**定级键盘-确认刷卡设备通信协议**

# 简介

收购线组成框图

|  |  |
| --- | --- |
| 收购线1  交烟刷卡  定级键盘  磅码主机  RS485通信  Zigbee无线通信 | 收购线2  交烟刷卡  定级键盘  磅码主机  RS485通信  Zigbee无线通信 |

交烟刷卡设备通过Zigbee无线与定级键盘通信，定级键盘通过RS485与磅码主机通信。

磅码主机与交烟刷卡之间的数据交互则**通过定级键盘转发**。Zigbee的设备地址由设备中的拨码开关决定。磅码主机与哪个交烟刷卡设备通信由数据帧中的地址决定。

# 1 通信帧格式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字节数 | 类型 | 取值 | 描述 |
| header | 1 | 字节 | 0xAA | 帧开始标记 |
| receiver | 1 | 字节 | **注1** | 接收者，标识数据要发给谁 |
| sender | 1 | 字节 | 发送者，标识数据是谁发出 |
| address | 1 | 字节 | 1~15 | Zigbee通信地址（同收购线号） |
| msgsn | 1 | 字节 | 0~255 | 帧序号（成功发送后加1） |
| cmd | 1 | 字节 | \* | 命令码（参见各章节） |
| size | 4 | 整数 | >=0 | 数据区字节数（小端格式 **注2**） |
| data | \* | 字节 | \* | 数据区，可无(*size*等于0，*size*后紧接校验码) |
| checksum | 1 | 字节 | \* | 从*receiver*到整个*data*的**和校验** |

**注1:** 1磅码主机、2定级键盘、3确认刷卡。

**注2:** 低字节在前，如0x12345678在数组中存放为{0x78, 0x56, 0x34, 0x12}

**注2.1:** 下文中如无特别说明所有表达数值的类型都是小端格式

下面以磅码与刷卡的交互说明帧格式的含义：

AA header:帧开始标记，如果设备收到的第一个字节不是此值则可认为无效

03 receiver:让交烟刷卡接收此帧数据

01 sender:此帧数据是磅码主机发出的

01 address:发给设备地址为1的zigbee模块（1号线的交烟刷卡设备）

02 msgsn:每帧数据的此值不同

11 cmd:读卡命令

02 00 00 00 size:数据区2字节

00 01 data:数据

1B checksum:校验码

由于没读到卡，刷卡设备仅回复一条消息表示收到请求：

AA

01 :发给磅码主机

03 :是交烟刷卡发的数据

01 :来自1号线的设备

46

12 :对读卡请求的应答（表示刷卡设备在线）

00 00 00 00 :没有数据

5D

# 2定级键盘

## 2.1进入等待工作员刷卡状态

*磅码主机发送：*

**命令码：**0x05

**数 据：**0x01（状态码）

示例：

AA 02 01 01 03 05 01 00 00 00 01 0E

|  |  |
| --- | --- |
| AA | 帧头标记 |
| 02 | 发送给定级键盘 |
| 01 | 是磅码主机发出的 |
| 01 | 发给1号收购线 |
| 03 | 帧序号 |
| 05 | 命令 |
| 01 00 00 00 | 数据区长度为1 |
| 01 | 数据 |
| 0E | 校验 |

## 2.2进入等待烟农刷卡状态

*磅码主机发送：*

**命令码：**0x09

**数 据：**0x01（状态码）

示例：

AA 02 01 01 04 09 01 00 00 00 01 13

|  |  |
| --- | --- |
| AA | 帧头标记 |
| 02 | 发送给定级键盘 |
| 01 | 是磅码主机发出的 |
| 01 | 发给1号收购线 |
| 04 | 帧序号 |
| 09 | 命令 |
| 01 00 00 00 | 数据区长度为1 |
| 01 | 数据 |
| 13 | 校验 |

