

---

**User's  
Manual**

# 无纸记录仪

通讯协议

---

# 目录

第一章 通讯功能概要

第二章 通讯指令

附录 1 仪表寄存器地址

附录 2 CRC 循环冗余校验算法

附录 3 ASCII 码表



## 第一章 通讯功能概要

### 1.1 通信功能一览

采用 RS485 串行接口，使用标准 Modbus RTU 协议。

功能	协议	连接设备
从机	Modbus RTU	主机（测量仪器、PC、PLC 等）

### 1.2 串口通讯参数

#### RS485 接口规格

插座类型	绿色端子中 485A+、485B-
连接方式	多点，总线式拓扑网络
通信方式	半双工
波特率	9600, 19200, 38400, 57600, 115200[bps]
起始位	1 位（固定）
数据位	8 位（固定）
校验位	奇校验，偶校验，无校验
停止位	1 位、2 位
接收缓冲器大小	128 字节
通信距离	小于 1km
终端阻抗*2	外部：推荐 120Ω,1/2W 电阻

## 第二章 通讯指令

### 03H 读取保持寄存器

#### 描述

读取仪表保持寄存器，包含通道数据。对应 32 位浮点型数据，根据通讯组态中的字节交换进行调整。调整类型包含 2143、1234、4321、34124。

#### 发送

命令信息中包含了读取寄存器的起始地址和读取长度。下面是一个从地址为 1 的设备中读取第一通道数据（寄存器偏移量为 0x00）的例子。

#### 发送格式

名称	数据 (HEX)
从设备地址	01H
功能码	03H
起始地址高	00H
起始地址低	00H
寄存器数量高	00H
寄存器数量低	02H
CRC 校验低	C4H
CRC 校验高	0BH

## 返回

返回通道 1 数据，浮点型数据，默认 3412。比如获取数据按交换顺序为 00 E0 AB 44 (低在前，高在后)，表示 1375。

### 返回格式

名称	数据 (HEX)
从设备地址	01H
功能码	03H
字节数量	04H
高位数据 (寄存器偏移 0x00)	ABH
低位数据 (寄存器偏移 0x00)	44H
高位数据 (寄存器偏移 0x01)	00H
低位数据 (寄存器偏移 0x01)	E0H
CRC 校验低	A4H
CRC 校验高	6CH

## 04H 读取输入寄存器

### 描述

读取通道数据，返回有符号长整形数据。

### 发送

读取仪表保持寄存器，包含通道数据。对应长整形数据，根据通讯组态中的字节交换进行调整。字节顺序可通过仪表调整，类型包含 2143、1234、4321、34124。

### 发送格式

名称	数据 (HEX)
从设备地址	01H
功能码	04H
起始地址高	00H
起始地址低	00H
寄存器个数高位	00H
寄存器个数低位	02H
CRC 校验低	71H
CRC 校验高	CBH

## 返回

返回通道 1 数据，有符号长整形数据，字节排序根据仪表中字节交换的顺序。比如获取数据 00 00 00 64，若记录仪中交换顺序为 4321，交换结果为 00 00 00 64。若记录仪中小数点为 0，表示 100。

### 返回格式

名称	数据 (HEX)
从设备地址	01H
功能码	04H
字节数量	04H
高位数据 (偏移量为 0x00)	00H
低位数据 (偏移量为 0x00)	00H
高位数据 (偏移量为 0x01)	00H
低位数据 (偏移量为 0x01)	64H
CRC 校验低	FAH
CRC 校验高	6FH



## 附录 1 仪表寄存器地址

### 03 命令对应地址

类型	寄存器偏移量	备注
通道 1 实时量	0000H	32 位浮点型数据
通道 2 实时量	0002H	32 位浮点型数据
通道 3 实时量	0004H	32 位浮点型数据
通道 4 实时量	0006H	32 位浮点型数据
通道 5 实时量	0008H	32 位浮点型数据
通道 6 实时量	000AH	32 位浮点型数据
通道 1 累积量	000CH	32 位浮点型数据
通道 2 累积量	000EH	32 位浮点型数据
通道 3 累积量	0010H	32 位浮点型数据
通道 4 累积量	0012H	32 位浮点型数据
通道 5 累积量	0014H	32 位浮点型数据
通道 6 累积量	0016H	32 位浮点型数据
通道 1 下下限报警	0200H	16 位无符号数据 1 报警 0 不报警
通道 1 下限报警	0201H	16 位无符号数据 1 报警 0 不报警
通道 1 上限报警	0202H	16 位无符号数据 1 报警 0 不报警
通道 1 上上限报警	0203H	16 位无符号数据 1 报警 0 不报警
通道 2 下下限报警	0204H	16 位无符号数据 1 报警 0 不报警
通道 2 下限报警	0205H	16 位无符号数据 1 报警 0 不报警
通道 2 上限报警	0206H	16 位无符号数据 1 报警 0 不报警

