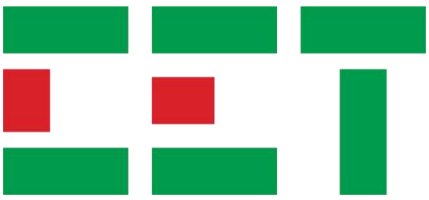
PMC-UP1 不间断电源

Modbus通信协议

（V1.1）



目 录

1 简介

本规约详细地描述了PMC-UP1产品在MODBUS通信模式下的输入和输出命令、信息和数据，为使用PMC-UP1产品的Modbus通信规约提供参考。

1.1 通信协议的目的

通信协议的作用是使信息和数据在上位机主站和装置从站之间有效地传递，它包括：

（1）允许主站访问从站装置的测量值信息，开关量状态，测控信息；

（2）允许访问和设置从站装置的参数；

（3）允许主站对从站装置进行遥控操作；

（4）允许主站访问和设置从站装置的相关装置信息；

（5）允许主站访问从站装置的SOE等。

1.2 通信协议的版本

该通信协议适用于PMC-UP1产品，以后若有改动会特别说明。

2 MODBUS串行通信协议详细说明

2.1 协议基本规则

以下规则确定在RS485或者RS232C回路控制器和其他RS485串行通信回路中设备的通信规则：

（1）所有RS485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最多128个从站监控设备之间传递；

（2）主站将初始化和控制所有在RS485通信回路上传递的信息；

（3）只有主站向从站发送通信命令后，从站才能进行数据通信；

（4）RS485环路以数据“打包”方式进行通信，一个数据包就是一个简单的字符串(每个字符8位)，一个数据包中最多可含255个字节。组成这个数据包的字节构成标准异步串行数据，并按预先设置的传送模式进行传递。串行数据流由类似于RS232C中使用的设备产生；

（5）主站发送数据包称为请求，从站发送数据包称为响应；

（6）任一时刻仅一个从站响应主站的请求。

2.2 传送模式

MODBUS协议可以采用ASCII或者RTU模式传送数据，PMC装置支持RTU模式。

通信波特率： 1200bps、2400bps、4800bps、9600bps、19200bps、38400bps，默认设置9600bps；

校验方式：支持偶校验、奇校验、无校验，默认设置偶校验；数据按照1位启动位、8位数据位、1位校验位、1位或2位停止位传送。

2.3 数据包结构描述

每个MODBUS 数据包都由以下几个部分组成：

（1）地址域

（2）功能码域

（3）数据域

（4）校验域

2.3.1 地址域

MODBUS的从站地址域包含数据包传送的从站地址，长度为一个字节，有效的从站地址范围为1~247。地址0为广播地址，仅应用于广播对时。如果从站接收到数据包中的地址信息与本地地址相同，则执行数据包中所包含的命令。从站所响应的数据包中，该地址域为从站本地地址。

2.3.2 功能码域

MODBUS数据包中功能码域长度为一个字节，用以通知从站应当执行何操作。从站响应数据包中应当包含主站所请求操作的相同功能码域字节。有关本装置的功能码参照下表：

功能码	含义	功能
0x03	读取寄存器	DO 状态、电池电压、电池温度、事件指针数等数据；
0x10	设置一个或多个寄存器、设置线圈	参数设置、清除等操作；

2.3.3 数据域

MODBUS 数据域长度不定，依据其具体功能而定。MODBUS数据域采用“BIG INDIAN”模式，即高位字节在前低位字节在后。例如：

1个16位寄存器包含数值为0x12AB 寄存器数值发送顺序为：

高位字节= 0x12

低位字节= 0xAB

本规约上传的数据类型有char(字符型)、uint16(无符号短整形16位)、int16(有符号短整形16位)、uint32(无符号长整形32位)、int32(有符号长整形32位)、float(浮点数32位)、int64(有符号长整形64位)状态字，具体的上传数据格式说明如下：

