | 2 | Unsigned | 1：alarm  2：normal | 0x0112 | R |
| 20 | Hardware version | 2 | Unsigned |  | 0x0113 | R |
| 21 | Software version | 2 | Unsigned |  | 0x0114 | R |
| 22 | Running state | 2 | Unsigned | 0: stop  1: run | 0x0115 | R |
| 23 | Save current wire feeder data | 2 | Unsigned | XX(Arbitrary value) | 0x013F | W |

Note: R/W - indicates that the parameter can be read and written; R - indicates that the parameter can only be read. W - indicates that this parameter can only be written

Starting from the V1.4 version of the protocol, support for hand-held welding query and setting of wire feeder parameters.

The functional definitions of each register are shown in Table 2.3:

Table 2.3 Functional Definitions of Each Register

|  |  |  |
| --- | --- | --- |
| NO. | Function name | Functional definition |
| Handheld laser welding system related parameters | | |
| 1 | Scan speed | The scanning speed is 2～6000mm/s, and the sanning width is 0～6mm. The scanning speed is limited by the scanning width, and the limit relationship is:  10 ≤ scanning speed / (scanning width \* 2) ≤ 1000. if the limit is exceeded, it will automatically become the limit value. When the scanning width is set to 0, it will not scan(i.e. point light sourece). |
| 2 | Scan width |
| 3 | Peak power | The peak power should be less than or equal to the laser power on the parameter page. |
| 4 | Duty cycle | Duty cycle range: 0～100. |
| 5 | Frequency | The rang of pulse frequency is 5～100000Hz, and 5～5000Hz is recommended. |
| 6 | Save current handheld welding data | This command will save all current process data of handheld welding. Suggestion: Do not save unless necessary to avoid affecting the lifespan of the flash. |
| Wire feeder related parameters | | |
| 1 | Automatic wire feeding speed | Means the speed of automatic feeding wire, the range is: 15~600cm/min. |
| 2 | Manual feed speed | Means the speed of manually feeding wire, the range is: 15~600cm/min. |
| 3 | Manual Withdrawal speed | Means the manually draw back speed, the range is: 15~600cm/min. |
| 4 | Start-up delay | Means the delay time from pressed down the feeding wire switch to start feeding wire, the range is: 0~2000ms. |
| 5 | Withdrawal length | Means the length of draw back welding wire after feeding wire switch loosen (assist to broken wire), the range is: 0~100mm. |
| 6 | Supplement length | means the length of filling wire after draw back welding wire, the range is 0~100mm. |
| 7 | Supplement delay | Means the waiting time between draw back and filling wire, the range is: 0~2000ms. (Used to strengthen the broken wire effect) |
| 8 | Pulse period | Used for the pulse mode, reflects the size of the individual fishtail pattern, the larger the value, the larger the fishtail pattern. The range is: 100 to 1000ms. |
| 9 | Smoothness | Used for the pulse mode, reflects the smoothness of the fish scale pattern in fish scale welding, the larger the value, the smoother the pattern, the less obvious the fish scale effect. Its range is: 25%～80%. |
| 10 | Average speed | Used for pulse control mode, representing the average wire feeding speed in pulse mode, with a range of 15～150cm/min. |
| 11 | Language selection | The range is: 0～8 |
| 12 | reserve |  |
| 13 | Mode setting | Used to set the current wire feeding mode:  0: Continuous mode  1: Pulse mode |
| 14~18 | reserve |  |
| 19 | Over current alarm | After the wire feeder generates an overcurrent alarm, it automatically stops wire feeding. If no abnormality is found, restart the wire feeder to resume normal operation. |
| 20 | Hardware version | Hardware version of the wire feeder currently in use. |
| 21 | Software version | Software version of the main control board of the wire feeder in use. |
| 22 | Running state | The running state of automatic wire feeding, stopped or running. |
| 23 | Save current wire feeder data | This command will save all current process data of handheld welding. Suggestion: Do not save unless necessary to avoid affecting the lifespan of the flash. |

