易事特集团股份有限公司	文件编号		文件版本	**V1.0**
	文件密级	非密	生效日期	**2022.10**
	制定部门	软件部		

EA990 G5 SNMP 卡（内置）
TCP/IP Modbus/SNMP
通信协议

易事特集团股份有限公司	文件编号		文件版本	**V1.0**
	文件密级	非密	生效日期	**2022.10**
	制定部门	软件部		

序号	版本	修改内容	修改人	修改时间	备注
1	**Ver1.0**	初始版本	Wuzf	2022-10-10	

目 录

第一章：TCP/IP Modbus 协议

一、协议相关说明

1、协议适用范围

本协议文档第一章节规范了设备 EA900 G4 6-20K UPS，由 SNMP 卡（内置）通过 TCP/IP 连接提供 Modbus 接口需求。

2、规范性引用文件

1. RFC791，互联网协议，Sep81 DARPA

2. MODBUS 协议参考指南 Rev J,MODICON，1996 年 6 月，doc#PI_MBUS_300

MODBUS 是一项应用层报文传输协议，用于在通过不同类型的总线或网络连接的设备之间的客户机/服务器通信。目前，使用下列情况实现 MODBUS：

1) 以太网上的 TCP/IP。

2) 各种媒体（有线：EIA/TIA-232-E、EIA/TIA-485-A；光纤、无线等等）上的异步串行传输。

3、Modbus 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。

4、Modbus TCP

Modbus rtu 和 Modbus tcp 两个协议的本质都是 MODBUS 协议，都是靠 MODBUS 寄存器地址来交换数据，但所用的硬件接口不一样，Modbus RTU 一般采用串口 RS232C 或 RS485/422，而 Modbus TCP 一般采用以太网口。ModbusTCP 模式没有额外规定校验，因为 TCP 协议是一个面向连接的可靠协议。TCP 和 RTU 协议非常类似，只要把 RTU 协议的两个字节的校验码去掉，然后在 RTU 数据帧前加上六个字节的前缀并通过 TCP/IP 网络协议发送出去即可。所有的请求通过 TCP 从寄存器端口 502 发出。

请求和响应带有六个字节的前缀，如下：

Modbus TCP 数据帧可分为两部分：MBAP + PDU

MBAP 报文头（7byte）：
byte 0: 事务处理标识符
byte 1: 事务处理标识符
byte 2: 协议标识符
byte 3: 协议标识符
byte 4: 长度字段
byte 5: 长度字段
byte 6: 单元标识符 (原“从站地址”)
PDU 数据：
byte 7: MODBUS 功能码
byte 8 : 传输数据

5、响应信息分类

在出错时，有一系列定义过的异常代码被从站送回。注意到主站会“投机地”发送指令，利用接收到的成功或异常代码来确定支配设备的哪一个 MODBUS 愿意响应以及从站不同可用数据区的大小。

所有的异常通过添加 0x80 到请求的功能代码来标记，跟随此字节的是一个单一的原因字节 如下例所示：

03 12 34 00 01 => 83 02
当索引 0x1234 ，响应异常类型 02“非法的数据地址”
异常情况列举如下：

01 非法的功能 对从站来说，在询问过程中收到的功能代码是不允许的行为。这可能是由于功能代码只适用 于新近的控制器，而不能在所选的单元使用。也可推断出从站处于错误的状态而发出这样的 一种请求，例如未经配置而被要求返回寄存器值。

02 非法的数据地址 对从站来说，在询问过程中收到的数据地址不是允许的地址。更明确一点，参考数值和传输 长度的结合是无效的。对于一个有 100 个寄存器的控制器来说，具有偏移 96 和长度 4 的请
求将能成功，而具有偏移 96 和长度 5 的请求将产生异常 02。

6、功能代码

功能码	名称	作用
0x02	读离散量输入	读从机离散量输入寄存器中的二进制数据
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
0x04	读取输入寄存器	在一个或多个输入寄存器取得当前的二进制值
0x06	写单个保持寄存器	写从机上的单个寄存器（可作为设置单个参数的功能码）
0x10	写多个保持寄存器	写从机上的多个寄存器（可作为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**）

