# 1.PED

## API索引

|  |  |
| --- | --- |
| * [PdkPed\_GetVer ()](#_获取密码键盘类模块版本号) * [PdkPed\_WriteMkey ()](#_主密钥明文写入) * [PdkPed\_WriteWkey ()](#_工作密钥写入) | * [PdkPed\_GetPin ()](#_PIN输入) * [PdkPed\_GetMac ()](#_MAC计算) * [PdkPed\_Des ()](#_DES加/解密) * [PdkPed\_EraseAllKey ()](#_擦除所有密钥)\* |

## 结构体等声明

PinPad类型枚举:

typedef enum

{

UPP\_INT\_PED =0, /\* 内置密码键盘PED \*/

UPP\_EXT\_SP20, /\* 外置密码键盘SP20\*/

UPP\_EXT\_S200, /\* 外置密码键盘S200\*/

}PinPadType;

加密模式:

typedef enum

{

UDES\_TYPE\_SINGLE =1, /\* 单DES \*/

UDES\_TYPE\_TRIPLE, /\* 3DES \*/

}DesType;

密钥类型:

typedef enum

{

UKEY\_TYPE\_MST =0, /\* 主密钥 \*/

UKEY\_TYPE\_PIN, /\* PIN工作密钥 \*/

UKEY\_TYPE\_MAC, /\* MAC工作密钥 \*/

UKEY\_TYPE\_DES , /\* DES工作密钥\*/}KeyType;

PP返回值定义:

#define UPP\_COMM\_FAIL (-100)

typedef enum

{

UPP\_PIN\_NONE = UPP\_COMM\_FAIL-33, //没有输入密码(密码键盘按确认键)

UPP\_PIN\_TIMEOUT = UPP\_COMM\_FAIL-32, //密码输入超时

UPP\_PIN\_ESC = UPP\_COMM\_FAIL-31, // 退出输入密码

UPP\_PIN\_ERROR = UPP\_COMM\_FAIL-30, // 密码输入错误

UPP\_API\_PARA = UPP\_COMM\_FAIL-12, //调用API传参错误

UPP\_PINPAD = UPP\_COMM\_FAIL-11, //密码键盘类型错误

UPP\_KEY\_NOEXIST= UPP\_COMM\_FAIL-10, //密钥不存在

UPP\_KEY\_EXIST = UPP\_COMM\_FAIL-9, //密钥组已存在

UPP\_DECLINED = UPP\_COMM\_FAIL-8, //操作被拒绝

UPP\_ERROR = UPP\_COMM\_FAIL-7, // 错误

UPP\_FAILURE = UPP\_COMM\_FAIL-6, // 系统级别的失败

UPP\_ESCAPE = UPP\_COMM\_FAIL-5, // 操作被退出

UPP\_TIMEOUT = UPP\_COMM\_FAIL-4, // 密码键盘通讯超时(外置键盘通讯)

UPP\_MAC\_ERROR = UPP\_COMM\_FAIL-3, // 计算MAC出错

UPP\_ERR\_PAN = UPP\_COMM\_FAIL-2, // 主账号有误

UPP\_ABORT = UPP\_COMM\_FAIL-1, // 用户放弃

UPP\_SUCCESS = 0, // 成功

}PPStatus;

## API函名称

### 获取密码键盘模块版本号

|  |  |  |
| --- | --- | --- |
| 原型 | **int PdkPed\_GetVer(uchar \*pszVer);** | |
| 功能 | 获取密码键盘类模块版本号 | |
| 参数 | pszVer  **【输出】** | 模块版本号存储缓冲区地址,缓冲区大小至少应为26bytes |
| 返回 | 见PPStatus | |
| 用法 |  | |

### 密码键盘初始化

|  |  |  |
| --- | --- | --- |
| 原型 | **int PdkPed\_Init (uchar ucType);** | |
| 功能 | 初始化密码键盘：连接密码键盘、个性化参数设置等 | |
| 参数 | ucType**【输入】** | 密码键盘类型,参见enum PinPadType |
| 返回 | 见PPStatus | |
| 用法 |  | |

### 主密钥写入

|  |  |  |
| --- | --- | --- |
| 原型 | **int PdkPed\_WriteMKey (uchar ucKeyIdx,**  **const uchar \*psKeyData,**  **int iKeyDataLen);** | |
| 功能 | 写入终端主密钥 | |
| 参数 | **ucKeyIdx**  **【输入】** | 密钥组索引 |
| **psKeyData**  **【输入】** | 主密钥明文 |
| **iKeyDataLen**  **【输入】** | 主密钥明文长度 |
| 返回 | 见PPStatus |  |
| 用法 | KeyDataLen用来决定密钥的加密模式，必须是8的整数倍，否则出错  KeyDataLen = 8：单DES密钥  KeyDataLen = 16：3DES密钥 | |
| 具体实现 |  | |

