Mantunsci

让电对人类不再有伤害

RS485 模块 Modbus 通讯协议 V0.2

深圳曼顿科技有限公司

协议修改记录

版本	记录	日期
V0.2	1.更新 03 命令码的描述 2.更新 04 命令码的描述 3.更新 05 命令码举例的描述 4.更新 06 命令码举例的描述 5.更新 16 命令码举例的描述	2019 年 6 月
V0.1	初版	

通信模块采用 485 通信接口，采用国际标准的 Modbus 通讯协议进行的主从通讯。用户可通过网络模块/PC 实现集中控制（监控模块数据、设置模块参数、开启/关闭 1P 和 2P 等），灵活适应特定的应用要求。（注：以下“主机”是指网络模块或个人计算机 PC、PLC 等，“从机”是指 485 通信模块），**由于内存的限制本从机仅支持向下通信 11 路开关地址。**

1 协议内容

该 Modbus 串行通信协议定义了串行通信中异步传输的帧内容及使用格式。其中包括：主机轮询及广播帧、从机应答帧的格式；主机组织的帧内容包括：从机地址(或广播地址)、执行命令、数据和错误校验等。从机的响应也是采用相同的结构，内容包括：动作确认，返回数据和错误校验等。如果从机在接收帧时发生错误，或不能完成主机要求的动作，它将组织一个故障帧作为响应反馈给主机。

2 总线结构

(1)接口方式

485 通信接口

(2)传输方式

异步串行，半双工传输方式。在同一时刻主机和从机只能有一个发送数据而另一个接收数据。数据在串行异步通信过程中，是以报文的形式，一帧一帧发送。网络中只有一个主机能够建立协议，称为“查询/命令”。从机只能通过提供数据响应主机的“查询/命令”，或根据主机的“查询/命令”做出相应的动作。主机既能对某个从机单独进行通信，也能对所有从机发布广播信息。对于单独访问的主机“查询/命令”，从机都要返回一个信息，称为响应，对于主机发出的广播信息，从机不能反馈响应信息给主机。

(3)拓扑结构

单主机多从机系统，使用一条总线连接。从机地址的设定范围为 1～247，0 和 255 为广播通信地址。网络中的每个从机的地址都具有唯一性。这是保证 Modbus 通讯的基础。

3 通讯帧结构

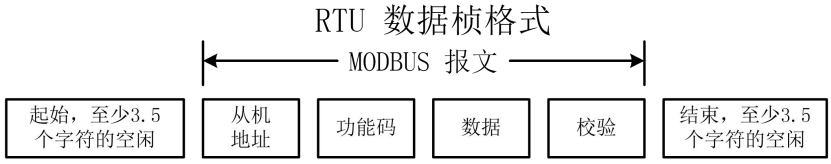
从机支持标准的 Modbus 协议通信数据格式 RTU/ASCII（1-0），传输速率支持 2400/4800/9600/14400/19200/28800/38400/56000（1-8），奇偶校验（0-ODD，1-EVEN，2-2 停止位，3-1 停止位）1 起始 bit、8 数据 bit。

基本字节数据格式的描述如下表：起始位、8 个数据位、1 个停止位。

11-bit 字符帧：

起始位	bit0	bit1	bit2	bit3	bit4	bit5	bit6	bit7	奇偶校验	停止位

在 RTU 模式中（ASCII 方式不做介绍），新帧总是以至少 3.5 个字节的传输时间静默作为开始。在以波特率计算传输速率的网络上，3.5 个字节的传输时间可以轻松把握。紧接着传输的数据域依次为：从机地址、操作命令码、数据和 CRC 校验字。从机始终监视着通讯总线的活动。当接收到第一个域（地址信息），每个从机设备都对该字节进行确认。随着最后一个字节的传输完成，又有一段类似的 3.5 个字节的传输时间间隔，用来表识本帧的结束，在此以后，将开始一个新帧的传送。



