RS-485 Communication Protocol

1. Communication Protocol

1.1 Protocol overview

(1) Protocol type

This is the MODBUS RTU protocol. The protocol aims to stipulate that the data exchange between the terminal device (slave station) and the bus interface unit (master station) is implemented in the MODBUS RTU (Remote Terminal Unit) mode. The protocol uses asynchronous master-slave half-duplex communication. The communication is initiated by the master station, and the slave station makes a corresponding response after receiving the master station request. Communication response time <0.1s.

(2) Physical Layer

- Transmission Interface： RS-485
- MODBUS Address： 0～247（Single device: 06H）
- Baud rate：9600bps~38400bps（This device is 19200bps）
- Communication Media: shielded twisted pair

(3) DLL (data link layer)

- Transmission mode: asynchronous master-slave half-duplex
- Data frame format：One start bit, 8 data, no check bit, one stop bit
- Data packet format：

Address	Function	Data	CRC
8bits	8bits	n×8bits	16bits

The transmission sequence of the data packet is always the same address, function code, data and check code. Each data packet needs to be transmitted as a continuous bit stream. When the master data packet arrives at the slave station, the slave station that matches the address field in the data packet will receive the data. If there is no error, it executes the request in the data packet and sends the response data packet to the master station. The response data packet returned from the slave station contains the following content: the address of the slave station address, the request data generated by the function execution, and check code (CRC).

- Address field

The address field is composed of an eight-bit data at the beginning of the data packet. This data represents the address of the slave station specified by the master station, and each slave station address on the bus are unique. The effective address range of the slave is within 0 to 247. After the master station sends a data packet, only the slave

station with the same address as the master station's query will respond.(When the batteries are connected in series and parallel to the network, the battery will automatically assign the device address to prevent mutual interference when responding, the device address is 01H~10H. In the case of a single machine connected to the network, the default device address is 06H)

- Function

  The function field describes what kind of function the slave station performs. The following table explains the meaning of all function codes.

Code	Definition	Specific functions
04H	Read BMS data	Read the data from the BMS data list

- Data field

The data field contains the data required by the slave to perform a specific function or the data collected by the slave in response to the query of the master. The content of these data includes the starting address of the data field required for the query and the number of registers required for the query.

- CRC field

The check code is the 16-bit check data formed by the master station and the slave station when the CRC check transmits the data. Due to various interferences in the communication, the data transmitted in the communication may change. The CRC check can effectively ensure that the master station and the slave station will not respond to the distorted data during the transmission process, which improves the security and efficiency of the system.

1.2 Application layer function detail

Note: this device address is 06H.

(1) Read data command (Function Code 04H)

- Read data downstream frame format

Device communication address （0~247） 06H	Function code (04H)	High byte of Start address of data field	Low byte of start address of data field	Data field length high byte	Data field length low byte	CRC Check low byte	CRC Check high byte

- Read data response frame format

Device communication address （0~247） 06H	Function code (04H)	Data length byte	Data content	Data content	CRC Check low byte	CRC Check high byte

- Example

Read the total voltage data downstream frame：

Address 06H	04H	00H	17H	00H	01H	Low check	High check

The response frame：

Address 06H	04H	02H	Total voltage data high byte	Total voltage data low byte	Low check	High check

Send：06 04 00 17 00 01 80 79
Receive：06 04 02 04 BA 8F 83
Total voltage：04BAH, decimalize 1210　unit：0.1V

Read the downstream frame of system fault state data:

Address 06H	04H	00H	81H	00H	03H	Low check	High check

Reply frame:

Address 06H	04H	06H	Fault status 0 high Byte	Fault status 0 low byte	Fault status 1 high Byte	Fault status 1 low byte	Fault status 2 high Byte
Fault status 2 high Byte	Low check	High check					

Send：06 04 00 81 00 03 E1 94
Receive：06 04 06 00 00 00 00 00 00 46 A3
Fault status 0：0000H
Fault status 1：0000H
Fault status 2：0000H

(2) Reading Device Information

- Example

Address 06H	04H	00H	0BH	00H	10H	Low check	High check

Reply frame:

Address 06H	04H	20H	Software and hardware versions	Maximum charging voltage	Minimum discharge voltage	Maximum charging current	Maximum discharge current	Full charge capacity	Full discharge capacity
Current	Total voltage	RSOC	State of charge	Low check	High check				

Send：06 04 00 0b 00 10 81 b3
Receive：06 04 20 00 31 00 10 00 8C 00 70 05 DC 05 DC 00 03 18 F8 00 03 18 F8 00 00 00 00 04 BA 01 F4 00 01 8C E0 7E D3
Software and hardware versions："0x00310010:
Software Version: 0x0031 -> V0.03.1
Hardware Version: 0x0010 -> V0.01.0

1.3 CRC Check method

The redundant cyclic code (CRC) contains 2 bytes, that is, 16-bit binary. The CRC code is calculated by the sending device and placed at the end of the sent information. The receiving device recalculates the CRC code of the received information, and compares the calculated CRC code with the received one. If the two do not match, it indicates an error.

The calculation method of the CRC code is to preset 16-bit registers all to 1.Then gradually process each 8-bit data information. In the CRC code calculation, only 8-bit data is used; start bits and stop bits, if there is a parity check bit, the parity check bit is also included, and they are not involved in the CRC code calculation.

In the CRC code calculation, 8-bit data is XOR with the data of the register, the resulting result is shifted one byte to lower byte, filling in the highest bit with 0. Check the lowest bit again, if the lowest bit is 1, XOR the content of the register with the preset value. If the lowest bit is 0, no XOR operation is performed.

Repeat this process for 8 times. After the 8th shift, the next 8-bit data is XOR with the content of the current register again. Repeated this process 8 times as above. When all the data information is processed, the content of the last register is the CRC code value. When sending and receiving data in the CRC code, the low byte is first.

The steps to calculate the CRC code:

1. Preset the 16-bit register as hexadecimal FFFF (that is all 1), and call this register as CRC register.
2. XOR the first 8-bit data with the lower bits of the 16-bit CRC register, and put the result in the CRC register.
3. Move the content of the register one bit to the right (toward the lower bit), fill the highest bit with 0, and check the lowest bit.
4. If the lowest bit is 0: repeat step 3 (shift again); if the lowest bit is 1: the CRC register is XOR with the polynomial A001 (1010 0000 0000 0001).
5. Repeat steps 3 and 4 until the right shift for 8 times, so that the entire 8-bit data has been processed.
6. Repeat steps 2 to 5 for the next 8-bit data processing.
7. The final CRC register is the CRC code.

Adopted CRC parameter model	CRC-16/MODBUS
Polynomial formula	x16+x15+x2+1
Width	16
Polynomial	0x8005
Initial value	0xFFFF
Result XOR value	0x0000
Input value inversion	true
Output value inversion	true

CRC algorithm reference ：

```
static const uint8_t chCRCHTalbe[] =
{
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40
};
static const uint8_t chCRCLTalbe[] =
{
    0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7,
    0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E,
    0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9,
    0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
    0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
    0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32,
    0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D,
    0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
    0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF,
    0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
    0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1,
    0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
    0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB,
    0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA,
    0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
```

```
    0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
    0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97,
    0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E,
    0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89,
    0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83,
    0x41, 0x81, 0x80, 0x40
};
uint16_t calculate_crc16(uint8_t *crc_buf, const uint8_t crc_count)
{
    uint16_t wIndex;
    uint16_t chCRCHi = 0xFF;
    uint16_t chCRCLo = 0xFF;
    uint16_t i;
    for(i=0;i<crc_count;i++)
    {
        wIndex = chCRCLo ^   crc_buf[i];
        chCRCLo = chCRCHi ^ chCRCHTalbe[wIndex];
        chCRCHi = chCRCLTalbe[wIndex] ;
    }
    return ((chCRCHi << 8) | chCRCLo) ;
}
```