### 工作密钥写入

|  |  |  |
| --- | --- | --- |
| 原型 | int PaxPp\_WriteWKey (ushort MKeyIndex, ushort WKeyIndex,byte \*KeyData,int KeyDataLen, int iDesType，uchar KeyType); | |
| 功能 | 下载工作密钥 | |
| 参数 | MKeyIndex**【输入】** | 主密钥组索引（1-100） |
| WKeyIndex**【输入】** | 工作密钥索引（1-100） |
| KeyData**【输入】** | 工作密钥密文 |
| KeyDataLen**【输入】** | 工作密钥密文长度 |
| iDesType**【输入】** | 加密算法,参见enum DesType |
| KeyType**【输入】** | 工作密钥类型，参见enum KeyType |
| 返回 | 见PPStatus |  |
| 用法 | KeyDataLen用来决定密钥的加密模式，必须是8的整数倍，否则出错  KeyDataLen = 8：单DES工作密钥  KeyDataLen = 16：3DES工作密钥  KeyDataLen = 24：7DES工作密钥 | |
| 具体实现 |  | |

### PIN输入

|  |  |  |
| --- | --- | --- |
| 原型 | **int PdkPed\_GetPin(**uchar \*PAN, uchar \*PinData, uchar MinPINLen, uchar MaxPINLen, ushort TimeOut, uchar KeyIndex, ushort iDesType**);** | |
| 功能 | Pin的输入 | |
| 参数 | **psPAN【输入】** | 指向卡号移位后生成的16位主帐号/或全’0’ |
| **psPinData【输出】** | 8bytes 指向生成的PINBlock |
| MinPINLen**【输入】** | 密码最小长度0 |
| MaxPINLen**【输入】** | 密码最大长度12 |
| TimeOut**【输入】** | 输入超时时间 单位秒。 |
| **ucKeyIdx【输入】** | PIN工作密钥组索引（1-100） |
| **ucDesType【输入】** | 加密算法,参见enum DesType |
| 返回 | 见PPStatus |  |
| 用法 |  | |
| 具体实现 |  | |

### MAC计算

|  |  |  |
| --- | --- | --- |
| 原型 | **int PdkPed\_GetMac(uchar ucCalMode,**  **uchar \*psMAC,**  **const uchar \*psData,**  **int iDataLen,**  **uchar ucKeyIdx,**  **uchar ucDesType,**  **const uchar \* psInitVct);** | |
| 功能 | 计算MAC | |
| 参数 | **ucCalMode**  **【输入】** | MAC算法类型 0-算法0 1-算法1 2-算法2 3-算法3（具体参见补充说明） |
| **psMAC**  **【输出】** | 输出的8字节MAC值 |
| **psData**  **【输入】** | 参与计算MAC的数据 |
| **iDataLen**  **【输入】** | 参与计算MAC的数据长度 |
| **ucKeyIdx**  **【输入】** | MAC工作密钥组索引 |
| **ucDesType**  **【输入】** | 加密算法,参见enum DesType |
| **psInitVct**  **【输入】** | 初始向量，用于和BLOCK1异或，生成新的BLOCK1 |
| 返回 | 见PPStatus |  |
| 用法 | * Data为原始数据，不用补足8的倍数，由API自行补0 * CalMode:   算法0：将BLOCK1用MAC密钥做DES/TDES加密,加密结果与BLOCK2进行逐位异或后再用TAK做DES/TDES加密,依次进行得到8字节的加密结果  算法1：将BLOCK1和BLOCK2进行逐位异或,异或结果与BLOCK3进行逐位异或,依次进行,最后得到8字节的异或结果,将该结果用TAK进行DES/TDES加密运算  算法2：ANSIX9.19规范,将BLOCK1用TAK做DES加  密(只取前8个字节的key),加密结果与BLOCK2进行逐  位异或后再用TAK做DES加密,依次进行得到8字节的  加密结果,直到最后一次采用DES/TDES加密  算法3：(银联POS终端MAC算法)将BLOCK1和BLOCK2进行逐位异或,异或结果与BLOCK3进行逐位异或,依次进行,最后得到8字节的异或结果。将异或运算后的最后8个字节（BCD to ASC）转换成16 个HEXDECIMAL, 取前8 个字节用TAK进行DES/TDES加密运算, 将加密后的结果与后8 个字节异或; 对异或的结果用TAK再进行DES/TDES加密运算，将运算后的结果作为最终的MAC值(标准的银联算法是需要进行BCD转ASC，然后取前8个字符，这一步留给应用来完成)  DataLen: 算法3时最大长度为10240，其余最大长度为1024 | |
| 具体实现 |  | |

### DES加/解密

|  |  |  |
| --- | --- | --- |
| 原型 | **int PdkPed\_Des(const uchar \*psInData,**  **uchar \*psOutData,**  **uchar ucKeyIdx,**  **uchar ucDesType,**  **uchar ucEncrypt);** | |
| 功能 | 用指定密钥加密或解密 | |
| 参数 | **psInData**  **【输入】** | 输入数据(8Bytes) |
| **psOutData**  **【输出】** | 输出数据(8Bytes) |
| **ucKeyIdx**  **【输入】** | DES工作密钥组索引 |
| **ucDesType**  **【输入】** | 加密算法,参见enum DesType |
| **ucEncrypt【输入】** | 1-加密 0-解密 |
| 返回 | 见PPStatus |  |
| 用法 |  | |
| 具体实现 |  | |

