MODBUS-RTU

Communication Protocol Manual

Ultrasonic Meters

Catalogue

1. Communication Definition

Startbit	1 bit
Data Bit	8bit
Parity Check Bit	No(Default) N/O/E
Stop Bit	1bit
Baud Rate	9600(Default) 600-19200bps

Table 1

2. Communication Mode and Protocol

The communication mode of the instrument and the external equipment is RS485. The MODBUS protocol-RTU mode was used.

3. Introduction of Protocol

MODBUS is an access/reply protocol that provides services specified by function codes. Used for host/slave communication between devices connected through different types of buses or networks. The host sends commands to access the slave, and after receiving the commands, the slave takes corresponding actions and responds to the host.

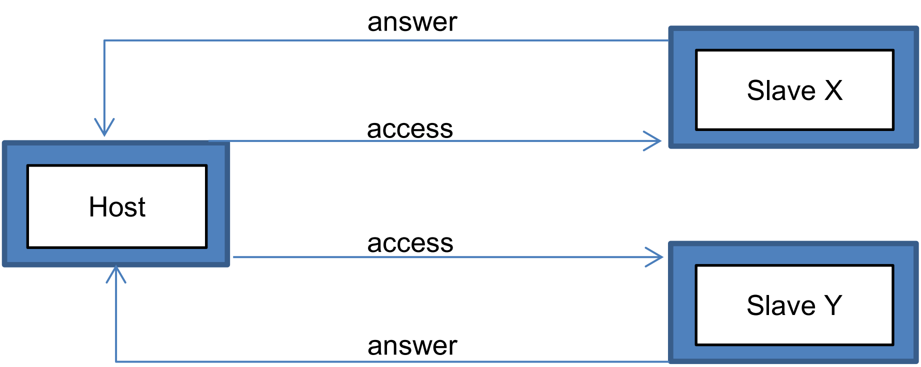

Figure 1

4. Description of Protocol

MODBUS has two transmission modes, RTU or ASCII. This instrument adopts RTU transmission mode.

Address	Function Code	Data	CRC Verification

Table 2

4.1 RTU Transmission Mode

4.1.1 RTU mode serial bit

Start bit	1	2	3	4	5	6	7	8	Check out	Stop bit

Table 3

4.1.2 The MODBUS-RTU Data Frame

Address	FC	Data	CRC Verification
1 Bytes	1 Bytes	Of The 0-252 Bytes	2 Bytes

Table 4

4.1.3 CRC Checkout

CRC inspection is an error detection domain based on the cyclic redundancy inspection algorithm. Perform this test regardless of whether there are data frames or parity checks. The CRC contains a 16 bit value composed of two 8-bit bytes for the clothing moth. As the final data attachment of the data frame, CRC first appends the bottom byte and then appends the high byte after calculation.

The CRC value is calculated by the host and recalculated by the slave after receiving the data frame. The calculation result is compared with the received CRC value. If it is not equal, it is an error.

4.2 MODBUS Communication model

Host **Slave**

Start access

(Function code Data request)

Initiate a response, perform an action

(Opcode Data response)

Receive response

4.3 Protocol Support

*The function code of this agreement indicates the current function to be executed.

Function code type	Function code (decimal)	Functional code description
Reading data	03	Read the menu data
Reading data	04	Read the variable (liquid level or other) data
Write data	10	Write the data to the menu register
Save parameter	12	The written data is saved to the EEPROM

Table 5

*Description of internal register addresses for commonly used variables (all register addresses below are starting addresses)

Meter type	Register (ten- scale system)	Register definition
Ultrasonic level meter	0	Level value
	2	Temperature scale
ultrasonic	0	Liquid level value
Open channel flow meter(Type I)	22	Instantaneous flow
	24	Hourly flow
	26	Cumulative flow
	28	Cumulative times
Ultrasonic Open channel meter (Type II)	0	Liquid level value
	28	Instantaneous flow
	30	hourly flow
	32	Cumulative flow
	34	Cumulative times

Table 6

4.4 The storage form of data in the internal register

The variable data stored in the internal register is an **IEEE754** standard single-precision floating-point number. Expressed in four-digit hexadecimal, it occupies four internal register addresses. For example, the data read back is **42 F9 80 00**.

