E 换电 V2.0 CCU 设计文档

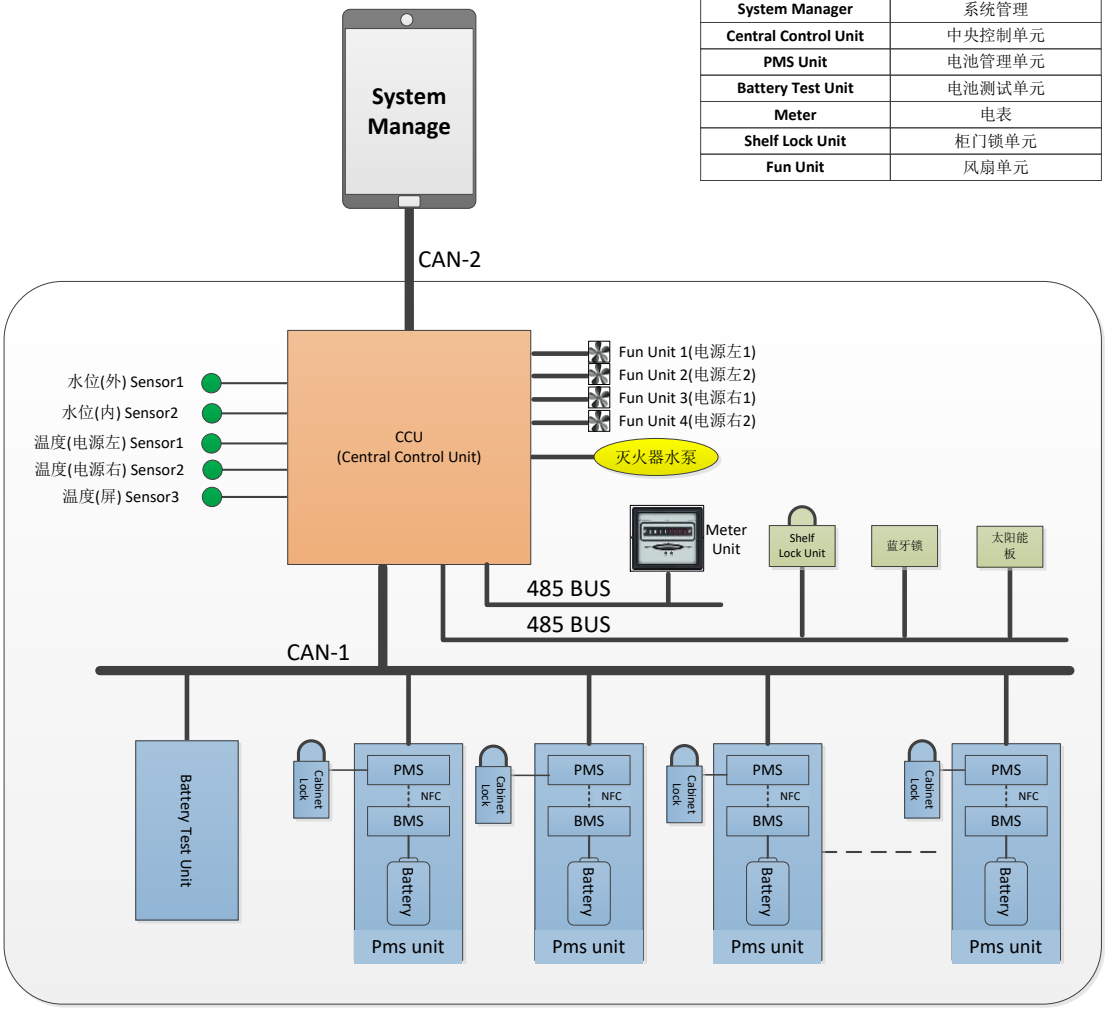
版本号	日期	修改记录	作者
0.1	2019-02-05	创建	Allen

1. 概述

换电柜系统支持单柜 CAN 组网，也支持多柜 CAN 单元一起组网，多柜组网示意图如下：组网单元最多 3 个。

1.1. 硬件模块架构图

名称	说明
System Manager	系统管理
Central Control Unit	中央控制单元
PMS Unit	电池管理单元
Battery Test Unit	电池测试单元
Meter	电表
Shelf Lock Unit	柜门锁单元
Fun Unit	风扇单元



