

## MODBUS 通讯协议及编程

### MODBUS 通讯协议及编程

ModBus 通讯协议分为 RTU 协议和 ASCII 协议，我公司的多种仪表都采用 ModBus RTU 通讯协议，下面就 ModBus RTU 协议简要介绍如下：

#### 一、通讯协议

##### （一）、通讯传送方式：

通讯传送分为独立的信息头，和发送的编码数据。以下的通讯传送方式定义也与 MODBUS RTU 通讯规约相兼容：

编 码 8 位二进制

起始位 1 位

数据位 8 位

奇偶校验位 1 位（偶校验位）

停止位 1 位

错误校检 CRC（冗余循环码）

初始结构 =  $\geq 4$  字节的时间

地址码 = 1 字节

功能码 = 1 字节

数据区 = N 字节

错误校检 = 16 位 CRC 码

结束结构 =  $\geq 4$  字节的时间

**地址码：**地址码为通讯传送的第一个字节。这个字节表明由用户设定地址码的从机将接收由主机发送来的信息。并且每个从机都有具有唯一的地址码，并且响应回送均以各自的地址码开始。主机发送的地址码表明将发送到的从机地址，而从机发送的地址码表明回送的从机地址。

**功能码：**通讯传送的第二个字节。ModBus 通讯规约定义功能号为 1 到 127。本仪表只利用其中的一部分功能码。作为主机请求发送，通过功能码告诉从机执行什么动作。作为从机响应，从机发送的功能码与从主机发送来的功能码一样，并表明从机已响应主机进行操作。如果从机发送的功能码的最高位为 1（比如功能码大与此同时 127），则表明从机没有响应操作或发送出错。

**数据区：**数据区是根据不同的功能码而不同。数据区可以是实际数值、设置点、主机发送给从机或从机发送给主机的地址。

**CRC 码：**二字节的错误检测码。

##### （二）、通讯规约：

当通讯命令发送至仪器时，符合相应地址码的设备接通讯命令，并除去地址码，读取信息，如果没有出错，则执行相应的任务；然后把执行结果返送给发送者。返送的信息中包括地址码、执行动作的功能码、执行动作后结果的数据以及错误校验码。如果出错就不发送任何信息。

## 1. 信息帧结构

地址码 功能码 数据区 错误校验码

8 位 8 位  $N \times 8$  位 16 位

**地址码：**地址码是信息帧的第一字节(8 位)，从 0 到 255。这个字节表明由用户设置地址的从机将接收由主机发送来的信息。每个从机都必须有唯一的地址码，并且只有符合地址码的从机才能响应回送。当从机回送信息时，相当的地址码表明该信息来自于何处。

**功能码：**主机发送的功能码告诉从机执行什么任务。表 1-1 列出的功能码都有具体的含义及操作。

代码 含义 操作

03 读取数据 读取当前寄存器内一个或多个二进制值

06 重置单一寄存器 把设置的二进制值写入单一寄存器

**数据区：**数据区包含需要从机执行什么动作或由从机采集的返送信息。这些信息可以是数值、参考地址等等。例如，功能码告诉从机读取寄存器的值，则数据区必需包含要读取寄存器的起始地址及读取长度。

对于不同的从机，地址和数据信息都不相同。

**错误校验码：**主机或从机可用校验码进行判别接收信息是否出错。有时，由于电子噪声或其它一些干扰，信息在传输过程中会发生细微的变化，错误校验码保证了主机或从机对在传送过程中出错的信息不起作用。这样增加了系统的安全和效率。错误校验采用 CRC-16 校验方法。

注：信息帧的格式都基本相同：地址码、功能码、数据区和错误校验码。

## 2. 错误校验

冗余循环码（CRC）包含 2 个字节，即 16 位二进制。CRC 码由发送设备计算，放置于发送信息的尾部。接收信息的设备再重新计算接收到信息的 CRC 码，比较计算得到的 CRC 码是否与接收到的相符，如果两者不相符，则表明出错。

CRC 码的计算方法是，先预置 16 位寄存器全为 1。再逐步把每 8 位数据信息进行处理。在进行 CRC 码计算时只用 8 位数据位，起始位及停止位，如有奇偶校验位的话也包括奇偶校验位，都不参与 CRC 码计算。