### DES明文密钥下装

|  |  |  |
| --- | --- | --- |
| 原型 | int PaxPp\_WritePlainDesKey (ushort KeyIndex,byte \*Key, int KeyDataLen); | |
| 功能 | 明文Key下装 | |
| 参数 | KeyIndex**【输入】** | 写入密码键盘中的DES密钥索引（1-100） |
| Key**【输入】** | 密钥明文 |
| KeyDataLen**【输入】** | 密钥长度 |
| 返回 | 见PPStatus |  |
| 用法 |  | |
| 具体实现 |  | |

# 2.CARD

## 返回值列表

#deifne CARD\_RET\_OK 0x00 //操作成功

#define CARD\_ERR\_BASE (0-200) //错误返回基数

#define CARD\_ERR\_INVAL\_PARAM (CARD\_ERR\_BASE-1) //参数错误

#define CARD\_ERR\_PROTOCOL (CARD\_ERR\_BASE-2) //协议错误

#define CARD\_ERR\_NO\_CARD (CARD\_ERR\_BASE-3) //没刷卡或者没有插入IC或者未搜寻到卡片(感应区内无指定类型的卡片)

#define CARD\_ICC\_ERR\_PULL\_OUT (CARD\_ERR\_BASE-4) //卡拔出(对于SAM卡座,如无SAM卡或未插好,则会返回0X33错误代码)

#define CARD\_ICC\_ERR\_CHANNEL (CARD\_ERR\_BASE-5) //通道号错误

#define CARD\_ICC\_ERR\_COMM\_TIMEOUT (CARD\_ERR\_BASE-6) //通信超时

#define CARD\_ICC\_ERR\_ODD\_EVEN (CARD\_ERR\_BASE-7) //奇偶错误

#define CARD\_ICC\_ERR\_SELECT\_ CHANNEL (CARD\_ERR\_BASE-8) //选择通道错误

#define CARD\_ICC\_ERR\_DEVICE\_BUSY (CARD\_ERR\_BASE-9) //设备占用

#define CARD\_ICC\_ERR\_NO\_RESET (CARD\_ERR\_BASE-10) //没有复位

#define CARD\_ICC\_ERR\_SEND\_DATA\_TOO\_LONG (CARD\_ERR\_BASE-11) //没有复位

#defien CARD\_PICC\_ERR\_NOT\_OPEN (CARD\_ERR\_BASE-12) //射频模块未开启

#define CARD\_PICC\_ERR\_CARD\_TOO\_MUCH (CARD\_ERR\_BASE-13) //感应区内卡片过多(出现通讯冲突)

#define CARD\_PICC\_ERR\_RESPOND\_IN\_PROTOCOL (CARD\_ERR\_BASE-14) //协议错误(卡片应答中出现违反协议规定的数据)

#define CARD\_PICC\_ERR\_NOT\_ACTIVATION (CARD\_ERR\_BASE-15) //卡片未激活

#define CARD\_PICC\_ERR\_CONFLICT (CARD\_ERR\_BASE-16) //多卡冲突

#define CARD\_PICC\_ERR\_COMM\_TRANS (CARD\_ERR\_BASE-17) //通信传输错误

#define CARD\_PICC\_ERR\_M1\_AUTH\_FAILE (CARD\_ERR\_BASE-18) //M1卡认证失败或block错误

#define CARD\_PICC\_ERR\_SECTION\_NOT\_AUTH (CARD\_ERR\_BASE-19) //扇区未认证

#define CARD\_PICC\_ERR\_DATA\_FORMAT (CARD\_ERR\_BASE-20) //数值块数据格式有误

#define CARD\_PICC\_ERR\_ CHIPS \_EXIST (CARD\_ERR\_BASE-21) //卡片仍在感应区内

#defien CARD\_PICC\_ERR\_STATUS (CARD\_ERR\_BASE-22) //卡片状态错误(如A/B卡调用M1卡接口, 或M1卡调用PaxPicc\_IsoCommand接口)