## Data Frame Format

(1) Basic data format for sending and receiving (successful response)

The basic data format for sending and receiving receipts is shown in Table 2.4:

Table 2.4 Basic Data Format for Sending and Receipt

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Slave address  (1 Byte) | Function code  (1 Byte) | Data area  (n Bytes) | | | Check code  (Low Bytes) | Check code  (High Bytes) |
| 0x09 | 0x03/0x06/0x10 | 0xXX | …… | 0xXX | 0xXX | 0xXX |

**Slave address**: The default address for handheld soldering is 0x09, which can be modified through the display screen. The modification method is as follows:

① Enter the authorization password input page of the display screen monitoring page;

② Enter ffffffdd001~ffffffdd247 (the last 3 bits represent the slave address) to modify the slave address to 1~247. As shown in Figure 2-1, modify the handheld welding address to 1:

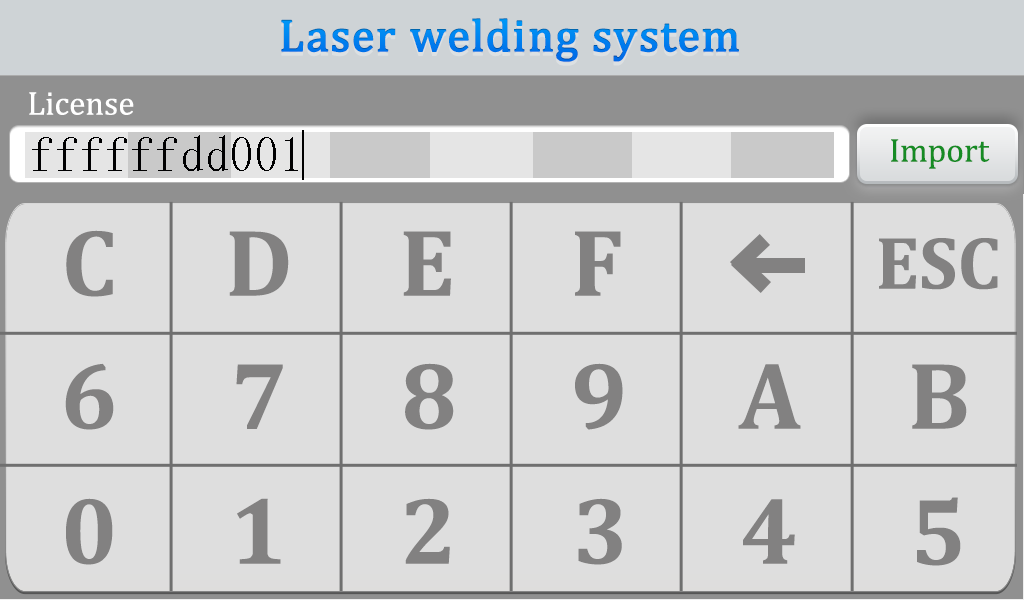


Figure 2-1 Change the handheld welding address to 1

**Function code**: see Table 2.5.

0x03: reads one or more registers, and the corresponding exception receipt function code is 0x83;

0x06: writes a single register, and the corresponding exception receipt function code is 0x86.

0x10: writes multiple registers, and the corresponding exception receipt function code is 0x90.

**Data area**: n bytes, in case of double byte data, the high byte comes first and the low byte comes last.

**Error verification code**: The verification method adopts CRC16 (Modbus), with high bytes first and low bytes last. CRC calculation includes all data from the "slave address" until the CRC checksum. The calculation method is shown in Appendix A.

CRC verification process: After the sender calculates the CRC of the data to be sent, the calculated CRC value is added to the sending frame. After receiving the data frame, the receiver recalculates the CRC value of the received data and compares it with the received CRC value. If the two CRC values are the same, it is considered normal, otherwise it is abnormal.

