通讯协议（ THM-V6）

Ver2.5

1、概述

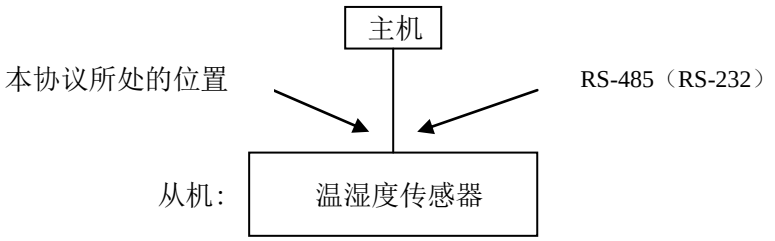
通信协议详细地描述了温湿度传感器的输入和输出命令、信息和数据，以便第三方使用和开发。

1.1 通信协议的作用

使信息和数据在上位机（主站）和温湿度传感器之间有效地传递，允许访问温湿度传感器的所有测量数据。

温湿度传感器可以实时采集现场温湿度的值,具备一个 RS-485（RS-232）通讯口，能满足小型温湿度监控系统的要求。温湿度传感器的功能和技术指标参见产品规格书。

温湿度传感器通信协议采用 MODBUS RTU 协议,本协议规定了应用系统中主机与温湿度传感器之间，在应用层的通信协议，它在应用系统中所处的位置如下图所示：



1.2 物理接口：

连接上位机的主通信口，采用标准串行 RS-485（RS-232）通讯口。

信息传输方式为异步方式，起始位 1 位，数据位 8 位，停止位 1 位，无校验。

数据传输缺省速率为 9600b/s

2、MODBU RTU 通信协议详述

2．1 协议基本规则

以下规则确定在回路控制器和其他串行通信回路中设备的通信规则。

1）所有回路通信应遵照主/从方式。在这种方式下，信息和数据在单个主站和从站（监控设备）之间传递。

2）主站将初始化和控制所有在通信回路上传递的信息。

3）无论如何都不能从一个从站开始通信。

4）所有环路上的通信都以“打包”方式发生。一个包裹就是一个简单的字符串（每个字符串 8 位),一个包裹中最多可含 255 个字节。组成这个包裹的字节构成标准异步串行数据，并按 8 位数据位，1 位停止位，无校验位的方式传递。串行数据流由类似于 RS-232 中使用的设备产生。

5）所有回路上的传送均分为两种打包方式：

A) 主/从传送

B) 从/主传送

6）若主站或任何从站接收到含有未知命令的包裹，则该包裹将被忽略，且接收站不予响应。

2．2 数据帧结构描述

每个数据帧组成如下：（RTU 模式）

地址　功能代码　数据数量　数据 1　...　数据 n　CRC16 位校验

3、传输格式

(1)命令报文格式

主机发送读温湿度数据命令：

地址	功能码	数据起始地址高位	数据起始地址低位	数据个数高位	数据个数低位	CRC 16 位校验
xx	03	00	02	00	02	xxxx 低位在前

从机传感器返回温湿度数据值：

地址	功能码	字节长度	温度返回数据	湿度返回数据	CRC 16 位校验
xx	03	04	xxxx 高位在前	xxxx 高位在前	xxxx 低位在前

主机发送读地址命令：

地址	功能码	数据起始地址高位	数据起始地址低位	数据个数高位	数据个数低位	CRC 16 位校验
00	03	00	00	00	01	xxxx 低位在前

从机传感器返回地址值：

地址	功能码	字节长度	地址高位	地址低位	CRC 16 位校验
00	03	02	00	xx	xxxx 低位在前

主机发送地址设置命令：

地址	功能码	写入位置高位	写入位置低位	操作数高位	操作数低位	字节长度	写入内容高位	写入内容低位	CRC 16 位校验
00	10	00	00	00	01	02	00	xx	xxxx 低位在前

