Linux

智能卡读写器 API 函数说明

2024/02/20

目录

1 概述

智能 IC 卡读写器采用了串口、USB 接口通讯，提供的接口函数库可满足用户二次开发的需要，其完善、可靠的接口函数，支持访问全部卡片的所有功能。

动态库链接库：libHcReader-1.0.so

USB 设备依赖库：libusb-1.0.so libusb-1.0.so.0 libusb-1.0.so.0.1.0

使用麒麟系统 deb 安装包后动态库存放位置如下：

/opt/hc/reader-driver-hc-common/app/lib

函数调用规则：

1、 程序开始或操作设备之前首先要调用打开读写器函数（不同通信方式的设备函数不同）建立读写器与 PC 之间的连接

2、 调用设备函数对设备进行操作/调用卡片操作函数对卡片进行操作

3、 程序正常退出或结束操作之前，要用关闭函数断开读写器与 PC 之间的连接

2 Api 说明

2.1 设备操作函数

2.1.1 初始化设备

函 数：**short hc_init(char* spName, unsigned long baud);**

说 明：该函数用于建立读写器与 PC 机之间的连接，适用于串口通信方式的读写器。

参 数：

spName：串口设备文件名，如“/dev/ttys0”。

Baud： 为通讯波特率 9600～115200

返 回：>= 0 成功；< 0 失败

2.1.2 初始化设备

函 数：**short hc_init_usb(void);**

说 明：打开 usb 通信读写器,此函数只适用于 USB 通讯设备.

参 数：无

返 回：>= 0 成功；< 0 失败

2.1.3 初始化设备

函 数：**short hc_net_init(char *ip,int m_iPort);**
功 能：该函数用于建立读写器与 PC 机之间的连接，TCP/IP 网口连接专用函数
参 数：
Ip：设备 IP 地址。
m_iPort： 网络端口号。
返 回：>= 0 成功；< 0 失败

2.1.4 关闭设备

函 数：**short hc_exit();**
功 能：断开 PC 机与读写器之间的连接，并释放相关设备描述符
参 数：无
返 回：>= 0 成功；< 0 失败

2.1.5 网络设备重启

函 数：**short dv_reset(void);**
功 能：用于更改网络参数后远程重启设备(仅适用于网络通信设备)
参 数：无
返 回：>= 0 成功；< 0 失败

2.1.6 设置本机网络参数

函 数：**short dv_setlocnet(char *loc_ip, int loc_port,**
char *gateway, char *mask);
功 能：设置读写器本机网络参数(仅适用于网络通信设备)
参 数：
loc_ip：要设置的设备 ip 地址
loc_port：要设置的设备网络端口号
gateway：要设置的设备网关
mask：要设置的子网掩码
返 回：>= 0 成功；< 0 失败

2.1.7 设置服务器网络参数

函 数: **short dv_setsernet(char *server_ip,**
unsigned int server_port);

功 能: 当读写器作为客户端自动上传数据时，设置服务器的网络参数(仅适用于网络通信设备)

参 数:

server_ip: 服务器 ip 地址

server_port: 服务器网络端口号

返 回: >= 0 成功; < 0 失败

2.1.8 读取网络设备参数

函 数: **short dv_srdnet(unsigned char *mode, char *loc_ip,**
unsigned int *loc_port, char *gateway,char *mask,
char *server_ip, unsigned int *server_port);

功 能: 读取设备网络参数(仅适用于网络通信设备)

参 数:

mode：读取到的网络设备工作模式

0 — 被动模式，设备作为服务器

1 — 主动读卡模式。设备作为客户端并主动读卡，然后把读到的卡数据发送至服务器

loc_ip: 读取到的读写器 IP 地址。

loc_port: 读取到的读写器网络端口号

gateway: 读取到的设备网关

mask: 读取到的设备子网掩码

server_ip: 读取到的服务器 IP 地址

server_port: 读取到的服务器端口号

返 回: >= 0 成功; < 0 失败

2.1.9 设置网络设备工作模式

函 数: **short dv_setnetmode(unsigned char mode);**
功 能: 设置网络设备工作模式(仅适用于网络通信设备)
参 数:
mode: 网络设备工作模式
0 — 被动模式，设备作为服务器
1 — 主动读卡模式。设备作为客户端并主动读卡，然后把读到的卡数据发送至服务器
返 回: >= 0 成功; < 0 失败

2.1.10 设置网络设备主动读卡模式

函 数: **short dv_setnetreadcard(unsigned char mode,**
unsigned char block_Adr, unsigned char block_Num,
unsigned char *_Key);
功 能: 当读写器设置为主动读卡模式时，需要设置主动读卡模式参数(仅适用于网络通信设备)
参 数:
mode: 主动读卡的模式.最高位表示蜂鸣及指示灯是否有效,1 为有效,0 为无效其余位取值如下
0 — 读取非接触卡片 UID，4 字节/7 字节
1 — 读取 M1 卡内扇区数据，
block_Adr: M1 卡数据起始块地址，当 mode = 0 时此参数无效
block_Num: 连续读取块长度，当 mode = 0 时此参数无效
_Key: 读取数据的块号所在扇区的密码，当 mode = 0 时此参数无效
返 回: >= 0 成功; < 0 失败

