RS485 Modbus Communication Protocol V1.5

1、Preview

This protocol define the communication between the monitoring devices and the upper machine in the communication interface, data definition, exchange methods and so on. In this document, DCE replace the monitoring device and DTE replace the upper machine for short, also customer can think DCE as our device, DTE as customer's device.

2、Data Transfer

2.1Transfer method

The communication interface between DCE and DTE is RS485，10bits asynchronous mode，1bit start symbol, 8bit data, 1bit stop symbol， no parity， baud rate can be chosen（default 9600bps）.

DTE and DCE use ask-answer mode to communicate, it means that DTE must ask DCE first, then DCE will answer DTE. The length of each frame of data must be no more than 255 bytes。

If DCE receive the right address, protocol type, protocol data and check code, DCE will respond normal protocol in 500 ms.

If DCE receive the wrong address or check code, DCE will ignore.

If DCE receive the right address, check code ,but wrong protocol type or wrong protocol data, DCE will respond abnormal protocol in 500 ms.

2.2 Protocol type

DTE → DCE

protocol type	code
Read Register	03H

DCE → DTE

protocol type	Normal/Abnormal code
Read Register	03H/83H

（Read Register）

DTE → DCE

protoco type	code
Write Register	10H

DCE → DTE

Data type	Normal/Abnormal code
Write Register	10H/90H

（Write Register）

2.3 Protocol format

DCE address	Protocol type	Data	Check code

2.3.1DCE address

DCE address is 1—255, 0 is broadcast address， DCE will not response when receive broadcast address。

2.3.2 Protocol type

2.3.2.1 Read Register

DTE ⟶ DCE

Byte1	DCE address
Byte2	Protocol type(03H)
Byte3	Start register address(high byte)
Byte4	Start register address(low byte)
Byte5	Register number N(high byte)
Byte6	Register number N(low byte)
Byte7	Check code(low byte)
Byte8	Check code(high byte)

DCE ⟶ DTE

Byte1	DCE address
Byte2	Protocol type(03H)
Byte3	Length of data (N×2)
Byte4	Register1 data(high byte)
Byte5	Register1 data(low byte)
Byte6	Register2 data(high byte)
Byte7	Register2 data(low byte)
	◦ ◦ ◦ ◦ ◦ ◦ ◦ ◦
	◦ ◦ ◦ ◦ ◦ ◦ ◦ ◦
N*2+2	Register N data (high byte)
N*2+3	Register N data (low byte)
N*2+4	Check code(low byte)
N*2+5	Check code(high byte)

2.3.2.2 Write Register, not support yet

DTE ⟶ DCE

Byte1	DCE address
Byte2	Protocol type(10H)
Byte3	Start register address(high byte)
Byte4	Start register address(low byte)
Byte5	Register number N(high byte)
Byte6	Register number N(low byte)
Byte7	Length of data(N×2)
Byte8	Register1 data(high byte)
Byte9	Register1 data(low byte)
	◦ ◦ ◦ ◦ ◦ ◦ ◦ ◦
	◦ ◦ ◦ ◦ ◦ ◦ ◦ ◦
N*2+6	Register N data(high byte)
N*2+7	Register N data(low byte)
N*2+8	Check code (low byte)
N*2+9	Check code (high byte)

DCE ⟶ DTE

DCE address
Protocol type(10H)
Start register address(high byte)
Start register address(low byte)
Length of data(N×2)
Check code (low byte)
Check code (high byte)

2. 3. 2. 3 Abnormal data

If DCE receive the right address and check code, but the protocol type or the data is wrong, it will response abnormal data. Abnormal data's protocol type will set the bit7 of DTE's protocol type to 1. For example, if DTE send 03H protocol type, DCE will response 83H, if DTE send 10H protocol type, DCE will response 90H.

