

广东易事特电源股份有限公司	文件编号		文件版本	V1.0
	文件密级	秘密	生效日期	2010.10
	制定部门	软件部		

电力 UPS 产品 MODbus 通讯协议

广东易事特电源股份有限公司	文件编号		文件版本	V1.0
	文件密级	秘密	生效日期	2010.10
	制定部门	软件部		

序号	版本	修改内容	修改时间	备注
1	Ver 1.0	确定基本的电气量	2010-10	

1、通讯接口

串口：RS485

波特率：初始化 2400bps，可设置

起始位：1 bit

数据长度：8 Bits

校验位：无

停止位：1 bit

UPS 访问地址：初始化 0x01，可设置

2、报文格式

起始位	设备地址	功能代码	数据域	CRC校验	结束符
T1-T2-T3-T4	8Bit	8Bit	N个8Bit	16Bit	T1-T2-T3-T4

T1-T2-T3-T4 为 4 字符的时间间隔；

1、UPS 请求命令格式：

定义	地址	功能码	寄存器地址		寄存器个数		CRC校验	
数据	ADDR	03H	高位	低位	高位	低位	低位	高位
字节数	1	1	2		2		2	

2、UPS 请求响应格式：

定义	地址	功能码	应答数据字节数		返回数据		CRC校验	
数据	ADDR	03H	高位	低位	高位	低位	低位	高位
字节数	1	1	2		2		2	

特别说明：

消息发送至少需要 3.5 个字符时间的停顿间隔开始。在最后一个传输字符之后，需要至少 3.5 个字符时间的停顿来标定消息的结束。一个新的消息可在此停顿后开始。

整个消息帧必须作为一连续的流转输。如果在帧完成之前两个字符间有超过 1.5 个字符空闲的停顿时间，认为帧错误，停止接收，并重新启动接收。也就是要保证两个帧间的间隔至少大于 3.5 个字符的时间，1.5 个字符时间和 3.5 个字符时间与具体的通信波特率有关，计算方法如下：如通信波特率为 9600，那么

$$1.5 \text{ 个字符间隔时间} = (1/9600) \times 11 \times 1.5 \times 1000 = 1.72 \text{ ms}$$

$$3.5 \text{ 个字符间隔时间} = (1/9600) \times 11 \times 3.5 \times 1000 = 4.01 \text{ ms}$$

实际在发送数据前要求总线静止时间即无数据发送时间大于或等于 5ms（波特率为 9600bps

时)

地 址：UPS 监控模块的通讯地址，每台 UPS 的地址可设置；

数据长度：命令帧中数据长度为寄存器个数，响应帧中数据长度为返回的数据字节数；

数据格式：寄存器地址、寄存器个数、应答数据字节数和返回数据都是高字节先发，低字节后发；

CRC16 校验：错误检测域包含 16Bits 值。错误检测域的内容是通过将消息内容进行循环冗长检测方法得出的，消息内容包括地址、功能码、起始寄存器地址或应答数据字节数、寄存器个数或返回数据，CRC 域附加在消息的最后，添加时先是低字节然后是高字节。故 CRC 的高位字节是发送消息的最后一个字节。

3、遥测寄存器定义

遥测量可以用功能码 0x03 读取 0x0000~0x000E 一个或多个寄存器的数据；

寄 存 器	名 称	系 数	单 位
0000	输入A相电压Hi	0.1	V
	输入A相电压Lo		
0001	输入B相电压Hi	0.1	V
	输入B相电压Lo		
0002	输入C相电压Hi	0.1	V
	输入C相电压Lo		
0003	输出电压Hi	0.1	V
	输出电压Lo		
0004	输出频率Hi	0.1	Hz
	输出频率Lo		
0005	输出电流Hi	0.1	A
	输出电流Lo		
0006	直流电压Hi	0.1	V
	直流电压Lo		
0007	电池充电电流Hi	0.1	A
	电池充电电流Lo		
0008	电池节数Hi	1	P
	电池节数Lo		
0009	机器温度Ho	1	℃
	机器温度Lo		

000A	Bit8~Bit15		未使用	
	Bit	信息名称	描述	
	0	市电输入	0: 正常	1: 输入异常
	1	电池低压	0: 正常	1: 电池低压
	2	逆变输出/旁路输出	0: 逆变输出	1: 旁路输出
	3	过载	0: 正常	1: UPS过载
	4	过温	0: 正常	1: UPS过温
	5	整流器工作状态	0: 正常	1: 故障
	6	逆变器工作状态	0: 正常	1: 故障
	7	市电相序	0: 正常	1: 故障