2.1.11 读取网络设备主动读卡模式

函 数: **short dv_srdnetreadcard(unsigned char *mode,**
unsigned char *block_Adr, unsigned char *block_Num,
unsigned char *_Key);
功 能: 读取主动读卡模式参数(仅适用于网络通信设备)
参 数:
mode: 读取到的主动读卡的模式. 最高位表示蜂鸣及指示灯是否有效,1 为有效,0 为无效。其余位取值如下

0 — 读取非接触卡片 UID，4 字节/7 字节

1 — 读取 M1 卡内扇区数据

block_Adr: 读取 M1 卡数据起始块地址，当 mode = 0 时此参数无效

block_Num: 读取连续读取块长度，当 mode = 0 时此参数无效

_Key: 读取数据的块号所在扇区的密码，当 mode = 0 时此参数无效

返 回: >= 0 成功; < 0 失败

2.1.12 获取设备用户卡槽状态

函 数: **short get_status(short *state);**

功 能: 获取用户卡槽是否有卡

参 数:

state: 插卡状态。 0 -- 无卡; 1 -- 有卡

返 回: >= 0 成功; < 0 失败

2.1.13 蜂鸣

函 数: **short dv_beep(short time);**

功 能: 读写器鸣响

参 数:

time: 蜂鸣时间，值范围 0-255（单位 10ms）

返 回: >= 0 成功; < 0 失败

2.1.14 读取设备硬件版本号

函 数: **short srd_ver(unsigned char *data_buffer);**

功 能: 读取设备硬件版本号

参 数:

data_buffer: 存放读取的硬件版本号字符串,最少分配 18 字节空间

返 回: >= 0 成功; < 0 失败

2.1.15 读取产品序列号

函 数: **short srd_snr(unsigned char *data_buffer);**

功 能: 读取产品序列号

参 数:

data_buffer: 存放读取的产品序列号

返 回: >= 0 成功; < 0 失败.

2.1.16 设置串口通讯波特率

函 数: **short set_baud(unsigned long baud);**
功 能: 设置串口通信波特率，设置完毕后,必须重新初始化通讯口
参 数:
baud: 要设置的串口波特率，取值 9600 ~ 115200
返 回: >= 0 成功; < 0 失败.

2.1.17 读取设备 eeprom 数据

函 数: **short srd_eeprom(short offset, short len,**
unsigned char *data_buffer);
功 能: 读取读写器 eeprom 中的数据
参 数:
offset: 读取的起始偏移地址,范围: 0 ~ 999
len: 要读取的数据长度,范围: 1 ~ 1000
data_buffer: 存放读取到的数据
返 回: >= 0 成功; < 0 失败

2.1.18 写入设备 eeprom 数据

函 数: **short swr_eeprom(short offset, short len,**
unsigned char *data_buffer);
功 能: 向读写器 eeprom 中写入数据
参 数:
offset: 写入的起始偏移地址, 范围: 0 ~ 999
len: 要写入的数据长度, 范围: 0 ~ 1000
data_buffer: 要写入的数据
返 回: >= 0 成功; < 0 失败

2.1.19 复位/关闭射频场强信号

函 数: **short rf_reset (unsigned short _Msec);**
功 能: 将 RF（射频 13.56MHz）模块的能量先关闭设置时间后在重新打开射频，此操作会让所有在天线区域的卡片都回到初始状态
参 数:
_Msec: 复位场强射频信号时间 (1~ 500ms)；0 —- 关闭射频场强信号
返 回: >= 0 成功; < 0 失败

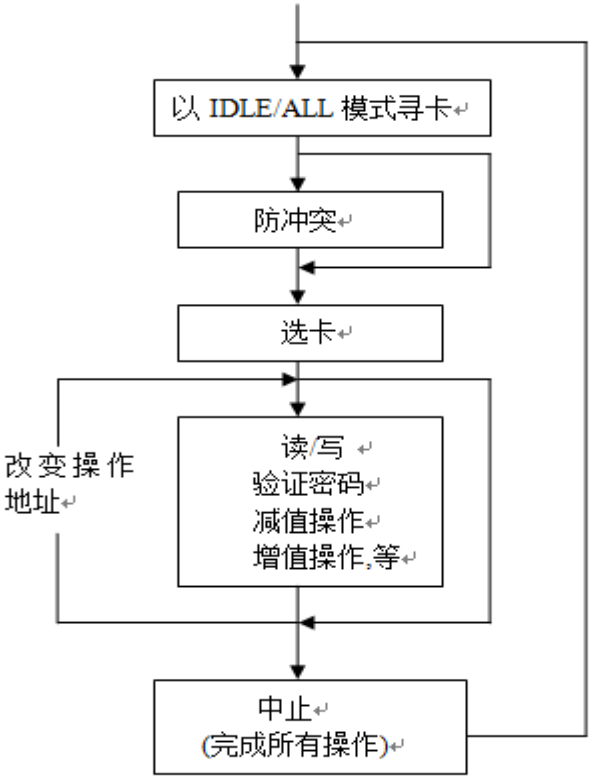
2.1.20 选择协议

函 数: **short rf_select_protocal(unsigned char protype);**
功 能: 选择要操作的卡片协议

参 数:
protype: 0 -- ISO14443 TypeA; 1 -- ISO14443 TypeB; 2 -- ISO15693
返 回: >= 0 成功; < 0 失败
注: 读写器默认协议为 ISO14443 TypeA 协议