uint16和int16：16位的短整形由两个字节组成byte0（bit0~bit7）、byte1(bit8~bit15),寄存器数据发送顺序为:byte1、byte0。

uint32和int32:32位的长整形由四个字节组成byte0(bit0~bit7)、byte1(bit8~bit15)、byte2(bit16~bit23)、byte3(bit24~bit31),寄存器数据发送顺序为：byte3、byte2、byte1、Byte0。

float：浮点数上传的是IEEE754 表示格式4字节，占两个寄存器。例如：2.66为0x71、

0x3D 0x2A、0x40，用两个寄存器上传这四字节的数据,上传的顺序为0x40、0x2A、0x3D、0x71。

int64:64位的长整型由八个字节组成byte0(bit0~bit7)、byte1(bit8~bit15)、byte2(bit16~bit23)、byte3(bit24~bit31)、byte4(bit32~bit39)、byte5(bit40~bit47)、byte6(bit48~bit55)、byte7(bit56~bit63),寄存器数据发送顺序为：byte7、byte6、byte5、byte4、byte3、byte2、byte1、byte0。

2.3.4 校验域

MODBUS RTU模式采用16位CRC校验。发送设备应当对数据包中的每一个数据都进行CRC16计算，最后结果存放入检验域中。接收设备也应当对数据包中的每一个数据(除校验域以外)进行CRC16计算，将结果域校验域进行比较。只有相同的数据包才可以被接受。具体的CRC校验算法参照附录A。

2.4 网络时间考虑

在RS485网络上传送数据包需要遵循以下有关时间的规定：

（1）9600bps下，主站请求数据包结束到从站响应数据包开始之间的时间最大为200毫秒，典型值为50毫秒；

（2）从站响应数据包结束后，主站必须经过至少3.5个字符的传输延迟后才能发送下一个数据请求；

（3）主站发送广播命令后，必须经过至少100ms的延迟才能发送下一个数据请求。

2.5 异常响应

如果主站发送了一个非法的数据包给装置或者是主站请求一个无效的数据寄存器时，异常的数据响应就会产生（如果接收到的数据CRC校验错误，则直接丢弃）。 这个异常响应数据包括从站地址、功能码、故障码和校验域。当功能码的高比特位置为1时，说明此数据包为异常响应。下表说明故障码的含义：

故障码名称	功能码	说 明
0x01(非法功能码)	0x80+原功能码	表示从站接收到不支持的功能码(除上面列举的功能码)。
0x02(非法数据地址)	0x80+原功能码	说明 PMC 装置接收到无效的数据地址或者是请求寄存器不在有效的寄存器范围内(例如对只读寄存器进行写操作，再比如请求寄存器偏移+长度无效)
0x03(非法数据值)	0x80+原功能码	读写数据时寄存器数量超出允许范围或字节数!=寄存器数*2； 写入的数据超出范围 必须连续写的寄存器写入不完整

0x04(操作寄存器失败)	0x80+原功能码	遥控操作失败，或异常码 02、03 规定之外的情况。

2.6 广播命令

广播命令的地址域为 0x00，仅在使用 0x10 功能码时有效。主站发送广播命令时，从站只接收数据包，不返回响应数据包，以防止网络内所有从站同时响应，造成网络堵塞。

3 MODBUS串行通信数据帧

标准的MODBUS协议仅支持16位数据模式，可传输最大为65535的数据，本装置支持浮点数传送。

本装置 MODBUS支持多个功能码，一个寄存器表示16位数据，对于超过一个寄存器的数据，则分成多个寄存器读写。比如，一个32位数据，高字占一个寄存器，低字占一个寄存器。64位数据则用4个寄存器表示。16位数据模式中，数据都是使用两个8位寄存器表示。