地址		数据内容	数据长度	类型	单位	系数	备注
HEX	DEC						
0x0101	257	电池状态	2bytes	USHORT	/	1	0x0001：电池未连接 0x0002：电池正常 0x0006：电池均充 0x0007：电池浮充 0x0008：电池放电 其他无效
0x0103	259	电池剩余时间	2bytes	USHORT	分	1	
0x0104	260	电池剩余容量	2bytes	USHORT	%	1	
0x0105	261	电池电压	2bytes	SHORT	V	0.1	
0x0106	262	电池电流	2bytes	SHORT	A	0.1	
0x0108	264	保留	2bytes	/	/	/	
0x0109	265	输入相数	2bytes	USHORT	/	1	
0x010A	266	输入电压_A	2bytes	SHORT	V	1	
0x010B	267	输入电压_B	2bytes	SHORT	V	1	
0x010C	268	输入电压_C	2bytes	SHORT	V	1	
0x010D	269	输入电流_A	2bytes	SHORT	A	0.1	
0x010E	270	输入电流_B	2bytes	SHORT	A	0.1	
0x010F	271	输入电流_C	2bytes	SHORT	A	0.1	
0x0110	272	输入频率_A	2bytes	USHORT	Hz	0.1	
0x0111	273	输入频率_B	2bytes	USHORT	Hz	0.1	

0x0112	274	输入频率_C	2bytes	USHORT	Hz	0.1	
0x0119	281	输出供电方式	2bytes	USHORT	/	1	0x0003：正常 0x0004：旁路 0x0005：电池 其他无效
0x011A	282	输出频率	2bytes	USHORT	Hz	0.1	
0x011B	283	输出相数	2bytes	USHORT	/	1	
0x011C	284	输出电压_A	2bytes	SHORT	V	1	
0x011D	285	输出电压_B	2bytes	SHORT	V	1	
0x011E	286	输出电压_C	2bytes	SHORT	V	1	
0x011F	287	输出电流_A	2bytes	SHORT	A	0.1	
0x0120	288	输出电流_B	2bytes	SHORT	A	0.1	
0x0121	289	输出电流_C	2bytes	SHORT	A	0.1	
0x0122	290	输出有功功率_A_高十六位	2bytes	USHORT	W	1	例：高十六位0x0001，低十六位0x010A，合为0x0001010A 转为十进制即 65802W
0x0123	291	输出有功功率_A_低十六位	2bytes	USHORT	W	1	
0x0124	292	输出有功功率_B_高十六位	2bytes	USHORT	W	1	例：高十六位0x0001，低十六位0x010A，合为0x0001010A 转为十进制即 65802W
0x0125	293	输出有功功率_B_低十六位	2bytes	USHORT	W	1	
0x0126	294	输出有功功率_C_高十六位	2bytes	USHORT	W	1	例：高十六位0x0001，低十六位0x010A，合为0x0001010A 转为十进制即 65802W
0x0127	295	输出有功功率_C_高十六位	2bytes	USHORT	W	1	
0x012B	299	旁路频率	2bytes	USHORT	Hz	0.1	
0x012C	300	旁路相数	2bytes	USHORT	/	1	
0x012D	301	旁路电压_A	2bytes	SHORT	V	1	

0x0130	304	旁路电流_A	2bytes	SHORT	A	0.1	
0x0139	313	额定输入电压	2bytes	USHORT	V	1	
0x013A	314	额定输入频率	2bytes	USHORT	Hz	0.1	
0x013B	315	额定输出电压	2bytes	USHORT	V	1	
0x013C	316	额定输出频率	2bytes	USHORT	Hz	0.1	
0x013D	317	额定容量_高十六位	2bytes	USHORT	W	1	例：高十六位0x0001，低十六位0x010A，合为0x0001010A 转为十进制即 65802W
0x013E	318	额定容量_ 低十六位	2bytes	USHORT	W	1	
0x0161	353	UPS 工作模式	2bytes	USHORT	/	1	0x0001：待机模式

