

基本性能参数

Ta = 25 °C

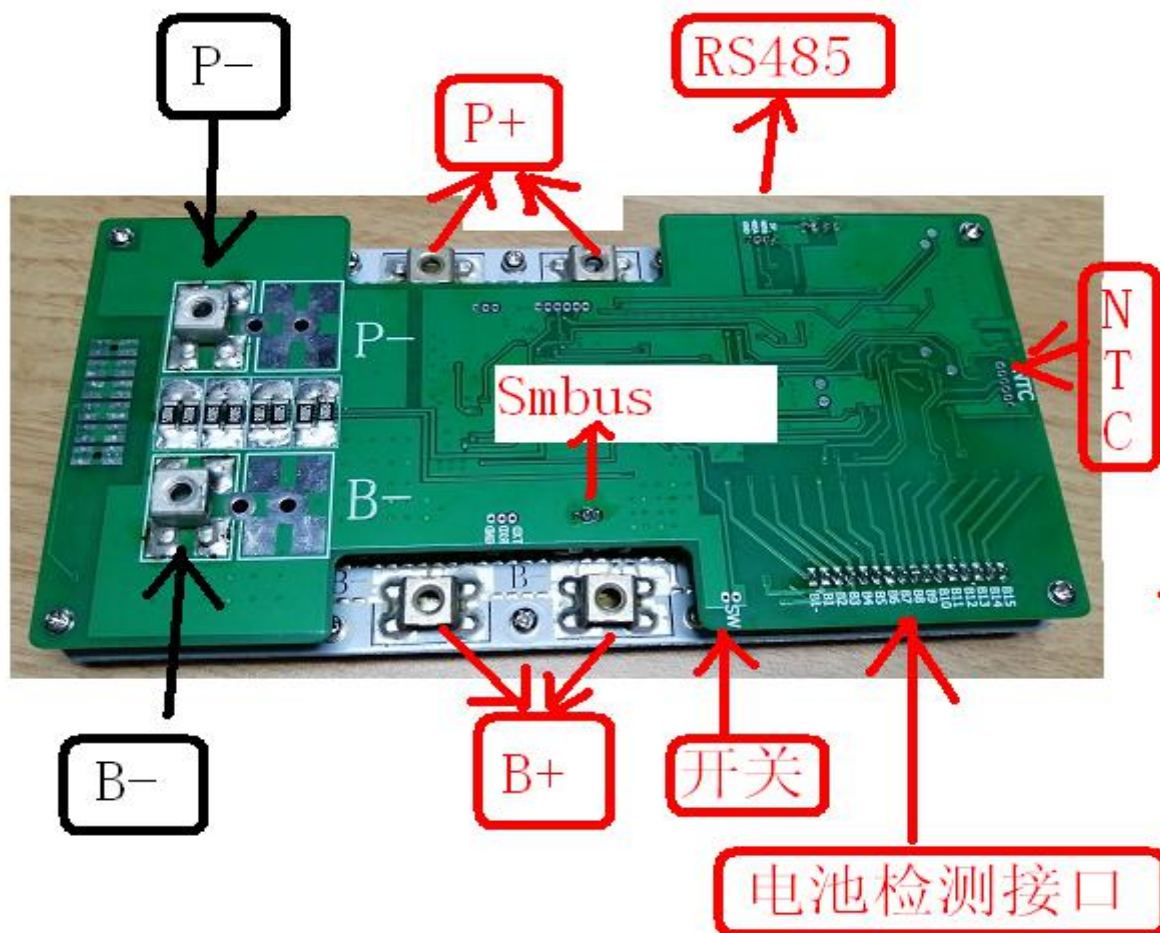
www.libatt.com

Details	Min.	Typ.	Max	Error	Unit
Battery					
Battery Gas （电池组类型）	Lion				
Battery Links	14S/1P				
Absolute Maximum Rating					
Input Charging Voltage		58.8		±1%	V
Input Charging Current		10			A
Output Discharging Voltage	39.2		58.8		V
Output Discharging Current		50	120		A
Continuous Output Discharging Current（可持续工作电流）	50				A
Ambient Condition					
Operating Temperature	-10		65		℃
Humidity (No Water-Drop)					RH
Storage					
Temperature	-40		75		℃
Humidity (No Water-Drop)					RH
Protection Parameters (for Individual Cell).					
Over-Charge Voltage Protection (OVP) 过充电压保护		4.25			V
Over-Charge Voltage Protection Release (OVPR)		4.10			V
Over-Discharge Voltage Protection (UVP) 过放电压保护		2.50			V
Over-Discharge Voltage Protection Release (UVPR)		3.00			V
Over-Current Discharge Protection (OCDP) 放电过流保护		150			A
Over-Current Protection Delay Time(OCPDT)过流保护延时		20			mS
Over-Current Discharge Protection Release	延时约 15S 自动恢复				
Short circuit current protection	有				A