3.1 读寄存器（0x03）

由主站发送的数据包请求，本装置响应所有有效的寄存器(在起始寄存器和终止寄存器之间)。命令格式如下：

请求格式(主站→PMC 装置)		响应格式(PMC 装置→主站)	
从站地址	1 字节	从站地址	1 字节
功能码	1 字节	功能码	1 字节
寄存器起始地址高位	1 字节	字节数 n	1 字节
寄存器起始地址低位	1 字节	Data1 高位	1 字节
寄存器数量高位	1 字节	Data1 低位	1 字节
寄存器数量低位	1 字节	……	
		Datan/2 高位	1 字节
		Datan/2 低位	1 字节
CRC 校验码低位	1 字节	CRC 校验码低位	1 字节
CRC 校验码高位	1 字节	CRC 校验码高位	1 字节

备注：一个数据包最少读 1 个寄存器，最多可读 125 个寄存器。寄存器地址本文档均已十进制表示，实际发送报文需要将寄存器地址转换为十六进制。

3.2 写寄存器（0x10）

该命令允许主站设置本装置工作参数，命令格式如下：

请求格式(主站→PMC 装置)		响应格式(PMC 装置→主站)	
从站地址	1 字节	从站地址	1 字节
功能码	1 字节	功能码	1 字节
寄存器起始地址高位	1 字节	寄存器起始地址高位	1 字节
寄存器起始地址低位	1 字节	寄存器起始地址低位	1 字节
寄存器数量高位	1 字节	寄存器数量高位	1 字节
寄存器数量低位	1 字节	寄存器数量低位	1 字节
字节数(n)	1 字节	CRC 校验码低位	1 字节
Data1 高位	1 字节	CRC 校验码高位	1 字节
Data1 低位	1 字节		
……			
Data(n/2) 高位	1 字节		
Data(n/2) 低位	1 字节		
CRC 校验码低位	1 字节		
CRC 校验码高位	1 字节		

备注：一个数据包最少写 1 个寄存器，最多可写 123 个寄存器。

4 装置寄存器说明

本规约规定，请求本装置中一个地址为 xx 寄存器的数据时，主站实际读取寄存器地址 xx 的数据。例如：请求本装置中寄存器地址为 10 的数据，主站传送实际寄存器地址为 10 的数据。详细的寄存器说明如下所示。

4.1 实时测量数据寄存器

刷新周期：1s

地址	类型	描述	数据格式	单位/系数	备注
0	RO	电池电压	float	V	
2	RO	电池温度	float	℃	
4	RO	事件指针数	uint32		
6	RO	DO状态	uint16		0：无市电 1：有市电

4.2 通信参数寄存器

地址	类型	描述	数据格式	单位/范围	备注
6400	RW	通信口 1 地址	uint16	1~247	默认值 100
6401	RW	通信口 1 波特率	uint16	0~5	0:1200，1:2400，2:4800，3:9600，4:19200，5:38400；默认 3
6402	RW	通信口 1 校验方式	uint16	0~5	0: 8N2，1: 8O1，2: 8E1，3: 8N1，4: 8O2，5: 8E2 默认 2

4.3 装置信息寄存器

地址	类型	描述	数据格式	备注
9800～9819	RO	设备类型	uint16	【注1】
9820	RO	程序版本号	uint16	如10000，表示版本V1.00.00
9821	RO	MODBUS规约版本	uint16	如10表示规约版本1.0
9822	RO	程序版本更新日期（年-2000）	uint16	如：100110表示版本日期为2010年1月10日
9823	RO	程序版本更新日期（月）	uint16	
9824	RO	程序版本更新日期（日）	uint16	
9825	RO	装置序列号	uint32	

【注 1】：设备类型寄存器，共含 20 个寄存器，每个寄存器地址存放一个字符的 ASCII 码，多余的寄存器填 0x0020，用于将来扩展。比如其内容为“PMC-UP1”字符串的 ASCII 码，则读 9800～9806 寄存器的值分别是：0x0050、0x004D、0x0043、0x002D、0x0055、0x0050、0x0031，读 9807～9819 寄存器的值均为 0x0020。