2.2 非接触式高频卡片操作函数(13.56MHz)

2.2.1 ISO14443 TypeA 卡片操作函数（S50/S70）



Mifare 标准非接触卡操作流程图

2.2.1.1 请求卡片

函 数: **short rf_request(unsigned char _Mode,**
unsigned short *TagType);

功 能: 向卡片发出寻卡命令，开始选择一张新卡片时需要执行该函数

参 数:

_Mode: 寻卡模式:

0 -- IDLE mode, 只有处在 IDLE 状态的卡片才响应读写器的命令

1 -- ALL mode, 处在 IDLE 状态和 HALT 状态的卡片都将响应读写器的命令

TagType: 返回卡片类型

Mifare std. 1k(S50): 0x0004

Mifare std. 4k(S70): 0x0002

UltraLight: 0x0044

MIFARE LIGHT：0x0010

MIFARE PRO：0x0008

MIFARE PLUS：0x00B4

FM005：0x0005

SHC1102：0x3300

返 回：>= 0 成功；< 0 失败

2.2.1.2 防冲突

函 数：**shrot rf_anticoll(unsigned char *_Snr);**

功 能：激活读写器的防冲突队列。如果有几张 mifare 卡片在感应区内，将会选择一张卡片并返回卡片的序列号供后续调用 rf_select 函数时使用.

参 数：

_Snr： 返回卡片序列号

返 回：>= 0 成功；< 0 失败

2.2.1.3 选择卡片

函 数：**short rf_select(unsigned char* _Snr, unsigned char *_Size);**

功 能：用指定的序列号选择卡片，将卡片的容量返回给 PC 机

参 数：

_Snr：rf_anticoll 函数返回的卡片序列号

_Size：卡片容量的地址指针，目前该值不能使用

返 回：>= 0 成功；< 0 失败

2.2.1.4 高级寻卡

函 数：**short rf_card(unsigned char _Mode, unsigned char *_Snr);**

功 能：寻卡并返回卡片的序列号,它可以完成函数 rf_request, rf_anticoll 和 rf_select 3 个函数的功能

参 数：

_Mode：寻卡模式，同 rf_request

_Snr：返回卡片序列号

返 回：>= 0 成功；< 0 失败

2.2.1.5 高级寻卡（7 字节 UID）

函 数: **short rf_card_7uid(unsigned char _Mode,**
unsigned char *_Snr);

功 能: 寻卡并返回卡片的序列号，此函数仅用于 7 字节序列号的卡片

参 数:

_Mode: 寻卡模式，同 rf_request

_Snr: 返回 7 字节的卡片序列号

返 回: >= 0 成功; < 0 失败.

2.2.1.6 将卡片置于 halt 状态

函 数: **short rf_halt();**

功 能: 将选中的卡片设为“Halt”停止状态，只有当该卡再次复位或用 ALL 模式调用 rf_request 函数时，读写器才能够再次操作它

参 数: 无

返 回: >= 0 成功; < 0 失败.

2.2.1.7 装载密码

函 数: **short rf_load_key(unsigned char _Mode,**
unsigned char _SecNr,unsigned char *_NKey);

功 能: 向读写器装载指定扇区的新密码（不与卡片进行通讯），读写器中有 16 个扇区的密码（0~15），每个扇区有两组密码(KEY A 和 KEY B)

参 数:

_Mode: 装载密码类型，0 -- 装载 KEY A; 4 -- 装载 KEY B

_SecNr: 装载密码的扇区号(0～15)

_Nkey: 6 字节密码

返 回: >= 0 成功; < 0 失败.

2.2.1.8 校验密码

函 数: **short rf_authentication(unsigned char _Mode,**
unsigned char _SecNr);

功 能: 验证卡片的密码与读写器中装载的密码(同一扇区)是否一致。如果卡片的密码与读写器中的密码（可用 rf_load_key 函数修改）相匹配，则密码验证通过

参 数:

_Mode: 验证密码类型，0 -- 用 KEY A 验证; 4 -- 用 KEY B 验证

_SecNr: 已装载密码的扇区号(0～15)

返 回: >= 0 成功; < 0 失败.

注: 密码验证成功后，所验证的扇区内所有块均可操作.

2.2.1.9 校验密码

函 数: **short rf_authentication_2(unsigned char _Mode,**
unsigned char KeyNr, unsigned char Adr);

功 能: 验证卡片所要访问扇区的密码与读写器中装载的密码（指定扇区）是否一致。如果卡片的密码与读写器中的密码（可用 rf_load_key 函数修改）相匹配，则密码验证通过。**主要用于验证扇区号大于 15 的扇区.**

参 数:

_Mode: 验证密码类型， 0 -- 用 KEY A 验证; 4 -- 用 KEY B 验证

KeyNr: 已装载密码的扇区号(0～15)

Adr: 卡片数据**块地址**

返 回: >= 0 成功; < 0 失败.