一个帧的信息必须以一个连续的数据流进行传输，如果整个帧传输结束前有超过 1.5 个字节以上的间隔时间，从设备将清除这些不完整的信息，并错误认为随后一个字节是新一帧的地址域部分，同样的，如果一个新帧的开始与前一个帧的间隔时间小于 3.5 个字节时间，接收设备将认为它是前一帧的继续，由于帧的错乱，最终 CRC 校验值不正确，从设备将会抛弃该帧，导致通信中断。RTU 帧的标准结构：

帧头START	3.5个字节的传输时间，此时间作为总线空闲时间，无需传输数据
从机地址域 ADDR	通讯地址：0～247（十进制）（0、256为广播地址）
功能域CMD	0x03：读从机实时数据；0x04：读从机参数；0x06：单个写从机参数；0x16：批量写从机参数
数据域 DATA	2*N个字节的数据，该部分为通讯的主要内容，也是通讯中，数据交换的核心。
CRC低位	CRC校验值（16bit）
CRC高位	
帧尾END	3.5个字节的传输时间，此时间作为总线空闲时间，无需传输数据

4 命令码及通讯数据域描述（控制和修改指令请不要以很小的间隔发送，由于内存有限，超过缓存池的指令将不会执行）

从机支持共 9 种命令码，分别是：

01：读取开关合/分闸状态；

02：读取是否能被远程控制

01：(特殊）读取哪些开关地址存在

02：（特殊）读取哪些开关是 3 相开关

03：读从机实时数据；

04：读从机参数；

05：控制开关合/分闸；

06：写单个从机参数；

15：批量写控制开关合/分闸；

16：批量写从机参数；

4.1 CMD 命令码：01（0000 0001）读取开关合/分闸状态，读取 N 位（bit），根据底层开关地址顺序排列：

主机发送：起始地址（从 1 开始，小于参数地址上限，实际发送数据减 1），返回数据个数（字 bit 数，大于 0，小于上限）

从机返回：返回**字节**数（8bit 为一字节，bit 位不足一个字节按照一个字节计算），从机参数数据，返回 bit1 表示合闸

Bit	7	6	5	4	3	2	1	0
数据字节1 对应开关地址	8	7	6	5	4	3	2	1
数据字节2 对应开关地址	16	15	14	13	12	11	10	9
…（以此类推）	24	23	22	21	20	19	18	17

例：从机地址为 0x01，从开关地址 1 开始连续读取 9 个开关，则该帧的结构描述如下：

RTU主机命令信息		RTU从机回应信息	
ADDR	0x01	ADDR	0x01
CMD	0x01	CMD	0x01
启始地址高位	**0x00**	**字节个数**	**0x02**
启始地址低位	**0x00（底层开关地址1）**	**数据字节1**	**0x13（对应开关地址1-8）**
数据个数高位	**0x00**	**数据字节2**	**0x00（对应开关地址9-16）**
数据个数低位	**0x09（bit）**	CRC低位	0xB4
CRC低位	0xFC	CRC高位	0xCC
CRC高位	0x0C		