2. BMS Data List

Parameter name	Address	Data type	Length	Read-write	Data range	Unit
Software and hardware versions	000B	unsigned long int	4 bit	Only read	0~4294967295	/
Maximum charging voltage	000D	unsigned int	2 bit	Only read	0~65535	0.1V
Minimum discharge voltage	000E	unsigned int	2 bit	Only read	0~65535	0.1V
Maximum charging current	000F	unsigned int	2 bit	Only read	0~65535	0.1A
Maximum discharge current	0010	unsigned int	2 bit	Only read	0~65535	0.1A
Full charge capacity	0011	unsigned long int	4 bit	Only read	0~4294967295	mAH
Full discharge capacity	0013	unsigned long int	4 bit	Only read	0~4294967295	mAH
Current	0015	signed long int	4 bit	Only read	-2147483647~2147483647	mA

Total voltage	0017	unsigned int	2 bit	Only read	0~65535	0.01V
RSOC	0018	unsigned int	2 bit	Only read	0~65535	0.1%
State of charge	0019	unsigned long int	4 bit	Only read	0~4294967295	mAH
Fault state 0	0x81	unsigned int	2 bit	Only read	0~65535	BIT
Fault state 1	0x82	unsigned int	2 bit	Only read	0~65535	BIT
Fault state 2	0x83	unsigned int	2 bit	Only read	0~65535	BIT
BMS state 0	0x100	unsigned int	2 bit	Only read	0~65535	BIT

3. Fault state 0 bytes

Data definition： 0 -> Normal 1 -> Protection

Bit address	Definition
Bit0	Each cell over-discharge primary protection
Bit1	Each cell over-charge primary protection
Bit2	Total voltage over-discharge primary protection
Bit3	Total voltage over-charge primary protection
Bit4	Discharge overcurrent primary protection
Bit5	Charge overcurrent primary protection
Bit6	Discharge high temperature primary protection
Bit7	Discharge low temperature primary protection
Bit8	Charge high temperature primary protection
Bit9	Charge low temperature primary protection
Bit10	MOS high temperature primary protection
Bit11	Ambient high temperature primary protection
Bit12	Ambient low temperature primary protection
Bit13	Excessive temperature differentials primary protection
Bit14	Excessive voltage differentials primary protection
Bit15	None

4. Fault state 1 bytes

Data definition： 0 -> Normal 1 -> Alarm

Bit address	Definition

Bit0	None
Bit1	Short circuit
Bit2	None
Bit3	None
Bit4	Discharge overcurrent secondary protection
Bit5	Charge overcurrent secondary protection
Bit6	None
Bit7	None
Bit8	Charging is not allowed
Bit9	None
Bit10	Heating failure
Bit11	None
Bit12	None
Bit13	None
Bit14	None
Bit15	None

5. Fault state 2 bytes

Bit address	Definition	Data parsing
Bit0	None	
Bit1	None	
Bit2	NTC failure	0b：Normal 1b：Fault
Bit3	Cell failure	0b：Normal 1b：Fault
Bit4	Sampling failure	0b：Normal 1b：Fault
Bit5	None	
Bit6	Heating failure	0b：Normal 1b：Fault
Bit7	AFE failure	0b：Normal 1b：Fault
Bit8	Bluetooth communication failure	0b：Normal 1b：Fault
Bit9	RS485 communication failure	0b：Normal 1b：Fault
Bit10	CAN communication failure	0b：Normal 1b：Fault
Bit11	None	
Bit12	BMS hardware failure	0b：Normal 1b：Fault
Bit13	None	
Bit14	None	
Bit15	None	

6. BMS status byte

Data definition：0 -> Off 1 -> On

Bit address	Definition	Data parsing
Bit0	Charge MOS state	0b：Off 1b：On
Bit1	Discharge MOS state	0b：Off 1b：On
Bit2~Bit3	Charge and discharge state	00b：Off 01b：Charging 10b：Discharging 11b：Abnormal state
Bit4	None	
Bit5	None	
Bit6	Heating MOS state	
Bit7	None	
Bit8	None	
Bit9	None	
Bit10	None	
Bit11	None	
Bit12	None	
Bit13	Charge permission	
Bit14	Discharge permission	
Bit15		