注: 密码验证成功后，所验证块的扇区内所有块均可操作.

2.2.1.10 校验密码（建议使用此函数）

函 数: **short rf_authentication_key(unsigned char _Mode,**
unsigned char _BlockNr, unsigned char *_Key);

功 能: 利用函数参数中提供的密码对卡片指定数据块进行认证。如果参数中提供的密码与卡片的密码匹配，则认证成功，反之则认证失败。使用此函**无需配合 rf_load_key 函数使用.**

参 数:

_Mode: 验证密码类型， 0 -- 用 KEY A 验证; 4 -- 用 KEY B 验证

_BlockNr: 卡片数据**块地址**

_Key: 6 字节密码

返 回: >= 0 成功; < 0 失败.

注: 密码验证成功后，所验证块的扇区内所有块均可操作.

2.2.1.11 读数据

函 数: **short rf_read(unsigned char _Adr, unsigned char *_Data);**

功 能: 读取卡中指定块数据

参 数:

_Adr: 卡片数据块地址

_Data: 读取到的数据。最少分配 16 字节的空间

返 回: >= 0 成功; < 0 失败.

2.2.1.12 写数据

函 数: **short rf_ write(unsigned char _Adr, unsigned char *_Data);**

功 能: 向卡中指定块写入数据

参 数:

_Adr: 卡片数据块地址

_Data: 要写入的数据

返 回: >= 0 成功; < 0 失败.

2.2.1.13 初始化值

函 数: **short rf_initval(unsigned char _Adr, unsigned long _Value);**

功 能: 初始化块值

参 数:

_Adr: 要初始化值的块地址

_Value:初始值

返 回: >= 0 成功; < 0 失败.

注:在进行值操作时，必须先执行初始化值函数，然后才可以读、减、加的操作

2.2.1.14 加值

函 数: **short rf_increment(unsigned char _Adr, unsigned long val);**

功 能: 块加值

参 数:

_Adr: 要加值的块地址

val: 要增加的值

返 回: >= 0 成功; < 0 失败.

2.2.1.15 减值

函 数: **short rf_decrement (unsigned char _Adr, unsigned long val);**

功 能: 块减值

参 数:

_Adr: 要减值的块地址

val: 要减少的值

返 回: >= 0 成功; < 0 失败.

2.2.1.16 读值

函 数：**short rf_readval(unsigned char _Adr, unsigned long *_Value);**

功 能：读取块值

参 数：

_Adr：要读值的块地址

_Value:读取到的值

返 回：>= 0 成功；< 0 失败.

2.2.2 非接触 CPU 卡操作

2.2.2.1 非接触 CPU 卡复位（TypeA）

函 数：**short rf_pro_reset(unsigned char *rbuff);**

功 能：非接触式 CPU 卡复位。在此操作之前需要先执行 rf_request，rf_anticoll 和 rf_select 函数或 rf_card 高级函数

参 数：

rbuff：卡片返回的复位信息

返 回：>= 0 卡片返回的复位信息长度；< 0 失败.

2.2.2.2 非接触 CPU 卡发送指令（TypeA/B）

函 数：**short rf_pro_trn(unsigned int slen,unsigned char *sbuff, unsigned char *rbuff);**

功 能：非接触式 CPU 卡发送 COS 指令，支持 TypeA 及 B 协议卡片

参 数：

slen：指令 sbuff 的长度

sbuff：发送的指令

rbuff：接收到卡片返回的应答信息

返 回：>= 0 卡片返回的应答信息长度；< 0 失败.

2.2.2.3 将卡片置于 halt 状态（TypeA）

函 数：**short rf_pro_halt();**

功 能：该函数用于解除 CPU 卡激活状态，卡片进入 HALT 停止状态。向卡片传输任何指令，卡片都将不做任何响应，必须重新激活卡片

参 数：无

返 回：>= 0 成功；< 0 失败.

2.2.3 ISO14443 TypeB 专用函数

说明： 读写器默认协议为 ISO14443 TypeA 协议,在操作 TypeB 协议卡片时，应先使用 rf_select_protocal()函数选择相应的协议.

2.2.3.1 请求卡片（TypeB）

函 数: **short rf_request_b(unsigned char _Afi, unsigned char _Param, unsigned char *ATQB);**

功 能: 向 TypeB 卡片发出寻卡命令，开始选择一张新卡片时需要执行该函数

参 数:

_Afi: 默认 0

_Param: 默认 0

ATQB: 返回的 4 字节卡号.

返 回: >= 0 成功; < 0 失败.

2.2.3.2 防冲突（TypeB）

函 数: **short rf_slotmarker(unsigned char _N, unsigned char *ATQB);**

功 能: TypeB 卡防冲突

参 数:

_N: 时隙编号，默认 0.