							0x0002：旁路模式 0x0003：市电模式 0x0004：电池模式 其他无效
0x0201 ~0x0210	513 ~528	厂家名称	2bytes*16	CHAR	/	/	
0x0211 ~0x0230	529 ~560	UPS 型号	2bytes*32	CHAR	/	/	

2、读离散量（功能码 **0x02**）

地址		告警/故障	数据长度/格式	说明
HEX	DEC			
0x0101	257	通讯异常	1 bit	通信设备与 UPS 之间
0x0103	259	电池供电	1 bit	1：发生
0x0104	260	电池电压低	1 bit	1：发生
0x0105	261	电池即将耗尽	1 bit	1：发生
0x0106	262	UPS 过温	1 bit	1：发生
0x0107	263	输入(市电)故障	1 bit	1：发生
0x0108	264	输出故障	1 bit	1：发生
0x0109	265	输出过载	1 bit	1：发生
0x010A	266	旁路供电	1 bit	1：发生
0x010B	267	旁路故障	1 bit	1：发生
0x010E	270	充电模块故障	1 bit	1：发生
0x0111	273	风扇故障	1 bit	1：发生
0x0118	280	测试中	1 bit	1：发生
0x011B	283	电池放电	1 bit	1：发生
0x011D	285	电池空开断开	1 bit	1：发生
0x011E	286	输入空开断开	1 bit	1：发生
0x011F	287	输出空开断开	1 bit	1：发生
0x0120	288	旁路空开断开	1 bit	1：发生

三、通信内容

1、MODBUS/TCP 数据格式

本部分阐述了通过 MODBUS/TCP 网络携带的 MODBUS 请求和或响应封装的一般格式。（注：请求和响应数据帧中，从功能代码到末尾的结构和 MODBUS RTU 格式相对应）

Modbus TCP 数据帧可分为两部分：MBAP + PDU（以 03 功能码为例）

请求：00 00 00 00 00 06 09 03 00 04 00 01				
	示例	长度/byte	说明	备注
MBAP 报文头	0x00	1	事务处理标识符（高八位）	客户机发起，服务器复制，用于事务处理配对
	0x00	1	事务处理标识符（低八位）	
	0x0000	2	协议标识符号	客户机发起，服务器复制，Modbus 协议=0
	0x0006	2	长度	从本字节下一个到最后
	0x09	1	单元标识符	从机地址
功能码	0x03	1	Modbus 功能码	
数据	0x0004	2	起始地址	
	0x0001	2	寄存器数量	

响应：00 00 00 00 00 05 09 03 02 00 05				
	示例	长度/byte	说明	备注
MBAP 报文头	0x00	1	事务处理标识符（高八位）	客户机发起，服务器复制，用于事务处理配对
	0x00	1	事务处理标识符（低八位）	
	0x0000	2	协议标识符号	客户机发起，服务器复制，Modbus 协议=0
	0x0005	2	长度	从本字节下一个到最后
	0x09	1	单元标识符	从机地址
功能码	0x03	1	Modbus 功能码	
数据	0x02	1	字节数	具体个数以实际读取为准（参考 Modbus）
	0x0005	2	数据	

MODBUS/TCP 中不需要“CRC-16”或“LRC”检查字段。而是采用 TCP/IP 和链路层（以太网）校验和机制来校验分组交换的准确性。

2、读输入寄存器示例（功能码 **0x04**）

假设 UPS 设备地址设置为 0x18，查询寄存器起始地址值为 0x0043，寄存器个数为 2 个，假设 查询“电池电压”和“电池电流”的值，此时“电池电压”的值为 54.1V，“电池电流”的值为 30.9A，根 据该值的系数为 0.1，那么：

寄存器 0x0043 的值为：(541)D = (021D)H

寄存器 0x0044 的值为：(309)D = (0135)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	0x00	0x43	0x00	0x02	0x82	0x16
字节数	1	1	2		2		2	

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	0x021D	0x0135	0x23	0x7C
字节数	1	1	1	4		2	

