**智能充电柜单元集成协议**

# 1 数据包格式

数据包的格式及详细定义如表1-1、表1-2所示。

*表1-1 数据包格式*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Header** | **Len** | **Cmd** | **Enc** | **Seq** | **CRC-16** | **Data** |
| 1-byte | 2-byte | 1-byte | 1-byte | 1-byte | 2-byte | n-byte |

*表1-2 数据包详细定义*

|  |  |  |  |
| --- | --- | --- | --- |
| **域** | **偏移位置** | **大小（字节）** | **详细描述** |
| **Header** | 0 | 1 | 数据包起始字节，固定为0x55 |
| **Len** | 1 | 2 | [15:10]保留  [9:0]数据包长度  注：LSB在前，MSB在后 |
| **Cmd** | 3 | 1 | 命令码，用于表示具体的操作 |
| **Enc** | 4 | 1 | 用于表示Data域是否加密，附录加密算法 |
| **Seq** | 5 | 1 | 数据序列号,应答时与发送一致 |
| **CRC-16** | 6 | 2 | 包头检验码，是Header到Seq的crc-16 校验值,附录crc计算C程序  注：LSB在前，MSB在后 |
| **Data** | 8 | n(n<=1015) | 数据内容 |

# 2 命令码分配

通过can总线，将多个电池柜单元集成为电池柜产品，需要先对每个电池柜单元的can id 做设置，通过区分接收数据的can id以区分电池柜单元间位置关系，每个电池柜单元可以通过两个can id,一个是通用can id 0x7ff，一个是独立设置的can id。Cmd 0x00、0x01 使用通用can id 通信，其他命令使用独立设置的can id 通信，一个电池柜中的多个单元要使用不同的can id。

*表2-1 智能充电柜命令码分配*

|  |  |  |
| --- | --- | --- |
| **Cmd** | **说明** | **备注** |
| 0x00 | 电池柜单元CAN ID 设置 | 使用通用can id |
| 0x01 | 电池柜单元CAN ID 读取 | 使用通用can id |
| 0x02 | 保留 |  |
| 0x03 | 保留 |  |
| 0x04 | 认证，每次上电只有认证成功才会处理下面命令 | 使用独立设置的can id,下同 |
| 0x05 | 电池柜单元起始模式设置 | 从这里开始的命令只有认证成功才会处理，因此都使用非加密 |
| 0x06 | 电池柜单元可同时充电最大个数n(0<n<=15) 默认15 |  |
| 0x07 | 充电开关 |  |
| 0x08 | 请求电池柜单元信息 | loader版本、app版本、产品SN |
| 0x09 | 请求电池总静态数据 |  |
| 0x0a | 请求或请求推送 电池总动态数据 |  |
| 0x0b | 设置电池自放电天数 |  |
| 0x0c | 设置指示灯黄灯闪烁，提示用户拿取电池 |  |
| 0x0d | 保留 |  |
| 0x0e | 保留 |  |
| 0x0f | 保留 |  |
| 0x10 | 请求进入升级 | Enc = 0 支持loader版本1.3.0及以上版本 |
| 0x11 | 发送固件头信息0 | Enc = 0 |
| 0x12 | 发送固件头信息1 | Enc = 0 |
| 0x13 | 传输固件 | Enc = 0 |
| 0x14 | 固件传输完成 | Enc = 0 |

# 3 命令返回码

*表3-1 智能充电柜命令返回码解释*

|  |  |  |
| --- | --- | --- |
| 返回码 | 说明 | 相关CMD |
| 0x00 | 执行成功 |  |
| 0x01 | 一般错误 | 升级 |
| 0x02 | Flash不够 | 升级 |
| 0x03 | Flash 写失败 | 升级 |
| 0x04 | Flash 擦除错误 | 升级 |
| 0x05 | 长度校验出错 | 升级 |
| 0x06 | 固件检验出错 | 升级 |
| 0xD0 | 未认证 | 认证 |
| 0xD1 | 认证失败 |  |
| 0xD2 | 电池不在位 | 充电开关 |
| 0xD3 | 电池有异常，具体异常查看总动态数据 | 充电开关 |
| 0xD4 | 充电柜单元温度过高 | 充电开关 |
| 0xD5 | 超出最大同时充电设置个数 |  |
| 0xD6 | 电池已充满 |  |
| 0xD7 | 写入参数非法 |  |
| 0xD8 | Flash 写失败 |  |
| 0xD9 | 未满电，不可设置指示灯 | 设置指示灯 |