ATQB: rf_request_b()返回的卡号

返 回: >= 0 成功; < 0 失败.

2.2.3.3 非接触 CPU 卡复位（TypeB）

函 数: **short rf_attrib(unsigned char *PUPI, unsigned char CID, unsigned char _BrTx, unsigned char _BrRx);**

功 能: 非接触式 CPU 卡复位，在此之前需要先执行 rf_request_b()函数

参 数:

icdev: hc_init()返回的设备描述符

PUPD: rf_request_b()返回的 4 字节卡号

CID: 取值 0 -- 14，需要卡片支持 CID 才可以使用

_BrTx: 默认 0

_BrRx: 默认 0

返 回: >= 0 成功; < 0 失败.

注: 复位成功后使用 rf_pro_trn()函数进行指令操作

2.2.3.4 将卡片置于 halt 状态（TypeB）

函 数：**short rf_haltb (unsigned char *PUPI);**

功 能：该函数用于解除 CPU 卡激活状态，卡片进入 HALT 停止状态。向卡片传输任何指令，卡片都将不做任何响应，必须重新激活卡片

参 数：

PUPI：rf request b()返回的 4 字节卡号

返 回：>= 0 成功；< 0 失败.

2.3 非接触式低频卡片操作函数(125kHz)

2.3.1 打开 125KHz 射频信号

函 数：**short Open_Mod();**

功 能：打开 125KHz 射频信号，需操作此函数才可以对低频卡进项操作

返 回：>= 0 成功；<0 失败

2.3.2 关闭 125KHz 射频信号

函 数：**short Close_Mod();**

功 能：关闭 125KHz 射频信号，操作此函数后将无法对低频卡片进行操作

返 回：>= 0 成功；<0 失败

2.3.3 设置读写器接收数据频率

函 数：**short E5557_Set_Datarate(short Datarate);**

功 能：设置读写器接收数据频率。

参 数：

Datarate：要设置的频率，取值范围为 0 ~ 7，每个值代表如下：

0 -- 1/8; 1 -- 1/16; 2 -- 1/32; 3 -- 1/40;

4 -- 1/50; 5 -- 1/64; 6 -- 1/100; 7 -- 1/128.

返 回：>= 0 成功；<0 失败

2.3.4 T5557 卡函数

2.3.4.1 写数据

函 数: **short E5557_Write_Free(short Page_Num, short Block_Num, short LockBit, unsigned char *data_buffer);**

功 能: 向射频卡中写入数据（不加密）

参 数:

Page_Num: 写入数据所在的数据页地址。对于 E5557 卡该参数的取值范围应该为 1（厂商代码区）或者为 0（用户数据区）

Block_Num: 写入数据的块号，当 page_num = 1 时取值 1 或 2; 当 page_num = 2 时取值为 0 ~ 7。

LockBit: 是否在写入时将该数据块锁定，如果锁定该块数据将不能再进行修改。该参数为 0 则不锁定，如果该参数为 1 则锁定。

Data_buffer: 要写入射频卡的数据内容（4 字节）

返 回: >= 0 成功; <0 失败

2.3.4.2 写数据

函 数: **short E5557_Write_Pwd(short Page_Num, short Block_Num, short LockBit, unsigned char *Password, unsigned char *data_buffer);**

功 能: 向射频卡中写入数据（加密）

参 数:

Page_Num: 写入数据所在的数据页地址。对于 E5557 卡该参数的取值范围应该为 1（厂商代码区）或者为 0（用户数据区）

Block_Num: 写入数据的块号，当 page_num = 1 时取值 1 或 2; 当 page_num = 2 时取值为 0 ~ 7。

LockBit: 是否在写入时将该数据块锁定，如果锁定该块数据将不能再进行修改。该参数为 0 则不锁定，如果该参数为 1 则锁定。

Password: 卡密码

Data_buffer: 要写入射频卡的数据内容（4 字节）

返 回: >= 0 成功; <0 失败

2.3.4.3 读数据

函 数: **short E5557_Direct_Read(short Page_Num, short Block_Num, short AorBit, unsigned char *Password, unsigned char *data_buffer);**

功 能: 读取 E5557 卡中指定数据页，指定块的数据

参 数:

Page_Num: 读取数据所在的数据页地址。对于 E5557 卡该参数的取值范围应该为 1（厂商代码区）或者为 0（用户数据区）

Block_Num: 读取数据的块号，当 page_num = 1 时取值 1 或 2; 当 page_num = 2 时取值为 0 ~ 7。

AorBit: 是否在读取之前先用密码唤醒卡片。该参数为 0 则不唤醒卡片，如果该参数为 1 则唤醒卡片

Password: 卡密码（如卡片密码无效，则可提供任意值）

Data_buffer: 读取到的数据内容（4 字节）

返 回: >= 0 成功; <0 失败

2.3.4.4 唤醒 AOR 模式加密卡片

函 数: **short E5557_Aor(unsigned char *Password);**

功 能: 使用密码唤醒使用 AOR 模式进行读加密的射频卡

参 数:

Password: 4 个字节的卡密码

返 回: >= 0 成功; <0 失败

2.3.4.5 选择工作数据页

函 数: **short E5557_Select_Page(short Page_Num);**

功 能: 选择卡片当前的工作数据页。选择数据页 0 则卡片循环发送厂商数据页的数据。选择数据页 1 则循环发送用户数据区的数据。发送数据的长度由配置区中的参数设定

参 数:

Page_Num: 读取数据所在的数据页地址。对于 E5557 卡该参数的取值范围应该为 1（厂商代码区）或者为 0（用户数据区）

返 回: >= 0 成功; <0 失败

2.3.4.6 卡片复位

函 数: **short E5557_Reset_Card();**
功 能: 卡片复位，使卡片返回上电时的工作状态。
返 回: >= 0 成功; <0 失败

2.3.5 EM 卡函数

2.3.5.1 读卡

函 数: short Read_EM_Card(unsigned char *data_buffer);
功 能: 读取 EM4001 或兼容卡数据，5 字节卡号
参 数:
data_buffer: 读取到的 5 字节卡号
返 回: >= 0 成功; <0 失败

2.4 接触式卡片操作函数

2.4.1 通用函数

2.4.1.1 对用户卡上电

函 数: **short turn_on();**
功 能: 对用户卡上电（主卡槽）
参 数: 无
返 回: >= 0 成功; < 0 失败.

2.4.1.2 对用户卡下电

函 数: **short turn_off();**
功 能: 对用户卡下电（主卡槽）
参 数: 无
返 回: >= 0 成功; < 0 失败.

2.4.2 接触式 CPU/SAM 卡操作

2.4.2.1 接触 CPU/SAM 卡复位

函 数: **short sam_slt_reset(unsigned char CardType,**
unsigned char *rbuffer);

功 能: 接触式 CPU 卡复位

参 数:

CardType: 卡座编号,取值如下

0 -- CPU 大卡座;

1 -- SAM1;

2 -- SAM2;

3 -- SAM3;

4 -- SAM4

rbuffer: 卡片返回的复位信息

返 回: >= 0 卡片返回的复位信息长度; < 0 失败.

2.4.2.2 接触 CPU/SAM 卡发送指令

函 数: **short sam_slt_protocol(unsigned char CardType,**
unsigned int slen, unsigned char *sbuffer,
unsigned char *rbuffer);

功 能: 向接触式 CPU 卡发送 COS 指令

参 数:

CardType: 卡座编号,同 sam_slt_reset

slen: 发送的指令长度

sbuffer: 发送的指令

rbuffer: 接收到的卡片应答信息

返 回: >= 0 卡片应答信息长度; < 0 失败.

2.4.2.3 接触 CPU/SAM 卡下电

函 数: **short sam_slt_power_down(unsigned char cardType);**

功 能: 接触 cpu/SAM 卡下电,下电后需要重新复位才可操作卡片.

参 数:

cardType: 卡座编号,同 sam_slt_reset

返 回: >= 0 成功; < 0 失败.

2.4.2.4 设置接触 CPU/SAM 卡通信速率

函 数: **short set_card_baud(unsigned char CardType,**
unsigned char BaudCode);

功 能: 设置读写器与卡片的通讯波特率

参 数:

cardType: 卡座编号,同 sam slt reset

BaudCode: 波特率编码。取值如下

01 -- 9600BPS

02 -- 19200BPS

03 -- 38400BPS

04 -- 57600BPS

05 -- 115200BPS

返 回: >= 0 成功; < 0 失败.

注: 此函数是为特殊 CPU 卡使用专有 COS 指令更改通讯速率,而不是通过复位信息 PPS 方式更改通讯速率而预留的。使用时，先发 COS 指令修改卡片的通讯波特率，然后使用此函数设置读写器的通讯波特率.

2.4.3 AT24C 系列卡片操作

2.4.3.1 检测卡片

函 数: **short chk_24c01a();**

功 能: 检查读写设备中卡片是否 24c01a 卡

参 数: 无

返 回: >= 0 成功; < 0 失败.

其它 AT24C 系列卡的测卡函数为:

short chk_24cxx();

其中 xx 为 02、04、08、16、32、64、128、256、512、1024

2.4.3.2 写数据

函 数: **short swr_24c01a(short offset, short len,**
unsigned char *data_buffer);

功 能: 向指定地址写数据

参 数:

offset: 偏移地址，范围 0～127

len: 写入数据的长度，范围 1～128

data_buffer: 写入的数据

返 回: >= 0 成功; < 0 失败.

其它 AT24C 系列卡的写卡函数为:

```
short swr_24cxx(short offset, short len,
                unsigned char *data_buffer);
```

其中 xx 为 02、04、08、16、32、64、128、256、512、1024

offset 和 len 的值跟它的容量相关

2.4.3.3 读数据

函 数: **short srd_24c01a(short offset, short len,**
unsigned char *data_buffer);

功 能: 从指定地址读数据

参 数:

offset: 偏移地址，范围 0～127

len: 读取数据的长度，范围 1～128

data_buffer: 读取到的数据

返 回: >= 0 成功; < 0 失败.