3、读离散量示例（功能码 **0x02**）

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，离散量个数为 1 个，假设查询该点的离散量为“旁路空开闭合”，此时“旁路空开闭合发生”，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。

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	起始地址		离散量个数		CRC 校验	
数据	0x18	0x02	0x00	0x33	0x00	0x01	0x4B	0xCC
字节数	1	1	2		2		2	

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	0x67	0x14
字节数	1	1	1	1	2	

4、读取设备寄存器（功能码 **0x03**）

假设 UPS 设备地址设置为 0x18，请求读取寄存器 43-44 的内容：

寄存器 0x0043 的值为：(541)D = (021D)H

寄存器 0x0044 的值为：(309)D = (0135)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	0x00	0x43	0x00	0x02	0x37	0xD6
字节数	1	1	2		2		2	

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	0x021D	0x0135	0x22	0xCB
字节数	1	1	1	4		2	

5、设置设备寄存器（功能码 **0x06/0x10**）

假设 UPS 设备地址设置为 0x18，需要把寄存器地址为 0x0001 的值设置为 0x00FF。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	0x00	0x01	0x00	0xFF	0x9A	0x43
字节数	1	1	2		2		2	

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	0x00	0x01	0x00	0xFF	0x9A	0x43
字节数	1	1	2		2		2	

附录 A　CRC 校验

CRC 循环冗余校验

循环冗余校验 (CRC) 域为两个字节，包含一个二进制 16 位值。附加在报文后面的 CRC 的值由发送设备计算。接收设备在接收报文时重新计算 CRC 的值，并将计算结果于实际接收到的 CRC 值相比较。如果两个值不相等，则为错误。

CRC 的计算, 开始对一个 16 位寄存器预装全 1. 然后将报文中的连续的 8 位子节对其进行后续的计算。只有字符中的 8 个数据位参与生成 CRC 的运算，起始位，停止位和校验位不参与 CRC 计算。CRC 的生成过程中， 每个 8–位字符与寄存器中的值异或。然后结果向最低有效位 (LSB) 方向移动(Shift) 1 位，而最高有效位 (MSB) 位置充零。然后提取并检查 LSB：如果 LSB 为 1， 则寄存器中的值与一个固定的预置值异或；如果 LSB 为 0，则不进行异或操作。这个过程将重复直到执行完 8 次移位。完成最后一次（第 8 次）移位及相关操作后，下一个 8 位字节与寄存器的当前值异或，然后又同上面描述过的一样重复 8 次。当所有报文中子节都运算之后得到的寄存器中的最终值，就是 CRC。

生成 **CRC** 的过程为**:**

1. 将一个 16 位寄存器装入十六进制 FFFF (全 1). 将之称作 CRC 寄存器.
2. 将报文的第一个 8 位字节与 16 位 CRC 寄存器的低字节异或，结果置于 CRC 寄存器.
3. 将 CRC 寄存器右移 1 位 (向 LSB 方向)， MSB 充零. 提取并检测 LSB.
4. (如果 LSB 为 0): 重复步骤 3 (另一次移位).

(如果 LSB 为 1): 对 CRC 寄存器异或多项式值 0xA001 (1010 0000 0000 0001).

5. 重复步骤 3 和 4，直到完成 8 次移位。当做完此操作后，将完成对 8 位字节的完整操作。
6. 对报文中的下一个字节重复步骤 2 到 5，继续此操作直至所有报文被处理完毕。
7. CRC 寄存器中的最终内容为 CRC 值.
8. 当放置 CRC 值于报文时，如下面描述的那样，高低字节必须交换。

将 CRC 放置于报文当 16 位 CRC (2 个 8 位字节) 在报文中传送时，低位字节首先发送，然后是高位字节。

例**:**执行 CRC 生成的 C 语言的函数在下面示出。所有的可能的 CRC 值都被预装在两个数组中，当计算报文内容时可以简单的索引即可。一个数组含有 16 位 CRC 域的所有 256 个可能的高位字节，另一个数组含有低位字节的值。这种索引访问 CRC 的方式提供了比对报文缓冲区

的每个新字符都计算新的 CRC 更快的方法。

注意: 此函数内部执行高/低 CRC 字节的交换。此函数返回的是已经经过交换的 CRC 值。 也就是说，从该函数返回的 CRC 值可以直接放置于报文用于发送。

函数使用两个参数:

unsigned char *puchMsg;指向含有用于生成 CRC 的二进制数据报文缓冲区的指针

unsigned short usDataLen; 报文缓冲区的字节数.

