

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	2011.07
	制定部门	软件部		

# UPS 产品 EA88 系列

## 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	2011.07
	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	2011-7-8	

## 目 录

一、协议相关说明 .....	1
1、协议简介 .....	1
2、接口方式 .....	1
3、协议格式 .....	1
3.1 RTU 模式的帧格式 .....	1
4、响应信息分类 .....	2
5、功能代码 .....	4
二、通信内容 .....	5
1、遥测量（功能码 0x03） .....	5
2、遥信量（功能码 0x04） .....	7
附录 A LRC/CRC 校验 .....	10
LRC 纵向冗余校验 .....	10
CRC 循环冗余校验 .....	10
附录 B 高低位字节表 .....	11
高位字节表 .....	11
低位字节表 .....	11

## 一、协议相关说明

### 1、协议简介

Modbus 协议是应用于控制器上的一种通用语言。通过该协议使控制器经由网络和其他 UPS 设备之间可以进行通信。本通信采用应答方式，由主机发起请求（发送遥测、遥信信息），从机执行请求并且应答。从机需通过地址设置加以区分，从机可设置的地址范围为 0x01~0xFF。

### 2、接口方式

RS485 接口：异步，半双工；如图 1 所示为监控板接口示意图，RS485 接口如(c)所示；

波特率：可设置为 1200bps、2400 bps、4800 bps、9600 bps、14400 bps 、19200 bps

数据长度：RTU 模式时为 8 位、ASCII 模式时为 7 位

奇偶校验位：可设置为奇校验、偶校验或者无校验

停止位：1 位

### 3、协议格式

本协议支持 MODbus 通信可选，支持 RTU 模式

#### 3.1 RTU 模式的帧格式

控制器以 RTU 模式在 Modbus 总线上进行通讯时，信息中的每个字节按十六进制。RTU 模式中每个字节的格式为：

编码系统：8 位二进制；

起始位：1 位；

数据位：8 位；

奇/偶校验：奇校验或者偶校验时为 1 位；无奇偶校验时该位为 1 位停止位；

停止位：1 位；

错误校验区：循环冗余校验(CRC)；

RTU 模式的请求帧格式为：

起始	设备地址	功能代码	寄存器 起始地址	寄存器 个数	CRC 高字节	CRC 低字节	结束
至少 3.5 个 字符空闲时间	1 byte	1 byte	2 bytes	2 bytes	1 byte	1 byte	至少 3.5 个 字符空闲时间

其中 RTU 模式字符传输格式采用 11 位传输，其中数据位为 8 位；位序列为：

起始位	1	2	3	4	5	6	7	8	停止位	停止位
-----	---	---	---	---	---	---	---	---	-----	-----

RTU 模式的响应帧格式为：

起始	设备地址	功能代码	数据	CRC 高字节	CRC 低字节	结束
至少 3.5 个字符空闲时间	1 byte	1 byte	N bytes	1 byte	1 byte	至少 3.5 个字符空闲时间

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传输。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

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

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

【例如】\*\*\*

请求帧信息：请求 1 号机的数据，位置为：寄存器起始地址 0002，寄存器个数为 1 个

	地址	功能码	寄存器起始地址		寄存器个数		CRC 校验	
数据	0x01	0x03	0x00	0x02	0x00	0x01	0x25	0xCA
字节数	1	1	2		2		2	

响应帧信息：1 号机的响应帧

	地址	功能码	返回数据字节数	数据内容		CRC 校验	
数据	0x01	0x03	0x02	0x12	0x22	0xE9	0x5C
字节数	1	1	1	2		2	

## 4、响应信息分类

主机向从机设备发送查询并希望有一个正常响应，主机查询中有可能产生 4 种事件：

- (1) 从机接收查询，无通讯错误，正常处理信息，则返回一个正常响应事件。
- (2) 由于通讯出错，从机不能接收查询数据，因而不返回响应。此时，主机依靠处理程序判定为查询超时。
- (3) 若从机接收查询，发现有 (LRC 或 CRC) 通讯错误，不返回响应，此时依靠主机处