其它 AT24C 系列卡的读卡函数为:

```
short srd_24cxx(short offset, short len,
                unsigned char *data_buffer);
```

其中 xx 为 02、04、08、16、32、64、128、256、512、1024

offset 和 len 的值跟它的容量相关

2.4.4 SLE4442/SLE4432 卡操作

2.4.4.1 检测卡片

函 数: **short chk_4442();**
功 能: 检查读写设备中卡片是否 SLE4442 卡
参 数: 无
返 回: >= 0 成功; < 0 失败.

2.4.4.2 写数据

函 数: **short swr_4442(short offset, short len,**
unsigned char *data_buffer);
功 能: 向指定地址写数据
参 数:
offset: 偏移地址，范围 0～255
len: 写入数据的长度，范围 1～256
data_buffer: 写入的数据
返 回: >= 0 成功; < 0 失败.

2.4.4.3 读数据

函 数: **short srd_4442(short offset, short len,**
unsigned char *data_buffer);
功 能: 从指定地址读数据
参 数:
offset: 偏移地址，范围 0～255
len: 读取数据的长度，范围 1～256
data_buffer: 读取到的数据
返 回: >= 0 成功; < 0 失败.

2.4.4.4 校验密码

函 数: **short csc_4442(unsigned char *data_buffer);**

功 能: 验证卡片密码，密码核对出错 1 次,便减 1,若计数器值为 0,则卡自动锁死,数据只可读出,不可再进行更改也无法再进行密码核对;若不为零时,其中核对正确 1 次,可恢复到初始值 3

参 数:

data_buffer: 3 字节的卡片密码

返 回: >= 0 成功; < 0 失败.

2.4.4.5 更改密码

函 数: **short wsc_4442(unsigned char *data_buffer);**

功 能: 更改卡片密码

参 数:

data_buffer: 3 字节的卡片新密码

返 回: >= 0 成功; < 0 失败.

2.4.4.6 读密码

函 数: **short wsc_4442(unsigned char *data_buffer);**

功 能: 读取卡片密码.卡片密码在校验成功后才可读取.在没有校验成功之前此函数可能依然返回正确,但实际未读取到卡片密码

参 数:

data_buffer: 读取到的 3 字节的卡片密码

返 回: >= 0 成功; < 0 失败.

2.4.4.7 读密码错误计数

函 数: **short rsct_4442(short *counter);**

功 能: 读出密码错误计数器的值.初始值为 3,密码核对出错 1 次,便减 1,若计数器值为 0,则卡自动锁死,数据只可读出,不可再进行更改也无法再进行密码核对;若不为零时, 其中核对正确 1 次,可恢复到初始值 3

参 数:

counter: 读取到的密码错误计数器的值

返 回: >= 0 成功; < 0 失败.

2.4.4.8 读保护位

函 数: **short prd_4442(short len, unsigned char *data_buffer);**
功 能: 读保护位
参 数:
len: 保护长度,其值为 4
data_buffer: 保护位标志
返 回: >= 0 成功; < 0 失败.

2.4.4.9 保护数据

函 数: **short pwr_4442(short offset, short len,**
unsigned char *data_buffer);
功 能: 保护指定地址的数据
参 数:
offset: 偏移地址，范围 0～31
len: 保护数据的长度，范围 1～32
data_buffer: 要保护的数据，必须和卡中已存在的数据一致
返 回: >= 0 成功; < 0 失败.

2.4.5 SLE4428/SLE4418 卡操作

2.4.5.1 检测卡片

函 数: **short chk_4428();**
功 能: 检查读写设备中卡片是否 SLE4428 卡
返 回: >= 0 成功; < 0 失败.

2.4.5.2 写数据

函 数: **short swr_4428(short offset, short len,**
unsigned char *data_buffer);
功 能: 向指定地址写数据
参 数:
offset: 偏移地址，范围 0～1023
len: 写入数据的长度，范围 1～1024
data_buffer: 写入的数据
返 回: >= 0 成功; < 0 失败.

2.4.5.3 读数据

函 数: **short srd_4428(short offset, short len,**
unsigned char *data_buffer);

功 能: 从指定地址读数据

参 数:

offset: 偏移地址，范围 0～1023

len: 读取数据的长度，范围 1～1024

data_buffer: 读取到的数据

返 回: >= 0 成功; < 0 失败.

2.4.5.4 校验密码

函 数: **short csc_4428(unsigned char *data_buffer);**

功 能: 验证卡片密码，密码核对出错 1 次,便减 1,若计数器值为 0,则卡自动锁死,数据只可读出,不可再进行更改也无法再进行密码核对;若不为零时,其中核对正确 1 次,可恢复到初始值 8

参 数:

data_buffer: 2 字节的卡片密码

返 回: >= 0 成功; < 0 失败.

2.4.5.5 更改密码

函 数: **short wsc_4428(unsigned char *data_buffer);**

功 能: 更改卡片密码

参 数:

data_buffer: 要更改的 2 字节卡片新密码

返 回: >= 0 成功; < 0 失败.