Table 2.5 List of Function Codes

|  |  |  |  |
| --- | --- | --- | --- |
| Function code  (Sending) | Function | Receipt function code  (Normal) | Receipt function code  (Abnormal) |
| 0x03 | Read 1 or more registers | 0x03 | 0x83 |
| 0x06 | Write a single register | 0x06 | 0x86 |
| 0x10 | Write multiple registers | 0x10 | 0x90 |

(2) Error receipt basic data format:

The basic data format of the error receipt is shown in Table 2.6:

Table 2.6 Basic data format of error receipt

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Slave address  (1 Byte) | Function code  (1 Byte) | Error code  (1 Byte) | CRC16  (low byte) | CRC16  (high byte) |
| 0x09 | 0x83/0x86/0x90 | 0x01/0x02/0x03 | 0XX | 0xXX |

Abnormal function code corresponding to the receipt relationship:

0x03-0x83: If the function value is abnormal, the upper position of the return function code is 1, that is, 0x83.

0x06-0x86: If the function value is abnormal, the upper position of the return function code is 1, that is, 0x86.

0x10-0x90: If the function value is abnormal, the upper position of the return function code is 1, that is, 0x90.

Error code:

0x01: Invalid function;

0x02: illegal register address;

0x03: Invalid register value.

Note: When the slave machine address is wrong, CRC check error, function code error, hand-held welding does not respond.

By default, the CRC is in small-endian mode (low bytes before, high bytes after). You can change the sequence of CRC high and low bytes on the display as follows:

① Enter the authorization password input page of the monitor page;

② Enter ffffffee111 to change CRC to big-endian mode (high bytes before, low bytes after), and enter ffffffee222 to change CRC to small-endian mode (low bytes before, high bytes after). As shown in Figure 2-2, the CRC is set to big-end mode:

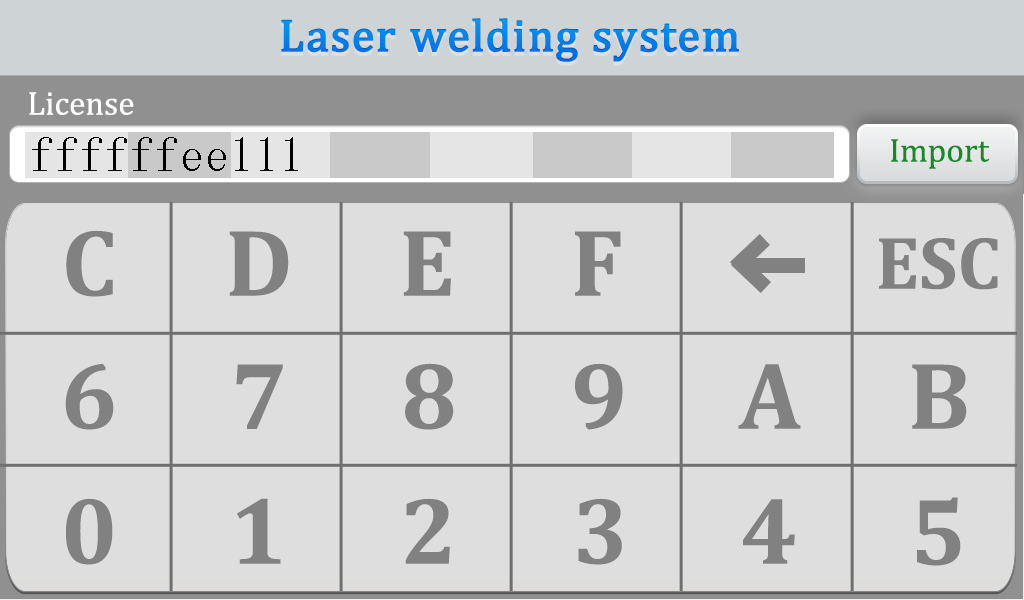


Figure 2-2 Setting the CRC to big-end mode

## Read Register instruction - Function code 0x03

### 2.4.1 Function Code 0x03 Instruction structure

(1) Function code 0x03 can read the value of one or more registers at the same time, and its instruction structure is shown in Table 2.7:

Table 2.7 Function code 0x03 instruction structure

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Read register (host request) | | | | | | | | |
| Data direction | Slave address  (1 Byte) | Function code  (1 Byte) | Data area | | | | Check code | |
| Host → Slave | Register Start Address  (2 Bytes) | | Number of Registers  (2 Bytes) | | CRC16 (2 Bytes) | |
| Register Start Address  (High Byte) | Register Start Address  (Low Byte) | Number of Registers  (High Byte) | Number of Registers  (Low Byte) | CRC16 (low byte) | CRC16 (high byte) |
| 0x09 | 0x03 | 0xXX | 0xXX | N | | 0xXX | 0xXX |

Note: When reading multiple registers, it means reading N consecutive registers starting from the starting address of the register, where continuity refers to the continuity of register addresses.

(2)The function code 0x03 successfully responded, and the instruction structure is shown in Table 2.8:

Table 2.8 Function Code 0x03 Successful Response Command Structure

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Read register (slave response) | | | | | | | | | | |
| Data direction | Slave address  (1 Byte) | Function code  (1 Byte) | Data area | | | | | | Check code | |
| Slave → Master  (Successfully responded) | Return Bytes  (1 Byte) | Data1 (2 Bytes) | | …… | DataN (2 Bytes) | | CRC16 (2 Bytes) | |
| Data1  (High Bytes) | Data1  (Low Bytes) | …… | DataN (High Bytes) | DataN (Low Bytes) | CRC16 (Low Bytes) | CRC16 (High Bytes) |
| 0x09 | 0x03 | 2\*N | 0xXX | 0xXX |  | 0xXX | 0xXX | 0xXX | 0xXX |

Note:

① Return Bytes "represents the total number of bytes from" Data1 "to" DataN ".

② Data1 is the value of the register corresponding to the starting address, Data2 is the value of the register corresponding to the starting address+1, and so on.

(3) The error response instruction structure is shown in Table 2.6.

### 2.4.2 Example of Function Code 0x03 Instruction

(1) Example commands for reading "peak power" and "duty cycle" are shown in Table 2.9:

Table 2.9 Example of Reading "Peak Power" and "Duty Cycle

|  |  |
| --- | --- |
| Data direction | Instruction |
| Host → Slave | 09 03 00 02 00 02 64 83 |
| Slave → Master | 09 03 04 00 64 00 32 B3 F9 |

Example explanation is shown in Table 2.10:

Table 2.10 Example Explanation of Reading "Peak Power" and "Duty Cycle

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Data direction | Slave address  (1 Byte) | Function code  (1 Byte) | Register Address  (2 Bytes) | Number of Registers  (2 Bytes) | CRC16 (2 Bytes) |  |
| Host → Slave | 0x09 | 0x03 | 0x0002 | 0x0002 | 0x6483 |  |
|  |  |  |  |  |  |  |
| Data direction | Slave address  (1 Byte) | Function code  (1 Byte) | Return Bytes  (1 Byte) | Data1 (2 Bytes) | Data2 (2 Bytes) | CRC16 (2 Bytes) |
| Slave → Master | 0x09 | 0x03 | 0x04 | 0x0064 | 0x0032 | 0xB3F9 |

As shown in Table 2.10:

Host → Slave: The function code for reading registers is 0x03, the address for "peak power" is 0x0002, and the address for "duty cycle" is 0x0003. Their addresses are consecutive. Therefore, by directly reading the values of the two registers with address 0x0002 as the starting address, the values for "peak power" and "duty cycle" can be read.

Slave → Host: Successfully responded to register reading with unchanged function code of 0x03; Returns the values of two registers, with byte count=2 \* 2=4; Data1 is 0x0064, corresponding to a decimal form of 100, indicating that the current peak power is 100W; Data2 is 0x0032, corresponding to a decimal form of 50, indicating that the current duty cycle is 50%.

(2) Example of error response:

As shown in Table 2.11, when the host sends a read from a non-existent register address 0x0010 to the slave, the slave responds with error function code 0x83 and error code 0x02 (indicating an illegal register address).