#define CARD\_PICC\_ERR\_CHIPS\_NOT\_EXIST (CARD\_ERR\_BASE-23) //接口芯片不存在或异常

typedef struct{

uchar szTrack1[256]; /\* 一磁道\*/

uchar szTrack2[256]; /\* 二磁道\*/

uchar szTrack3[256]; /\* 三磁道\*/

uchar szSerial[33]; /\* IC卡序列号\*/

} **STCARD\_INFO**；

## ReadCard

|  |  |  |
| --- | --- | --- |
| 原 型 | **int PdkCard\_Read(uchar EnterMode, uchar\* ActuralEnter,**  **STCARD\_INFO \* CardInfo, int iTimeout)** | |
| 功 能 | 卡片读入（支持刷卡/插卡/挥卡） | |
| 参数 | **EnterMode**【输入】 | |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **B7** | **B6** | **B5** | **B4** | **B3** | **B2** | **B1** | **B0** | **备注** | |  |  |  |  |  |  |  | x | 是否启用刷卡 | |  |  |  |  |  |  | x |  | 是否启用插卡 | |  |  |  |  |  | x |  |  | 是否启用非接挥卡 | |  |  |  |  | x |  |  |  | 是否启用手动输入(保留) | |  |  |  | x |  |  |  |  | 是否FALLBACK(保留) | |  |  | x |  |  |  |  |  | 是否为转入卡(保留) | |  | x |  |  |  |  |  |  | 保留 | | x |  |  |  |  |  |  |  | 保留 | |
|  | **ActuralEnter**【输出】 | |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **B7** | **B6** | **B5** | **B4** | **B3** | **B2** | **B1** | **B0** | **备注** | |  |  |  |  |  |  |  | x | 刷卡 | |  |  |  |  |  |  | x |  | 插卡 | |  |  |  |  |  | x |  |  | 非接挥卡 | |  |  |  |  | x |  |  |  | 保留 | |  |  |  | x |  |  |  |  | 保留 | |  |  | x |  |  |  |  |  | 保留 | |  | x |  |  |  |  |  |  | 保留 | | x |  |  |  |  |  |  |  | 保留 | |
| **CardInfo**【输出】 | 获取到的卡片信息（磁道信息等） |
| **iTimeout**【输入】 | 卡片读入超时时间（秒） |
| 返回 | 0 | 成功 |
| 其它 | 读卡失败  超时  用户取消 |
| 用 法 |  | |

# 3.PRINTER

## 错误码及宏定义

打印返回码定义

#define PRNT\_FAIL\_BASE (-300)

typedef enum

{

PRNT\_DEV\_TYPE\_ERR = PRNT\_FAIL\_BASE-18, //打印机设备类型错

PRNT\_DEV\_NONE = PRNT\_FAIL\_BASE-17, //无打印机

PRNT\_OVERHEAT = PRNT\_FAIL\_BASE-16, //打印机过热

PRNT\_RECV\_TIMEOUT = PRNT\_FAIL\_BASE-15, //接收超时

PRNT\_CMD\_TIMEOUT = PRNT\_FAIL\_BASE-14, //处理命令超时

PRNT\_DEV\_UNFINISHED = PRNT\_FAIL\_BASE-13, //通道占用

PRNT\_LOSE\_FONT = PRNT\_FAIL\_BASE-12, //缺少字库

PRNT\_DATA\_ERR = PRNT\_FAIL\_BASE-11, //打印数据错

PRNT\_LOSE\_DRIVER = PRNT\_FAIL\_BASE-10, //缺少驱动

PRNT\_USER\_CANCEL = PRNT\_FAIL\_BASE-9, //用户取消

PRNT\_BUSY = PRNT\_FAIL\_BASE-8, //打印机忙

PRNT\_CACHE\_ERR = PRNT\_FAIL\_BASE-7, //存缓存失败

PRNT\_HANDLE\_ERR = PRNT\_FAIL\_BASE-6, //分体机手柄未放回

PRNT\_FILE\_FAIL = PRNT\_FAIL\_BASE-5, //打开文件失败

PRNT\_LOSE\_COMMAND = PRN\_FAIL\_BASE-4, //未获得打印句柄

PRNT\_OUTOF\_PAPER = PRN\_FAIL\_BASE-3, //打印机缺纸

PRNT\_DEV\_FAIL = PRN\_FAIL\_BASE-2, //打印机设备故障

PRNT\_FAIL = PRN\_FAIL\_BASE-1, //打印机未知故障

PRNT\_SUCCESS = 0, //成功

} PrintRet;

/\*打印机类型定义\*/

#define PRT\_TYPE\_ERR (-1)

#define PRT\_TYPE\_S (1)

#define PRT\_TYPE\_T (2)

/\*格式化打印参数定义\*/

#define PRT\_STYLE\_NORMAL 0x00 //默认(左靠)

#define PRT\_STYLE\_LEFT 0x01 //左靠打印

#define PRT\_STYLE\_RIGHT 0x02 //右靠打印

#define PRT\_STYLE\_CENTER 0x04 //居中打印

## 初始化打印模块

|  |  |  |
| --- | --- | --- |
| 函数原型 | int PaxPrnt\_Init(void) | |
| 函数功能 | 初始化打印模块. | |
| 参数说明 | 入参 | 无 |
| 出参 | 无 |
| 返回值 | =0：成功  <0：失败  (详键打印类错误返回码定义) | |
| 补充说明 | 1. 每次打印前调用PaxUprint\_Init初始化  2. 函数调用GetTermInfo读取打印机类型,分体式POS手机检测提示放回座机.  3. 函数调用PaxPrnt\_DetectPrinterType检测打印机类型(热敏/针打)  4. 调用PrnInit初始化打印机 | |

## 启动打印机打印数据

|  |  |  |
| --- | --- | --- |
| 函数原型 | int PaxPrnt\_StartPrint(unsigned char ucEnableExit) | |
| 函数功能 | 启动打印机,将缓冲区里的数据打印出来. | |
| 参数说明 | 入参 | ucEnableExit打印中是否允许按取消键退出(包括缺纸出错时)  0- 不允许  1- 允许 |
| 出参 |  |
| 返回值 | 0：成功  <0：忙、缺纸、故障 | |
| 补充说明 | 分体式机型的座机检测.  PrnStart函数调用  缺纸检测与提示. | |