## 2.3恢复定级信息到定级键盘

### 2.3.1定级键盘请求恢复定级信息

当定级键盘重启后会发送恢复自己上次工作状态的请求。

*定级键盘发送：*

**命令码：**0x06

**数 据：**无

示例：

AA 01 02 01 01 06 00 00 00 00 0B

### 2.3.2发送定级信息到定级键盘

当磅码主机进入收购后如果定级键盘重启则需要将收购数据恢复到定级键盘。

*磅码主机发送：*

**命令码：**0x05

**数 据：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | | 类型 | 字节 | 取值 | 描述 |
| 状态 | | 字节 | 1 | 2 | 2正在定级、3定级完成 |
| 合同号 | | 字符 | 19 | 0 | 在旧的设计中用定级键盘保存收购数据，以便在需要时恢复到磅码设备中。  如果不用此功能可以将这些项置零。 |
| 初检员编号 | | 字符 | 16 | 0 |
| 流水号 | | 整数 | 4 | 0 |
| 总金额 | | 整数 | 4 | 0 |
| 总重量 | | 整数 | 4 | 0 |
| 合同总量 | | 整数 | 16(4\*4) | 0 |
| 合同余量 | | 整数 | 16(4\*4) | 0 |
| 今日余量 | | 整数 | 4 | 0 |
| 品种、等级信息 | | 字节 | 104 | 0 |
| 定级信息个数 | | 整数 | 4 | 0~72 | 定级键盘最多可保存72条记录 |
| 定  级  信  息  **注4** | 等级ID | 字节 | 0~72 | 1~43 |  |
| 定级状态 | 字节 | 1 | 0 |  |
| 重量 | 短整数 | 2 | 0 |  |
| 定级时间 | 整数 | 4 | 0 |  |
| 磅码时间 | 短整数 | 2 | 0 | 与定级时间的差值 |
| 皮重 | 短整数 | 2 | 0 |  |

**注4：每定一个级就会有一个***定级信息***结构**

示例：向定级键盘恢复1个等级（B1K）

AA 02 01 01 04 05 CC 00 00 00 02 36 32 31 34 35 37 30 32 38 31 30 30 30 34 39 38 38 37 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 01 00 00 00 00 00 00 00 00 00 00 00 90 5F 01 00 C0 3E 01 00 00 00 00 00 D0 20 00 00 45 0F 01 00 31 0F 01 00 00 00 00 00 14 00 00 00 45 0F 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 26 38 00 00 82 9F 48 58 D9 18 00 00 A3

|  |  |
| --- | --- |
| AA 02 01 01 04 05 | （部分包头） |
| CC 00 00 00 | 数据区长度204字节 |
| 02 | 进入定级状态 |
| 36 32 31 34 35 37 30 32 38 31 30 30 30 34 39 38 38 37 30 | 银行卡号 |
| 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 | 定级员卡号 |
| 01 00 00 00 | 流水号为1 |
| 00 00 00 00 | 总金额 |
| 00 00 00 00 | 总重量 |
| 90 5F 01 00 C0 3E 01 00 00 00 00 00 D0 20 00 00 | 合同总量4个 |
| 45 0F 01 00 31 0F 01 00 00 00 00 00 14 00 00 00 | 合同余量4个 |
| 45 0F 01 00 | 今日余量 |
| 数据略（共104字节） | 品种、等级信息 |
| 01 00 00 00 | 等级个数为1 |
| 26  38 00 00 82 9F 48 58 D9 18 00 00 | B1K的等级ID |

## 2.4定级键盘报告定级消息

*定级键盘发送：*

**命令码：**0x08

**数 据：**第0字节为状态（2正在定级、3定级完成），后续字节为已定级的等级ID。

示例：定级B1K

AA 01 02 01 08 08 02 00 00 00 02 26 3E

|  |  |
| --- | --- |
| AA 01 02 01 08 08 |  |
| 02 00 00 00 | 数据区两字节 |
| 02 | 正在定级 |
| 26 | B1K等级ID |
| ...... | 更多等级则附加到后面，相应数据区大小增加 |
| 3E | 校验码 |