4.2 CMD 命令码：02（0000 0010）读取开关是否能被远程控制状态，读取 N 位（bit），根据底层开关地址顺序排列：

主机发送：起始地址（从 1 开始，小于参数地址上限，实际发送数据减 1），返回数据个数（字 bit 数，大于 0，小于上限）

从机返回：返回**字节**数（8bit 为一字节，bit 位不足一个字节按照一个字节计算），从机参数数据返回 bit1 表示不能远控

（同上）

4.3 CMD 命令码：01（0000 0001）读取开关是否存在，**读取全部可能存在的位**（bit），根据底层开关地址顺序排列：

主机发送：**起始地址固定为 0，“数据个数”固定为 0x00FF。(这里属于特殊的参数)**

从机返回：字节个数：为最大支持开关数量/8 取整数部分（有小数存在则+1），数据字节： bit1 表示存在此开关

（同上）

CMD 命令码：02（0000 0010）读取开关是否为三相开关，**读取全部可能存在的位**（bit），根据底层开关地址顺序排列：

主机发送：**起始地址固定为 0，“数据个数”固定为 0x00FF。(这里属于特殊的参数)**

从机返回：字节个数：为最大支持开关数量/8 取整数部分（有小数存在则+1）字节数，数据字节：bit1 表此示开关为三相开关

（同上）

4.4 CMD 命令码：03（0000 0011）**读从机实时数据**，读取 N 个字（Word）对应**数据域**格式（具体地址含义见附表 1）：

主机发送：起始地址（从 0 开始，小于数据地址上限），返回数据个数（字 Word 数，大于 0，小于上限）

从机返回：返回**字节**数，从机实时数据（远程模式增加返回：参数地址，底层开关起始地址）

例：从机地址为 0x01，功能地址 0x03，从开关地址 2 开始连续读取两个开关（连续读取为按顺序排列的底层开关地址），则该帧结构如下：

RTU主机命令信息		RTU从机回应信息		RTU从机回应信息（远程模式）	
ADDR	0x01	ADDR	0x01	ADDR	0x01
CMD	0x03	CMD	0x03	CMD	0x03
启始地址高位	**0x04（具体功能地址0-255）**	**字节个数**	**0x04**	**启始地址高位**	**0x04**
启始地址低位	**0x01（底层开关地址2）**	**底层开关2，数据地址0004H高位**	**0x13**	**启始地址低位**	**0x01**
数据个数高位	**0x00**	**底层开关2，数据地址0004H低位**	**0x88**	**字节个数**	**0x04**
数据个数低位	**0x02**	**底层开关3，数据地址0004H高位**	**0x13**	**底层开关2，数据地址0004H高位**	**0x13**
CRC低位	0x94	**底层开关3，数据地址0004H低位**	**0x88**	**底层开关2，数据地址0004H低位**	**0x88**
CRC高位	0xFB	CRC低位	0x73	**底层开关3，数据地址0004H高位**	**0x13**
		CRC高位	0xCB	**底层开关3，数据地址0004H低位**	**0x88**
				CRC低位	0x7B
				CRC高位	0x5C

4.5 CMD 命令码：04（0000 0100）**读从机参数**，读取 N 个字（Word）对应**数据域**格式（具体地址含义见附表 2）：

主机发送：起始地址（从 0 开始，小于参数地址上限，底层开关地址从 0 开始表示地址 1），返回数据个数（字 Word 数，大于 0，小于上限）

从机返回：返回**字节**数，从机参数数据（远程模式增加返回：参数地址，底层开关起始地址）

此命令与上条类似，请参考上一条命令举例。

4.6 CMD 命令码：05（0000 0101）写开关合/分闸状态，写取 1 位（bit）：

主机发送：底层开关地址（从 0 开始，小于参数地址上限，发送 0xFF 表示广播），合发送 0xFF00，分发送 0x0000

从机返回：镜像信息表示响应成功。，**当命令缓存池满以后会返回 0x06 从设备忙异常代码**，可延迟一段时间再试（可能仅部分指令执行）

例：从机地址为 0x01，底层开关地址 5，控制合闸，则该帧的结构描述如下：

RTU主机命令信息		RTU从机回应信息	
ADDR	0x01	ADDR	0x01
CMD	0x05	CMD	0x05
启始地址高位	**0x00**	**启始地址高位**	**0x00**
启始地址低位	**0x04（底层开关地址5）**	**启始地址低位**	**0x04（底层开关地址5）**
数据高字节	**0xFF（合闸）**	**数据高字节**	**0xFF（合闸）**
数据低字节	**0x00**	**数据低字节**	**0x00**
CRC低位	0x9C	CRC低位	0x48
CRC高位	0x3B	CRC高位	0xFD

4.7 CMD 命令码：06（0000 0110）**写从机参数**，写 1 个字（Word）对应**数据域**格式（具体地址含义见附表 3）：

主机发送：目标地址（从 0 开始，小于参数地址上限，**底层开关地址从 0 开始表示地址 1，底层开关地址 0xF0 表示从机自身参数设定）**，数据内容（字 Word）