本系统的设备 CCU，PMS，Battery Test Unit 使用 CAN-1 总线互联。
每个模块表示一个 CAN 节点设备。

1.2. 功能描述

E 换电系统实现以下功能
1） 充电功能。
2） 换电功能。
3） 系统管理功能，手机所有的传感器状态，设备运行状态和执行系统管理命令。
4） 灭火功能，一旦发现火警，启动灭火功能。
5） 照明功能。
6） 散热/制冷功能，包括启动风扇，空调等。

1.3. 设备地址分配

本系统所有的 CAN 设备节点地址分配采用预置方案，并且在运行过程中不可以修改。具体的设备地址分配如下表所示。

地址(Hex)	设备	备注
0x01	CCU	中央控制单元
0x02	Cool Unit	制冷单元
0x03	Shelf Lock Unit	柜门锁单元
0x04~0x5	保留	
0x06~0x15	PMS Unit	PMS 单元，每个柜子最多 16 个单元。
0x16	Battery Test Unit	电池测试单元
0x17	保留	保留
0x18	Meter Unit	电表单元（暂时不支持，待扩展）
0x19~0x1F	保留	
0xFF	所有设备	全局地址，仅用于目标地址（DA）

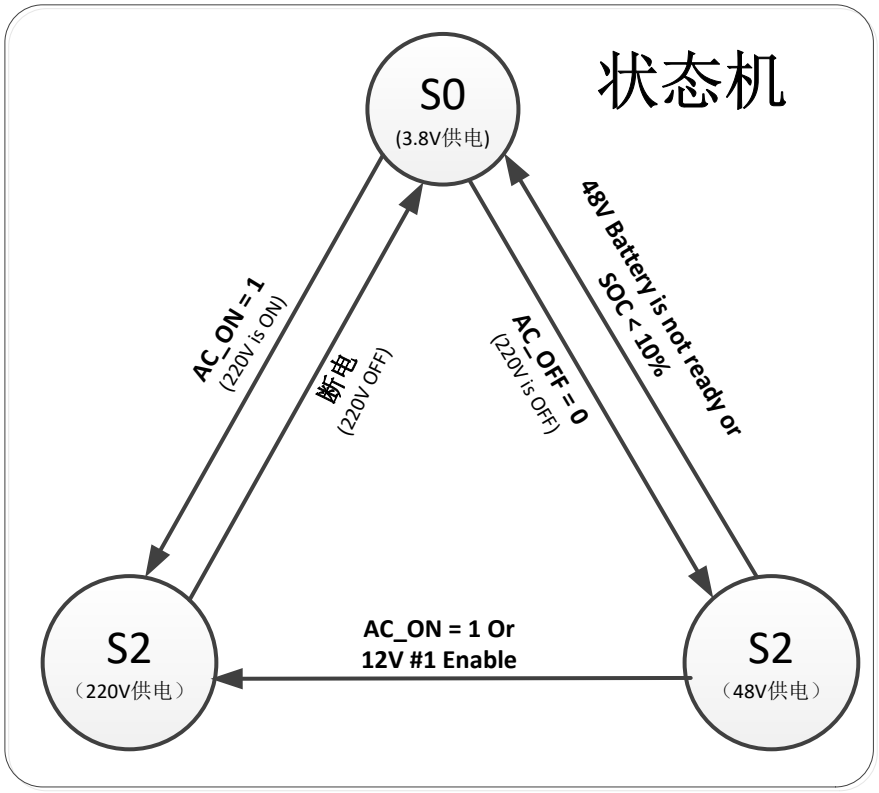
表 1.1-1 设备地址分配

本文档定义 CCU 和 PMS 之间的通信协议：

2. 状态机

根据供电电源不同，定义 CCU 共 3 种工作状态；
1） SO：18650 电池供电，供电电压 3.8V。
2） S1：备用电池供电，供电电压 48V。
3） S2,：市电供电，供电电压 22V。