## 设置打印字体

|  |  |  |
| --- | --- | --- |
| 函数原型 | void PaxPrn\_FmtSetFont(unsigned char AscFont,unsigned char HZFont) | |
| 函数功能 | 设置打印字体 | |
| 参数说明 | 入参 | AscFont – ASCII打印字体  HZFont -汉字打印字体 |
| 出参 | 无 |
| 返回值 | 无 | |
| 补充说明 | 1. 使用PaxPrnt\_LineStr / PaxPrnt\_LineStrStr/ PaxPrn\_PrnStr  格式化打印前需调用本函数设置字体  2. 从未设定默认调用ASC-1号字体,汉字-1号字体.  3. 本次格式化打印前不设置字体,沿用上次设置的字体. | |
| 调用举例 | PaxPrn\_FmtSetFont (2,2);  PaxPrnt\_LineStrStr("居中字体",PRNT\_STYLE\_CENTER); | |

## 设置打印间距

|  |  |  |
| --- | --- | --- |
| 函数原型 | void PaxPrnt\_SetSpace(unsigned char ucCharSpace,unsigned char ucLineSpase) | |
| 函数功能 | 设置打印间距 | |
| 参数说明 | 入参 | ucCharSpace – 字间距  ucLineSpase – 行间距 |
| 出参 | 无 |
| 返回值 | 无 | |
| 补充说明 | 1. 调用PaxPrnt\_LineStrStr格式化打印前需先设置行间距,字间距.  2. 默认字间距0.  3. 默认行间距0,  4. 不设置间距,沿用上次设置 | |
| 调用举例 | PaxPrnt\_SetSpace(0,2); | |

## 格式化打印行(字符串1)

|  |  |  |
| --- | --- | --- |
| 函数原型 | void PaxPrnt\_LineStr (unsigned char \*pPrnMsg, unsigned char ucPrintStyle) | |
| 函数功能 | 格式化打印一行(一个ASCII字符串) | |
| 参数说明 | 入参 | unsigned char \*pPrnMsg–打印字符串的指针  unsigned char ucPrintStyle –打印格式  1. PRNT\_STYLE\_NORMAL --默认(左靠)  2. PRNT\_STYLE\_LEFT --左靠打印  3. PRNT\_STYLE\_RIGHT --右靠打印  4. PRNT\_STYLE\_CENTER --居中打印 |
| 出参 | 无 |
| 返回值 | 无 | |
| 补充说明 | 1. 调用函数需PaxPrnt\_Init初始化  2. 调用函数前调用PaxPrn\_FmtSetFont选择打印字体.  3. 函数重置字间距为0.  4. 基于标准字库,热敏0~7号字体. | |
| 调用举例 | PaxPrn\_FmtSetFont (2,2);  PaxPrnt\_LineStr ("居中字体",PRNT\_STYLE\_CENTER);  PaxPrn\_FmtSetFont (0,0);  PaxPrnt\_LineStr ("左靠打印",PRNT\_STYLE\_LEFT);  PaxPrn\_FmtSetFont (3,3);  PaxPrnt\_LineStr ("右靠打印",PRNT\_STYLE\_RIGHT); | |

## 格式化打印行(字符串2)

|  |  |  |
| --- | --- | --- |
| 函数原型 | void PaxPrnt\_LineStrStr(unsigned char \*pPrnMsg1, unsigned char \*pPrnMsg2,unsigned char ucPrintStyle,unsigned char ucPrintStyle2) | |
| 函数功能 | 格式化打印一行(2个ASCII字符串) | |
| 参数说明 | 入参 | unsigned char \*pPrnMsg1–打印字符串1的指针  unsigned char \*pPrnMsg2–打印字符串2的指针  unsigned char ucPrintStyle1 –字符串1格式  unsigned char ucPrintStyle2 –字符串1格式  1. PRNT\_STYLE\_NORMAL --默认(左靠)  2. PRNT\_STYLE\_LEFT --左靠打印  3. PRNT\_STYLE\_RIGHT --右靠打印  4. PRNT\_STYLE\_CENTER --居中打印  目前支持组合:  1. (靠左 – - 靠右)  2. (靠左 – 居中 - )  3. (靠左 – 靠左 - ) |
| 出参 | 无 |
| 返回值 | 无 | |
| 补充说明 | 1. 调用函数需PaxPrnt\_Init预初始化  2. 调用函数前调用PaxPrn\_FmtSetFont选择打印字体.  3. 函数重置字间距为0.  4. 基于标准字库,热敏0~7号字体. | |
| 调用举例 | PaxPrn\_FmtSetFon (2,2);  PaxPrnt\_LineStrStr ("左靠字符1","右靠字符2",PRNT\_STYLE\_LEFT,PRNT\_STYLE\_RIGHT); | |

## 打印字符串

|  |  |  |
| --- | --- | --- |
| 函数原型 | void PaxPrnt\_PrnStr(unsigned char ucPrintStyle,const char \* pszFormat, ...) | |
| 函数功能 | 打印字符串(带格式) | |
| 参数说明 | 入参 | ucPrintStyle –打印格式  1. PRNT\_STYLE\_NORMAL --默认(左靠)  2. PRNT\_STYLE\_LEFT --左靠打印  3. PRNT\_STYLE\_RIGHT --右靠打印  4. PRNT\_STYLE\_CENTER --居中打印  pszFormat – 打印字符串的指针 |
| 出参 | 无 |
| 返回值 | 无 | |
| 补充说明 | 1. 支持变参,无自动换行. 2. 支持打印串中出现‘\n’[换行]、‘\f’[换页]控制字符 3. 如果打印数据包过长,则程序将溢出; 4. 缓冲区最大为4096字节。 5. 将str存入打印缓冲区,在调用PrnStart()后,按照写入缓冲区的顺序,开始打印缓冲区里的数据。 | |
| 调用举例 | PaxPrnt\_PrnStr(PRNT\_STYLE\_CENTER, "测试d%d\n",9999); | |
|  |  | |