从机返回：修改地址，从机修改后数据（字 Word），**当命令缓存池满以后会返回 0x06 从设备忙异常代码**，可延迟一段时间再试

例如：将 5000（1388H）写到从机地址 0x02，0x00 功能地址，底层开关地址 5。则该帧的结构描述如下：

RTU主机命令信息		RTU 从机回应信息	
ADDR	0x02	ADDR	0x02
CMD	0x06	CMD	0x06
写数据地址高位	**0x00（具体功能地址0-255）**	**写数据地址高位**	**0x00（具体功能地址0-255）**
写数据地址低位	**0x04（底层开关地址5）**	**写数据地址低位**	**0x04（底层开关地址5）**
数据内容高位	**0x13**	**数据内容高位**	**0x13**
数据内容低位	**0x88**	**数据内容低位**	**0x88**
CRC低位	0x94	CRC低位	0x94
CRC高位	0xAE	CRC高位	0xAE

4.8 CMD 命令码：16（0001 0000）**批量写从机参数**，写指定个数的字（Word）对应**数据域**格式（具体地址含义见附表 3）：

主机发送：目标地址（从 0 开始，小于参数地址上限），数据内容（字 Word）

从机返回：修改地址，从机修改后数据（字 Word），**当命令缓存池满以后会返回 0x06 从设备忙异常代码**，可延迟一段时间再试（可能仅部分指令执行）

例如：从机地址 0x01，0x00 功能地址，从底层开关地址 1 开始连续写入 3 个参数。则该帧的结构描述如下：

RTU主机命令信息		RTU从机回应信息	
ADDR	0x01	ADDR	0x01
CMD	0x10	CMD	0x10
写数据地址高位	**0x00（具体功能地址0-255）**	**写数据地址高位**	**0x00（具体功能地址0-255）**
写数据地址低位	**0x00（底层开关地址1）**	**写数据地址低位**	**0x00（底层开关地址1）**
数据个数高位	**0x00**	**数据个数高位**	**0x00**
数据个数低位	**0x03**	**数据个数低位**	**0x03**
控制字节数	**0x06**	CRC低位	0x80
数据1高位	**0x00**	CRC高位	0x08
数据1低位	**0xFA**		
数据2高位	**0x00**		
数据2低位	**0xFF**		
数据3高位	**0x01**		
数据3低位	**0x04**		
CRC低位	0xCA		
CRC高位	0xF6		

4.9 CMD 命令码：15（0000 1111）批量控制开关合/分闸状态，写 N 位（bit，缓存限制不能一次控制超过 8 个），根据底层开关地址顺序排列：

主机发送：底层开关起始地址（从 1 开始，小于参数地址上限，实际发送数据减 1），控制开关数量，以及控制字节数量（不足 8bit，按 1 算），具体对应开关的控制数据 1 代表合闸，0 代表分闸（按照 bit0-bit7 顺序排列）。

从机返回：底层开关起始地址（从 1 开始，小于参数地址上限，实际发送数据减 1），控制开关数量。**当命令缓存池满以后会返回 0x06 从设备忙异常代码**，可延迟一段时间再试（可能仅部分指令执行）

例：从机地址为 0x01，从开关地址 3 开始连续改变 5 个开关状态（3 号分，4 号合，5 号分，6 号合，7 号分，0000 1010），则该帧的结构描述如下：

RTU主机命令信息		RTU从机回应信息	
ADDR	0x01	ADDR	0x01
CMD	0x0F	CMD	0x01
启始地址高位	**0x00**	**启始地址高位**	**0x00**
启始地址低位	**0x02（底层开关地址3）**	**启始地址低位**	**0x02（对应开关地址3）**
数据个数高位	**0x00**	**数据个数高位**	**0x00**
数据个数低位	**0x05（bit）**	**数据个数低位**	**0x05**
控制字节数	**0x01**	CRC低位	0x34
控制数据1	0x0A(0000 1010)	CRC高位	0x08
CRC低位	0x96		
CRC高位	0x91		

4.10 通讯帧错误校验方式

在标准模式中，帧的错误校验方式主要包括两个部分的校验，即字节的位校验（奇/偶校验）和帧的整个数据校验（CRC 校验或 LRC 校验）。在这里经简化，只采用整个数据的 CRC 校验。

