广东易事特电源股份有限公司	文件编号		文件版本	V1.2
	文件密级	秘密	生效日期	2011.11
	制定部门	软件部		

UPS 产品 EA66 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	V1.2
	文件密级	秘密	生效日期	2011.11
	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	2012-4-6	
2	Ver 1.1	修改 04 功能代码	2012-09-01	
3	Ver 1.2	修改 02 功能码	2014-04-07	

目 录

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 1~247。

2、接口方式

RS485 接口：异步，半双工

波特率：　　可设置为 1200bps、2400 bps、4800 bps、9600 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 RTU 模式和 ASCII 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 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 个字符间隔时间 =（1/9600）×11×1.5×1000 = 1.72 ms

3.5 个字符间隔时间 =（1/9600）×11×3.5×1000 = 4.01 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	

3.2 ASCII 模式的帧格式

当控制器以 ASCII 模式在 Modbus 总线上进行通讯时，一个信息中的每 1 字节作为 2 个 ASCII 字符传输，ASCII 码每个字节的格式为：

编码系统 ：16 进制，ASCII 字符‘0’-‘9’，‘A’-‘F’

起始位 ：1 位；

数据位 ：7 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　：纵向冗余校验（LRC）；

ASCII 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	LRC	结束
: (0x3A)	2 bytes	2 bytes	4 bytes	4 bytes	2 bytes	**CRLF (0x0D0A)**

其中 ASCII 模式字符传输格式，采用 10 位传输，其中 7 位数据位，位序列为：

起始位	1	2	3	4	5	6	7	停止位（奇偶校验位）	停止位

ASCII 模式的响应帧格式为：

起始	设备地址	功能代码	数据内容	LRC	结束
: (0x3A)	2 bytes	2 bytes	4N bytes	2 bytes	**CRLF (0x0D0A)**

ASCII 模式帧格式的帧头为“0x3A”，帧尾为”0x0D”和”0x0A”。字符之间发送的最大间隔为 1s，若大于 1s，则接收设备认为出现了一个错误。在 ASCII 模式下，数据字节全部以 ASCII 码方式发送，先发送高 4 位，然后发送低 4 位。例如：0x01，会传输 0x30，0x31 两个 ASCII 字符。此时数据采用 LRC 校验，校验涵盖从从机地址到数据的信息部分。校验和等于所有参与校验数据的字符和(舍弃进位位)的补码+1。

【例如】***

请求帧信息：请求 1 号机 002 参数的数据帧，数据个数为 1 个：

	起始	地址	功能码	寄存器起始地址		寄存器个数		LRC	结束
字符	:	0x01	0x03	0x00	0x02	0x00	0x01	F9	CRLF
ASCII	0x3A	0x3031	0x3033	0x3030	0x3032	0x3030	0x3031	0x4639	0x0D0A

响应帧信息为：写入 4000(即 0x0FA0)到从机 1 的内部寄存器 0002 如下表：

	起始	地址	功能码	返回数据字节数	数据内容		LRC	结束
字符	:	0x01	0x06	0x02	0x0F	0xA0	0x48	CRLF
ASCII	0x3A	0x3031	0x3036	0x3032	0x3046	0x4130	0x3438	0x0D0A

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。
0x03	不合法数据	查询数据区的值是从机不允许的值。
0x04	从机设备故障	从机执行主机请求的动作时出现不可恢复的错误。
0x08	内存奇偶校验错误	从机读扩展内存中的数据时，发现有奇偶校验错误，主机按从机的要求重新发送数据请求。

【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
0x02	读离散量输入	读从机离散量输入中的二进制数据 （获取告警功能码）
0x04	读输入寄存器	在一个或多个保持寄存器取得当前的二进制值 （获取模拟量功能码）
0x06	写单个寄存器	写从机上的单个寄存器 （设置单个参数的功能码）
0x10	写多个寄存器	写从机上的多个寄存器 （设置多个参数的功能码）