UPS 遥信寄存器定义

遥信量可以用功能码 0x04 读取 0x0000~0x0007 一个或多个寄存器的数据;

地址	信息名称	描述	
0000	市电输入	0000: 正常	0001: 输入异常
0001	电池低压	0000: 正常	0001: 电池低压
0002	逆变输出/旁路输出	0000: 逆变输出	0001: 旁路输出
0003	过载	0000: 正常	0001: UPS过载
0004	过温	0000: 正常	0001: UPS过温
0005	整流器工作状态	0000: 正常	0001: 故障
0006	逆变器工作状态	0000: 正常	0001: 故障
0007	市电相序	0000: 正常	0001: 故障

UPS 遥控量

遥信量可以用功能码 0x06 向一个寄存器写数据, 实现控制和设置功能;

地址	信息名称	描述
0001	通讯地址设置	0x0001~0x00FF (1~255) 初始化: 0x0001
0002	通讯波特率设置	0001:1200bps 0002:2400bps 0003:4800bps 0004:9600bps 初始化: 2400bps
0003	远程开机	写入0x00FF 实现

0004	远程关机	写入0x00FF 实现
0005	消除报警（声音）	写入0x00FF 实现
0006	电池测试	写入0x00FF 实现

5、功能代码

03	读寄存器数据	读取一个或多个寄存器数据
04	读遥信量功能码	读取一个或多个寄存器数据
06	遥控（设置功能）	对寄存器写数据实现控制、设置功能

例如：主机要读取从机地址为 03 的 UPS 的输入三相电压；

PC 查询指令：03030000000185E8

03：UPS 机器地址；

03：功能码；

0000：查询寄存器首地址；

0001：查询寄存器个数；

85E8：CRC 校验码，低位在前高位在后；

UPS 返回报文：0303020898C7EE

03：UPS 机器地址；

03：功能码；

02：返回查询数据的字节数；

0898：寄存器 0000 的数值，输入电压 2200；

C7EE：CRC 校验码，低位在前高位在后；

主机查询

读保持寄存器 0000H 以后 1 个寄存器的内容；查询信息规定了要读的寄存器起始地址及寄存器的数量，寄存器寻址起始地址为 0000H。

主机发送	发送信息	字节数	备注
从机地址	03	1	02号机
功能代码	03	1	读寄存器
开始地址高位	00	2	输入电压 起始地址
开始地址低位	00		
寄存器数量高位	00	2	1个寄存器
寄存器数量低位	01		

CRC16	XX	2	低位在前
	XX		高位在后
字节总数		8	

从机响应

当对应从机接收到主机发出的查询命令后，回返回一串信息作为响应信息，响应信息中的寄存器数据为二进制数据，每个寄存器分别对应 2 个字节，第一个字节为高位值数据，第二个字节为低位数据。

从机响应	返回数据	字节数	备注
从机地址	03	1	02号机
功能代码	03	1	读寄存器
应答数据字节数	02	1	3个寄存器6字节
DATA Hi (0000)	08	2	输入A相电压 220V
DATA Li (0000)	98		
DATA Hi (0001)	08	2	输入B相电压 220V
DATA Li (0001)	98		
DATA Hi (0002)	08	2	输入C相电压 220V
DATA Li (0002)	98		
CRC16	XX	2	低位在前
	XX		高位在后
字节总数		11	

6、CRC 检测

CRC 码的计算方法是：

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。

7、通讯错误信息及数据的处理：

当从机检测到除了 CRC 码出错以外的错误时，必须向主机回送信息，功能码的最高位置为 1，即从机返送给主机的功能码是在主机发送的功能码的基础上加 128。以下的这些代码表明有意外的错误发生。

- 从机如果从主机接收到的信息有 CRC 错误，则不返回任何信息；
- 从机如果从主机接收到的信息有除了 CRC 错误以外的错误，则返回以下格式的信息：

地址码： 1 字节

功能码： 1 字节（最高位为 1）

错误码： 1 字节

CRC 码： 2 字节。

从机响应如下错误码(十六进制)：

81. 非法的功能码。

接收到的功能码从机不支持。

82. 读取非法的数据地址。

指定的数据位置超出从机的可读取的地址范围。

83.

接收到主机发送的数据值超出从机相应地址的数据范围。

附件：CRC 校验算法程序（查表方式）

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,
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,

```