Short circuit current protection delay time 短路保护时间	200	500	1000		uS
Short circuit protection Release	延时约 15S 自动恢复				
Charging and Discharging Temperature					℃
Charging and Discharging Temperature Protection Release					℃
Balance Temperature					℃
Balance Temperature Protection Release					℃
Cell balance					
BleedStartPoint					mV
BleedAccuracy					mV
BleedCurrent					mA
Balance Mode	Charging				
Current Consumption					
Normal Mode 正常模式			20000		uA
Idle mode 空载模式			20000		uA
Sleep mode 睡眠模式		500			uA

备注：以上参数为推荐值，用户可依据实际应用进行修改。

VP15RS485 V1.0

PCB 接线图



接线方法:

1. 首先使用黑色粗线连接电池组的“负极”到保护板的“B-”。
2. 接连接排线的第一根 (B-) 到第一节电芯的负极, 接连接排线的第二根 (B1) 到第一节电芯的正极, 第三根 (B2) 到第二节电芯的正极, 依次类推。。。Bn (最上边那根) 也刚好接到最高节电芯的正极, 也就是电池组的正极 (B+)。
3. 使用表检测排线端子的每相邻两个端子的电压是否一一对应电池的电压
4. 电池总负极 B-, 总正极 B+, 输出 P+/C+, 输出 P-/C- 分别接到保护板对应的焊盘点上。
5. 软件智能 BMS 需要充电激活。
6. 如上面步骤都正确后, 插入排线, 检测一下输出是否和电池组总电压相等。

输出端意思也就是:

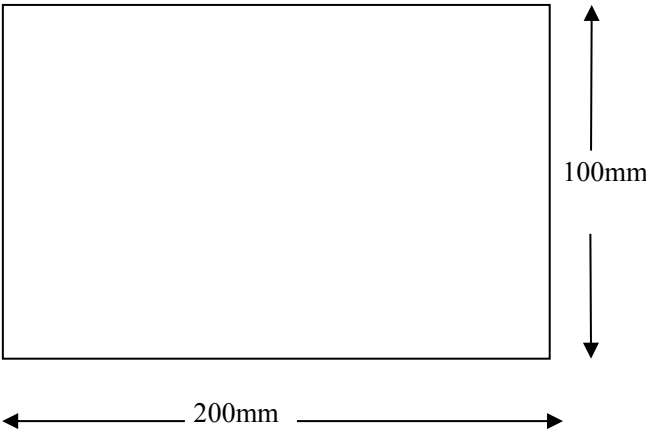
1.) 充电器负极接线端和 B+ 之间的电压;
2.) 负载负极接线端和 B+ 之间的电压;

注意: 1. 安装保护板时请自行加固定螺丝, 注意螺丝不要因为震动等原因造成松动脱落!

2. 先要焊接好每一根排线到相应的电芯极点上, 才能把排线端子插入保护板座子中, (请不要插入排线端子再焊接电芯, 防止焊接时出现错误损坏保护板!)