各个状态机状态切换条件定义如下图所示：

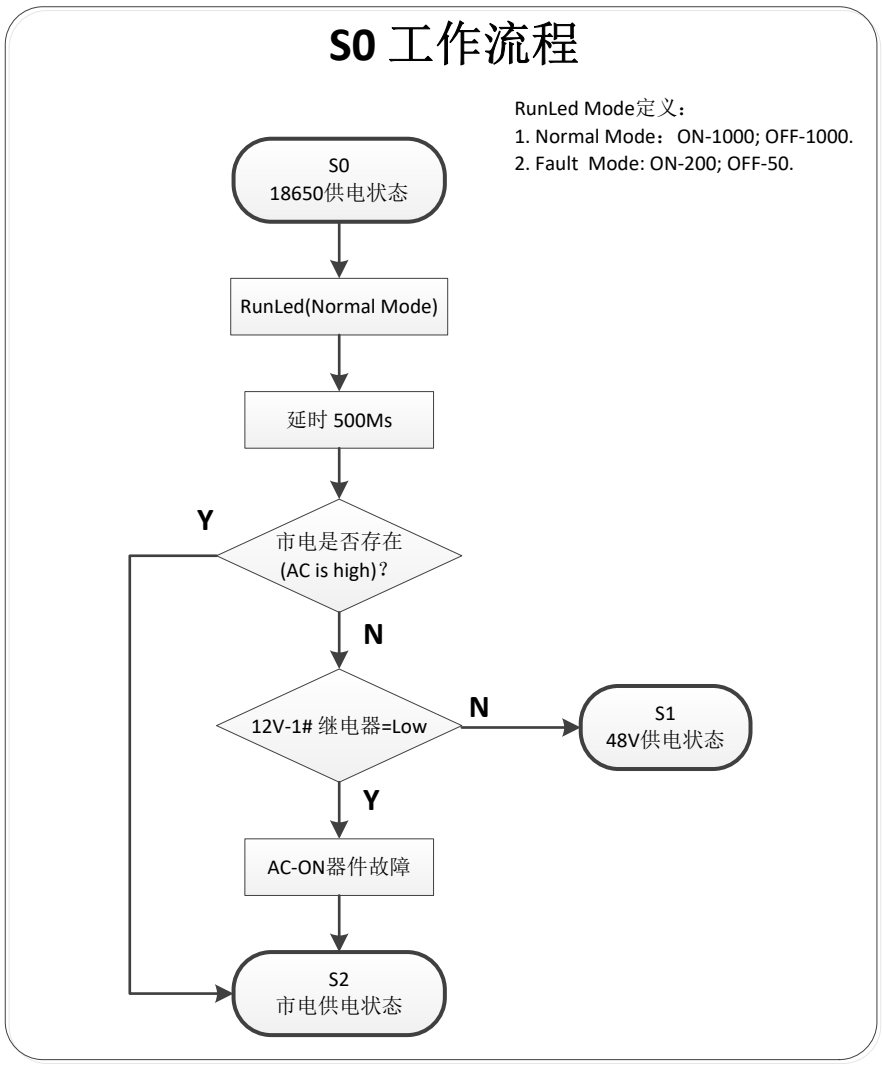


2.1. S0（18650 供电状态）

当系统刚启动时，需要先由 18650 电池供电启动 CCU，进入 S0 状态。

S0 状态主要功能：检测市电是否存在，如果存在则切换到 22V 供电，否则启动备用电池供电。

具体的工作流程如下图所示：



2.2. S1（48V 备电供电状态）

当 E 换电系统没有 220V 市电时，会尝试使用备用电池 48V 供电。
