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 |
|  |  |  |  |

# 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。



## 硬件模块架构图



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。

## 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。

## 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：

# 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：



## 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：



## 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。**



## 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。



# 功能模块设计

E换电系统的内部模块架构如下图所示



## 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：



## 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：



## 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”。

## 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”。

## 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”。

## 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”。

## 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”。

## 日志模块

### 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 |

### 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 |
|  |  |  |  |  |

# 固件升级设计

## 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;

}

## 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 |