## 打印字符串2

|  |  |  |
| --- | --- | --- |
| 函数原型 | void PaxPrnt\_PrnStrStr(unsigned char ucPrintStyle2,unsigned char \*pPrnMsg1,const char \* pszFormat, ...) | |
| 函数功能 | 打印字符串(带格式) | |
| 参数说明 | 入参 | ucPrintStyle2 –打印字符串2格式  1. PRNT\_STYLE\_NORMAL --默认(左靠)  2. PRNT\_STYLE\_LEFT --左靠打印  3. PRNT\_STYLE\_RIGHT --右靠打印  4. PRNT\_STYLE\_CENTER --居中打印  pPrnMsg1 – 字符串1的指针  pszFormat – 字符串2的指针 |
| 出参 | 无 |
| 返回值 | 无 | |
| 补充说明 | 1. 字符串2支持变参,无自动换行 2. 支持打印串中出现‘\n’[换行]、‘\f’[换页]控制字符 3. 如果打印数据包过长,则程序将溢出; 4. 字符串2缓冲区最大为4096字节。字符串1最大100字节,多余会被截掉. 5. 将str存入打印缓冲区,在调用PrnStart()后,按照写入缓冲区的顺序,开始打印缓冲区里的数据。 | |
| 调用举例 | PaxPrnt\_PrnStrStr(PRT\_STYLE\_LEFT, "商户号:","%s\n","112233445566778899"); | |
|  |  | |

## 选定字符填充满一行

|  |  |  |
| --- | --- | --- |
| 函数原型 | void PaxPrnt\_DrawLineByChar (unsigned char InAscChar,unsigned char ucAscFont) | |
| 函数功能 | 选定字符填充满一行 | |
| 参数说明 | 入参 | InAscChar–需打印的字符  ucAscFont – ASCII字体 |
| 出参 | 无 |
| 返回值 | 无 | |
| 补充说明 | 1. 基于标准字库,热敏0~7号字体,针式0~4号字体.  2. 调用函数需PaxPrnt\_Init初始化  3. 非标准字库无法对齐  4．支持字间距设置,支持字体设置 | |

# 4．SYS

## 错误码及宏定义

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* macros declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//获取终端信息的宏定义

#define TERM\_MODEL "MODEL"

#define TERM\_PRINTER "PRINTER"

#define TERM\_MODEM "MODEM"

//#define TERM\_PCI "PCI"

#define TERM\_USB "USB"

#define TERM\_LAN "LAN"

#define TERM\_GPRS "GPRS"

#define TERM\_CDMA "CDMA"

#define TERM\_WIFI "WIFI"

#define TERM\_RF "RF"

#define TERM\_IC "IC"

#define TERM\_MAG "MAG"

#define TERM\_TILT "TILT"

#define TERM\_WCDMA "WCDMA"

#define TERM\_TOUCH\_SCR "TOUCHSCR"

//获取应用信息的宏定义

#define APP\_APP\_INFO "AppInfo"

#define APP\_SO\_INFO "SoInfo"

#define APP\_IP\_INFO "IPInfo"

#define APP\_WL\_SIGNAL "WLSignal"

#define APP\_BATTERY\_INFO "BatteryInfo"

#define APP\_DHCP\_FLAG "DHCPFlag"

#define APP\_RF\_PARAM "RFParam"

/\*\* 系统错误定义 \*/

#define SYS\_RET\_OK 0

#define SYS\_ERR\_BASE (0-400)

#define SYS\_ERR\_FAIL (SYS\_ERR\_BASE-1) //失败

#define SYS\_ERR\_PARAERR (SYS\_ERR\_BASE-2) //参数错误

#define SYS\_ERR\_NO\_FIRST (SYS\_ERR\_BASE-3) //应用不是第一次远行

//#define PARAM\_ERR\_READ\_FILE (SYS\_ERR\_BASE-3)

//#define PARAM\_ERR\_WRITE\_FILE (SYS\_ERR\_BASE-4)

//#define PARAM\_ERR\_NO\_KEY (SYS\_ERR\_BASE-5)

//#define PARAM\_ERR\_NO\_INDEX (SYS\_ERR\_BASE-6)

//#define PARAM\_ERR\_SYS (SYS\_ERR\_BASE-7)

//#define PARAM\_ERR\_FILE\_NOEXIST (SYS\_ERR\_BASE-8)

//#define PARAM\_ERR\_FORMAT (SYS\_ERR\_BASE-9)

## 读取终端序列号

|  |  |  |
| --- | --- | --- |
| 原 型 | **int PdkSys\_ReadSN(uchar \*pszSN)** | |
| 功 能 | 获取终端序列号 | |
| 参数 | **pszSN**【输出】 | 终端SN号 |
| 返回 | 0 | 成功 |
| 其它 | 见[返回码列表](#_返回码列表_1) |
| 用 法 |  | |