# 4 CAN 总线说明

电池柜单元间使用can总线通信，can 波特率1M BPS，标准数据帧，每个电池柜单元可通过两个ID,一个是通用的0x7ff,用于设置与获取特殊设置的ID。

电池柜单元发送的ID 与特殊设置的ID一致;

ID 排列方式为右对齐

# 5 命令详细说明

## 5.1 电池柜单元CAN ID 设置（0x00）

使用通用can id 通信,enc =0x01,加密算法需要16字节补齐，can id 使用标准can id(0~0x7ff),由于0x7ff被用作通用can id,因此用UINT32的can id(0~0x7fe)+12字节补齐，再通过加密算法，使用专门的私钥加密

注意：UINT16 LSB在前，MSB在后

*表5-1电池柜单元CAN ID 设置*

|  |  |  |  |
| --- | --- | --- | --- |
|  | 字节 | 说明 | 备注 |
| 请求数据 | 1：2 | UINT16 can id | Enc = 1 |
|  | 3:16 | 补齐加密用 |  |
| 应答数据 | 1 | 返回码 | Enc = 0 |

## 5.2 电池柜单元CAN ID 读取 (0x01)

读取到的enc = 0x01 ,通过解密算法，使用私钥解密

*表5-2电池柜单元CAN ID 读取*

|  |  |  |  |
| --- | --- | --- | --- |
|  | 字节 | 说明 | 备注 |
| 请求数据 | - | - | Enc = 0 |
| 应答数据 | 1 | 返回码 | Enc = 1 |
|  | 2:3 | UINT16 can id |  |
|  | 4:16 | 补齐加密用 |  |

## 5.3 电池柜单元认证 (0x04)

认证流程是，先向电池柜单元请求32字节的随机数A，再产生128字节的随机明文，用A加密明文产生新的128字节，用私钥B对256字节（128字节随机明文+新的128字节）做加密，即。发送整体的256字节认证内容。由电池柜单元验证后，给出返回码。

*表5-3电池柜单元认证命令*

|  |  |  |  |
| --- | --- | --- | --- |
|  | 字节 | 说明 | 备注 |
| 请求数据 | 1 | 操作类型：  0 – 请求随机数，即32字节随机数A  1 – 发送256字节认证内容，其中128字节随机明文，加128字节的密文，密文由A加密随机明文，再用私钥B加密 | 操作类型为0时  Enc = 0  操作类型为1时  Enc = 1 |
|  | 2:257 | 数据内容 | 操作类型为1时需要 |
|  | 258:272 | 使用加密，需补齐数据长度 | 操作类型为1时需要 |
| 应答数据 | 1 | 返回码 | Enc = 0 |
|  | 2 | 操作类型：  0 – 发送32字节随机数；  1 – 应答认证内容 |  |
|  | 3:34 | 数据内容 | 操作类型为0时有效 |

## 5.4 电池柜单元起始模式 (0x05)

模式1：起始存储模式，电池插上充电至存储状态，电量在50%以上，退出起始模式，低于50%，充电至50%退出起始模式。

模式2：起始满充模式，电池插上充电至满电，满电未拿走也将进入存储状态。

*表5-4 充电柜模式*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | 1 | 操作类型：  0 – 获取当前模式  1 – 写入模式 |
|  | 2 | 写入模式参数  1 – 模式1  0xff – \*模式2 |
| 应答数据 | 1 | 返回码 |
|  | 2 | 操作类型：  0 – 应答模式获取  1 – 应答写入 |
|  | 3 | 应答模式参数  1 – 模式1  0xff – \*模式2 |