CRC 生成函数

```
unsigned short CRC16 ( puchMsg， usDataLen ) /* 函数以 unsigned short 类型返回 CRC */
unsigned char *puchMsg ; /* 用于计算 CRC 的报文 */
unsigned short usDataLen ; /* 报文中的字节数 */

{
    unsigned char uchCRCHi = 0xFF ; /* CRC 的高字节初始化 */
    unsigned char uchCRCLo = 0xFF ; /* CRC 的低字节初始化 */
    unsigned uIndex ;                    /* CRC 查询表索引 */
    while (usDataLen--)                  /* 完成整个报文缓冲区 */
    {
        uIndex = uchCRCLo ^ *puchMsgg++ ; /* 计算 CRC */
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex] ;
        uchCRCHi = auchCRCLo[uIndex] ;
    }
     return (uchCRCHi << 8 | uchCRCLo) ;
```

附录 B 高低位字节表

高字节表

/* 高位字节的 CRC 值 */

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

} ;

低字节表

/* 低位字节的 CRC 值 */

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

};

第二章：SNMP 协议（简单网络管理协议）

一、协议相关说明

1、协议适用范围

本协议文档第二章节规范了设备 EA900 G4 6-20K UPS，由 SNMP 卡（内置）通过 TCP/IP 连接提供

SNMP 协议用于网络管理节点信息的接口需求。

2、SNMP 协议简介

简单网络管理协议(SNMP：Simple Network Management Protocol)是专门设计用于在 IP 网络管理网络节点（服务器、工作站、路由器、交换机及 HUBS 等）的一种标准协议，是一种应用层协议。SNMP 使网络管理员能够管理网络效能，发现并解决网络问题以及规划网络增长。通过 SNMP 接收随机消息（及事件报告）网络管理系统获知网络出现问题。

SNMP 的前身是简单网关监控协议（SGMP），用来对通信线路进行管理。随后，人们对 SGMP 进行了很大的修改，特别是加入了符合 Internet 定义的 SMI 和 MIB，改进后的协议就是著名的 SNMP。基于 TCP/IP 的 SNMP 网络管理框架是工业上的现行标准，由 3 个主要部分组成，分别是管理信息结构 SMI、管理信息库 MIB 和管理协议 SNMP，定义如下：

1）SMI 定义了 SNMP 框架所用信息的组织和标识，为 MIB 定义管理对象及使用管理对象提供模板。

2）MIB 定义了可以通过 SNMP 进行访问的管理对象的集合。

3）SNMP 协议是应用层协议，定义了网络管理者如何对代理进程的 MIB 对象进行读写操作。

SNMP 中的 MIB 是一种树状数据库，MIB 管理的对象，就是树的端节点，每个节点都有唯一位置和唯一名字.IETF 规定管理信息库对象识别符（OID，Object Identifier）唯一指定，其命名规则就是父节点的名字作为子节点名字的前缀。