理程序判定为查询超时。

(4) 从机接收查询，无通讯错误，但无法处理( 如读不存在的寄存器地址或错误的寄存器个数 )时，向主机报告错误的性质。

向主机报告错误的响应信息有 2 个与正常响应不相同的区域：

**功能代码区：**正常响应时，从机的响应功能代码区，带原查询的功能代码。所有功能代码的 MSB 为 0(其值低于 80H)。不正常响应时，从机把功能代码的 MSB 置为 1，使功能代码值大于 80H，高于正常响应的值。这样，主机应用程序能识别不正常响应事件，能检查不正常代码的数据区。

**数据区：**正常响应中，数据区含有(按查询要求给出的) 数据或统计值，在不正常响应中，数据区为一个不正常代码，它说明从机产生不正常响应的条件和原因。

不正常代码及含义如下表所示：

代码	名称	含义
0x01	不合法功能代码	从机接收的是一种不能执行功能代码。发出查询命令后，该代码指示无程序功能
0x02	不合法数据地址	接收的数据地址，是从机不允许的地址。如：寄存器起始地址错误，查询的寄存器个数错误。
0x03	不合法地址代码	从机接收的访问地址长度超过本机长度错误。
0x04	不合法CRC校验	接收的数据正常，但是也许接收过程受到干扰，导致 CRC 校验出错。

【例如】\*\*\*

RTU 模式：（ASCII 模式类似）

命令信息：请求 1 号机的数据，位置为：寄存器起始地址 0066，寄存器个数为 2 个

	地址	功能码	寄存器起始地址		寄存器个数		CRC 校验	
数据	0x01	0x03	0x00	0x66	0x00	0x02	0x24	0x14

响应信息：1 号机的响应帧，因为寄存器起始地址错误，因此返回信息为不合法的数据地址

	地址	功能码	数据内容	CRC 校验	
数据	0x01	0x83	0x02	0xC0	0xF1

## 5、功能代码

功能码	名称	作用
0x03	读取保持寄存器	在一个或多个保持寄存器取得当前的二进制值（可作为获取模拟量功能码）
0x04	读输入寄存器	读从机输入寄存器中的二进制数据 （可作为获取告警和状态量功能码）

## 二、通信内容

### 1、遥测量（功能码 0x03）

序号 (寄存器)	名称	DATA类型(Hi-Lo)	系数	单位	备注
0	交流输入电压R	Unsigned int	0.1	伏特V	
1	交流输入电压S	Unsigned int	0.1	伏特V	若为单机型此项 为0
2	交流输入电压T	Unsigned int	0.1	伏特V	若为单机型此项 为0
3	交流输入最大值	Unsigned int	0.1	伏特V	
4	交流输入最小值	Unsigned int	0.1	伏特V	
5	交流输入错误电压值	Unsigned int	0.1	伏特V	
6	交流输入频率	Unsigned int	0.1	安培Hz	
7	交流输出电压	Unsigned int	0.1	伏特V	
8	交流输出电流	Unsigned int	0.1	安培A	
9	交流输出频率	Unsigned int	0.1	伏特V	
10	交流旁路电压	Unsigned int	0.1	伏特V	
11	系统温度	Unsigned int	0.1	摄氏度℃	
12	交流旁路频率	Unsigned int	0.1	安培Hz	
13	电池电压	Unsigned int	0.1	伏特V	
14	直流母线电压	Unsigned int	0.1	伏特V	
15	(保留)				
16	直流母线电流	Unsigned int	0.1	安培A	
17	旁路负载电流	Unsigned int	0.1	安培A	
18	逆变负载电流	Unsigned int	0.1	安培A	
19	(保留)				
20	(保留)				
21	输出有功功率	Unsigned int	0.1	安培W	
22	输出总功	Unsigned int	0.1	安培VA	
23	输出负载百分比	Unsigned int	1	%	
24	(保留)	Unsigned int	0.1	安培A	
25	Ups模式				注1