（\* 为默认状态，可配置参数修改）

注意：此命令作用于电池第一次放入电池槽，且充电开关状态为自动。

## 5.5 最大同时充电个数 (0x06)

考虑到受工作环境电源功率，加此保护；更有效的保护应当在电池柜的工控板，协调控制多个电池柜单元的总功率。

*表5-5 电池柜单元最大同时充电个数*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | 1 | 操作类型：  0 – 获取当前最大充电个数  1 – 写入最大充电个数 |
|  | 2 | 写入最大充电个数参数n  0<n<=15 |
| 应答数据 | 1 | 返回码 |
|  | 2 | 操作类型：  0 – 应答最大充电个数获取  1 – 应答写入，失败原因查看返回码 |
|  | 3 | 应答最大充电个数参数n  0<n<=15 |

注意：在充电过程中修改最大同时充电个数会使电池序号小且充电开关状态是自动的电池停止充电，如所有充电开关为自动的已经关闭还不满足最大同时充电个数，将关闭电池序号小、充电开关状态为开的，充电开关被设置自动,

总而言之，充电开关状态为打开的比充电开关为自动的有更高的优先级打开充电开关

## 5.6 充电开关 (0x07)

可控制电池柜单元的15个充电开关，关闭一定关闭，打开要满足条件才能打开，自动充电柜单元会工作在存储状态（控制电量在40%~60%），电池在充满后会将开关置为自动，在自动状态下，会适应最大同时充电个数的限制。

*表5-6 充电开关控制*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | 1 | 操作类型：  0 – 获取充电开关状态  1 – 写入开关状态 |
|  | 2 | 写入开关状态参数  1 – 打开  2 – 关闭  0xff – \*自动 |
|  | … | … |
|  | 16 | 1 – 打开  2 – 关闭  0xff – \*自动 |
| 应答数据 | 1 | 返回码 |
|  | 2 | 操作类型：  0 – 应答充电开关状态  1 – 应答写入,失败原因查看返回码 |
|  | 3 | 应答充电开关状态  1 – 打开  2 – 关闭  0xff – \*自动 |
|  | … | … |
|  | 17 | 1 – 打开  2 – 关闭  0xff – \*自动 |

（\* 为默认状态，可配置参数修改）

## 5.7 请求电池柜单元信息 (0x08)

*表5-7 请求电池柜单元信息*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | - | - |
| 应答数据 | 1 | 返回码 |
|  | 2:11 | Loader版本  2:3 UINT16主版本  4:5 UINT16次版本  6:7 UINT16测试版本编号，发布版本为0  8:9 UINT16版本发布日期：年  10 UINT8版本发布日期：月  11 UINT8版本发布日期：日 |
|  | 12:21 | App版本  12:13 UINT16主版本  14:15 UINT16次版本  16:17 UINT16测试版本编号，发布版本为0  18:19 UINT16版本发布日期：年  20 UINT8版本发布日期：月  21 UINT8版本发布日期：日 |
|  | 22 | UINT8 电池类型   |  |  | | --- | --- | | 0x00 | P4系列电池 | |  |  | |
|  | 23:36 | 产品编号Char[14] |