Byte1	DCE address
Byte2	Protocol type(1*******)
Byte3	Abnormal type code
Byte4	Check code(low byte)
Byte5	Check code(high byte)

Abnormal type code table

Type code	Definition
01	Invalid protocol type
02	Invalid register address(include invalid length of data)
03	Invalid written data
04	Device is busy

2.4 Check code

DCE address	Protocol type	Data	Check code

Check code calculate range

Here is the check code calculate function written by C language below:

```
const unsigned 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 };

   const unsigned    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 };

unsigned int CRC16(unsigned char *puchMsg, unsigned int DataLen)
{
      unsigned char uchCRCHi = 0xFF;
      unsigned char uchCRCLo = 0xFF;
      unsigned char *pt;
      unsigned int Len;
      unsigned char uIndex;

      pt = puchMsg;
      Len = DataLen;

      while(Len--)
      {
            uIndex = uchCRCLo ^ (*pt++);
            uchCRCLo = uchCRCHi ^ (auchCRCHi[uIndex]);
            uchCRCHi = auchCRCLo[uIndex];
      }

      return (uchCRCHi << 8 | uchCRCLo);
}
```

3 Register Address Dispatch

Register address range is 1～9999,each address have two bytes value。

Register address list：

Register address	Register value	Actual value	Unit	Remark
Temperature data				
31	Channel1Temperature data	* 0.1	℃	Bit15: temperature is positive(+) or negative(-) 0-positive, 1-negative; Bit14-0: temperature data; Actual value ＝ (data&0x7fff)*0.1 ; When data is 0xffff means this channel is offline or have no temperature;
32	Channel2Temperature data	* 0.1	℃	Ditto
33	Channel3Temperature data	* 0.1	℃	Ditto
34	Channel4Temperature data	* 0.1	℃	Ditto
35	Channel5Temperature data	* 0.1	℃	Ditto
。。。	。。。	。。。	。。。	。。。
。。。	。。。	。。。	。。。	。。。
。。。	。。。	。。。	。。。	。。。
126	Channel96Temperature data	* 0.1	℃	Ditto
127	Channel97Temperature data	* 0.1	℃	Ditto
128	Channel98Temperature data	* 0.1	℃	Ditto
129	Channel99Temperature data	* 0.1	℃	Ditto
130	Channel100Temperature	* 0.1	℃	Ditto

	data			
Humidity data				
131	Channel1Humidity data	* 1 / *0.1	%	The unit is 0.1% or 1% depending on the sensor data type; When data is 0xffff means this channel is offline; When unit is 1% and the data is 0x00ff means this channel have no humidity; When unit is 0.1% and the data is 0x8000 means this channel have no humidity;
132	Channel2Humidity data	* 1 / *0.1	%	Ditto
133	Channel3Humidity data	* 1 / *0.1	%	Ditto
134	Channel4Humidity data	* 1 / *0.1	%	Ditto
135	Channel5Humidity data	* 1 / *0.1	%	Ditto
。。。	。。。	。。。	。。。	。。。
。。。	。。。	。。。	。。。	。。。
。。。	。。。	。。。	。。。	。。。
226	Channel96Humidity data	* 1 / *0.1	%	Ditto
227	Channel97Humidity data	* 1 / *0.1	%	Ditto
228	Channel98Humidity data	* 1 / *0.1	%	Ditto
229	Channel99Humidity data	* 1 / *0.1	%	Ditto
230	Channel100Humidity data	* 1 / *0.1	%	Ditto
Pressure sensor temperature data				
231	Channel1Temperature data	* 0.1	℃	Bit15: temperature is positive(+) or negative(-) 0-positive, 1-negative; Bit14-0: temperature data;

				Actual value = (data&0x7fff)*0.1 ; When data is 0xffff means this channel is offline or have no temperature;
232	Channel2Temperature data	* 0.1	℃	Ditto
233	Channel3Temperature data	* 0.1	℃	Ditto
。。。	。。。	。。。	。。。	。。。
。。。	。。。	。。。	。。。	。。。
329	Channel99Temperature data	* 0.1	℃	Ditto
330	Channel100Temperature data	* 0.1	℃	Ditto
Pressure sensor pressure data				
331	Channel1 Pressure data	*1	kPa	When data is 0xffff means this channel is offline or have no pressure;
332	Channel2 Pressure data	*1	kPa	Ditto
333	Channel3 Pressure data	*1	kPa	Ditto
。。。	。。。	。。。	。。。	。。。
。。。	。。。	。。。	。。。	。。。
429	Channel99 Pressure data	*1	kPa	Ditto
430	Channel100 Pressure data	*1	kPa	Ditto
Voltage data (Only V2.19 and above is supported)				
431	Channel1Voltage data	* 1	mV	When data is 0xffff means this channel is offline or have no voltage;
432	Channel2 Voltage data	* 1	mV	Ditto
433	Channel3 Voltage data	* 1	mV	Ditto
。。。	。。。	。。。	。。。	。。。
。。。	。。。	。。。	。。。	。。。
529	Channel99 Voltage data	* 1	mV	Ditto
530	Channel100 Voltage ata	* 1	mV	Ditto
RSSI data (Only V2.19 and above is supported)				
531	Channel1 RSSI data	* 1	dBm	When data is 0xffff