**注 1:**

UPS 模式代表符号

0 上电模式 1 待机模式 2 旁路模式 3 市电模式 4 电池模式  
5 电池测试模式 6 错误模式 7 整流模式 8 关机模式

**【例如】\*\*\***

假设 UPS 设备地址设置为 0x18，查询寄存器起始地址值为 0x0008，寄存器个数为 2 个，即查询“交流输出电流 B”和“交流输出电流 C”的值；假设此时“交流输出电流 B”的值为 89.2A，交流输出电流 C”的值为 88.9A，根据该值的系数为 0.1，那么：

寄存器 0x0008 的值为：(892)D = (037C)H

寄存器 0x0009 的值为：(889)D = (0379)H

则返回数据的字节数为 4 个，RTU 模式时，对数据查询的请求帧信息和响应帧信息为：

请求帧信息为：

	地址	功能码	寄存器起始地址	寄存器个数	CRC校验
数据	0x18	0x03	0x0008	0x0002	0xC047

响应帧信息为：

	地址	功能码	返回数据字节数	数据内容	CRC校验
数据	0x18	0x03	0x04	0x037C 0x0379	0x7C72

对上述情况采用 ASCII 模式时，对数据查询的请求帧信息和响应帧信息为：

请求帧信息为：

	起始	地址	功能码	寄存器起始地址	寄存器个数	LRC	结束
数据	:	0x18	0x03	0x0008	0x0002	0xDB	CRLF
ASCII	0x3A	0x3138	0x3033	0x3030 0x3038	0x3030	0x3032 0x4442	0x0D0A

响应帧信息为：

	起始	地址	功能码	返回数据字节数	数据内容	LRC	结束
数据	:	0x18	0x03	0x04	0x037C 0x0379	0xE6	CRLF
ASCII	0x3A	0x3138	0x3033	0x3034	0x3033 0x3743 0x3033 0x0D0A	0x4536	0x0D0A

## 2、遥信量（功能码 0x04）

序号(寄存器)	名称	类型	备注
100	输入模式	Unsigned int	0x0000: 单相输入 0x0001: 三相输入
101	过载报警状态	Unsigned int	0x0000: 正常 0x0001: 过载报警
102	电池状态	Unsigned int	0x0000: 正常 0x0001: 电池开路
103	市电频率状态	Unsigned int	0x0000: 正常 0x0001: 频率过低 0x0002: 频率过高
104	输出频率状态	Unsigned int	0x0000: 正常 0x0001: 频率过低 0x0002: 频率过高
105	旁路频率状态	Unsigned int	0x0000: 正常 0x0001: 频率过低 0x0002: 频率过高
106	市电状态	Unsigned int	0x0000: 正常 0x0001: 市电丢失
107	逆变软启动状态	Unsigned int	0x0000: 正常 0x0001: 逆变软启动超时
108	继电器状态	Unsigned int	0x0000: 正常 0x0001: 继电器错误
109	逆变器状态	Unsigned int	0x0000: 正常 0x0001: 逆变电压短路 0x0002: 逆变电压过低 0x0003: 逆变电压过高
110	过载故障状态	Unsigned int	0x0000: 正常 0x0001: 过载故障
111	温度状态	Unsigned int	0x0000: 正常 0x0001: 过温故障
112	(保留)		
113	直流母线状态	Unsigned int	0x0000: 正常 0x0002: 母线电压过低