## 5.8 电池总静态数据 (0x09)

区分电池柜的数据类型，静态数据在充电过程中不会变化，在读到总动态数据总电池数有变化时，再次请求该命令更新数据。或请求为变化时推送，及时获取最新的总静态数据。

*表5-8获取电池总静态数据*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | 1 | 操作类型  1 – 请求一次  2 – 请求一次并要求变化时主动推送 |
| 应答数据 | 1 | 返回码 |
|  | 2 | 操作类型  1 – 请求一次  2 – 请求一次并要求变化时主动推送 |
|  | 3：4 | UINT16  BIT0:BIT14 分别表示电池1~15的在位信息  1 – 电池在位  0 – 电池不在位  通过这个，可以知道电池在位数量，还可以知道是哪些位置，根据电池数的不同，携带不定长数据结构 |
|  | 5 | UINT8电池位置序号 （1~15）  后   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 1 | 2 | 3 | 4 | 5 | | 6 | 7 | 8 | 9 | 10 | | 11 | 12 | 13 | 14 | 15 |   前 |
|  | 6:7 | UINT16 电池生产日期  Bit0:Bit4 day  Bit5:Bit8 month  Bit9:Bit15 year-1980 |
|  | 8:9 | UINT16 循环次数 |
|  | 10:11 | UINT16 满充容量 mAh |
|  | 12:19 | 版本号  12:15 loader 版本号  16:19 app 版本号 |
|  | 20 | UINT8 寿命百分比 |
|  | 21:34 | UINT8[14] 流水码，与电池底部标签一致 |
|  | … | … |

## 5.9 电池总动态数据 (0x0a)

*表5-9 电池总动态数据请求或请求推送*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | 1 | 操作类型：  1 – 请求一次  2 – 按1Hz推送  3 – 停止推送 |
| 应答数据 | 1 | 返回码 |
|  | 2 | 操作类型：  1 –请求一次  2 – 按1Hz 推送  3 – 停止推送 |
|  | 3 | UINT8 电池柜单元警告信息   |  |  | | --- | --- | | BITO | 适配器故障 | | BIT1 | 电池柜单元温度过高 | | BIT2 | 电池柜单元温度过低 | | BIT3 | 电池柜单元风扇异常 | | BIT4:BIT7 | 保留 | |
|  | 4：5 | UINT16  BIT0:BIT14 分别表示电池位置序号1~15  1 – 电池在位  0 – 电池不在位  通过这个，可以知道电池在位数量，还可以知道是哪些位置，根据电池数的不同，携带不定长数据结构 |
|  | 6 | UINT8电池位置序号 （1~15）  后   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 1 | 2 | 3 | 4 | 5 | | 6 | 7 | 8 | 9 | 10 | | 11 | 12 | 13 | 14 | 15 |   前 |
|  | 7:8 | UINT16 电压 mV |
|  | 9:10 | UINT16 温度 0.1℃ |
|  | 11:12 | UINT16 相对容量百分比 |
|  | 13:20 | UINT16[4]单节电压 mV |
|  | 21 | UINT8预计满充所需时间 分钟 |
|  | 22 | UINT8 开始自放电时间 d(默认10天) |
|  | 23:24 | 报警原因   |  |  | | --- | --- | | BIT0 | 电池通信异常 | | BIT1 | 电池一级放电过流 | | BIT2 | 电池二级放电过流 | | BIT3 | 电池一级放电过温 | | BIT4 | 电池二级放电过温 | | BIT5 | 电池一级放电低温 | | BIT6 | 电池二级放电低温 | | BIT7 | 电池放电短路 | | BIT8 | 电池电芯欠压 | | BIT9 | 电池电芯损坏 | | BIT10:BIT15 | 保留 | |
|  | … | … |

## 5.10 ***电池自放电天数 (0x0b)***

*表5-10 电池自放电天数设置*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | 1:2 | UINT16  BIT0:BIT14 分别表示电池位置序号1~15  1 – 要设置自放电天数  0 – 不需设置，忽略下面携带的参数  （若全都设置，填0x7fff） |
|  | 3 | 电池序号1的设置  n (0~10) UINT8 天 |
|  | … | … |
|  | 17 | 电池序号15的设置  n (0~10) UINT8 天 |
| 应答数据 | 1 | 返回码 |

