

# JMK Bootloader 通信协议

苏州捷敏科电子科技有限公司

## 修订历史

版本	日期	说明
Ver1.00	2019/03/27	创建文档。
Ver1.01	2019/04/04	增加 CRC32 的详细描述。
Ver1.02	2019/04/19	Bootloader 使用示例、CRC32 参考代码。
Ver1.03	2019/08/31	获取硬件信息命令，增加电池信息说明。

## 1. 概述

在哈喽换电项目中，BMS Bootloader 程序采用 RS485 串行接口通信，主要用于固件在线升级，通信参数：9600bps,8,N,1。

该协议支持以下 BMS： JMK05-BMS20 系列，JMK01-BMS15 系列

## 2. 通信格式

文档约定：

- 多字节传输过程中，低字节在前，高字节在后。
- 所有序号、ID，编号从 0 开始。

### 2.1 RS485 数据帧格式（兼容 BMS 通信协议）

帧头	地址	通信命令	数据长度	数据包内容	校验资料	帧尾
单字节	单字节	单字节	单字节	多字节	2 字节	2 字节
0x3A	0x16	0xF0	-	-	-	0x0D,0x0A

说明：

- 帧头： 单字节， 内容为 0x3A， 为固定值。
- 地址： 单字节， 内容为 0x16， 为固定值。
- 通信命令： 单字节， Boot Loader 命令为 0xF0， 更多信息参考 BMS 通信协议。
- 数据长度： 单字节， 内容为该通讯数据帧内数据缓冲区内的数据长度。 主机发送数据时， 如果不包含其他数据则统一设置为 1。
- 数据内容： 多字节， 内容为具体各个命令对应的数据字节， 字节数是不固定的， 数量由数据长度部分的数值确定。 主机发送数据时， 如无特别命令要求， 建议设置为 0。
- 校验资料： 两字节， 内容为通讯数据的累加校验和数据， 包括： 地址、通信命令、数据长度、数据内容的累加和， 低字节在前， 高字节在后。
- 帧尾： 两字节， 内容为结束标识 1（为固定值 0x0D） 和结束标识 2（为固定值 0x0A）。

### 2.2 Boot Loader 数据包格式

命令	参数
单字节	多字节

说明：

- 命令： 单字节， 见 Boot loader 命令描述。
- 参数： 多字节， 见 Boot loader 命令描述。

### 3. Boot Loader 命令列表

命令	命令代码	命令说明
进入 Boot Loader 模式	0xF1	发送特殊字节序列，进入 BootLoader 模式。
获取硬件信息	0xF2	包括：硬件版本、信息、编号。
MCU 复位	0xF3	MCU 软件复位。
程序跳转	0xF4	跳转到内部 Flash 的应用程序。
读取当前固件信息	0xF5	包括：固件版本、CRC32 校验、固件大小、总帧数。
发送新固件信息	0xF6	包括：CRC32 校验、固件大小、总帧数。
发送新固件数据	0xF7	包括：当前帧号、帧数据。

### 4. Boot Loader 协议说明

#### 4.1. 进入 Boot Loader 模式（0xF1）

发送特殊字节序列，进入 Boot Loader 模式。

主机	数据包大小：14	命令	参数
		0xF1	12 字节
BMS	数据包大小：2	命令	参数
		0xF1	1 字节
说明	参数说明：		
	主机发送	固定字符串“JMK-BMS-BL00”。	
	BMS 返回	应答：0x00(OK)，其它(Error)。	

#### 4.2. 获取硬件信息（0xF2）

包括：硬件版本，MCU Flash 大小、唯一 ID。

主机	数据包大小：2	命令	参数
		0xF2	1 字节
BMS	数据包大小：29	命令	参数
		0xF2	28 字节

说明	参数说明:	
	主机发送	固定为 0x00。
	BMS 返回	产品硬件信息，见下面表格。

产品硬件信息:

字节 ID	名称	备注
1~2	硬件版本	1:VER_L, 2:VER_H.
3~4	软件版本 (Boot Loader)	3:VER_L, 4:VER_H.
5~10	硬件信息 (6 字节)	
11~22	硬件编号 (12 字节)	
23~26	电池信息 (4 字节)	见下表说明。
27~28	保留 (2 字节)	未使用。

电池信息:

字节 ID	名称	备注
23	BMS 厂商	0x00:捷敏科, 其它: 保留。
24	电池厂商	0x00:洁能劲, 其它: 保留。
25	电芯类型	0x00:磷酸铁锂, 0x01:三元锂, 其它: 保留。
26	电池型号	0x00:60V16AH, 其它: 保留。

#### 4.3. MCU 复位 (0xF3)

软件复位。

主机	数据包大小: 2	命令	参数
		0xF3	1 字节
BMS	数据包大小: 2	命令	参数
		0xF3	1 字节
说明	参数说明:		
	主机发送	固定为 0x00。	
	BMS 返回	应答: 0x00-OK, 其它-Error。	

