2100/2000F/1807/2001F/2000H
Modbus 协议

1、主设备发送命令

格式	从设备地址(*)	功能码	数据起始寄存器（*）		数据寄存器数（*）		CRC 校验（*）	
字节长度	1 字节	1 字节	高字节 1 字节	低字节 1 字节	高字节 1 字节	低字节 1 字节	低字节 1 字节	高字节 1 字节
示例	01H	03H	00H	01H	00H	06H	94H	08H

注：带（*）的项可根据实际情况修改

从设备地址：即设备的设备号。

1.1、功能码：

0X03：①返回节点温度（MB节点号1--997）与传感器ID号没有直接关系，要通过软件把传感器ID号下载到产品，一台最多存储240个节点值。

②此功能码传输节点值时可传输存储区域的最大值、最小值、全部值、时基，具体上传模式取决于“无线参数设置”→“MODBUS传输模式”的设定值。注：只有2000F V3.4/2100 V3.4版本有此功能。

③时基（带节点号与时间）：数据长度2字节、节点号4字节、温度值2字节、时间4字节，此功能所有高字节在前、低字节在后

0X04：返回环境温度（设备编号 1--90）只有 2000F 带蓝牙、2001F 产品有此功能。

0X05：返回环境湿度（设备编号 1--90）只有 2000F 带蓝牙、2001F 产品有此功能。

0X09：①母线槽传感器环境温湿度（MB 节点号 1--997），每组（一屏是一组）温湿度 2 个字节，第一个环境温度，第二个环境湿度，只有 2001F 产品有此功能。

②无线环境温湿度传感器，按节点号显示，每个传感器共 4 个字节，前 2 个字节环境温度（高字节在前、低字节在后），后 2 个字节环境湿度（高字节在前、低字节在后），一次最多返回 80(0X50)个环境温湿度,2000H 产品有此功能

1.2、数据起始寄存器：对应 MB 节点起始地址（填满两个字节，最小 01，00 无效不返回）（高字节在前，低字节在后）。

1.3、数据寄存器数：用上。（填满两个字节，最大 120 个节点，十六进制 0X78）（高字节在前，低字节在后）。

1.4、CRC 校验：从设备地址到数据寄存器数的校验和（低字节在前）。

1.5、串口参数：波特率：2400～9600；数据位：8；校验位：无；停止位：1。

2、从设备返回

地址（设备地址）	功能码（3）	数据数量（数据长度）	节点温度（高字节在前，低字节在后，类型 signed short）	。。。。。。。signed short	节点温度（高字节在前，低字节在后，类型）signed short	CRC 校验（低字节在前）
8 位/1	8 位/1	8 位/1 字节	16 位/2 字节		16 位/2 字	16 位/2 字节

字节	字节			。。。。。。	节	

注：

2.1、设备地址：设备的设备号。

2.2、功能码：即主设备发送报文中的功能码。

2.3、数据长度：返回报文一包最长为 120 个字节（0X78）。大于 120 个字节时无数据返回。

2.4、节点：从发送报文 MB 节点起始地址开始，依次递增的 N 个节点值（如无数据返回为 00，第一包数据接收后 1 小时之内再无数据上传，产品显示通讯断，对应节点号值填写 00）。

如果报文解析错误，则不返回任何数据。等待下次的数据请求。

2.5、负数表示方法，返回数据换算为二进制并取反再加 1，例如-22.5 度返回十六进制为 FF 1F，二进制为 1111 1111 0001 1111 取反后为 0000 0000 1110 0000 再加 1 为：0000 0000 1110 0001，最终十进制为 225，除 10 后为 22.5。

2.6、上传的温度值全部是放大 10 倍后的数据，接收后要除以 10 才是真实温度值，例如 00 F1 换成 10 进制是 241，温度值即是 24.1（241/10）。

2.7、环境温度、湿度值上传时放大 10 倍，后台接收后要做除以 10 才是真实的环境温湿度。

3、读取节点温度举例

3.1、主机下发报文：

01 03 00 01 00 50 14 36

3.2、01 为设备号（通讯地址）

03 为功能能码

00 01 MB节点起始地址为1

00 50 读取个数为80个节点

即此报文是获取 1 –80的节点温度

3.3、公司测试返回数据（举例）

01 03 A0 02 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 02 2C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 0E 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 02 01 00 00 00 00 00 00 02 22 1D 8E