## 5.11 ***设置指示灯 (0x0c)***

电池在正常充满电后是绿灯常亮，在此状态下可设置蓝灯闪烁，提示用户拿取对应电池，便于资产管理

当电池取出，此设置失效；

当电池一直未取出，到了未满电，此设置失效。

*表5-11 充电柜单元指示灯设置*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | 1 | UINT8 灯的序号  后   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 1 | 2 | 3 | 4 | 5 | | 6 | 7 | 8 | 9 | 10 | | 11 | 12 | 13 | 14 | 15 |   前 |
|  | 2 | UINT8 操作类型  1设置为蓝灯闪烁  2 取消控制 |
| 应答数据 | 1 | 返回码 |
|  |  |  |

## 5.12 请求进入升级***(0x10)***

在app收到不应答，软复位进入loader

Loader收到应答，可以执行下一步

*表5-12 请求进入升级*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | - | - |
| 应答数据 | 1 | 返回码 |

## 5.13 发送固件头信息0 ***(0x11)***

固件文件下标从0开始

*表5-13 发送固件头信息0*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | 1:32 | 发送固件字节[2:33] |
| 应答数据 | 1 | 返回码 |

## 5.14 发送固件头信息1 ***(0x12)***

固件文件下标从0开始

*表5-14 发送固件头信息1*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | 1:32 | 发送固件字节[34:65] |
| 应答数据 | 1 | 返回码 |

## 5.15 传输固件***(0x13)***

固件文件下标从0开始

*表5-15 传输固件*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | 1:N | 从固件bin[66]开始读，N必须为16的倍数，例如每次发1008字节，注意不能超出协议规定最大字节数，直到发完 |
| 应答数据 | 1 | 返回码 |

## 5.16 固件传输完成***(0x14)***

固件传输完成后，发送此命令，固件开始校验并写入flash

*表5-16 固件传输完成*

|  |  |  |
| --- | --- | --- |
|  | 字节 | 说明 |
| 请求数据 | - | - |
| 应答数据 | 1 | 返回码 |

# 6 应用示例

业务流程：

1. 每个电池柜单元单独设置独立CAN ID，例如一个柜装三个电池柜单元，可以分别设置为0x444,0x555,0x666.
2. 设置完成后，工控机分别与三个电池槽通信认证。
3. 请求电池柜单元起始模式、最大同时充电个数、电池柜单元信息。
4. 请求总静态数据变化时主动推送、请求1Hz的总动态数据推送。
5. 若考虑功率限制，只需设置每个电池柜单元的最大同时充电个数
6. 电池柜单元会检测内部温度过高时报警并停止充电。电池柜如果检测到环境异常，可以设置充电开关为关闭。
7. 根据电池预约使用要求，打开充电开关，注意自放电天数设置大于0天，不然电池充满后很快开始自放电，为了使电池的使用更加平衡，优先使用循环次数低的电池。
8. 一次预约充电后，读到预计满充时间为0，则表示电池已经充满，这个条件下可以设置提示灯提醒用户取走对应电池。

# 7 版本变更记录

*表7-1 智能充电柜集成协议版本维护记录表*

|  |  |  |
| --- | --- | --- |
| 变更日期、版本号 | 变更内容 | 维护人 |
| 2018/03/06 v1.0 | 创建初始文档 |  |
| 2018/03/08 v1.1 | 1修改总静态数据内容，增加流水码，去除序列号  2修正字节描述错误 |  |
| 2018/03/09 v1.2 | 命令码增加flash写错误 |  |
| 2018/03/10 | 1修改认证的描述  2 canid使用4字节描述 |  |
| 2018/05/09 | 1修改返回码  2修改默认起始模式  3充电开关默认值设为0xff |  |
| 2018/05/11 | enc的使用 |  |
| 2018/05/15 v1.3 | 新增5.11对指示灯的控制  新增5.7里电池类型描述 |  |
| 2018/05/28 v1.4 | 在应用实例里增加使用流程，指示灯由黄灯闪烁改为蓝灯闪烁  修改起始模式判断 |  |
| 2018/06/08 v1.5 | 新增描述最大同时充电个数受限制时，关闭的优先级，新增升级协议 |  |

# 8 附录

见智能充电柜单元开发文件夹下。