CRC 校验方式：

使用 RTU 帧格式，帧包括了基于 CRC 方法计算的帧错误检测域。CRC 域检测了整个帧的内容。CRC 域是两个字节，包含 16 位的二进制值。它由传输设备计算后加入到帧中。接收设备重新计算收到帧的 CRC，并与接收到的 CRC 域中的值比较，如果两个 CRC 值不相等，则说明传输有错误。

CRC 是先存入 0xFFFF，然后调用一个过程将帧中连续的 6 个以上字节与当前寄存器中的值进行处理。仅每个字符中的 8Bit 数据对 CRC 有效，起始位和停止位以及奇偶校验位均无效。

CRC 产生过程中，每个 8 位字符都单独和寄存器内容相异或（XOR），结果向最低有效位方向移动，最高有效位以 0 填充。LSB 被提取出来检测，如果 LSB 为 1，寄存器单独和预置的值相异或，如果 LSB 为 0，则不进行。整个过程要重复 8 次。在最后一位（第 8 位）完成后，下一个 8 位字节又单独和寄存器的当前值相异或。最终寄存器中的值，是帧中所有的字节都执行之后的 CRC 值。

CRC 的这种计算方法，采用的是国际标准的 CRC 校验法则，用户在编辑 CRC 算法时，可以参考相关标准的 CRC 算法，编写出真正符合要求的 CRC 计算程序。

现在提供 CRC 计算的简单函数给用户参考（用 C 语言编程）分别为直接计算法和查表法：

直接计算法

```
unsigned int CRC_cal_value(unsigned char *data_value,unsigned char data_length)
{
    int i;
    unsigned int crc_value=0xffff;
    while(data_length--)
    {
        crc_value ^= *data_value++;
        for(i=0;i<8;i++)
        {
            if(crc_value&0x0001)
            crc_value=(crc_value>>1)^0xa001;
            else
            crc_value=crc_value>>1;
        }
    }
    return(crc_value);
}
```

查表法

```
// 低字节 CRC 值表
const UCHAR 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
};
// 高字节 CRC 值表
const UCHAR 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
    };
    UINT CRC_cal_value(UCHAR * tmp, UINT len)
    {
        WORD uchCRC; UINT uIndex;
        uchCRC._word=0xFFFF;
       while (len--)
       {
            uIndex = uchCRC.byte0 ^ *tmp;
            uchCRC.byte0 = uchCRC.byte1 ^ auchCRCHi[uIndex];
            uchCRC.byte1 = auchCRCLo[uIndex];
            tmp++;
       }
       return uchCRC._word;
    }
```

在阶梯逻辑中，CKSM 根据帧内容计算 CRC 值，采用查表法计算，这种方法程序简单，运算速度快，但程序所占用 ROM 空间较大，对程序空间有要求的场合，请谨慎使用。