Table 2.11 Example of Reading Register Error Response

|  |  |
| --- | --- |
| Data direction | Instruction |
| Host → Slave | 09 03 00 10 00 01 84 87 |
| Slave → Master | 09 83 02 41 33 |

## Write Register Instruction - Function Code 0x06 and Function Code 0x10

### 2.5.1 Introduction to Function Code 0x06 Instruction Structure

(1) The function code 0x06 can be written to a single register, and its instruction structure is shown in Table 2.12:

Table 2.12 Function Code 0x06 Instruction Structure

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Write a single register (host request) | | | | | | | | |
| Data direction | Slave address  (1 Byte) | Function code  (1 Byte) | Data area | | | | Check code | |
| Host → Slave | Register Address  (2 Bytes) | | Register value  (2 Bytes) | | CRC16 (2 Bytes) | |
| Register Address  (High Byte) | Register Address  (Low Byte) | Register value  (High Byte) | Register value  (Low Byte) | CRC16 (Low Byte) | CRC16 (High Byte) |
| 0x09 | 0x06 | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX |

As shown in Table 2.12, the function code 0x06 is used to write data to a single register address.

(2) Function code 0x06 successful response, instruction structure is shown in Table 2.13:

Table 2.13 Function code 0x06 Successful response instruction structure

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Write a single register (slave response) | | | | | | | | |
| Data direction | Slave address  (1 Byte) | Function code  (1 Byte) | Data area | | | | Check code | |
| Slave → Master  (Successfully responded) | Register Address  (2 Bytes) | | Register value  (2 Bytes) | | CRC16 (2 Bytes) | |
| Register Address  (High Byte) | Register Address  (Low Byte) | Register value  (High Byte) | Register value  (Low Byte) | CRC16 (Low Byte) | CRC16 (High Byte) |
| 0x09 | 0x06 | 0xXX | 0xXX | N | | 0xXX | 0xXX |

(3) The structure of the error response instruction is shown in Table 2.6.

### 2.5.2 Example of Function Code 0x06 Instruction

An example of writing 'peak power' is shown in Table 2.14:

Table 2.14 Example of Writing "Peak Power"

|  |  |
| --- | --- |
| Data direction | Instruction |
| Host → Slave | 09 06 00 02 00 64 28 A9 |
| Slave → Master | 09 06 00 02 00 64 28 A9 |

Example explanation is shown in Table 2.15:

Table 2.15 Example Explanation of Writing "Peak Power"

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data direction | Slave address  (1 Byte) | Function code  (1 Byte) | Register Address  (2 Bytes) | DATA1 (2 Bytes) | CRC16 (2 Bytes) |
| Host → Slave | 0x09 | 0x06 | 0x0002 | 0x0064 | 0x28A9 |
|  |  |  |  |  |  |
| Data direction | Slave address  (1 Byte) | Function code  (1 Byte) | Register Address  (2 Bytes) | Number of Registers  (2 Bytes) | CRC16 (2 Bytes) |
| Slave → Master | 0x09 | 0x06 | 0x0002 | 0x0064 | 0x28A9 |

As shown in Table 2.15, the address of the "Peak Power" register is 0x0002. In the example, writing the value 0x0064 (corresponding to a decimal number of 100) to address 0x0002 indicates setting the "Peak Power" to 100W.

### 2.5.3 Function code 0x10 instruction structure

(1) Function code 0x10 can write multiple registers simultaneously, and its instruction structure is shown in Table 2.16:

Table 2.16 Function Code 0x10 Instruction Structure

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Write register (host request) | | | | | | | | | | | | | | |
| Data direction | Slave address (1 Byte) | Function code (1 Byte) | Data area | | | | | | | | | | Check code | |
| Host → Slave | Register Start Address (2 Bytes) | | Number of Registers (2 Bytes) | | Number of data bytes (1 Byte) | Data1 (2 Bytes) | | … | DataN (2 Bytes) | | CRC16 (2 Bytes) | |
| Register Start Address (High Byte) | Register Start Address (Low Byte) | Register Number  (High Byte) | Register Number(Low Byte) | Data1 (High Byte) | Data1 (Low Byte) | ... | DataN (High Byte) | DataN (Low Byte) | CRC16 (Low Byte) | CRC16 (High Byte) |
| 0x09 | 0x10 | 0xXX | 0xXX | N | | 2\*N | 0xXX | 0xXX |  | 0xXX | 0xXX | 0xXX | 0xXX |

As shown in Table 2.16, function code 0x10 can simultaneously write data to N consecutive registers starting with "register start address". Data Bytes "represents the total number of bytes from" Data1 "to" DataN ".