备用电池仓可根据配置：位于 1 号仓或者 2 号仓，或者 1 号仓和 2 号仓。

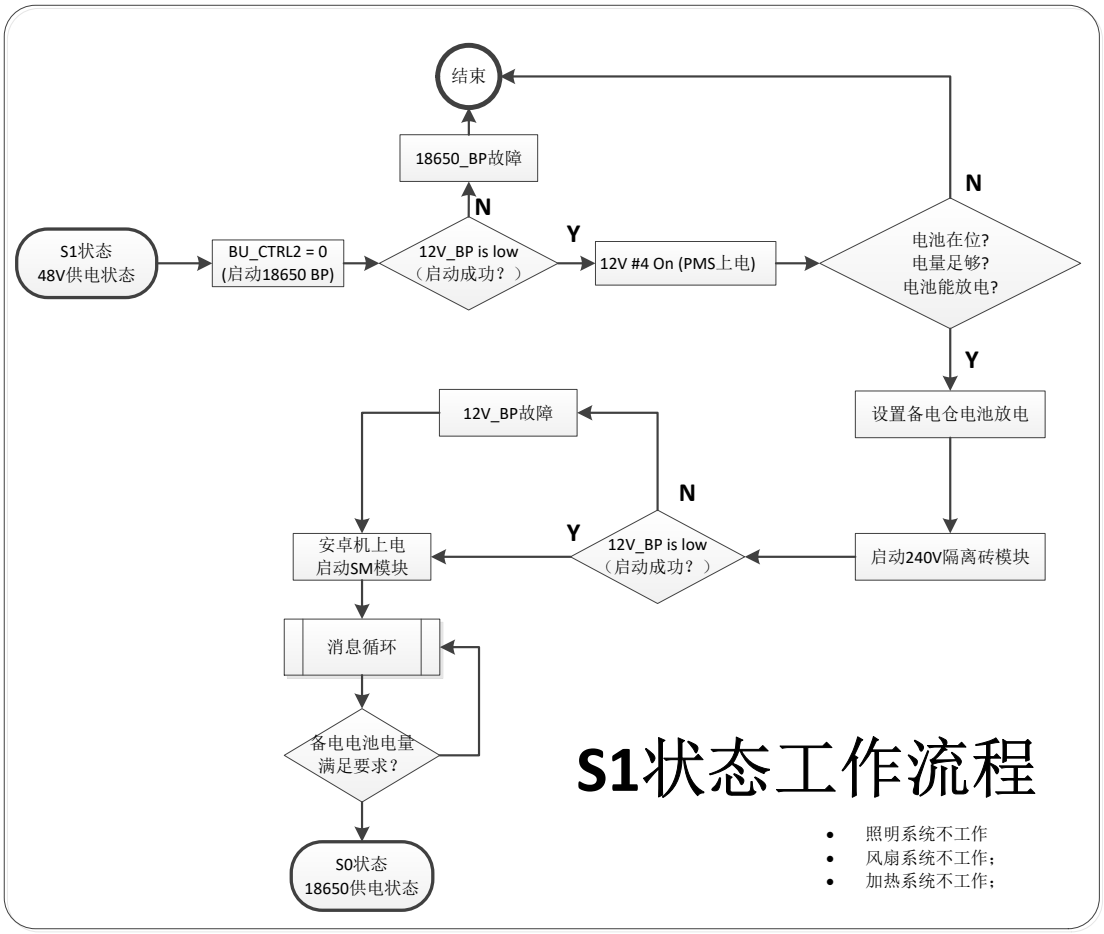
在 S1 状态下支持的功能如下：

1） 系统管理功能，包括基本的设备状态上报和执行系统管理命令。
2） 换电功能。
3） 灭火功能。

在 S1 状态下不支持的功能如下：

1） 照明功能。
2） 散热/制冷功能，包括启动风扇，空调等。
3） 充电功能。