4.11 通信数据地址的定义表

附表 1 CMD 命令码：0x03（0000 0011）**从机实时数据表。**只有标明开关类型的数据才有意义：

功能说明	开关类型	地址	数据意义说明														R/W 特性
线路电压	2P/3P/4P	0x00	16 位电压，单位 1V														R
漏电电流	2P/3P/4P	0x01	16 位漏电电流，单位 0.1mA														R
线路功率	2P/1P/3P/4P	0x02	16 位功率，单位 1W														R
模块温度	2P/1P/3P/4P	0x03	16 位温度，单位 0.1 度														R
线路电流	2P/1P/3P/4P	0x04	16 位电流，单位 0.01A														R
告警位	2P/1P/3P/4P	0x05	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	R
			电流预警	漏电预警	欠压预警	过压预警	欠压报警	打火报警	输入缺相报警 3P/4P	漏电保护自检未完成	漏电保护功能正常	过压报警	过流报警	漏电报警	温度报警	过载报警	浪涌报警	短路报警	
电量低	2P/1P/3P/4P	0x06	32 位电量，单位 0.001 度														R
电量高	2P/1P/3P/4P	0x07	32 位电量，单位 0.001 度														R
A 相电压	3P/4P	0x08	16 位电压，单位 1V														R
B 相电压	3P/4P	0x09	16 位电压，单位 1V														R
C 相电压	3P/4P	0x0A	16 位电压，单位 1V														R
A 相电流	3P/4P	0x0B	16 位电流，单位 0.01A														R
B 相电流	3P/4P	0x0C	16 位电流，单位 0.01A														R
C 相电流	3P/4P	0x0D	16 位电流，单位 0.01A														R
N 相电流	3P/4P	0x0E	16 位电流，单位 0.01A														R
A 相功率	3P/4P	0x0F	16 位功率，单位 1W														R
B 相功率	3P/4P	0x10	16 位功率，单位 1W														R
C 相功率	3P/4P	0x11	16 位功率，单位 1W														R
A 相告警位	3P/4P	0x12	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	R
			电流预警	保留	欠压预警	过压预警	欠压报警	打火报警	输入缺相	保留	保留	过压报警	过流报警	保留	保留	过载报警	保留	短路报警	
B 相告警位	3P/4P	0x13	同上														R
C 相告警位	3P/4P	0x14	同上														R
其他告警位	3P/4P	0x15	7（仅 3P/4P）		6（仅 3P/4P）		5		4		3（仅 3P/4P）		2（仅 3P/4P）		1（仅 3P/4P）		0		R
			相序 1： ACB 0： ABC		三相负载 1：不平衡 0：平衡		保留		内部报警		相序保护 1：禁止 0：允许		不平衡保护 1：禁止 0：允许		缺相保护 1：禁止 0：允许		网络控制 1：禁止 0：允许		R
			保留																R

附表 2 CMD 命令码：0x04（0000 0100）**从机参数地址表。**只有标明开关类型的数据才有意义：

功能说明	开关类型	地址	数据意义说明	R/W 特性
线路电压门限	2P/1P/3P/4P	0x00	16 位电压上限，单位 1V	R
线路电压门限	2P/1P/3P/4P	0x01	16 位电压下限，单位 1V	R
漏电电流门限	2P/4P（具有漏电保护的型号）	0x02	16 位漏电电流上限，单位 0.1mA	R
线路功率门限	2P/1P/3P/4P	0x03	16 位功率上限，单位 1W	R
模块温度门限	2P/1P/3P/4P	0x04	16 位温度上限，单位 0.1 度	R
模块电流门限	2P/1P/3P/4P	0x05	16 位电流，单位 0.01A	R
规格	2P/1P/3P/4P	0x06	10 16 20 25 32 40 45 60	R
型号			6-3P/4P，7-2P，8-1P	
版本	2P/1P/3P/4P	0x07	版本号	R

附表 3 CMD 命令码：0x06（0000 0110）**从机参数地址表。**只有标明开关类型的数据才有意义：

功能说明	开关类型	地址	数据域格式，以及命令意义说明	R/W 特性
配置电压门限	2P/3P/4P	0x0000	数据发送电压上门限，单位为 1V	W
配置电压门限	2P/3P/4P	0x0001	数据发送电压下门限，单位为 1V	W
漏电电流门限	2P/3P/4P	0x0002	16 位漏电电流上限，单位 0.1mA	W
配置功率门限	2P/1P/3P/4P	0x0003	数据发送功率门限，单位为 1W	W
配置温度门限	2P/1P/3P/4P	0x0004	数据发送温度上限，单位为 0.1 度	W
配置电流门限	2P/1P/3P/4P	0x0005	数据发送电流门限，单位为 0.01A	W
漏电测试指令	2P/4P（具有漏电保护的型号）	0x0008	数据发送 0x5A 表示测试	W
预警电压阈值上限	2P/3P/4P	0x0014	数据发送预警电压上门限，单位为 1V	W
预警电压阈值下限	2P/3P/4P	0x0015	数据发送预警电压下门限，单位为 1V	W
电量高	2P/1P/3P/4P	0x0016	修改开关电量数据，先发送电量高 16bit（工程指令）	W
电量低	2P/1P/3P/4P	0x0017	修改开关电量数据，再发送电量低 16bit（工程指令），才能修改成功	W