(2) The function code 0x10 successfully responded, and the instruction structure is shown in Table 2.17:

Table 2.17 Function Code 0x10 Successful Response Instruction Structure

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Write register (slave response) | | | | | | | | |
| Data direction | Slave address  (1 Byte) | Function code  (1 Byte) | Data area | | | | Check code | |
| Slave → Master  (Successfully responded) | Register Start Address  (2 Bytes) | | Number of Registers  (2 Bytes) | | CRC16 (2 Bytes) | |
| Register Start Address  (High Byte) | Register Start Address  (Low Byte) | Register Number  (High Byte) | Register Number(Low Byte) | CRC16 (Low Byte) | CRC16 (High Byte) |
| 0x09 | 0x10 | 0xXX | 0xXX | N | | 0xXX | 0xXX |

(3) The error response instruction structure is shown in Table 2.6.

### 2.5.4 Function code 0x10 instruction example

An example of writing 'pulse frequency' is shown in Table 2.18:

Table 2.18 Example of Setting "Pulse Frequency" to 2000Hz

|  |  |
| --- | --- |
| Data direction | Instruction |
| Host → Slave | 09 10 00 04 00 02 04 00 00 07 D0 DB 90 |
| Slave → Master | 09 10 00 04 00 02 01 41 |

Example explanation is shown in Table 2.19:

Table 2.19 Example explanation for writing "pulse frequency"

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Data direction | Slave address  (1 Byte) | Function code  (1 Byte) | Register Address  (2 Bytes) | Number of Registers  (2 Bytes) | Byte count  (1 Bytes) | DATA1 (2 Bytes) | DATA2 (2 Bytes) | CRC16 (2 Bytes) |
| Host → Slave | 0x09 | 0x10 | 0x0004 | 0x0002 | 0x04 | 0x0000 | 0x07D0 | 0xDB90 |
|  |  |  |  |  |  |  |  |  |
| Data direction | Slave address  (1 Byte) | Function code  (1 Byte) | Register Address  (2 Bytes) | Number of Registers  (2 Bytes) | CRC16 (2 Bytes) |  |  |  |
| Slave → Master | 0x09 | 0x10 | 0x0004 | 0x0002 | 0x0141 |  |  |  |

As shown in Table 2.19, the address of the "Pulse Frequency" register is 0x0004 (high 16 bits) and 0x0005 (low 16 bits), with 2 registers and 2 bytes=4 bytes. The hexadecimal number corresponding to 2000 is 0x000007D0.

## Usage Instructions

(1) Instruction interval: The baud rate is 19200bps by default, and the minimum instruction sending interval is 20ms.

(2) Parameter validity: After updating the process parameters through Modbus protocol, the screen interface will be updated immediately. If the equipment is in the welding working state, the parameters will take effect immediately.

(3) Parameter saving: Data is used as temporary parameters and is not saved after power failure. When high-frequency parameters change, it is not recommended to save parameters in real time, which will easily affect the data transmission efficiency and flash life. To save, send the save instruction, after saving, the screen process interface data will be updated.