## 获取终端信息

|  |  |  |
| --- | --- | --- |
| 原 型 | **int PdkSys\_GetTermInfo(char \*pszKey,**  **char \*pszValue )** | |
| 功 能 | 获取终端型号与配置信息 | |
| 参 数 | **pszKey**【输入】 | pszKey：模块名称，现有如下模块：  MODEL：终端型号,如“S500”。  PRINTER:‘0’-无,‘S’-针打,‘T’-热敏  MODEM:‘0’-无, ‘1’-有  USB:‘0’-无, ‘1’-有  LAN:‘0’-无, ‘1’-有  GPRS:‘0’-无, ‘1’-有  CDMA:‘0’-无，‘1’-有  WIFI：‘0’-无，‘1’-有  RF：非接模块，‘0’-无，‘1’-有  IC：‘0’-无，‘1’-有  MAG：磁卡模块,‘0’-无，‘1’-有  TILT：重力感应器，‘0’-无，‘1’-有  WCDMA：‘0’-无，‘1’-有  TOUCHSCR：触屏‘0’-无，‘1’-有 |
|  | **pszValue**【输出】 | 信息缓冲区 |
| 返 回 | 0 |  |
| 其它 | 见[返回码列表](#_返回码列表_1) |
| 用 法 | 入参为某个模块的名称，出参即返回这个模块的配置信息。 | |

# 5. EMV PROCESSOR

## 错误码及宏定义

#define EMV\_RET\_OK 0

#define EMV\_ERR\_BASE (0-500)

#define EMV\_CNTCT\_ISSU (EMV\_ERR\_BASE-1) //联系发卡行

#define EMV\_TRANS\_FAIL (EMV\_ERR\_BASE-2) //交易失败

#define EMV\_TRANS\_REF (EMV\_ERR\_BASE-3) //交易拒绝

//#define EMV\_TRANS\_ESC (EMV\_ERR\_BASE-4) //交易取消

typedef struct{

ulong ulPosSer; /\* POS流水号\*/

uchar szAmt[13]; /\* 交易金额\*/

uchar szDate[9]; /\* 交易日期\*/

uchar szTime[7]; /\* 交易时间\*/

} **STTRANS\_PARA**；

## EMV流程封装成一个API（含非接完整流程）+多个UI回调

|  |  |  |
| --- | --- | --- |
| 原 型 | **int PdkEMV\_Process (uchar ucProcType,**  **STTRANS\_PARA \* pstTransPara**  **uchar\* OnlTagList, uchar\* PrtTagList, uchar\* PrtTlvBack,**  **uchar\* ucTransResult)** | |
| 功 能 | EMV完整流程处理全过程 | |
| 参 数 | **ucProcType**【输入】 | EMV流程类型 0-完整流程 1-简易流程 2-小额支付 |
| **ucTransPara**【输入】 | 发起交易流程需要的参数（如：流水号、日期时间等） |
| **OnlTagList**【输入】 | 组联机交易包55域需要的tag列表  例：tag1-9f34,tag2-4f: “\x00\x03\x9f\x34\4f”（头两字节Hex长度） |
| **PrtTagList**【输入】 | 打印需要要素tag列表  例：tag1-9f34,tag2-4f: “\x00\x03\x9f\x34\4f” （头两字节Hex长度） |
| **PrtTlvBack**【输出】 | 实际打印的emv要素tlv  例：tlv1-95 05 0000080000,tlv2-5f2a 02 0156: “\x00\x0c\x95\x05\00\x00\x08\x00\x00\x5f\x2a\x02\x01\x56” （头两字节Hex长度） |
| **ucTransResult**【输出】 | POS\_ONLINE\_APPROVE 1  POS\_ONLINE\_DENIAL 2  POS\_OFFLINE\_APPROVE 3  POS\_OFFLINE\_DENIAL 4 |
| 返 回 | 0 | 成功 |
| 其它 | CNTCT\_ISSU 联系发卡行  TRANS\_FAIL 交易失败  TRANS\_REF 交易拒绝  TRANS\_ESC 交易取消 |
| 用 法 |  | |

|  |  |  |
| --- | --- | --- |
| 原 型 | **int PdkEmv\_cGetAmt(uchar \*Amt, uchar\* CashBackAmt)** | |
| 功 能 | (UI回调) 输入交易金额 | |
| 参数 | Amt【输出】 | APP输入的交易金额 |
| **CashBackAmt**【输出】 | APP输入的Cashback金额 |
| 返回 | 0 | 成功 |
| 其它 | 见[返回码列表](#_返回码列表_1) |
| 用 法 | **GPO之前回调SDK** | |