			0x0003: 母线电压过高
114	R相输入电压状态	Unsigned int	0x0000: 正常 0x0001: 输入电压过低 0x0002: 输入电压过高
115	S相输入电压状态	Unsigned int	0x0000: 正常 0x0001: 输入电压过低 0x0002: 输入电压过高
116	T相输入电压状态	Unsigned int	0x0000: 正常 0x0001: 输入电压过低 0x0002: 输入电压过高
117	旁路电压状态	Unsigned int	0x0000: 正常 0x0001: 输入电压过低 0x0002: 输入电压过高
118	(保留)		
119	故障模式	Unsigned int	0x0000: 正常 0x0001: UPS为故障模式

注:

以上 03、04 功能所提供的数据是本系列机型的通用数据, 由于各个容量及非标准机型可能有所差异。

【例如】\*\*\*

假设UPS设备地址设置为0x18, 查询寄存器起始地址值为103, 即0x0067, 寄存器个数为1个, 即查询“电池状态”; 假设此时“电池状态”为放电状态, 即0x08。

则返回数据的字节数为1个, RTU模式时, 对状态查询的请求帧信息和响应帧信息为:

请求帧信息为:

	地址	功能码	寄存器起始地址	寄存器个数	CRC校验
数据	0x18	0x04	0x0067	0x0001	0x1C82

响应帧信息为:

	地址	功能码	寄存器起始地址	数据内容	CRC校验
数据	0x18	0x04	0x02	0x0008	0x1347

对上述情况采用ASCII模式时, 对数据查询的请求帧信息和响应帧信息为:

请求帧信息为:

	起始	地址	功能码	寄存器起始地址	寄存器个数	LRC	结束
--	----	----	-----	---------	-------	-----	----

数据	:	0x18	0x04	0x0067		0x0001		0x7C	CRLF
ASCII	0x3A	0x3138	0x3034	0x3030	0x3637	0x3030	0x3031	0x3743	0x0D0A

响应帧信息为：

	起始	地址	功能码	返回数据字节数	寄存器个数		LRC	结束
数据	:	0x18	0x04	0x0067	0x0008		0xDB	CRLF
ASCII	0x3A	0x3138	0x3034	0x3032	0x3030	0x3038	0x4442	0x0D0A

## 附录 A LRC/CRC 校验

---

- LRC 校验
- CRC 校验

### LRC 纵向冗余校验

LRC 域是一个包含一个 8 位二进制值的字节。LRC 值由传输设备来计算并放到消息帧中，接收设备在接收消息的过程中计算 LRC，并将它和接收到消息中 LRC 域中的值比较，如果两值不等，说明有错误。

LRC 校验比较简单，它在 ASCII 协议中使用，检测了消息域中除开始的冒号及结束的回车换行号外的内容。它仅仅是把每一个需要传输的数据按字节叠加后取反加 1 即可。

### CRC 循环冗余校验

循环冗余校验 CRC 区为 2 字节，含一个 16 位二进制数据。由发送设备计算 CRC 值，并把计算值附在信息中，接收设备在接收信息时，重新计算 CRC 值，并把计算值与接收的在 CRC 区中实际值进行比较，若两者不相同，则产生一个错误。

CRC 开始时先把寄存器的 16 位全部置成“1”，然后把相邻 2 个 8 位字节的数据放入当前寄存器中，只有每个字符的 8 位数据用作产生 CRC，起始位、停止位和奇偶校验位不加入到 CRC 中。

产生 CRC 期间，每 8 位数据与寄存器中值进行异或运算，其结果向右移一位(向 LSB 方向)，并用“0”填入 MSB，检测 LSB，若 LSB 为“1”则与预置的固定值异或，若 LSB 为“0”则不作异或运算。

重复上述过程，直至移位 8 次，完成第 8 次移位后，下一个 8 位数据，与该寄存器的当前值异或，在所有信息处理完后，寄存器中的最终值为 CRC 值。

## 附录 B 高低位字节表

### 高位字节表

/\* Table of CRC values for high-order byte \*/

```
static unsigned int 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, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40 } ;
```

### 低位字节表

/\* Table of CRC values for low-order byte \*/

```
static unsigned int 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, 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 };