二、寄存器列表

1. 读输入寄存器（功能码 0x04）

模拟量表格

地址		数据内容	数据长度/格式	说明		
HEX	DEC			单位	系数	备注
0x0000	0	版本号	2bytes	1	0.1	程序版本号
0x0001	1	R 相输入电压	2bytes	V	1	
0x0002	2	S 相输入电压	2bytes	V	1	
0x0003	3	T 相输入电压	2bytes	V	1	
0x0004	4	R 相输入频率	2bytes	Hz	0.1	
0x0005	5	S 相输入频率	2bytes	Hz	0.1	
0x0006	6	T 相输入频率	2bytes	Hz	0.1	
0x0007	7	R 相旁路电压	2bytes	V	1	
0x0008	8	S 相旁路电压	2bytes	V	1	
0x0009	9	T 相旁路电压	2bytes	V	1	
0x000A	10	R 相旁路频率	2bytes	Hz	0.1	
0x000B	11	S 相旁路频率	2bytes	Hz	0.1	
0x000C	12	T 相旁路频率	2bytes	Hz	0.1	
0x000D	13	R 相输出电压	2bytes	V	0.1	
0x000E	14	S 相输出电压	2bytes	V	0.1	
0x000F	15	T 相输出电压	2bytes	V	0.1	
0x0010	16	R 相输出电流	2bytes	A	0.1	
0x0011	17	S 相输出电流	2bytes	A	0.1	
0x0012	18	T 相输出电流	2bytes	A	0.1	
0x0013	19	R 相输出频率	2bytes	Hz	0.1	
0x0014	20	S 相输出频率	2bytes	Hz	0.1	
0x0015	21	T 相输出频率	2bytes	Hz	0.1	
0x0016	22	R 相有功功率	2bytes	KW	1	
0x0017	23	S 相有功功率	2bytes	KW	1	
0x0018	24	T 相有功功率	2bytes	KW	1	
0x0019	25	R 相视在功率	2bytes	KVA	1	

0x001A	26	S 相视在功率	2bytes	KVA	1	
0x001B	27	T 相视在功率	2bytes	KVA	1	
0x001C	28	R 相负载率	2bytes	%	1	
0x001D	29	S 相负载率	2bytes	%	1	
0x001E	30	T 相负载率	2bytes	%	1	
0x001F	31	正充电电压	2bytes	V	1	
0x0020	32	负充电电压	2bytes	V	1	
0x0021	33	正充电电流	2bytes	A	0.1	
0x0022	34	负充电电流	2bytes	A	0.1	
0x0023	35	充电器温度	2bytes	℃	0.1	
0x0024	36	正电池电压	2bytes	V	1	
0x0025	37	负电池电压	2bytes	V	1	
0x0026	38	电池容量	2bytes	%	1	
0x0027	39	剩余放电时间	2bytes	min	1	
0x0028	40	电池温度 1	2bytes	℃	1	
0x0029	41	电池温度 2	2bytes	℃	1	
0x002A	42	电池温度 3	2bytes	℃	1	
0x002B	43	电池温度 4	2bytes	℃	1	
0x002C	44	模块温度	2bytes	℃	0.1	
0x002D	45	工作模式	2bytes			1：待机模式 2：旁路模式 3：市电模式 4：电池模式 5：电池自检 6：故障模式 7：变频模式 8：紧急关机模式 9：关机模式
0x002E	46	模块 1~16 存在标志	2bytes			第 N 个模块存在 1<<N-1
0x002F	47	模块 17~32 存在标志	2bytes			第 N 个模块存在 1<<N-1
0x0030	48	模块 1~16 故障标志	2bytes			第 N 个模块故障 1<<N-1

0x0031	49	模块 17~32 故障标志	2bytes			第 N 个模块故障 1<<N-1
0x0032	50	模块 1~16 告警标志	2bytes			第 N 个模块告警 1<<N-1
0x0033	51	模块 17~32 告警标志	2bytes			第 N 个模块告警 1<<N-1
0x0034	52	预留 1	2bytes			
0x0035	53	预留 2	2bytes			
0x0036	54	预留 3	2bytes			