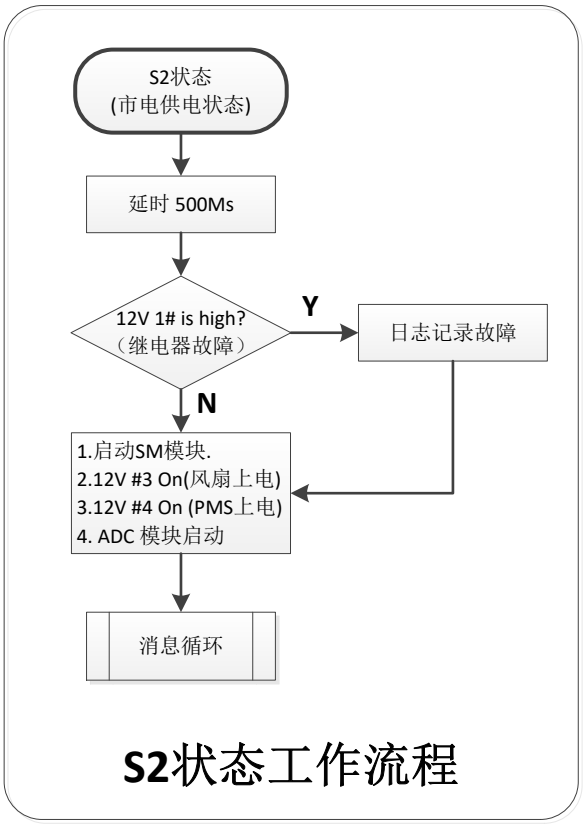
S1（48V 备电供电状态）的工作流程如下图所示。



2.3. S2（220V 供电状态）

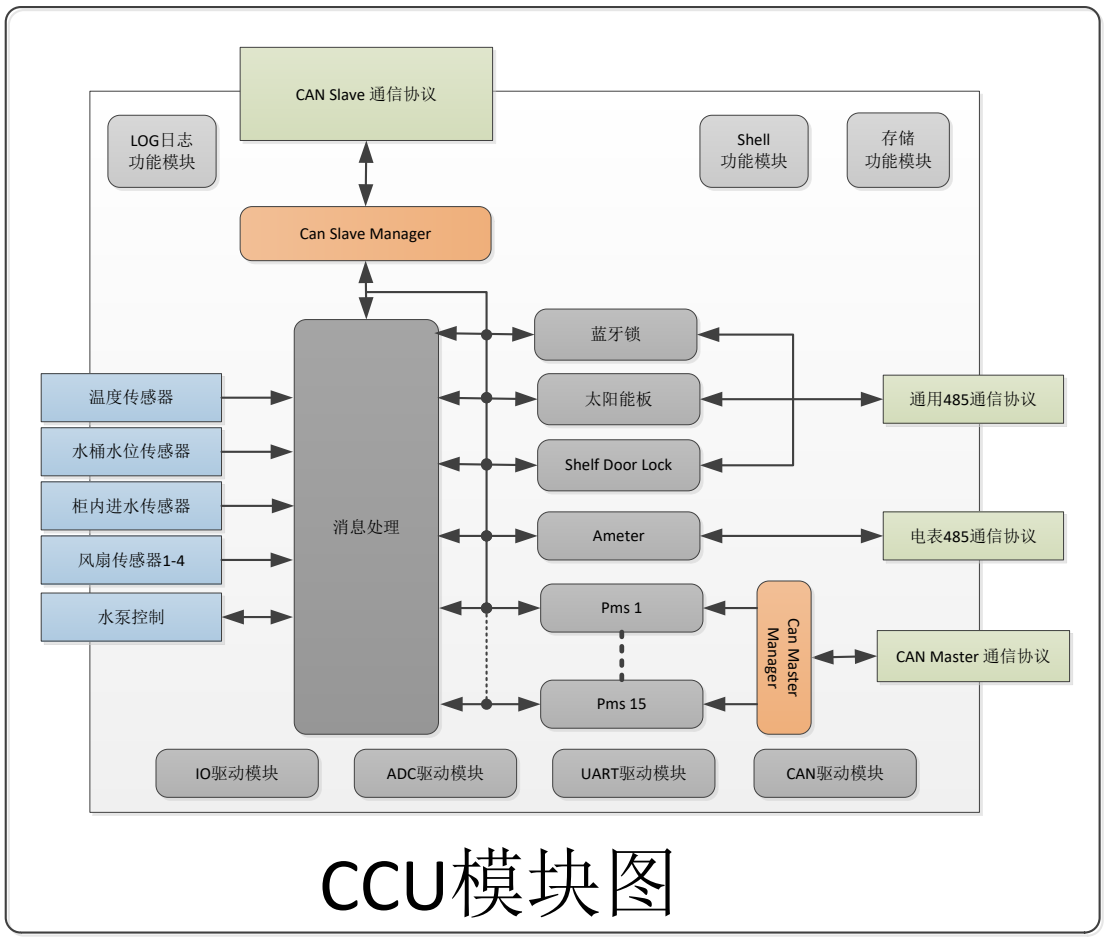
在 S2 状态下，所有的模块都要启动工作，包含以下功能

1） 系统管理功能，包括基本的设备状态上报和执行系统管理命令。
2） 换电功能。
3） 灭火功能。
4） 照明功能。
5） 散热/制冷功能，包括启动风扇，空调等。
6） 充电功能。