4.12 本从机数据地址的定义表（通信参数修改可能需要重启从机模块）

功能说明	地址	数据域格式，以及命令意义说明	R/W 特性
自动地址触发*	0x01	底层开关自动设定地址触发开关（发送 0xFF00 表示开始，发送 0 表示结束）	W
从机地址修改**	0x02	地址修改需发送 0xBA00+需要修改的地址（1）	W
通信波特率	0x03	发送 0xBA00+需要修改的代号（1-8）2400/4800/9600/14400/19200/28800/38400/56000	W
通信格式	0x04	发送 0xBA00+ 代号（1-RTU，0-ASCII）	W
校验方式	0x05	发送 0xBA00+代号（0-ODD，1-EVEN，2-2 停止位，3-1 停止位）	W
恢复出厂***	0x06	发送 0xFF5A 恢复默认设置	W
远程通讯模式****	0x07	发送 0xBA00+代号（0-正常模式，1-远程模式）	

注：上表红色表示默认值

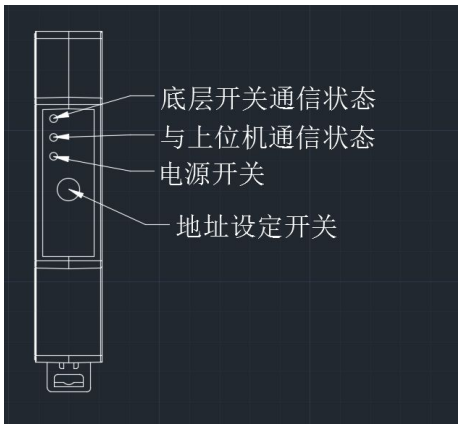
*自动地址流程：全部开关打到分闸，然后发送开始（此时所有开关黄灯会闪烁），然后按下电源后的第一个 On/Off 键 5 秒，然后开关会自动完成编址（所有黄灯熄灭），然后再发送停止命令。此模式可以手动触发，长按 set 键 8 秒即可进入/退出此模式。

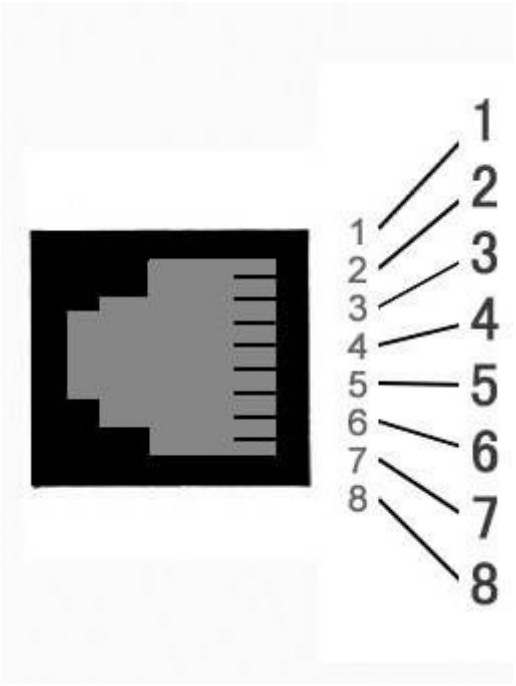
**手动修改从机地址：长按 set 键 10 秒（松开后 SYS 常亮），然后按下几次按键，就是设定成地址几，再长按 set 键 10 秒保存（松开后 SYS 熄灭）

***手动恢复出厂设定：长按 set 键 20 秒，松开后恢复出厂设定。

****针对大延时或远程通讯系统，在 3/4 指令返回时增加返回数据内容地址以及开关起始地址，但此模式不兼容标准格式。

5 硬件接口描述：





1 电源+（采用内置隔离方式的模块无需电源）
2 电源-
3 B
4 A
5 B
6 A
7 电源-
8 电源+（采用内置隔离方式的模块无需电源）

内置信号隔离，电源内置隔离方式的模块无需外部提供电源（默认）。
（如果无内置隔离的模块，需要外接 DC5V 电源，请勿接反。）

端子接线方式：（从左到右，从上到下）

序号	功能
1	485A
2	485B
3	屏蔽线