## Examples of Common commands

(1) An example of setting the "scan width" command is shown in Table 2.20:

Table 2.20 Example of Scan Width Setting Instructions

|  |  |
| --- | --- |
| Set “Scan width” | |
| Value （mm） | Instruction |
| 1 | 09 06 00 01 00 0a 59 45 |
| 2 | 09 06 00 01 00 14 D9 4D |
| 3 | 09 06 00 01 00 1e 59 4A |
| 4 | 09 06 00 01 00 28 D9 5C |
| 5 | 09 06 00 01 00 32 58 97 |

(2) An example of setting the "peak power" command is shown in Table 2.21:

Table 2.21 Example of Peak Power Setting Command

|  |  |
| --- | --- |
| Set “Peak power” | |
| Value （W） | Instruction |
| 100 | 09 06 00 02 00 64 28 A9 |
| 200 | 09 06 00 02 00 c8 28 D4 |
| 300 | 09 06 00 02 01 2c 29 0F |
| 400 | 09 06 00 02 01 90 28 BE |
| 500 | 09 06 00 02 01 f4 29 55 |
| 600 | 09 06 00 02 02 58 29 D8 |

(3) An example of setting the "Duty cycle" command is shown in Table 2.22:

Table 2.22 Example of Duty cycle Setting Instructions

|  |  |
| --- | --- |
| Set “Duty cycle” | |
| Value （%） | Instruction |
| 50 | 09 06 00 03 00 32 F9 57 |
| 100 | 09 06 00 03 00 64 79 69 |

(4)An example of setting the "Frequency" command is shown in Table 2.23:

Table 2.23 Example of Frequency Setting Instructions

|  |  |
| --- | --- |
| Set “Frequency” | |
| Value （Hz） | Instruction |
| 1000 | 09 10 00 04 00 02 04 00 00 03 e8 D8 82 |
| 2000 | 09 10 00 04 00 02 04 00 00 07 d0 db 90 |

(5) An example of setting both "scan width" and "scan speed" commands is shown in Table 2.24:

Table 2.24 Example of Command for Setting Scan Width and Scan Speed

|  |  |  |
| --- | --- | --- |
| Set Scan Width and Scan Speed | | |
| Scan width （mm） | Scan Speed （mm/S） | Instruction |
| 1 | 100 | 09 10 00 00 00 02 04 03 e8 00 0a d9 b8 |
| 2 | 200 | 09 10 00 00 00 02 04 07 d0 00 14 d9 4d |
| 3 | 300 | 09 10 00 00 00 02 04 0b b8 00 1e db c6 |

Appendix: CRC algorithm

(1) Table lookup method:

uint16\_t modbus\_CRC16\_Table (uint8\_t \*nData, uint16\_t wLength)