4.4 SOE 事件记录寄存器

地址	类型	描述	备注
10000～10007	RO	事件1	SOE LOG【注1】
10008～10015	RO	事件2	SOE LOG
10016～10023	RO	事件3	SOE LOG
10024～10031	RO	事件 4	SOE LOG

10032～10039	RO	事件 5	SOE LOG
10040～10047	RO	事件 6	SOE LOG
10048～10055	RO	事件 7	SOE LOG
10056～10063	RO	事件 8	SOE LOG
10064～10071	RO	事件 9	SOE LOG
10072～10079	RO	事件 10	SOE LOG
10080～10087	RO	事件 11	SOE LOG
10088～10095	RO	事件 12	SOE LOG
……		……	
12040~12047	RO	事件256	SOE LOG

【注 1】SOE LOG 定义如下，每条事件占用8个寄存器地址：

偏移地址	意义
+0	Hi：类 Lo：子类
+1	无效
+2	无效
+3	无效
+4	无效
+5	双位置信息
+6+7	记录值（浮点数）

一条完整的SOE包含8个寄存器，每次读取SOE必须读取完整的事件记录，并且要从每条事件记录的起始地址开始读取；SOE事件定义，详见附录B。

4.5 操作寄存器

地址	类型	描述	数据格式	范围/备注
9600	WO	清除事件记录	uint16	写 0xFF00，进行清除

附录 A、CRC-16 校验算法

使用RTU模式，消息包括了一基于CRC方法的错误检测域。CRC域检测了整个消息的内容。

CRC域是两个字节，包含一16位的二进制值。它由传输设备计算后加入到消息中。接收设备重新计算收到消息的CRC，并与接收到的CRC域中的值比较，如果两值不同，则有误。

CRC是先调入一值是全“1”的16位寄存器，然后调用一过程将消息中连续的8位字节各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或(OR)，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或一下，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或。最终寄存器中的值，是消息中所有的字节都执行之后的CRC值。

CRC添加到消息中时，低字节先加入，然后高字节。

CRC简单函数如下：

```
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg ；  /* 要进行CRC校验的消息 */
unsigned short usDataLen ；  /* 消息中字节数 */
{
    unsigned char uchCRCHi = 0xFF ；  /* 高CRC字节初始化 */
    unsigned char uchCRCLo = 0xFF ；  /* 低CRC 字节初始化 */
    unsigned uIndex ；  /* CRC循环中的索引 */
    while (usDataLen--) /* 传输消息缓冲区 */
    {
            uIndex = uchCRCHi ^ *puchMsgg++ ；  /* 计算CRC */
            uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex]；
            uchCRCLo = auchCRCLo[uIndex] ；
     }
     return (uchCRCHi << 8 | uchCRCLo) ；
}
/* CRC 高位字节值表 */
static unsigned 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

} ;

附录 B、事件定义

类别：

大类类别	大类表示意义	小类表示意义	双位置信息	记录值意义
3	告警事件	告警或返回事件类型	1：告警；0：返回；	告警值或返回值
5	操作事件	操作事件类型	0：无效	无记录值

详细描述：

类 (class)	子类 (sub-class)	双位置信息	记 录 值 (soe_value)	描述
3	1	1 / 0	告警值/返回值	温度越限/返回值
	2	1 / 0	告警值/返回值	电压越限/返回值
	3	1 / 0	预警值/返回值	温度越限/返回时的电压
	4	1 / 0	预警值/返回值	电压越限/返回时的温度
5	1	0	0	市电上电
	2	0	0	市电掉电
	3	0	0	预留
	4	0	0	通信清除事件记录
	5	0	0	装置掉电

注：温度越上限值：70度、返回值：66.5，电压越上限值：18V、返回值：17.1，越下限值：10V、返回值：10.5。

修订记录

序号	版本号	修改日期	修改摘要	修改人
1	V1.0	2022-10-12	第一版	杨凯雄
2	V1.1	2024-4-3	SOE3 大类中增加子类 2、3、4	张逸飞