				means this channel is offline or have no data;
532	Channel2 RSSI data	* 1	dBm	Ditto
533	Channel3 RSSI data	* 1	dBm	Ditto
◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦
◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦
629	Channel99 RSSI data	* 1	dBm	Ditto
630	Channel100 RSSI ata	* 1	dBm	Ditto
double temperature of the temperature 2 (Only V2.23 and above is supported)				
631	Channel1Temperature data	* 0.1	℃	Bit15: temperature is positive(+) or negative(-) 0-positive, 1-negative; Bit14-0: temperature data; Actual value = (data&0x7fff)*0.1 ; When data is 0xffff means this channel is offline or have no temperature;
632	Channel2Temperature data	* 0.1	℃	Ditto
633	Channel3Temperature data	* 0.1	℃	Ditto
634	Channel4Temperature data	* 0.1	℃	Ditto
635	Channel5Temperature data	* 0.1	℃	Ditto
◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦
◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦
◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦	◦ ◦ ◦
726	Channel96Temperature data	* 0.1	℃	Ditto
727	Channel97Temperature data	* 0.1	℃	Ditto
728	Channel98Temperature data	* 0.1	℃	Ditto
729	Channel99Temperature data	* 0.1	℃	Ditto

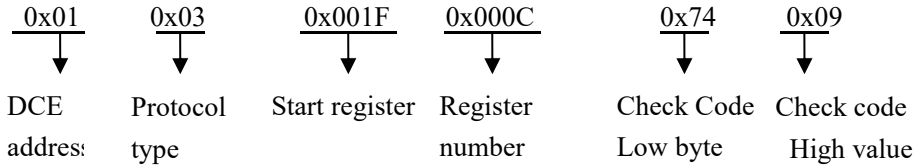
730	Channel100Temperature data	* 0.1	℃	Ditto

Notes:V1.07 and below version can just support 50 channel temperature and humidity.