2. 读离散量（功能码 0x02）

状态告警表格

地址		告警/故障	数据长度/格式	说明
HEX	DEC			
0x0000	0	模块 BUS 高压	1 bit	1 表示模块故障发生， 0 未发生
0x0001	1	模块 BUS 低压	1 bit	
0x0002	2	模块 BUS 不平衡	1 bit	
0x0003	3	模块 BUS 短路	1 bit	
0x0004	4	（预留）	1 bit	
0x0005	5	模块 BUS 软启超时	1 bit	
0x0006	6	模块逆变软启超时	1 bit	
0x0007	7	模块逆变高压	1 bit	
0x0008	8	模块逆变低压	1 bit	
0x0009	9	模块 R 相输出短路	1 bit	
0x000A	10	模块 S 相输出短路	1 bit	
0x000B	11	模块 T 相输出短路	1 bit	
0x000C	12	模块 RS 相输出短路	1 bit	
0x000D	13	模块 ST 相输出短路	1 bit	
0x000E	14	模块 TR 相输出短路	1 bit	
0x000F	15	模块 R 相输出负功异常	1 bit	
0x0010	16	模块 S 相输出负功异常	1 bit	
0x0011	17	模块 T 相输出负功异常	1 bit	
0x0012	18	模块过载故障	1 bit	
0x0013	19	模块电池异常	1 bit	
0x0014	20	（预留）	1 bit	
0x0015	21	模块过温故障	1 bit	
0x0016	22	模块同步信号故障	1 bit	
0x0017	23	模块同步脉冲故障	1 bit	
0x0018	24	模块继电器粘死	1 bit	

0x0019	25	市电 SCR 故障	1 bit	
0x001A	26	模块 CAN 总线故障	1 bit	
0x001B	27	模块总负功故障	1 bit	
0x001C	28	模块物理地址冲突	1 bit	
0x001D	29	模块 IGBT 短路	1 bit	
0x001E	30	逆变器异常	1 bit	
0x001F	31	旁路接线错误	1 bit	
0x0020	32	充电器 BUS 高压	1 bit	1 表示充电器故障发生， 0 未发生
0x0021	33	充电器 BUS 低压	1 bit	
0x0022	34	充电器 BUS 不平衡	1 bit	
0x0023	35	充电器 BUS 短路	1 bit	
0x0024	36	充电器 BUS 软启超时	1 bit	
0x0025	37	充电器 Buck 软启超时	1 bit	
0x0026	38	充电器过温	1 bit	
0x0027	39	充电器 Relay 异常	1 bit	
0x0028	40	充电器市电 SCR 异常	1 bit	
0x0029	41	CAN 通讯故障	1 bit	
0x002A	42	充电器过充自杀	1 bit	
0x002B	43	电池反接	1 bit	
0x002C	44	充电器 ID 错误	1 bit	
0x002D	45	（预留）	1 bit	
0x002E	46	（预留）	1 bit	
0x002F	47	（预留）	1 bit	
0x0030	48	紧急关机（模块）	1 bit	1 表示模块告警发生， 0 未发生
0x0031	49	过载异常（模块）	1 bit	
0x0032	50	通信故障（模块）	1 bit	
0x0033	51	UPS 过载（模块）	1 bit	
0x0034	52	电池未接（模块）	1 bit	
0x0035	53	（预留）	1 bit	
0x0036	54	UPS 过流（模块）	1 bit	
0x0037	55	市电旁路不一致（模块）	1 bit	