The internal storage form is :

Starting address+3	Starting address+2	Starting address+1	Register start address
42	F9	80	00

Table 7

5. Example

5.1 The master reads the liquid level display value data command of the slave as the format

Sample: 01 04 00 00 00 02 71 CB

Serial Address	Function Code	Register Address	Read Data Length	CRC Check Code
01	04	00 00	00 02	71 Cb

Table 8

The read data length is two words, and one word returns two bytes.

The command returned by the slave is in the format

Sample: 01 04 04 40 A0 00 00 EE 66

Serial Address	Function Code	Return Data Length	Read Data Length	CRC Check Code
01	04	04	40 A0 00 00	EE 66

Table 9

5.2 Master writes data command format to slave

Sample: 01 10 00 00 00 02 04 40 A0 00 00 E6 4D

Serial Address	Function Code	Register Address	Register Length	Write Data Length	Write Data	CRC Check Code
01	10	00 00	00 02	04	40 A0 00 00	71 Cb

Table 10

5.3The master asks the slave to save the data into the EEPROM command

01 12 00 A0 00 02 04 00 00 00 A1 99 A5

This is a special command, return after completion **01 13 00 00 00 00 00 09 63**

6. Cyclic Redundancy (Crc16) Check Method

The cyclic redundancy check CRC area is 2 bytes, containing a 16-bit binary data. The sending device calculates the CRC value and attaches the calculated value to the information. When the receiving device receives the information, it recalculates the CRC value and compares the calculated value with the received actual value in the CRC area. If the two are not the same, an error is generated. At the beginning of CRC, set all 16 bits of the register to "1", and then put the data of two adjacent 8-bit bytes into the current register. Only the 8-bit data of each character is used to generate CRC, the start bit , stop bits and parity bits are not added to the CRC.

During CRC generation, each 8-bit data is XORed with the value in the register, the result is shifted one bit to the right (towards the LSB direction), and the MSB is filled with "0", and the LSB is detected. If the LSB is "1", then the Exclusive OR of preset fixed value, if LSB is "0", no exclusive OR operation will be performed.

Repeat the above process until shifting 8 times. After the 8th shift is completed, the next 8-bit data is XORed with the current value of the register. After all information is processed, the final value in the register is the CRC value.

The process of generating CRC:

1. Set the 16-bit CRC register to FFFFH.

2. XOR the first 8-bit data with the lower 8 bits of the CRC register, and put the result into the CRC register.

3. The CRC register is shifted one bit to the right, the MSB is filled with zero, and the LSB is checked.

4. (If LSB is 0): repeat 3, and then move right one bit. (If LSB is 1): XOR operation between CRC register and A001H.

5. Repeat steps 3 and 4 until 8 shifts are completed, and the processing of 8-bit bytes is completed

6. Repeat steps 2 to 5 to process the next 8-bit data until all bytes are processed.

7. The final value of the CRC register is the CRC value.

8. When putting the CRC value into the message, the high 8 bits and the low 8 bits should be placed separately. When sending the 16-bit CRC value in the message, send the low 8 bits first, and then send the high 8 bits

7.Floating Point Data Format

7.1 Brief analysis of IEEE754 standard single-precision floating-point numbers

IEEE754 standard single-precision floating-point numbers are composed of 1 sign bit + 8-bit exponent code + 23-bit mantissa, and are represented by four hexadecimal numbers. For example, 124.75 is represented as 42 F9 80 00 in hexadecimal. The calculation method is:

24.75 converted to binary: 1111100.11

Expressed in scientific notation as: 1.11110011*2^6

The order code is 6+127=133, and 0 is used for positive and 1 for negative. So the binary number of 124.75 is:

0 10000101111100110000000000000000B= 42F98000H

7.2 Convert 4 bytes to floating point numbers based on C language

```
Union                                          // community
{
   Float          testData_float;              //floating point 4 bytes
   Unsigned       char       testArray[4];     //value
}
TData;
```

Note: In the union, floating-point numbers and four-byte character groups share a storage space.

Detailed explanation: Take the liquid level value of the ultrasonic liquid level gauge as an example, read back the data bit 42 F9 80 00, and convert the floating point number to 124.75. The internal register storage format is as follows:

Starting address+3	Starting address+2	Starting address+1	Register start address
03	02	01	00
42	F9	80	00

Table 11

Program:

```
Folat Tempfloat;
TData.testArray[3] = 0x42;                    // input high byte
TData.testArray[2] = 0xF9;
TData.testArray[1] = 0x80;
TData.testArray[0] = 0x00;                    // input low byte
Tempfloat = testData float;                   // get floating point number
```