2.4.5.6 读密码

函 数: **short wsc_4428(unsigned char *data_buffer);**

功 能: 读取卡片密码.卡片密码在校验成功后才可读取.在没有校验成功之前此函数可能依然返回正确,但实际未读取到卡片密码

参 数:

data_buffer: 读取到的 2 字节卡片密码

返 回: >= 0 成功; < 0 失败.

2.4.5.7 读密码错误计数

函 数: **short rsct_4428(short *counter);**

功 能: 读出密码错误计数器的值.初始值为 8,密码核对出错 1 次,便减 1,若计数器值为 0,则卡自动锁死,数据只可读出,不可再进行更改也无法再进行密码核对;若不为零时, 其中核对正确 1 次,可恢复到初始值 8

参 数:

counter: 读取到的密码错误计数器的值

返 回: >= 0 成功; < 0 失败.

2.4.5.8 保护数据

函 数: **short pwr_4428(short offset,short len,**
unsigned char *data_buffer);

功 能: 保护指定地址的数据

参 数:

offset: 偏移地址, 范围 0～1023

len: 保护数据的长度, 范围 1～1024

data_buffer: 要保护的数据, 必须和卡中已存在的数据一致

返 回: >= 0 成功; < 0 失败.

2.4.5.9 带保护位读数据

函 数: **short srd_4428(short offset, short len,**
unsigned char *data_buffer);

功 能: 从指定地址带保护位读数据

参 数:

offset: 偏移地址, 范围 0～1023

len: 读取数据的长度, 范围 1～1024

data_buffer: 读取到的数据, 其大小应为 2 * len

返 回: >= 0 成功; < 0 失败.

说 明: 从偏移地址 offset 开始带保护位读出

len 个字节数据放入 data_buff 中,每个字节的后面跟一个保护位标志字节,该字节值为 0x00 表示相应的字节已保护,0xff 表示未被保护

例 如: 从 0 字节开始带保护位读出 2 字节长度的数据为

data_buff[0]=0x13,data_buff[1]=0x00,

data_buff[2]=0x20,data_buff[3]=0xff;

表示偏移地址 0 字节被保护, 偏移地址 1 字节未被保护

2.4.5.10 写数据并保护

函 数: **short wrwpb_4428(short offset, short len,**
unsigned char *data_buffer);

功 能: 向指定地址写数据并保护,保护后的数据被固化,无法更改

参 数:

offset: 偏移地址，范围 0～1023

len: 写入数据的长度，范围 1～1024

data_buffer: 存放要写入并加保护的数据

返 回: >= 0 成功; < 0 失败.

2.5 工具函数

2.5.1 将 ASCII 码转换为十六进制数据

函 数: **short asc_hex(char *asc, unsigned char *hex, int pair_len);**

功 能: 将每两个 asc 码字符转换成一个 16 进制数据.如将字符串“a1a2a3a4a5”转换为数组{0xa1,0xa2,0xa3,0xa4,0xa5}

参 数:

asc: 要转换的字符串,必须为 16 进制字符,长度必须为偶数

hex: 存放转换后的数据

pair_len: asc 的实际长度

返 回: >= 0 成功; < 0 失败.

2.5.2 将十六进制数据转换为 ASCII 码

函 数: **short hex_asc(unsigned char *hex, char *asc, int length);**

功 能: 将 16 进制数据转换为对应的 asc 字符串，如将数组{0xa1,0xa2,0xa3,0xa4,0xa5}转换为字符串“a1a2a3a4a5”

参 数:

hex: 要转换的数据

asc: 存放转换后的数据

length: hex 的实际长度

返 回: >= 0 成功; < 0 失败.

注: 转换后的字符串长度为 2 * length.

3 附录

3.1 函数返回值错误类型代码

-0X0001	初始化通信端口失败
-0X0002	关闭通信端口失败
-0X0003	串口被占用
-0X0004	不支持的串口波特率
-0X0005	未扫描到设备
-0X0006	设备未连接
-0X0007	无效的句柄
-0X0008	通讯发生错误
-0X0009	通讯数据格式错误
-0X000A	参数错误
-0X000B	数据长度错误
-0X000C	地址越界
-0X000D	缓冲区溢出
-0X000E	操作超时
-0X000F	写(卡片)错误
-0X0010	读(卡片)错误
-0X0011	BCC 校验和错误
-0X0012	不支持的卡座编号
-0X0013	卡片类型错误
-0X0014	卡座有卡未上电
-0X0015	未检测到卡片
-0X0016	非法卡
-0X0017	装载密码出错
-0X0018	获取读卡器密码错误
-0X0019	读卡器未知错误
-0X001A	请求失败
-0X001B	防冲突失败
-0X001C	选卡失败
-0X001D	认证错误
-0X001E	非值存储区域

-0x001F	值溢出错误
-0X0020	restore 出错
-0X0021	transfer 出错
-0X0022	增值错误
-0X0023	减值错误
-0X0024	Mifare 值操作失败
-0X0025	磁条卡数据错误
-0X0026	CPU 卡复位出错
-0X0027	CPU 卡 APDU 失败
-0X0028	命令不支持
-0x0029	系统操作出错
-0X002A	密码错误