磅码主机收到定级请求后需要应答消息已收到，否则定级键盘会一直重复发送此消息。

*磅码主机发送：(表示已收到定级)*

**命令码：**0x07

**帧序号：**等于接收帧中的帧序号

**数 据：**无

示例：

AA 02 01 01 08 07 00 00 00 00 13

## 2.5进入定级状态

参见2.3节，此时***定级信息个数***为0

# 3交烟刷卡

## 3.1等待烟农卡（磅码反复发送此消息直到读到卡）

*磅码主机发送:*

**命令码：**0x11

**数 据：**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 数据类型 | 取值 | 说明 |
| 是否启用初检刷卡 | 字节 | 0/1 | 0:关闭 1:开启 |
|  | 字节 | 0x01 |  |

示例：

AA 03 01 01 02 11 02 00 00 00 00 01 1B

*① 刷卡设备发送:（应答收到读卡请求）*

**命令码：**0x12

**数 据：**无

示例：

AA 01 03 01 46 12 00 00 00 00 5D

*② 刷卡设备发送：(应答卡号)*

**命令码：**0x16

**数 据：**

|  |  |  |
| --- | --- | --- |
| ***字段*** | ***字节*** | ***描述*** |
| 状态 | 1 |  |
| 卡号 | 19 | ascii |
| 分隔符 | 1 | ‘,’（取值固定为ascii逗号） |
| 初检员编号 | 16 | ascii |

示例：

AA 01 03 01 A0 16 25 00 00 00 00 36 32 31 34 35 37 32 31 38 30 30 30 30 37 31 35 36 37 31 2C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 DB

|  |  |
| --- | --- |
| AA 01 03 01 A0 16 |  |
| 25 00 00 00 | 数据部分37字节 |
| 00 |  |
| 36 32 31 34 35 37 32 31 38 30 30 30 30 37 31 35 36 37 31 | 卡号 |
| 2C | 逗号分隔符 |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | 定级员卡号（为0表示未刷卡） |
| DB | 校验码 |

## .3.2验证烟农通过

*磅码主机发送：*

**命令码：**0x15

**数 据：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***字段*** | ***类型*** | ***字节*** | ***取值*** | ***描述*** |
| 结果码 | 字节 | 1 | 0 | 成功 |
| 合同余量 | 整数 | 4 | \* | 单位10g（例:1Kg取值为100） |
| 姓名 | 字符 | \* | \* | gb2312编码 |
| 结束符 | 字符 | 1 | 0 |  |

示例：

AA 03 01 01 D4 15 0E 00 00 00 00 45 0F 01 00 BC D6 B0 CD C2 B3 D6 BB 00 66

|  |  |
| --- | --- |
| AA 03 01 01 D4 15 |  |
| 0E 00 00 00 | 数据区14字节 |
| 00 | 成功 |
| 45 0F 01 00 |  |
| BC D6 B0 CD C2 B3 D6 BB |  |
| 00 | 字符串结束符 |
| 66 | 校验码 |

## 3.3切换到等待烟农卡状态（交售完成后磅码发送一次此消息）

*磅码主机发送:*

**命令码：**0x11

**数 据：**0x00,0xFF

示例：

AA 03 01 01 01 11 02 00 00 00 00 FF 18

|  |  |
| --- | --- |
| AA 03 01 01 01 11 |  |
| 02 00 00 00 | 数据区2字节 |
| 00 FF | 数据区 |
| 18 | 校验码 |

*刷卡设备发送:*

**参见3.1节**

# 附录：

## 和校验算法

unsigned char chksum(unsigned char \*buf, short len)

{

uint8\_t tmp = 0;

for (int i = 0; i < len; i ++)

{

tmp += buf[i];

}

return tmp;

}