RS485 communication protocol

Ver3.0

1, an overview of the

The communication protocol describes in detail the M702 input and output commands, information and data for use and development by third parties.

1.2 Physical Interfaces:

The main communication port of the upper computer is connected with the standard serial RS-485 communication port.

The information transmission mode is asynchronous. The start bit is 1 bit, the data bit is 8 bits, and the stop bit is 1 bit.

2. Details of M701 communication protocol

1)All loop communication shall follow the master/slave mode. In this way, information and data are passed between a single master station and a slave station (monitoring equipment).

2) Broadcast mode is not supported.

3) Under no circumstances can communication be started from a slave station.

4) If the master station or any slave station receives a package containing an unknown command, the package will be ignored and the receiving station will not respond.

2.2 Returned Data frame structure description

Each data frame consists of :(RTU mode)
address
Function code
Number of data
Data 1
.....
Data n
CRC 16-bit check 1

3. Transmission format

(1) Format of command packets
Host sends read data command:

address	Function code	The start address of data is high	Data start address low value	The number of returned data is high	Low number of returned data	Low number of returned data
××	03	00	02	00	07	Low in the first

(Currently, only all data can be read. Seven data values can be read starting from address 0002.) There are only seven data, corresponding to seven addresses, whose address and data high levels are not processed in the module. The default value is 0, indicating the start and individual high values of data on the host. 0X00 is recommended

0x0002	1	CO2 concentration
0x0004	1	Formaldehyde concentration
0x0006	1	TVOC concentration
0x0008	1	PM2.5 concentrations
0x000A	1	PM10 concentration
0x000C	1	Temperature value
0x000E	1	Humidity value

The start address must be the address listed above. If the number of data is exceeded, no response is given. For example, if the start address is 0X000C, only 1 or 2 data can be returned. If three data are returned, no response is given. Similarly, if the address 0X000E is used, the number of returned data can only be 1. If the number of returned data is 0, no response is received.

From the machine address	Function code	The number of data	Data N	Data N
XX	03	xx	xx	xxxx

The byte length indicates only the data length

N0	N1	N2	N3	N4	N5	N6	N7	N8
CO2 high	CO2 low	formaldehyde high	formaldehyde low	TVOC high	TVOC low	PM2.5 high	PM2.5 low	PM10 high

N9	N10	N11	N12	N13
PM10 low	The temperature high	The temperature low	humidity high	humidity low

Host sends read address command:

address	Function code	The start address of data is high	Data start address low value	High number of data	Low number of data	CRC16 bit check
00	02	00	00	00	02	Xxxx Low in the first

Slave return address:

address	Function code	bytes	Address high	Address the low	CRC16 bit check
00	02	02	00	xx	Xxxx Low in the first

4. Host data sampling frequency:

When reading data from the t/h sensor, the upper computer reads data at an interval of at least 500ms (1s is recommended)

5. Function code

03: Reads data
02: Reads the address

6. Command examples:

Serial port Settings: asynchronous communication, start bit 1, data bit 8, no check, stop bit 1
The default data transfer rate is 9600b/s

The host computer sends: 01 03 00 02 00 07 CRCL，CRCH
(Read 7 data from address 01 from 00 02, slave machine does not check check code)

M701 returns:

0 x01, 0 x01, 0 x07, CO2H, CO2L, formaldehyde H, L, formaldehyde TVOCH, TVOCL, PM2.5 H, PM2.5, L PM10H, PM10L, temperature H, L, temperature, humidity, H, L, humidity CRCL, CRCH

Case 1:

TX: 01 03 00 02 00 07 CRCL，CRCH

RX: 01 03 07 01 E2 00 05 00 24 00 2D 00 00 FF 03 11 CRCL，CRCH

CO2 concentration = CO2H x256 + CO2L PPM (BYTE3 x256 + BYTE4)
Formaldehyde concentration = formaldehyde H x256 + formaldehyde UG (BYTE5 x256 + BYTE6)
TVOC = TVOCH x256 ug (BYTE7 x256 + BYTE8)
Ug (BYTE9 X256 + BYTE10)
Ug (BYTE11 X256 + BYTE12)
Temperature = ((BYTE13)X256 + (BYTE14)) % 10

Above CO2=482, formaldehyde =5,TVOC=36,PM2.5=45,PM10=56, temperature =30.5, humidity =64.6

Example 2:

TX:01 03 00 0C 00 02 CRCL，CRCH (10:43:52:001)(Reads two data starting from 000C at address 01.) Two data starting from 0x000C, corresponding to temperature and humidity.