通道 2 上上限报警	0207H	16 位无符号数据 1 报警 0 不报警
通道 3 下下限报警	0208H	16 位无符号数据 1 报警 0 不报警
通道 3 下限报警	0209H	16 位无符号数据 1 报警 0 不报警
通道 3 上限报警	020AH	16 位无符号数据 1 报警 0 不报警
通道 3 上上限报警	020BH	16 位无符号数据 1 报警 0 不报警
通道 4 下下限报警	020CH	16 位无符号数据 1 报警 0 不报警
通道 4 下限报警	020DH	16 位无符号数据 1 报警 0 不报警
通道 4 上限报警	020EH	16 位无符号数据 1 报警 0 不报警
通道 4 上上限报警	020FH	16 位无符号数据 1 报警 0 不报警
通道 5 下下限报警	0210H	16 位无符号数据 1 报警 0 不报警
通道 5 下限报警	0211H	16 位无符号数据 1 报警 0 不报警
通道 5 上限报警	0212H	16 位无符号数据 1 报警 0 不报警
通道 5 上上限报警	0213H	16 位无符号数据 1 报警 0 不报警
通道 6 下下限报警	0214H	16 位无符号数据 1 报警 0 不报警
通道 6 下限报警	0215H	16 位无符号数据 1 报警 0 不报警
通道 6 上限报警	0216H	16 位无符号数据 1 报警 0 不报警
通道 6 上上限报警	0217H	16 位无符号数据 1 报警 0 不报警

注：字节顺序可在表中设置。16 位数据先高位后低位。

## 04 命令对应地址

类型	寄存器偏移量	备注
通道 1 实时量	0000H	有符号长整型
通道 2 实时量	0002H	有符号长整型
通道 3 实时量	0004H	有符号长整型
通道 4 实时量	0006H	有符号长整型
通道 5 实时量	0008H	有符号长整型
通道 6 实时量	000AH	有符号长整型
通道 1 累积量(整数部分)	000CH	无符号长整型
通道 1 累积量(小数部分)	000EH	无符号长整型, 默认 3 位小数
通道 2 累积量(整数部分)	0010H	无符号长整型
通道 2 累积量(小数部分)	0012H	无符号长整型, 默认 3 位小数
通道 3 累积量(整数部分)	0014H	无符号长整型
通道 3 累积量(小数部分)	0016H	无符号长整型, 默认 3 位小数
通道 4 累积量(整数部分)	0018H	无符号长整型

通道 4 累积量(小数部分)	001AH	无符号长整型, 默认 3 位小数
通道 5 累积量(整数部分)	001CH	无符号长整型
通道 5 累积量(小数部分)	001EH	无符号长整型, 默认 3 位小数
通道 6 累积量(整数部分)	0020H	无符号长整型
通道 6 累积量(小数部分)	0022H	无符号长整型, 默认 3 位小数

注：字节顺序可在表中设置，小数点与表中设置同步。

## 附录 2 CRC 循环冗余校验算法

### 1. CRC 校验概述

CRC 校验码的基本思想是利用线性编码理论，在发送端根据要传送的  $k$  位二进制码序列，以一定的规则产生一个校验用的监督码（既 CRC 码） $r$  位，并附在信息后边，构成一个新的二进制码序列数共  $(k+r)$  位，最后发送出去。在接收端，则根据信息码和 CRC 码之间所遵循的规则进行检验，以确定传送中是否出错。

### 2. CRC 校验算法

```
const uchar ucCRCHI[] =  
{  
    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 uchar ucCRCLo[] =
```

```
{
```

```
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 计算
```

```
ushort CalCrc(uchar *pucData , ushort usDataLen)
```

```
{
```

```
    uchar ucCrcLo = 0xFF ;
```

```
    uchar ucCrcHi = 0xFF ;
```

```
    uchar ucIndex ;
```

```
    while(usDataLen--)
```

```
    {
```

```
        ucIndex = ucCrcLo ^ *pucData++ ;
```

```
        ucCrcLo = ucCrcHi ^ ucCRCHI[ucIndex] ;
```

```
        ucCrcHi = ucCRCLo[ucIndex] ;
```

```
    };
```

```
    return (ucCrcHi * 0x100 + ucCrcLo) ;
```

```
}
```