#### 4.4. 程序跳转 (0xF4)

跳转到内部 Flash 的应用程序。

主机	数据包大小: 2	命令	参数
----	----------	----	----

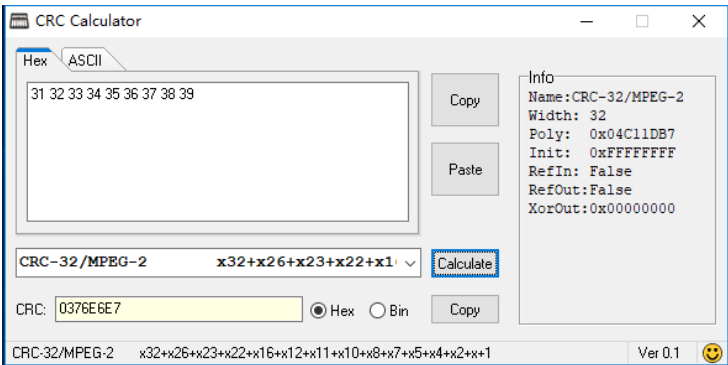
		0xF4	1 字节
BMS	数据包大小: 2	命令	参数
		0xF4	1 字节
说明	参数说明:		
	主机发送	固定为 0x00。	
	BMS 返回	应答: 0x00(OK), 其它(Error)。	

#### 4.5. 读取当前固件信息 (0xF5)

包括: 固件版本、CRC32 校验、固件大小、总帧数。

主机	数据包大小: 2	命令	参数
		0xF5	1 字节
BMS	数据包大小: 17	命令	参数
		0xF5	16 字节
说明	参数说明:		
	主机发送	固定为 0x00。	
	BMS 返回	当前 Flash 固件信息, 见下表说明。	

固件信息:

字节 ID	名称	备注
1~2	软件版本 (固件)	1:VER_L, 2:VER_H.
3~5	总字节数	低字节在前, 高字节在后。
6~7	总帧数	低字节在前, 高字节在后。
8~11	CRC32 校验码	<p>低字节在前, 高字节在后。 采用 CRC32-MPEG2 标准, 详细信息如下:</p> 
12~16	保留 (5 字节)	

#### 4.6. 发送新固件信息（0xF6）

此指令，上位机需获取本地固件的信息，发送给 BMS，BMS 依据此信息接收数据并判断数据是否准确和完整。

包括：固件版本、CRC32 校验、固件大小、总帧数。

主机	数据包大小：17	命令	参数
		0xF6	16 字节
BMS	数据包大小：2	命令	参数
		0xF6	1 字节
说明	参数说明：		
	主机发送	新固件信息	
	BMS 返回	应答：0x00(OK)，其它(Error)。	

固件信息：

字节 ID	名称	备注
1~2	保留	
3~5	总字节数	低字节在前，高字节在后。
6~7	总帧数	低字节在前，高字节在后。

8~11	CRC32 校验码	<p>低字节在前，高字节在后。 采用 CRC32-MPEG2 标准，详细信息如下：</p> 
12~16	保留（5 字节）	

#### 4.7. 发送新固件数据（0xF7）

包括：当前帧号、帧数据。

主机	数据包大小：N+1	命令	参数
		0xF7	多字节（N）
BMS	数据包大小：2	命令	参数
		0xF7	1 字节
说明	参数说明：最后一帧数据的接收完成后，自动完成新固件的升级。		
	主机发送	新固件数据，见下表说明。	
	BMS 返回	应答：0x00(OK)，其它(Error)。	

固件信息：

字节 ID	名称	备注
1~2	当前固件数据帧号。	范围：0~（总帧数-1）。
3~N	固件数据	每帧最长数据为 128 字节。

## 5. Boot Loader 使用实例

步骤 1: 主机发送“进入 Boot Loader 模式 (0xF1)”命令, 进入 BootLoader 模式。

步骤 2: 主机发送“发送新固件信息 (0xF6)”命令, 准备下载新固件。

步骤 3: 主机发送“发送新固件数据 (0xF7)”命令, 直到所有固件发送完成。

(BMS 在收到最后一帧数据后, 会做 CRC 校验, 回复时间在 2-3 秒)

步骤 4: 主机发送“发程序跳转 (0xF4)”命令, BMS 运行新的程序。

说明:

- 进入 BootLoader 模式后, 如果 10 秒内未收到有效 Bootloader 命令, 自动返回 BMS 程序。
- 下载固件数据失败后, 需要重新执行步骤 2~5。
- 建议上位机软件的指令超时时间为 5 秒。

## 6. 附件

### 6.1. CRC32-MPEG2 参考程序

C 实现:

```

/// <summary>
/// 获取文件的 CRC32 标识
/// </summary>
private static UInt32 CRC32_MPEG_2(byte[] data, int length)
{
    uint i;
    UInt32 crc = 0xffffffff, j = 0;

    while ((length-- != 0))
    {
        crc ^= (UInt32)data[j] << 24;
        j++;
        for (i = 0; i < 8; ++i)
        {
            if ((crc & 0x80000000) != 0)
                crc = (crc << 1) ^ 0x04C11DB7;
            else
                crc <<= 1;
        }
    }
    return crc;
}

```



Java 实现:

```
public int crc32_mpeg_2(byte[] data)
{
    int crcVal = 0xFFFFFFFF;
    for (byte item : data)
    {
        crcVal ^= item << 24;
        for (int k = 0; k < 8; k++) {
            if ((crcVal & 0x80000000) != 0)
                crcVal = (crcVal << 1) ^ 0x04C11DB7;
            else
                crcVal <<= 1;
        }
    }
    return crcVal;
}
```