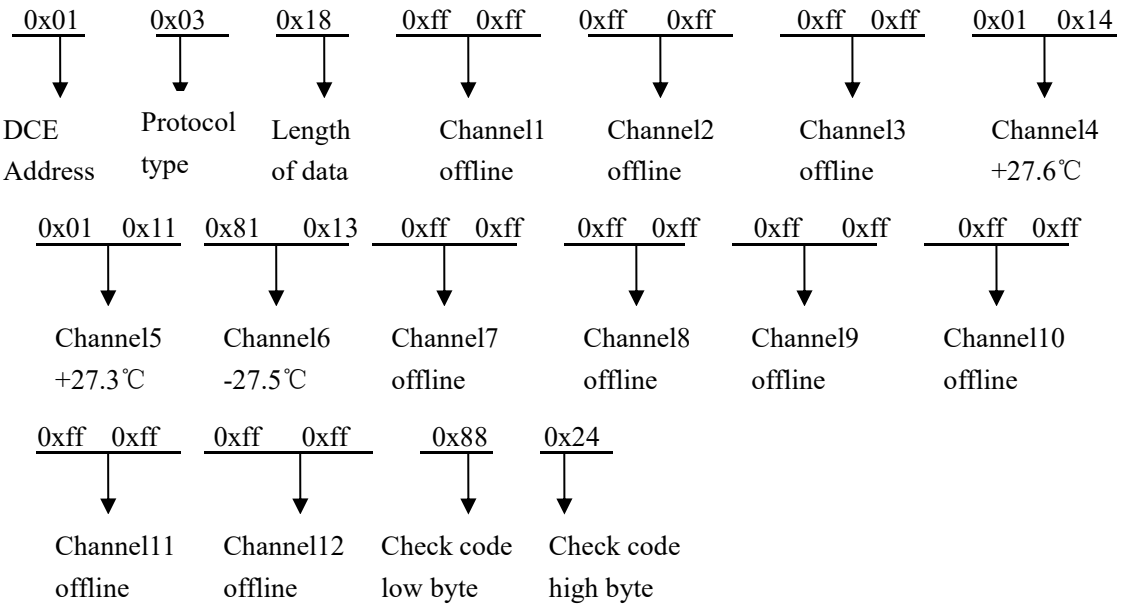
Appendix 1：Data examples

Read 12 channel temperature

DTE send:

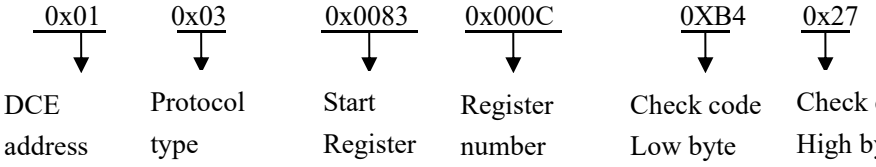


DCE response:

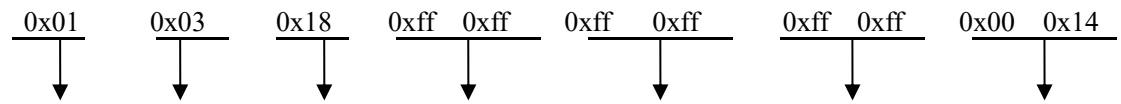


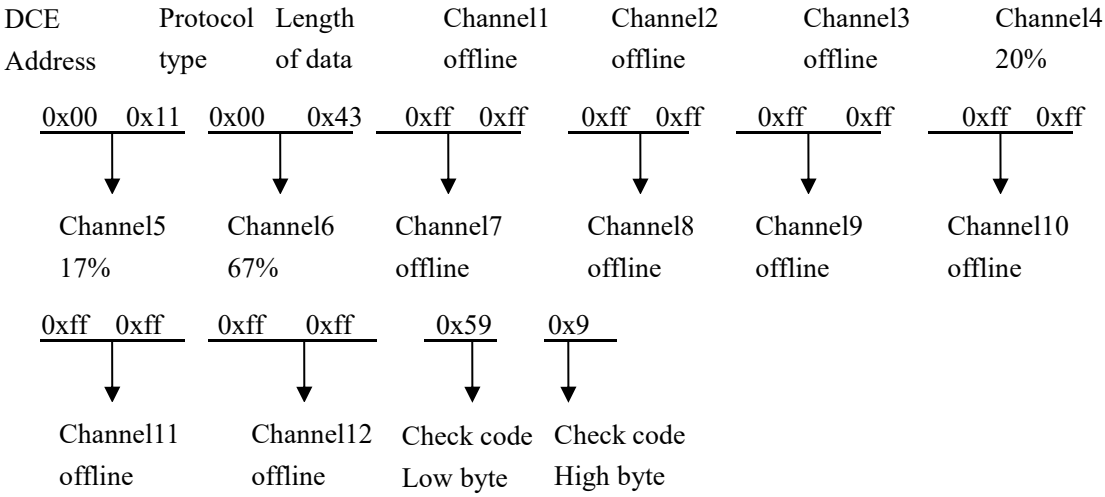
Read 12 channel humidity

DTE send:



DCE response:




DCE Address
Protocol type
Length of data
Channel1 offline
Channel2 offline
Channel3 offline
Channel4 20%
0x00 0x11
0x00 0x43
0xff 0xff
0xff 0xff
0xff 0xff
0xff 0xff
Channel5 17%
Channel6 67%
Channel7 offline
Channel8 offline
Channel9 offline
Channel10 offline
0xff 0xff
0xff 0xff
0x59
0x9
Channel11 offline
Channel12 offline
Check code Low byte
Check code High byte