3.4、 01 为设备号（通讯地址）

03为功能码

A0为数据长度（160）即80个节点

02 31 即为第一个节点数据（56.1℃）对应产品上第1个传感器值

00 00 为第2个传感器温度值0

…….两个字节对应一个传感器，以此类推。

02 22 为第80个传感器温度值54.6℃

1D 8E为CRC

3.5、读取最大值最小值举例说明：

上位机下发报文：01 03 00 01 00 01 D5 CA

正确返回报文为：01 03 02 00 E1 78 0C

数据长度只能是一位即 01，如果大于 1 时返回错误代码（01 83 02 C0 F1）。

4、大于 240 个点使用说明

4.1、MB 节点起始地址既为“数据起始寄存器”的开始值

4.2、一台产品最多读取 240 个节点

4.3、例如设置菜单里无线参数设置→①MB 节点起始地址为 1，既读取时“数据起始寄存器”要写为 01，回传时第一个温度值就对应液晶屏第一屏的第一个传感器号；②如果 MB 节点起始地址为 10，既读取时“数据起始寄存器”要写为 0A（十六进制），回传时第一个温度值还是对应液晶屏第一屏的第一个传感器号。

4.4、如何读取到一台产品的 240 个节点，MB 起始地址为 1，分成两个数据包读取（01 03 00 01 00 78 14 28 和 01 03 00 79　00 77 D4 35（MB 起地址不用更改为 121）），如果配置的节点没有大于 120 个，那用第二条读取时就返回错误代码（01 83 02 C0 F1）。

4.5、如果整个变电站的传感器数量大于 240 个节点，那么就需要 2 台 2000F，把第二台 2000F 的本机通讯地址设置为 2，无线参数设置→MB 节点起始地址设置为 241，既读取时“数据起始寄存器”要写为 F1（十六进制）；大于 480 个节点读取方法同上依此类推。

5、环境温度

读取方法与节点温度相同，环境温度 5 分钟上报一次，级联时存储到最后一台产品的寄存器里（一台产品时存储到本机数据寄存器），最多存储 90 台设备的环境温度（注意每台产品的“本机通讯地址”请设置为不同的编号（内存地址位置按此值存储）），返回值除 10 是温度值。例如设备的本机通讯地址为 12，用 MODBUS 读取环境温度的代码为：0C 04 00 0C 00 01 F0 D4;

6、环境湿度

读取方法与节点温度相同，环境湿度 5 分钟上报一次，级联时存储到最后一台产品的寄存器里（一台产品时存储到本机数据寄存器），最多存储 90 台设备的环境湿度（注意每台产品的“本机通讯地址”请设置为不同的编号（内存地址位置按此值存储）），返回值即是湿度值。例如设备的本机通讯地址为 20，用 MODBUS 读取环境温度的代码为：14 05 00 14 00 01 4E CB;

7、产品显示位置对应号

以每屏显示 6 个点为例，显示顺序为

左侧	1	右侧	4
	2		5
	3		6

8、公司通讯协议（设备主动上传）

```
struct STR_JDWDSEND
{
    unsigned char nStart1;              //0x09          1 字节
    unsigned char nStart2;              //0xaf           1 字节
    unsigned char nOrder=0x 02;         //温度控制字     1 字节
    unsigned long   nID;                //传感器编号     4 字节
    signed    short   nT;               //读取的温度     2 字节
    unsigned short   ntemp;             //发包序列号     2 字节
    unsigned   short    nCrc;           //CRC 校验码     2 字节
    unsigned char nEnd;                 //0x16          1 字节
```

};
nT=实际温度×10
09 af 编号（1 位） 节点 ID 低位（2 位）节点 ID 高位（2 位） 温度（2 位） 序列号（2 位） 校验位（2 位） 16

9、附 CRC 算法表

```
const unsigned char chCRCHTalbe[] =                                    // CRC 高位字节值表
{
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 chCRCLTalbe[] =                                    // CRC 低位字节值表
{
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 short int CRC16(unsigned char* pchMsg, int wDataLen)
{
        unsigned char chCRCHi = 0xFF; //  高 CRC 字节初始化
        unsigned char chCRCLo = 0xFF; //  低 CRC 字节初始化
        int wIndex;                   // CRC 循环中的索引

        while (wDataLen--)
        {
                //  计算 CRC
                wIndex = chCRCLo ^ *pchMsg++ ;
                chCRCLo = chCRCHi ^ chCRCHTalbe[wIndex];
                chCRCHi = chCRCLTalbe[wIndex] ;
        }

        return ((chCRCHi << 8) | chCRCLo) ;
}
```