备注: 请严格按照上电顺序操作, 如因上电顺序操作造成的保护板损坏, 我司不承担任何责任。

PCB 结构图(厚度=<30mm)



包装、运输指引

1. 包装

将成品 PCBA 平整放入干净的气泡袋中，封好。

2. 运输

用快递或派专人送货。运输过程中，应注意防潮、防湿，避免挤压、碰撞等，以免保护板变形。

注：请置于阴凉干燥的环境中保存

备注：

样板来源				VP15RS485				 E&J TECHNOLOGY GROUP CO., LTD
样板确认记录		1		产品承认书				
		2						
		3						
制 表		日期		型 号	板材	板厚	工 艺	
设 计		日期		VP15RS485	FR-4	1.6	喷锡	
审 核		日期						

电池组保护板 MODBUS 规约

1.物理接口

RS485 波特率为 9600BPS，字符格式采用 8 位数据位，1 位停止位，无校验。

2.帧结构

采用 Modbus 规约

8Bit 地址	8Bit 功能码	nX8Bit 数据	16BitCRC 校验码
---------	----------	-----------	--------------

 的 RTU (Remote Terminal Unit) 方式。
功能码
支持功能码 03（读数据）和功能码 06（写数据）。

CRC 校验码

CRC (Cyclical Redundancy Check) 对地址，功能码和数据进行校验，由两字节组成，CRC 由传输设备生成，附加在数据帧中，如果由接收到数据计算出来的校验和与附加在数据后的校验和不一致，则有错误发生。关于 CRC 生成函数，请参阅附录内容。

查询命令帧格式

Field Name	Example (HEX)	Description
Slave Address	03	地址 0x03
Function	03	功能码 3
Starting Address Hi	00	起始寄存器高字节
Starting Address Lo	01	起始寄存器低字节，起始地址 1
No. of Points Hi	00	读取寄存器个数高字节
No. of Points Lo	02	读取寄存器个数低字节，读两个寄存器值
CRCLO		CRC 低字节
CRCHI		CRC 高字节

从机响应帧格式

Field Name	Example (HEX)	Description
Slave Address	0x03	地址 0x03
Function	03	功能码 3
Byte Count	04	返回字节数
Data Hi	01	第 1 个数据高位值
Data Lo	2B	第 1 个数据低位值
Data Hi	01	第 2 个数据高位值
Data Lo	11	第 2 个数据低位值
CRCLO		CRC 低字节
CRCHI		CRC 高字节

动力电池 RS232 (MODBUS RTU) 通讯格式

帧结构

实时数据查询帧：

地址码	功能码	寄存器起始地址	寄存器长度	校验码低位	校验码高位
0X01	0X03	0X00 0X00	0X00 0X34	0X44	0X1D

返回数据如下：序号	内容	单位	操作
1	地址	0x01	1 字节
2	命令	0x03	1 字节
3	数据字节数	0x68	1 字节
4	电池包总电压	单位是 10mV (注 1)	2 字节
5	电池电流	单位是 10mA (注 2)	2 字节
6	单体 1 电压	mV	2 字节
7	单体 2 电压	mV	2 字节
8	单体 3 电压	mV	2 字节
9	单体 4 电压	mV	2 字节
10	单体 5 电压	mV	2 字节
11	单体 6 电压	mV	2 字节
12	单体 7 电压	mV	2 字节
13	单体 8 电压	mV	2 字节
14	单体 9 电压	mV	2 字节
15	单体 10 电压	mV	2 字节
16	单体 11 电压	mV	2 字节
17	单体 12 电压	mV	2 字节
18	单体 13 电压	mV	2 字节
19	单体 14 电压	mV	2 字节
20	单体 15 电压	mV	2 字节
21	单体 16 电压	mV	2 字节
22	单体 17 电压	mV	2 字节
23	单体 18 电压	mV	2 字节
24	单体 19 电压	mV	2 字节
25	单体 20 电压	mV	2 字节
26	单体 21 电压	mV	2 字节
27	单体 22 电压	mV	2 字节
28	单体 23 电压	mV	2 字节
29	单体 24 电压	mV	2 字节
30	单体最高电压	mV	2 字节
31	单体最低电压	mV	2 字节
32	平均电压	mV	2 字节
33	电蕊压差	mV	2 字节
34	单体最高电压串数		2 字节
35	单体最低电压串数		2 字节