0x0038	56	（预留）	1 bit	
0x0039	57	读写 EEROM 错误（模块）	1 bit	
0x003A	58	模块风扇故障	1 bit	
0x003B	59	市电相序错误（模块）	1 bit	
0x003C	60	旁路相序错误（模块）	1 bit	
0x003D	61	N 线未接（模块）	1 bit	
0x003E	62	同步信号故障（模块）	1 bit	
0x003F	63	同步脉冲故障（模块）	1 bit	
0x0040	64	市电异常（模块）	1 bit	
0x0041	65	旁路异常（模块）	1 bit	
0x0042	66	电池低压（模块）	1 bit	
0x0043	67	（预留）	1 bit	
0x0044	68	模块未检测到充电器	1 bit	
0x0045	69	（预留）	1 bit	
0x0046	70	旁路频率异常（模块）	1 bit	
0x0047	71	输出过压（模块）	1 bit	
0x0048	72	物理地址冲突（模块）	1 bit	
0x0049	73	R 相 PFC 异常（模块）	1 bit	
0x004A	74	S 相 PFC 异常（模块）	1 bit	
0x004B	75	T 相 PFC 异常（模块）	1 bit	
0x004C	76	继电器异常（模块）	1 bit	
0x004D	77	旁路 STS 异常（模块）	1 bit	
0x004E	78	预留	1 bit	
0x004F	79	预留	1 bit	
0x0050	80	电池过充（充电器）	1 bit	1 表示充电器告警发生， 0 未发生
0x0051	81	电池反接（充电器）	1 bit	
0x0052	82	电池未接（充电器）	1 bit	
0x0053	83	充电器异常	1 bit	
0x0054	84	充电器短路	1 bit	
0x0055	85	充电器未开	1 bit	
0x0056	86	充电器自杀	1 bit	
0x0057	87	充电器通讯异常	1 bit	

0x0058	88	充电器 ID 错误	1 bit	
0x0059	89	LCD 掉线（充电器）	1 bit	
0x005A	90	充电电压设置错误	1 bit	
0x005B	91	充电电流设置错误	1 bit	
0x005C	92	紧急关机（充电器）	1 bit	
0x005D	93	充电器风扇故障	1 bit	
0x005E	94	市电相序错误（充电器）	1 bit	
0x005F	95	N 线未接（充电器）	1 bit	

3. 预置寄存器列表（功能码 0x06，0x10）

预置寄存器

地址		寄存器内容	数据长度/格式	说明
HEX	DEC			
0x0000	0	(预留)	2 bytes	(预留)
0x0001	1	电池测试 10s	2 bytes	写入 0xFFFF 有效; 持续 10s 钟测试后返回。如果测试过程中电池电压低，系统立即返回初始状态。
0x0002	2	电池低压测试	2 bytes	写入 0xFFFF 有效， 系统测试直到电池电压低转逆变供电。
0x0003	3	蜂鸣器开关	2 bytes	写入 0xFFFF 有效， UPS 系统报警时，告警音可打开或取消。
0x0004	4	取消测试命令	2 bytes	写入 0xFFFF 有效， 取消所有正在测试的状态，系统立即恢复为输出状态。
0x0005	5	取消关机命令	2 bytes	写入 0xFFFF 有效， a.如果系统正处于关机等待状态，则可取消关机命令； b.系统若处于关机后的恢复状态，该命令立即恢复系统输出，但 UPS 必须最少维持 10s 的禁止状态。
0x0006	6	电池测试 N 分钟	2 bytes	写入时间 N，N∈[0,99]有效； 持续测试 n 分钟。如果测试过程中电池电压低，系统立即返回；
0x0007	7	定时 N 秒后关机	2 bytes	写入时间 N，N∈[12,600]有效； 在 N 秒后关闭 UPS 系统。
0x0008	8	定时关机后 N 分钟再重启 UPS	2 bytes	写入时间 N，N∈[1,9999]有效； UPS 系统关闭后，再过 N 分钟系统重启。 注意，单独向此寄存器地址操作无效，必须连续向 0x0007 至 0x0008 操作，此功能才会生效。
0x0009	9	设置年份	2 bytes	2012~2099 有效