RX:01 03 02 01 27 02 45 03 57 (10:43:52:159)

The temperature 0X0127 corresponds to base 295, indicating 29.5 ° C

The returned humidity value 0X0245 corresponds to base 581, indicating that the humidity is 58.1%RH

Note:

The temperature and humidity data returned from the machine are expressed in two bytes, with the highest value in the front and the lowest value in the back

The returned data range is -32768- 32767. The actual temperature and humidity data needs to be divided by 10

Such as:

Return the hexadecimal humidity value 0X0311, which corresponds to 785 in decimal notation, indicating that the humidity is 78.5%RH

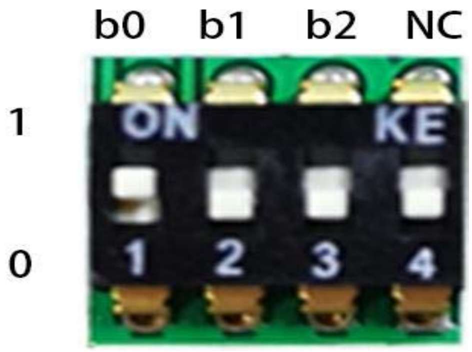
Returns the hexadecimal temperature value 0X00FF, which corresponds to 255 decimal notation, indicating that the temperature is 25.5 ° C

Returns the hexadecimal temperature data 0X8064, with the highest bit 1 indicating negative, corresponding to decimal -100 indicating temperature -10.0 ° C

Read address value:

The host computer sends:00 02 00 00 00 01 CRCL，CRCH
Transmitter returns: 00 02 02 00, address L, CRCL, CRCH

7. Address setting description



Dip switch a total of 4 bits, use 3 bits, the fourth bit can be ignored, the microcontroller does not read.
Pins 1,2, and 3 correspond to B0, B1,b2 respectively. As shown in the figure above, the upward dip switch (ON direction) is high, and the downward dip switch is low.
Address code = (b0 | b1 & lt; < 1 |b2< < 2) (Do not use 00 if possible)

The address used by the module may be different from that read by the command. The address used by the module may be read only during power-on. If the address is changed after power-on, the address used by the module will not change. It will change only after being powered on again.
Sending commands to read address codes is the real-time response of the current DIP switch. If the address is not changed after power-on, the address used by the module is the same as the address read by the command. If the address is changed, the address is changed when the command is read. The module uses the power-on address.

b2	b1	b0	ADDR
0	0	0	0x00
0	0	1	0x01
0	1	0	0x02
0	1	1	0X03
1	0	0	0X04
1	0	1	0X05
1	1	0	0X06
1	1	1	0X07

CRC check reference:

```
Static char auchCRCHi[] = static char auchCRCHi[] ={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 char auchCRCLo[] = static char auchCRCLo[] ={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};
```

CRC function calculation method:

1. Preset a 16-bit register to hexadecimal FFFF (that is, all 1); Call this register a CRC register;

2. The first 8-bit binary data (i.e. the first byte of a communication frame) is different or from the lower 8 bits of a 16-bit CRC register, and the result is placed in the CRC register;

3. Move the contents of the CRC register one right (toward the low) and fill the highest bit with a 0, and check the move out bit after the right move;

4. If move out is 0: repeat step 3 (move right again one bit); If the offbit is 1: the CRC register performs xor with polynomial A001 (1010 0000 0000 0001);

5. Repeat steps 3 and 4 until the right shift is 8 times so that the entire 8-bit data has been processed;

6. Repeat Step 2 to Step 5 to process the next byte of the communication information frame.

7. After all bytes of the communication information frame are calculated according to the above steps, the high and low bytes of the 16-bit CRC register obtained are exchanged;

8. The final CRC register content is: CRC code.

CRC function routines: //*pushMsg is the array pointer variable to be checked, usDataLen is the number of data variables to be checked

```
void CRC16(char *pushMsg,unsigned short usDataLen) {
            Char uchCRCHi = 0 XFF;    // High CRC byte initialization
            Char uchCRCLo = 0 XFF;     // Low CRC byte initialization
            Unsigned int uIndex;                // Index in CRC loop
            Unsigned int uIndex;                // Index in CRC loop
            uIndex=uchCRCHi^*pushMsg++;          / / CRC calculation
                    UchCRCHi = uchCRCLo ^ auchCRCHi [uIndex];
                          UchCRCLo = auchCRCLo [uIndex];
         * pushMsg++ = uchCRCHi;       // Check data is higher than that
   * pushMsg = uchCRCLo;     // The low value of the parity data is the first
```