{

static const uint16\_t wCRCTable[] = {

0X0000, 0XC0C1, 0XC181, 0X0140, 0XC301, 0X03C0, 0X0280, 0XC241,

0XC601, 0X06C0, 0X0780, 0XC741, 0X0500, 0XC5C1, 0XC481, 0X0440,

0XCC01, 0X0CC0, 0X0D80, 0XCD41, 0X0F00, 0XCFC1, 0XCE81, 0X0E40,

0X0A00, 0XCAC1, 0XCB81, 0X0B40, 0XC901, 0X09C0, 0X0880, 0XC841,

0XD801, 0X18C0, 0X1980, 0XD941, 0X1B00, 0XDBC1, 0XDA81, 0X1A40,

0X1E00, 0XDEC1, 0XDF81, 0X1F40, 0XDD01, 0X1DC0, 0X1C80, 0XDC41,

0X1400, 0XD4C1, 0XD581, 0X1540, 0XD701, 0X17C0, 0X1680, 0XD641,

0XD201, 0X12C0, 0X1380, 0XD341, 0X1100, 0XD1C1, 0XD081, 0X1040,

0XF001, 0X30C0, 0X3180, 0XF141, 0X3300, 0XF3C1, 0XF281, 0X3240,

0X3600, 0XF6C1, 0XF781, 0X3740, 0XF501, 0X35C0, 0X3480, 0XF441,

0X3C00, 0XFCC1, 0XFD81, 0X3D40, 0XFF01, 0X3FC0, 0X3E80, 0XFE41,

0XFA01, 0X3AC0, 0X3B80, 0XFB41, 0X3900, 0XF9C1, 0XF881, 0X3840,

0X2800, 0XE8C1, 0XE981, 0X2940, 0XEB01, 0X2BC0, 0X2A80, 0XEA41,

0XEE01, 0X2EC0, 0X2F80, 0XEF41, 0X2D00, 0XEDC1, 0XEC81, 0X2C40,

0XE401, 0X24C0, 0X2580, 0XE541, 0X2700, 0XE7C1, 0XE681, 0X2640,

0X2200, 0XE2C1, 0XE381, 0X2340, 0XE101, 0X21C0, 0X2080, 0XE041,

0XA001, 0X60C0, 0X6180, 0XA141, 0X6300, 0XA3C1, 0XA281, 0X6240,

0X6600, 0XA6C1, 0XA781, 0X6740, 0XA501, 0X65C0, 0X6480, 0XA441,

0X6C00, 0XACC1, 0XAD81, 0X6D40, 0XAF01, 0X6FC0, 0X6E80, 0XAE41,

0XAA01, 0X6AC0, 0X6B80, 0XAB41, 0X6900, 0XA9C1, 0XA881, 0X6840,

0X7800, 0XB8C1, 0XB981, 0X7940, 0XBB01, 0X7BC0, 0X7A80, 0XBA41,

0XBE01, 0X7EC0, 0X7F80, 0XBF41, 0X7D00, 0XBDC1, 0XBC81, 0X7C40,

0XB401, 0X74C0, 0X7580, 0XB541, 0X7700, 0XB7C1, 0XB681, 0X7640,

0X7200, 0XB2C1, 0XB381, 0X7340, 0XB101, 0X71C0, 0X7080, 0XB041,

0X5000, 0X90C1, 0X9181, 0X5140, 0X9301, 0X53C0, 0X5280, 0X9241,

0X9601, 0X56C0, 0X5780, 0X9741, 0X5500, 0X95C1, 0X9481, 0X5440,

0X9C01, 0X5CC0, 0X5D80, 0X9D41, 0X5F00, 0X9FC1, 0X9E81, 0X5E40,

0X5A00, 0X9AC1, 0X9B81, 0X5B40, 0X9901, 0X59C0, 0X5880, 0X9841,

0X8801, 0X48C0, 0X4980, 0X8941, 0X4B00, 0X8BC1, 0X8A81, 0X4A40,

0X4E00, 0X8EC1, 0X8F81, 0X4F40, 0X8D01, 0X4DC0, 0X4C80, 0X8C41,

0X4400, 0X84C1, 0X8581, 0X4540, 0X8701, 0X47C0, 0X4680, 0X8641,

0X8201, 0X42C0, 0X4380, 0X8341, 0X4100, 0X81C1, 0X8081, 0X4040 };

uint8\_t nTemp;

uint16\_t wCRCWord = 0xFFFF;

uint16\_t wCRCResult = 0;

while (wLength--)

{

nTemp = \*nData++ ^ wCRCWord;

wCRCWord >>= 8;

wCRCWord ^= wCRCTable[nTemp];

}

//Swap CRC high and low bytes

wCRCResult = wCRCWord>>8;

wCRCResult = wCRCResult | (wCRCWord << 8);

return wCRCResult;

}

(2) Direct calculation:

uint16\_t modbus\_CRC16\_Calculate(uint8\_t \*ndata, uint16\_t wLength)

{

uint16\_t wCRCWord = 0xFFFF;

uint16\_t polynomial = 0xA001;

uint16\_t wCRCResult = 0;

uint8\_t i,j;

for(i = 0;i < wLength;i++)

{

wCRCWord ^= ndata[i];

for(j = 0;j < 8;j++)

{

if((wCRCWord & 0x0001) != 0)

{

wCRCWord >>= 1;

wCRCWord ^= polynomial;

}

else

{

wCRCWord >>= 1;

}

}

}

//Swap CRC high and low bytes

wCRCResult = wCRCWord>>8;

wCRCResult = wCRCResult | (wCRCWord << 8);

return wCRCResult;

}