3. 功能模块设计

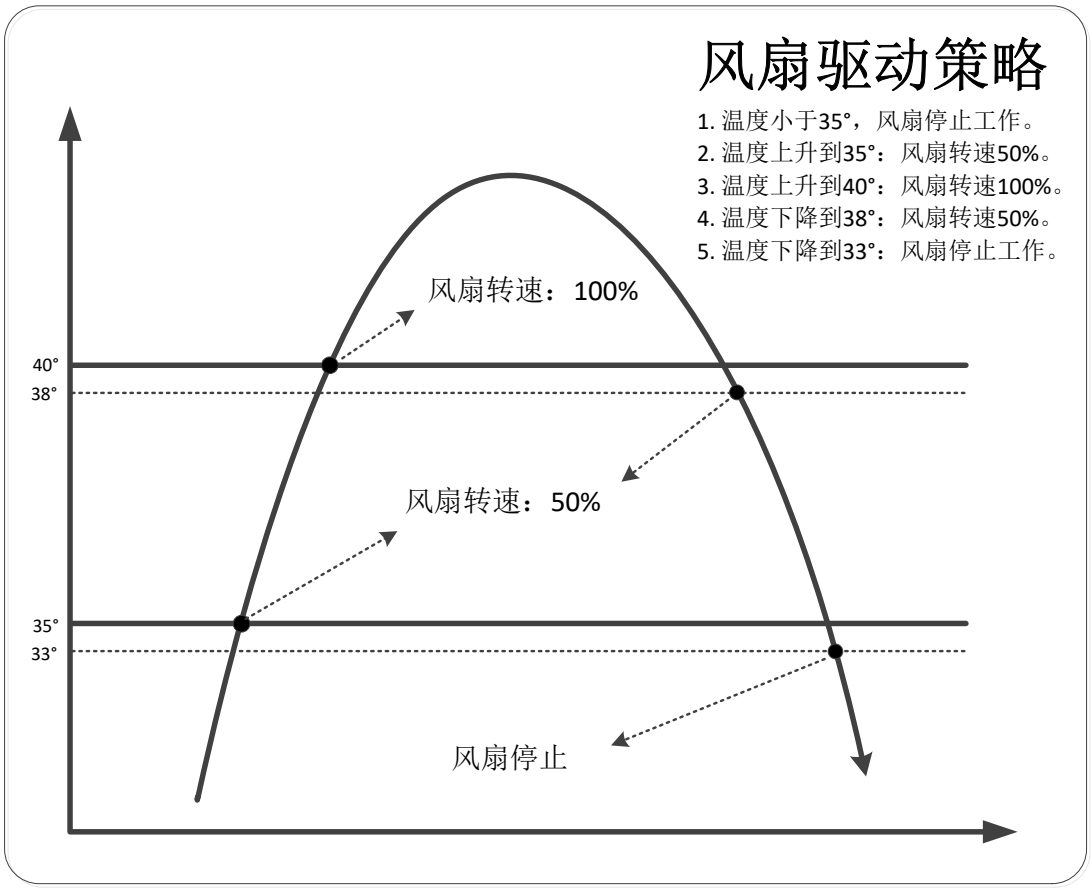
E 换电系统的内部模块架构如下图所示


CAN Slave 通信协议
LOG日志
功能模块
Shell
功能模块
存储
功能模块
Can Slave Manager
蓝牙锁
温度传感器
太阳能板
通用485通信协议
水桶水位传感器
Shelf Door Lock
柜内进水传感器
消息处理
Ameter
电表485通信协议
风扇传感器1-4
Pms 1
Can Master Manager
水泵控制
CAN Master 通信协议
Pms 15
IO驱动模块
ADC驱动模块
UART驱动模块
CAN驱动模块

CCU模块图

3.1. 降温功能

在本系统中，有 2-4 个温度传感器，通过采集温度传感器的值来驱动风扇降温。
风扇驱动策略如下：



3.2. 灭火功能

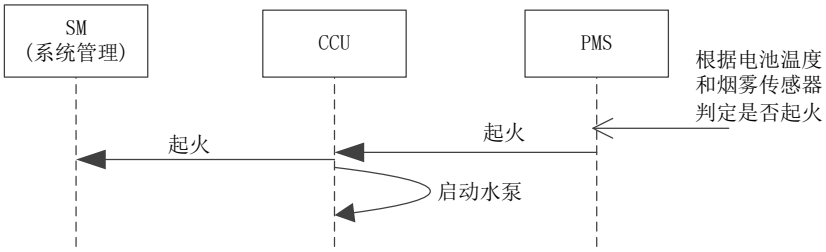
灭火功能定义如下：PMS 通过采集烟雾传感器和每个仓内电池温度传感器的值，PMS 判定是否起火，如果达到起火条件，则报告 CCU 有火警发生，CCU 启动水泵进行灭火。