36	电池剩余容量	单位是 10mAH(注 3)	2 字节
37	电池设计容量	单位是 10mAH(注 3)	2 字节
38	电池电量百分比	0~100%	2 字节
39	电池 CYCLE 次数		2 字节
40	探头 1 温度	0.1C	2 字节
41	探头 2 温度	0.1C	2 字节
42	探头 3 温度	0.1C	2 字节
43	过压状态 1		2 字节
44	过压状态 1		2 字节
45	欠压状态 1		2 字节
46	欠压状态 1		2 字节
47	工作状态	(注 4)	2 字节
48	均衡状态 1		2 字节
49	均衡状态 1		2 字节
50	出厂日期	(注 5)	2 字节
51	电芯类型	高字节 0x00-磷酸铁锂电芯 ;0x01-三元电芯; 0x10-钛锂电芯	
	厂商编码	低字节 0x00~0xFF	
52	电池组编号		2 字节
53	硬件版本号	(高字节)	2 字节
	软件版本号	(低字节)	
54	Box 工作模式	2 字节	
		00:单机; 01:并联 10:并联准备	
55	BMS 地址		2 字节 (注 6)
56	CRC 校验		2 字节

注 1: 整组电压单位采用 10mV 原因: 因为 2 个字节最大为 65535, 如果单位为 mV, 当电池串数为 16 串时, 有可能超过 65535mV, 所以采用 10mV 为单位发送, 最大就会支持 655.35V 的整组电压

注 2: 电流单位采用 10mA, 带符号位, 充电为正, 放电为负; 也是因为会存在电流会存在超出 32767mA 的范围, 所以采用 10mA 的单位, 最大支持范围为 327.67A, 比如充电电流为 10A, 则发送值为 1000; 如放电电流为 10A, 实际发送值为 $65535 - 1000 = 64535$;

注 3: 剩余容量单位采用 10mAh 也是因为数据最大为 65535, 如果单位为 mAh, 最大只支持 65.535Ah, 会存在超出范围, 则所以单位都采用 10mAh, 这样最大会支持 655.35Ah。

过压状态 1: 两字节 16 位位号	内容
0	单体 1 过压 1==过压 0==正常
1	单体 2 过压 1==过压 0==正常
2	单体 3 过压 1==过压 0==正常
3	单体 4 过压 1==过压 0==正常
4	单体 5 过压 1==过压 0==正常
5	单体 6 过压 1==过压 0==正常
6	单体 7 过压 1==过压 0==正常
7	单体 8 过压 1==过压 0==正常
8	单体 9 过压 1==过压 0==正常
9	单体 10 过压 1==过压 0==正常
10	单体 11 过压 1==过压 0==正常
11	单体 12 过压 1==过压 0==正常

12	单体 13 过压 1==过压 0==正常
13	单体 14 过压 1==过压 0==正常
14	单体 15 过压 1==过压 0==正常
15	单体 16 过压 1==过压 0==正常
过压状态 2: 两字节 16 位位号 内容	
0	单体 17 过压 1==过压 0==正常
1	单体 18 过压 1==过压 0==正常
2	单体 19 过压 1==过压 0==正常
3	单体 20 过压 1==过压 0==正常
4	单体 21 过压 1==过压 0==正常
5	单体 22 过压 1==过压 0==正常
6	单体 23 过压 1==过压 0==正常
7	单体 24 过压 1==过压 0==正常
8	保留为 0
9	保留为 0
10	保留为 0
11	保留为 0
12	保留为 0
13	保留为 0
14	保留为 0
15	保留为 0

欠压状态 1: 两字节 16 位位号 内容	
0	单体 1 欠压 1==欠压 0==正常
1	单体 2 欠压 1==欠压 0==正常
2	单体 3 欠压 1==欠压 0==正常
3	单体 4 欠压 1==欠压 0==正常
4	单体 5 欠压 1==欠压 0==正常
5	单体 6 欠压 1==欠压 0==正常
6	单体 7 欠压 1==欠压 0==正常
7	单体 8 欠压 1==欠压 0==正常
8	单体 9 欠压 1==欠压 0==正常
9	单体 10 欠压 1==欠压 0==正常
10	单体 11 欠压 1==欠压 0==正常
11	单体 12 欠压 1==欠压 0==正常
12	单体 13 欠压 1==欠压 0==正常
13	单体 14 欠压 1==欠压 0==正常
14	单体 15 欠压 1==欠压 0==正常
15	单体 16 欠压 1==欠压 0==正常