0x000A	10	设置月份	2 bytes	1~12 有效
0x000B	11	设置日期	2 bytes	根据是否闰月，设置正确的时间，否则会拒绝修改
0x000C	12	设置小时	2 bytes	0~23 有效
0x000D	13	设置分钟	2 bytes	0~59 有效
0x000E	14	设置秒钟	2 bytes	0~59 有效
0x000F	15	(预留)	2 bytes	(预留)
0x0010	16	(预留)	2 bytes	(预留)
0x0011	17	(预留)	2 bytes	(预留)
0x0012	18	(预留)	2 bytes	(预留)
0x0013	19	(预留)	2 bytes	(预留)
0x0014	20	(预留)	2 bytes	(预留)

三、通信内容

1、读输入寄存器（功能码 0x04）

【举例】

假设 UPS 设备地址设置为 0x18，查询寄存器起始地址值为 0x0010，寄存器个数为 2 个，即查询“R 相输出电流”和“S 相输出电流”的值；假设此时“R 相输出电流”的值为 89.2A，“S 相输出电流”的值为 88.9A，根据该值的系数为 0.1，那么：

寄存器 0x0010 的值为：$(892)_D = (037C)_H$

寄存器 0x0011 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	0x04	0x0010	0x0002	0x0772

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	0x04	0x04	0x037C	0x0379	0xCB73

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	0x0010		0x0002		0xD2	CRLF
ASCII	0x3A	0x3138	0x3034	0x3030	0x3130	0x3030	0x3032	0x4432	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	0x04	0x04	0x037C		0x0379		0xE5	CRLF
ASCII	0x3A	0x3138	0x3034	0x3034	0x3033	0x3743	0x3033	0x0D0A	0x4535	0x0D0A

2、读离散量（功能码 0x02）

【举例】

假设 UPS 设备地址设置为 0x18，查询寄存器起始地址值为 51，即 0x0033，寄存器个数为 1 个，即查询“UPS 过载状态” ；假设此时“UPS 已过载” ；即该值为 1。

返回数据时，在该字节中由低位向高位排列，直至 8 个位为止。下一个字节中的 8 个输入位也是从低位到高位排列。若返回的输入位数不是 8 的倍数，则在最后的数据字节中的剩余位直至字节的最高位全部填零。字节的最高位，字节数区。说明了全部数据的字节数

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	0x02	0x0033	0x0001	0xCC4B

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	0x02	0x01	0x01	0x1467

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	0x02	0x0033		0x0001		0xB2	CRLF
ASCII	0x3A	0x3138	0x3032	0x3030	0x3333	0x3030	0x3031	0x4232	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	0x02	0x01	0x01	0xE4	CRLF
ASCII	0x3A	0x3138	0x3032	0x3031	0x3031	0x4534	0x0D0A

3、预置寄存器（功能码 0x06,0x10）

【举例】

假设 UPS 设备地址设置为 0x18，预置寄存器起始地址值为 1，寄存器个数为 1 个，即电池测试 10S。

寄存器内容被预置后返回正常响应；

预置单个寄存器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	预置数据	CRC 校验
数据	0x18	0x06	0x0001	0xFFFF	0xDBB3

响应帧信息为：

	地址	功能码	寄存器地址	预置成功的数据	CRC 校验
数据	0x18	0x06	0x0001	0xFFFF	0xDBB3

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		预置数据		LRC	结束
数据	:	0x18	0x06	0x0001		0xFFFF		0xB2	CRLF
ASCII	0x3A	0x3138	0x3036	0x3030	0x3031	0x4646	0x4646	0x4232	0x0D0A

响应帧信息为：

	起始	地址	功能码	寄存器地址		预置数据		LRC	结束
数据	:	0x18	0x06	0x0001		0xFFFF		0xB2	CRLF
ASCII	0x3A	0x3138	0x3036	0x3030	0x3031	0x4646	0x4646	0x4232	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。

重复上述处过程，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 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 } ;

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, 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 };