起火条件由 PMS 判定，再通知 CCU，CCU 启动水泵灭火。
在灭火过程中如果关闭市电 220V，则启动备用电源系统继续灭火过程。

起火条件由 PMS 判定如下：

烟雾传感器 1 或者传感器 2 变化值	>=1V（待定）
电池内部温度或者链接器温度	>=60°（待定）

灭火模块协作图如下：



3.3. SM（系统管理）功能模块

SM，系统管理模块，实现的功能如下：
1） 负责和安卓机建立 CAN 通信
2） 上报整个系统所有硬件设备状态。
3） 接收安卓机的命令。
4） 执行命令。

具体的功能实现方法和细节参考协议文档“E 换电通信协议-SM”。

3.4. PMS 功能模块

PMS 功能模块主要实现以下功能：
1) 获取 PMS 设备及工作状态，包括烟雾传感器，温度传感器等状态。
2) 获取 BMS 设备及工作状态，包括电池电量，电池故障，工作电流等状态。
3) 开仓门功能。
4) 充电功能。

具体的功能实现方法和细节参考协议文档“E 换电通信协议-PMS”。

3.5. AmMemter 模块

AmMemter 模块实现获取电表电量功能，包含如下参数：
1） 即时电表读数。
2） 最大电流

具体的功能实现方法和细节参考协议文档“E 换电通信协议-Memter”。

3.6. Shelf Lock 模块

Shelf Lock 模块主要实现柜门锁的控制和状态管理。
具体的功能实现方法和细节参考协议文档“E 换电通信协议-ShelfLock”。

3.7. 太阳能模块

太阳能模块主要实现太阳能的充电控制和状态管理，具体的功能实现方法和细节参考协议文档“E 换电通信协议-太阳能”。

3.8. 日志模块

3.8.1. 日志格式定义

日志存储空间：为 1M。
日志存储方式：循环写入，当所有扇区全部写满内容，则删除最早的记录内容，写入新的记录内容。

日志由多条记录组成，每条记录的长度固定，为 12 个字节，因此，日志空间最多可存储 1MK/12= 87381 条记录。

记录格式如下：

Index	Name	Type	Value	Descriptor
0-3	dateTime	UINT32		日期时间，unix 时间戳格式，从 1970 年 1 月 1 日（UTC/GMT 的午夜）开始所经过的秒数，不考虑闰秒。
4	Type			日志类型
5	EventID	UINT8		时间 ID，参考下表。
6	ObjID	UINT8		对象 ID，参考下表。
7-10	Param	UINT32		参数 1
11	Tail	UINT8		结束标志

3.8.2. 日志 EventID 定义

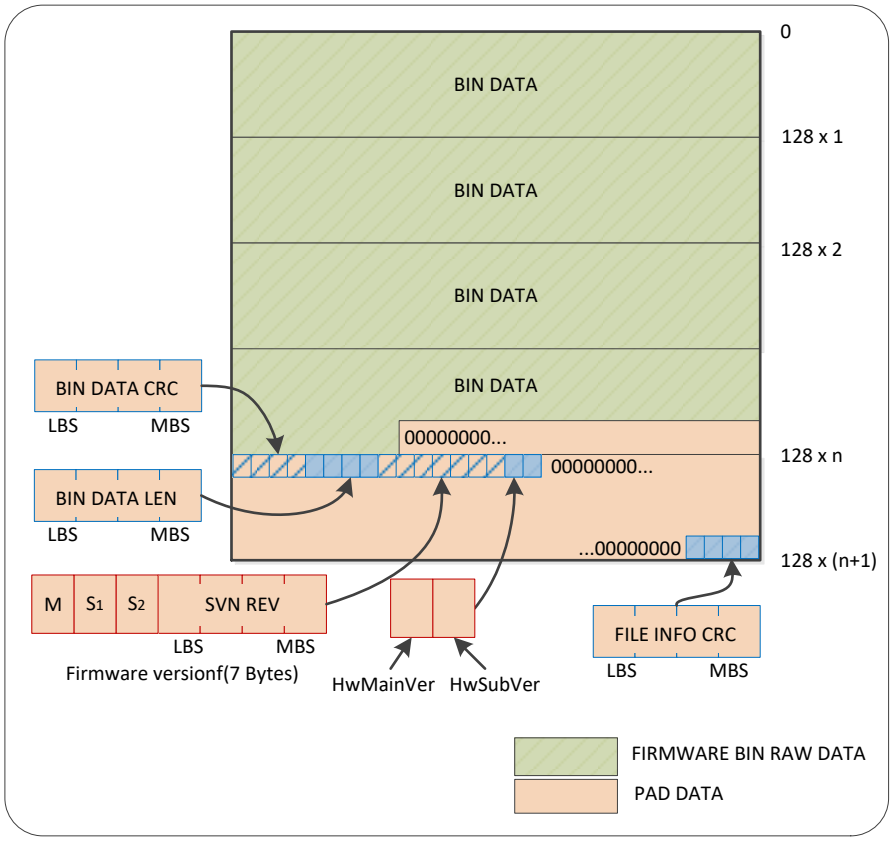
日志 Event ID	值	参数 1		说明
ET_SYS_RESET = 1 (硬复位源，软复位原因)	1	00000001	上电复位	系统复位
		00000010	复位脚复位	
		00000100	看门狗复位	
		00010000	欠压复位	