欠压状态 2: 两字节 16 位位号 内容	
0	单体 17 欠压 1==欠压 0==正常
1	单体 18 欠压 1==欠压 0==正常
2	单体 19 欠压 1==欠压 0==正常
3	单体 20 欠压 1==欠压 0==正常
4	单体 21 欠压 1==欠压 0==正常
5	单体 22 欠压 1==欠压 0==正常
6	单体 23 欠压 1==欠压 0==正常
7	单体 24 欠压 1==欠压 0==正常
8	保留为 0

9	保留为 0
10	保留为 0
11	保留为 0
12	保留为 0
13	保留为 0
14	保留为 0
15	保留为 0

注 4: 工作状态

16: 保留为 0

15: 放电 MOS 管导通状态 (1 为正常打开, 0 为异常关闭)

14: 充电 MOS 管导通状态 (1 为正常打开, 0 为异常关闭)

13: 保护板锁定 (0 为正常, 1 为异常)

12: 检测芯片错误 (0 为正常, 1 为异常)

11: 短路保护 (0 为正常, 1 为异常)

10: 放电过流保护 (0 为正常, 1 为异常)

9: 充电过流保护 (0 为正常, 1 为异常)

8: 放电低温保护 (0 为正常, 1 为异常)

7: 放电高温保护 (0 为正常, 1 为异常)

6: 充电低温保护 (0 为正常, 1 为异常)

5: 充电高温保护 (0 为正常, 1 为异常)

4: PACK 欠压保护 (0 为正常, 1 为异常)

3: PACK 过压保护 (0 为正常, 1 为异常)

2: 单体欠压保护 (0 为正常, 1 为异常)

1: 单体过压保护 (0 为正常, 1 为异常)

均衡状态 1: 两字节 16 位位号 内容

0	单体 1 均衡 1==均衡 0==正常
1	单体 2 均衡 1==均衡 0==正常
2	单体 3 均衡 1==均衡 0==正常
3	单体 4 均衡 1==均衡 0==正常
4	单体 5 均衡 1==均衡 0==正常
5	单体 6 均衡 1==均衡 0==正常
6	单体 7 均衡 1==均衡 0==正常
7	单体 8 均衡 1==均衡 0==正常
8	单体 9 均衡 1==均衡 0==正常
9	单体 10 均衡 1==均衡 0==正常
10	单体 11 均衡 1==均衡 0==正常
11	单体 12 均衡 1==均衡 0==正常
12	单体 13 均衡 1==均衡 0==正常
13	单体 14 均衡 1==均衡 0==正常
14	单体 15 均衡 1==均衡 0==正常
15	单体 16 均衡 1==均衡 0==正常

过压状态 2: 两字节 16 位位号 内容

0	单体 17 均衡 1==均衡 0==正常
1	单体 18 均衡 1==均衡 0==正常
2	单体 19 均衡 1==均衡 0==正常
3	单体 20 均衡 1==均衡 0==正常
4	单体 21 均衡 1==均衡 0==正常

5	单体 22 均衡 1==均衡 0==正常
6	单体 23 均衡 1==均衡 0==正常
7	单体 24 均衡 1==均衡 0==正常
8	保留为 0
9	保留为 0
10	保留为 0
11	保留为 0
12	保留为 0
13	保留为 0
14	保留为 0
15	保留为 0

注 5:

出厂日期:

Day 0...4 5 bit binary value 1 - 31 (corresponds to date)

Month 5...8 4 bit binary value 1 - 12 (corresponds to month number)

Year 9...15 7 bit binary value 0 - 127 (corresponds to year biased by 1980)

附录 CRC16 校验的计算方法

计算方法 1:

```

#define          CRC_16_POLYNOMIALS  0xa001
// X^16 +X^15 +X^2+1
uint16_t Check_CRC16(uint8_t* pchMsg,uint8_t wDataLen)
{
    uint8_t i, chChar;
    uint16_t wCRC = 0xFFFF;
    while (wDataLen--)
    {
        chChar = *pchMsg++;
        wCRC ^= (uint16_t) chChar;
        for (i = 0; i < 8; i++)
        {
            if (wCRC & 0x0001)
                wCRC = (wCRC >> 1) ^ CRC_16_POLYNOMIALS;
            else
                wCRC >>= 1;
        }
    }
    return wCRC;
}