在计算 CRC 码时，8 位数据与寄存器的数据相异或，得到的结果向低位移一字节，用 0 填补最高位。再检查最低位，如果最低位为 1，把寄存器的内容与预置数相异或，如果最低位为 0，不进行异或运算。

这个过程一直重复 8 次。第 8 次移位后，下一个 8 位再与现在寄存器的内容相异或，这个过程与上一样重复 8 次。当所有的数据信息处理完后，最后寄存器的内容即为 CRC 码值。CRC 码中的数据发送、

接收时低字节在前。

计算 CRC 码的步骤为：

预置 16 位寄存器为十六进制 FFFF（即全为 1）。称此寄存器为 CRC 寄存器；  
 把第一个 8 位数据与 16 位 CRC 寄存器的低位相异或，把结果放于 CRC 寄存器；  
 把寄存器的内容右移一位(朝低位)，用 0 填补最高位，检查最低位；  
 如果最低位为 0：重复第 3 步(再次移位)；如果最低位为 1：CRC 寄存器与多项式 A001  
 （1010 0000 0000 0001）进行异或；  
 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；  
 重复步骤 2 到步骤 5，进行下一个 8 位数据的处理；  
 最后得到的 CRC 寄存器即为 CRC 码。

3. 功能码 03，读取点和返回值：

仪表采用 Modbus RTU 通讯规约，利用通讯命令，可以进行读取点(“保持寄存器”)或返回值(“输入寄存器”)的操作。保持和输入寄存器都是 16 位（2 字节）值，并且高位在前。这样用于仪表的读取点和返回值都是 2 字节。一次最多可读取寄存器数是 60。由于一些可编程控制器不用功能码 03，所以功能码 03 被用作读取点和返回值。从机响应的命令格式是从机地址、功能码、数据区及 CRC 码。数据区中的寄存器数据都是每两个字节高字节在前。

#### 4. 功能码 06，单点保存

主机利用这条命令把单点数据保存到仪表的存储器。从机也用这个功能码向主机返送信息。

## 二、编程举例

下面是一个用 VC 编写的 ModBus RTU 通讯的例子

### （一）、通讯口设置

```

DCB dcb;
hCom=CreateFile("COM1",
GENERIC_READ|GENERIC_WRITE,
0,
NULL,
OPEN_EXISTING,
0,
NULL);
if(hCom==INVALID_HANDLE_VALUE)
{
    MessageBox("createfile error,error");
}
BOOL error=SetupComm(hCom,1024,1024);
if(!error)

```

```
    MessageBox("setupcomm error");
    error=GetCommState(hCom,&dcb);
    if(!error)
        MessageBox("getcommstate,error");
    dcb.BaudRate=2400;
    dcb.ByteSize=8;

    dcb.Parity=EVENPARITY;//NOPARITY;
    dcb.StopBits=ONESTOPBIT;

    error=SetCommState(hCom,&dcb);
```

## （二）、CRC 校验码计算

```
UINT crc
void calccrc(BYTE crcbuf)
{
    BYTE i;

    crc=crc ^ crcbuf;
    for(i=0;i<8;i++)
    {
        BYTE TT;
        TT=crc&1;
        crc=crc>>1;
        crc=crc&0x7fff;
        if (TT==1)
            crc=crc^0xa001;
        crc=crc&0xffff;
    }
}
```

## （三）、数据发送

```
zxaddr=11;//读取地址为 11 的巡检表数据
zxnum=10;//读取十个通道的数据
```

```
writebuf2[0]=zxaddr;
writebuf2[1]=3;
writebuf2[2]=0;
writebuf2[3]=0;
writebuf2[4]=0;
writebuf2[5]=zxnum;
crc=0xffff;
```

```
        calccrc(writebuf2[0]);  
        calccrc(writebuf2[1]);  
        calccrc(writebuf2[2]);  
        calccrc(writebuf2[3]);  
        calccrc(writebuf2[4]);  
        calccrc(writebuf2[5]);  
  
        writebuf2[6]=crc & 0xff;  
        writebuf2[7]=crc/0x100;  
        WriteFile(hCom,writebuf2,8,&comnum,NULL);
```

#### （四）、数据读取

```
ReadFile(hCom,writebuf,5+zxnum*2,&comnum,NULL);//读取 zxnum 个通道数据  
可增加错误处理程序，如地址码错误、CRC 码错误判断、通讯故障处理等。
```