		00100000	MO 复位信号	
		10000000	CPU 复位	
ET_STATE_SWITCH	2	旧设备状态值	新状态	状态切换

4. 固件升级设计

4.1. APPROM OTA 文件格式定义

APPROM 的文件格式定义如下：文件的前段数据是纯的固件数据，文件将会被逻辑划分成若干个块，每个块的大小是 64 字节对齐，文件尾部不完整块用 FFh 填充，文件的最后一个块包含文件的 CRC，文件有效长度信息，版本信息等。



说明：

1) BIN DATA CRC: 使用 CRC 校验算法对 BIN DATA 计算出一个 CRC 值，不包含补位的数据。
2) BIN DATA LEN: BIN DATA 的长度，不包含补位的数据。
3) Firwware version: 固件的版本号。
4) FILE INFO CRC: 使用 CRC 校验算法对文件最后的一个区块字节，从 BIN DATA CRC 位置开始，长度为 128 - 4，计算出的 CRC 值，用于校验最后的区块是否是有效的内容。
5) CRC 算法如下：

```
uin 电池仓 6_t crc16_compute(const uint8_t * p_data, uint32_t size, const uin 电池仓 6_t * p_crc)
{
    uint32_t i;
    uin 电池仓 6_t crc = (p_crc == NULL) ? 0xffff : *p_crc;

    for (i = 0; i < size; i++)
    {
        crc  = (unsigned char)(crc >> 8) | (crc << 8);
        crc ^= p_data[i];
        crc ^= (unsigned char)(crc & 0xff) >> 4;
        crc ^= (crc << 8) << 4;
        crc ^= ((crc & 0xff) << 4) << 1;
    }
    return crc;
}
```

4.2. 文件的升级过程

通过操作数据字典进行升级

附录 1： Firmware Version 定义

MCU Firmware version adopts **GNU** style(Including 4 parts):
Major_Version_Number.Minor_Version_Number.Revision_Number.Build_Number

- Main Version Number (**主版号，1字节**)
  从 1 开始，当项目在进行重大修改或局部修正较多，而导致项目整体发生全局变化时，主版本号加 1.
- Minor Version Number (**子版本号，1字节**)
  当项目在原有的基础上增加了部分功能时，主版本号不变，子版本号加 1，修正版本号复位成 0.
- Revision Number **(修正版本号，1字节)**
  当项目进行了局部修改或 bug 修正时，主版本号和子版本号都不变，修正版本号加 1.
- Build Number **(编译版本号，4字节)**

Build Number 是不断递增的，如果 IDE 比较智能的话，每次打包发布时，会自动加 1。如果不是自动加 1 的话，每次对外发布新的 firmware 时，都需要手动加 1 或者参考 SVN 修订号.

附录 2： Hardware Version 定义

Hardware adopts the below version style(Including 2 parts):

Device_Typer. Major_Version_Number

Product Typer 定义如下：

Product Typer	Description
1	电池仓 Smart
其他值	保留