二、通讯内容及 **OID**

本协议支持访问的 OID 及相关通信信息参见 MIB 库(RFC1628_UPS_MIB.mib)，如下表

数据内容	节点(oid)	节点名称	单位	系数	备注
电池状态	1.3.6.1.2.1.33.1.2.1	upsBatteryStatus	/	1	1：电池未连接 2：电池正常 6：电池均充 7：电池浮充 8：电池放电 其他无效
电池剩余时间	1.3.6.1.2.1.33.1.2.3	upsEstimated MinutesRemaining	分	1	
电池剩余容量	1.3.6.1.2.1.33.1.2.4	upsEstimatedCharge Remaining	%	1	
电池电压	1.3.6.1.2.1.33.1.2.5	upsBatteryVoltage	V	0.1	
电池电流	1.3.6.1.2.1.33.1.2.6	upsBatteryCurrent	A	0.1	
输入相数	1.3.6.1.2.1.33.1.3.2	upsInputNumLines	/	1	
输入电压	1.3.6.1.2.1.33.1.3.3.1.3	upsInputVoltage	V	1	
输入电流	1.3.6.1.2.1.33.1.3.3.1.4	upsInputCurrent	A	0.1	
输入频率	1.3.6.1.2.1.33.1.3.3.1.2	upsInputFrequency	Hz	0.1	
输出供电方式	1.3.6.1.2.1.33.1.4.1	upsOutputSource	/	1	3：正常 4：旁路 5：电池 其他无
输出频率	1.3.6.1.2.1.33.1.4.2	upsOutput Frequency	Hz	0.1	
输出相数	1.3.6.1.2.1.33.1.4.3	upsOutputNumLines	/	1	
输出电压	1.3.6.1.2.1.33.1.4.4.1.2	upsOutputVoltage	V	1	
输出电流	1.3.6.1.2.1.33.1.4.4.1.3	upsOutputCurrent	A	0.1	
输出有功功率	1.3.6.1.2.1.33.1.4.4.1.4	upsOutputPower	W	1	
负载百分比	1.3.6.1.2.1.33.1.4.4.1.5	upsOutput PercentLoad	%	1	
旁路频率	1.3.6.1.2.1.33.1.5.1	upsBypass	Hz	0.1	

		Frequency			
旁路电压	1.3.6.1.2.1.33.1.5.3.1.2	upsBypassVoltage	V	1	
旁路电流	1.3.6.1.2.1.33.1.5.3.1.3	upsBypassCurrent	A	0.1	
额定输入电压	1.3.6.1.2.1.33.1.9.1	upsConfig InputVoltage	V	1	
额定输入频率	1.3.6.1.2.1.33.1.9.2	upsConfig InputFreq	Hz	0.1	
额定输出电压	1.3.6.1.2.1.33.1.9.3	upsConfig OutputVoltage	V	1	
额定输出频率	1.3.6.1.2.1.33.1.9.4	upsConfig OutputFreq	Hz	0.1	
额定容量	1.3.6.1.2.1.33.1.9.5	upsConfigOutputVA	W	1	
厂家名称	1.3.6.1.2.1.33.1.1.1	upsIdent Manufacturer	/	/	
UPS 型号	1.3.6.1.2.1.33.1.1.2	upsIdentModel	/	/	
电池供电	1.3.6.1.2.1.33.1.6.3.2	upsAlarmOnBattery			1：发生
电池电压低	1.3.6.1.2.1.33.1.6.3.3	upsAlarmLowBattery			1：发生
电池即将耗尽	1.3.6.1.2.1.33.1.6.3.4	upsAlarm DepletedBattery			1：发生
UPS 过温	1.3.6.1.2.1.33.1.6.3.5	upsAlarmTempBad			1：发生
输入(市电)故障	1.3.6.1.2.1.33.1.6.3.6	upsAlarmInputBad			1：发生
输出故障	1.3.6.1.2.1.33.1.6.3.7	upsAlarmOutputBad			1：发生
输出过载	1.3.6.1.2.1.33.1.6.3.8	upsAlarm OutputOverload			1：发生
旁路供电	1.3.6.1.2.1.33.1.6.3.9	upsAlarmOnBypass			1：发生
旁路故障	1.3.6.1.2.1.33.1.6.3.10	upsAlarmBypassBad			1：发生
充电模块故障	1.3.6.1.2.1.33.1.6.3.13	upsAlarm ChargerFailed			1：发生
风扇故障	1.3.6.1.2.1.33.1.6.3.16	upsAlarmFanFailure			1：发生
测试中	1.3.6.1.2.1.33.1.6.3.24	upsAlarm TestInProgress			1：发生