从机传感器返回响应值：

地址	功能码	写入位置高位	写入位置低位	操作数高位	操作数低位	CRC 16 位校验
00	10	00	00	00	01	xxxx 低位在前

(2)帧格式（10 位）

起始位	D0	D1	D2	D3	D4	D5	D6	D7	停止位

4、 主机数据采样频率：

读取温湿度传感器数据时，上位机读取数据每次间隔时间不小于 500ms，推荐值 1s。

5、命令举例：

串口设置：异步通讯，起始位 1 位，数据位 8 位，无校验，停止位 1 位
数据传输速率缺省为：9600b/s

地址为 1 时，读取温湿度值：（CRC 校验低位在前）
上位机发送： 01 03 00 02 00 02 65 CB （读从数据起始地址为 0002H 开始的 2 个模拟量）
变送器返回： 01 03 04，温度 H，温度 L，湿度 H，湿度 L，CRC_L，CRC_H

读取地址码值：（CRC 校验低位在前）
上位机发送： 00 03 00 00 00 01 85 DB （读从数据起始地址为 0000H 开始的 1 个模拟量）
变送器返回： 00 03 02 00，地址 L ，CRC_L，CRC_H

为变送器设置新地址“2”：（CRC 校验低位在前）（地址设置范围：1～254）
上位机发送：00 10 00 00 00 01 02 00 02 2A 01
变送器返回：00 10 00 00 00 01 00 18

注意：

从机返回的温湿度数据分别用两个字节表示，高位在前，低位在后；
返回数据范围 -32768～32767，实际温湿度数据需要将返回值除以 10；
CRC16 位校验用两个字节表示，低位在前，高位在后；
地址设置范围：1～254

例如：

返回湿度 16 进制数据：0x0311，对应十进制 785，表示湿度为 78.5%RH
返回温度 16 进制数据：0x00FF，对应十进制 255，表示温度为 25.5℃
返回温度 16 进制数据：0x8064,最高位为 1 表示负数,对应十进制-100,表示温度为-10.0℃

附录：

CRC 高位字节值表：

```
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};

CRC 低位字节值表：

```
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 函数计算方法：

1. 预置 1 个 16 位的寄存器为十六进制 FFFF（即全为 1）；称此寄存器为 CRC 寄存器；
2. 把第一个 8 位二进制数据（既通讯信息帧的第一个字节）与 16 位的 CRC 寄存器的低 8 位相异或，把结果放于 CRC 寄存器；
3. 把 CRC 寄存器的内容右移一位（朝低位）用 0 填补最高位，并检查右移后的移出位；
4. 如果移出位为 0：重复第 3 步（再次右移一位）；

如果移出位为 1：CRC 寄存器与多项式 A001（1010 0000 0000 0001）进行异或；

5. 重复步骤 3 和 4，直到右移 8 次，这样整个 8 位数据全部进行了处理；
6. 重复步骤 2 到步骤 5，进行通讯信息帧下一个字节的处理；
7. 将该通讯信息帧所有字节按上述步骤计算完成后，得到的 16 位 CRC 寄存器的高、低字节进行交换；
8. 最后得到的 CRC 寄存器内容即为：CRC 码。

CRC 函数例程：

```
//*pushMsg 为需要校验的数组指针变量，usDataLen 为需要校验的数据个数变量
void CRC16(char *pushMsg,unsigned short usDataLen)
{
```

```
 char uchCRCHi=0xFF;             //高 CRC 字节初始化
 char uchCRCLo=0xFF;             //低 CRC 字节初始化
 unsigned int uIndex;            //CRC 循环中的索引
 while(usDataLen--)              //CRC 查表校验函数
  {
   uIndex  =uchCRCHi^*pushMsg++;//计算 CRC
   uchCRCHi=uchCRCLo^auchCRCHi[uIndex];
   uchCRCLo=auchCRCLo[uIndex];
  }
 *pushMsg++=uchCRCHi;            //校验数据高位在后
 *pushMsg=uchCRCLo;              //校验数据低位在前
}
```