```

计算方法 2:

/* CRC 高位字节值表 */

```

byte[] auchCRCHi = newbyte[] {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*/

```
byte[] auchCRCLo = newbyte[] {
    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
```

```
};
```

```
//校验函数
```

```
UInt16 crc16(byte[] puchMsg, UInt16 usDataLen)
```

```
{
```

```
    UInt16 clc;
```

```
    UInt16 i = 0;
```

```
    byte uchCRCHi = 0xFF; /* 高CRC字节初始化 */
```

```

byte uchCRCLo = 0xFF; /* 低CRC 字节初始化 */

byte uIndex; /* CRC循环中的索引 */

while (usDataLen-- > 0) /* 传输消息缓冲区 */
{
    uIndex = (byte) (uchCRCHi ^ puchMsg[i++]); /* 计算CRC */
    uchCRCHi = (byte) (uchCRCLo ^ auchCRCHi[uIndex]);
    uchCRCLo = auchCRCLo[uIndex];
}

clc = (UInt16) (uchCRCHi << 8 | uchCRCLo);

return (clc);

}

```

增加二个开或关充电放电 MOS 功能

3.2 写入数据，功能码 06，格式与查询格式一致。，增加 MOS 开关控制的指令，来控制充/放电 MOS，
 比如主机端发送：0x01 0x06 0x00 0x9d 0xAA 0xBB 0x26 0xF7,则会打开 MOS
 比如主机端发送：0x01 0x06 0x00 0x9c 0xAA 0xBB 0x77 0x33 则会关闭 MOS
 如果写入命令成功， 则从机 BMS 也会返回相同的数据给主机

增加修改 BMS 地址的功能（（注 6））

写入 BMS 地址数据，功能码 06，格式与查询格式一致。

比如主机端发送：0xF7 0x06 0x55 0x02 0xDC 0xBA 0xF4 0x23,

解释：

第一个字节：0xF7 固定地址码

第二个字节：0x06 功能码 06

第三个字节：0x55 命令识别码高字节数据 1

第四个字节：0x02 要写入的 BMS 的低字节地址数据

第 5 个字节：0xDC 写命令识别码高字节数据 2 第 6 个字节：0xBA 写命令识别码低字节数据

第 7 个字节：0xF4 CRC 高字节数据

第 8 个字节：0x23 CRC 低字节数据

如果写入地址成功， 则从机 BMS 也会返回相同的数据给主机

注 8：

读出 BMS 地址数据，功能码 06，格式与查询格式一致。（这功能是确认写入 BMS 的地址数据是否生效）

比如主机端发送：0xF7 0x06 0x55 0x00 0xAB 0xCD 0xF4 0x23,

解释：

第一个字节：0xF7 固定地址码

第二个字节：0x06 功能码 06

第三个字节：0x55 命令识别码高字节数据 1

第四个字节：0x00 保留

第 5 个字节：0xAB 写命令识别码高字节数据 2

第 6 个字节：0xCD 写命令识别码低字节数据

第 7 个字节：0x32 CRC 高字节数据

第 8 个字节：0x35 CRC 低字节数据

如果读取地址成功， 则从机 BMS 也会返回相应的地址数据给主机如下 0x02(BMS 地址)的数据：

BMS 端发送回复：0xF7 0x06 0x55 0x02 0xAB 0xCD 0x93 0xF5, (解释如下)

第一个字节：0xF7 固定地址码

第二个字节：0x06 功能码 06

第三个字节：0x55 命令识别码高字节数据 1

第四个字节：0x02 读出来的 BMS 地址码

第 5 个字节：0xAB 写命令识别码高字节数据 2

第 6 个字节：0xCD 写命令识别码低字节数据

第 7 个字节：0x93 CRC 高字节数据

第 8 个字节：0xF5 CRC 低字节数