**pszAppListInfo**？？？

|  |  |  |
| --- | --- | --- |
| 原 型 | **int PdkEmv\_cSelApp(uchar\* pszAppListInfo[], int iAppNum, int\* piSeledApp)** | |
| 功 能 | (UI回调)在应用列表中选择一个应用 | |
| 参数 | **ppucAppListInfo**【输入】 | 显示的应用名称组 |
| **iAppNum**【输入】 | 显示的应用名称个数 |
| **piSeledAID**【输出】 | 被选择的AID序号（按显示的应用名称排序，0表示选中**pszAppListInfo[0]**对应的App） |
| 返回 | 0 | 成功 |
| 其它 | 见[返回码列表](#_返回码列表_1) |
| 用 法 | 在**EMV\_II\_FinalSelect之前回调** | |

|  |  |  |
| --- | --- | --- |
| 原 型 | **int PdkEmv\_cCardHolderVerify(uchar CVMType,**  **uchar \* CVMPara,**  **uchar\* GetInfo,**  **uchar\* Result)** | |
| 功 能 | （UI回调）输入持卡人验证信息(联机pin、脱机pin、身份验证确认) | |
| 参数 | **CVMType**【输入】 | 持卡人验证方法类型：  EMV\_CVM\_CERTIFICATE(PBOC)、  EMV\_CVM\_SIGNATURE、  EMV\_CVM\_FAIL\_CVM、EMV\_CVM\_ONLINE\_PIN、  EMV\_CVM\_OFFLINE\_PIN |
| **CVMPara**【输入】 | CVMType匹配的参数（如脱机pin的重试次数、身份信息的身份证号） |
| **GetInfo**【输出】 | 填入的持卡人验证信息（如脱机明文pin） |
| **Result**【输出】 | 持卡人验证的结果  EMV\_OK 输入密码成功  EMV\_NO\_PASSWORD无密码或用户不希望输入密码 |
| 返回 | 0 | 成功 |
| 其它 | 见[返回码列表](#_返回码列表_1) |
| 用 法 | **EMV\_II\_StartCVM和EMV\_II\_CompleteCVM之间回调** | |

|  |  |  |
| --- | --- | --- |
| 原 型 | **int PdkEmv\_cReqOnl(unsigned char\* ReqTLV,**  **unsigned char\* RespInfo,**  **unsigned char\* OnlResult)** | |
| 功 能 | （回调）联机通讯处理 | |
| 参数 | **ReqTLV**【输入】 | 联机请求需要的IC卡数据 |
| **RespInfo**【输出】 | 联机应答返回的IC卡数据 |
|  | **OnlResult**【输出】 | 联机结果  ONLINE\_APPROVE 0x00 //联机返回码(联机批准)  ONLINE\_FAILED 0x01 //联机返回码(联机失败)  ONLINE\_REFER 0x02 //联机返回码(联机参考)  ONLINE\_DENIAL 0x03 //联机返回码(联机拒绝)  ONLINE\_ABORT 0x04 //兼容PBOC(交易终止) |
| 返回 | 0 | 成功 |
| 其它 | 见[返回码列表](#_返回码列表_1) |
| 用 法 | 在**EMV\_II\_CompleteTrans前回调** | |

|  |  |  |
| --- | --- | --- |
| 原 型 | **int PdkEmv\_cDispCardNo(uchar \*CardNo)** | |
| 功 能 | （UI回调）显示卡号并确认 | |
| 参数 | **CardNo**【输入】 | 卡号值 |
| 返回 | 0 | 成功 |
| 其它 | 见[返回码列表](#_返回码列表_1) |
| 用 法 | 在**EMV\_II\_ReadAppData之后回调** | |

## QPBOC流程封装成一个API+几个UI回调

|  |  |  |
| --- | --- | --- |
| 原 型 | **int PdkEmv\_QpbocProcess (uchar\* ReqTagList, uchar\* RespTlv, STTRANS\_PARA \*pst TransPara,**  **uchar\* ucTransResult)** | |
| 功 能 | **QPBOC**流程处理全过程 | |
| 参 数 | **ReqTagList**【输入】 | 应用需要的tag列表  例：  tag1-9f34,tag2-4f: “\x00\x03\x9f\x34\4f” （头两字节Hex长度） |
| **TransPara**【输入】 | 传入的交易参数（包括金额、日期时间、POS流水号） |
| **RespTlv**【输出】 | 实际打印的emv要素tlv  例：tlv1-95 05 0000080000,tlv2-5f2a 02 0156: “\x00\x0c\x95\x05\00\x00\x08\x00\x00\x5f\x2a\x02\x01\x56” （头两字节Hex长度） |
| **ucTransResult**【输出】 | AC\_AAC 0x00  AC\_TC 0x01  AC\_ARQC 0x02  AC\_AAC\_HOST 0x03 //used for VisaOnLine |
| 返 回 | 0 | 成功 |
| 其它 |  |
| 用 法 |  | |

EMV参数信息

| 参数名称 | 参数属性 | 参数长度（byte） | 参数tag值 | 参数含义 | 参数下载时间 | 参数适用场合 |
| --- | --- | --- | --- | --- | --- | --- |
| AID | b | 5－16 | 9F06 | 终端支持的借/贷记应用列表，如ISO/IEC 7816-5所述，指明应用 | 安装或调整时 | 交易应用 |
| 应用选择指示符（ASI） | b | 1 | DF01 | 指示应用选择时终端上的AID与卡片中的AID是完全匹配（长度和内容都必须一样），还是部分匹配（卡片AID的前面部分与终端AID相同，长度可以更长）。终端支持的应用列表中的每个AID仅有一个应用选择指示符。 | 安装或调整时 | 交易应用 |
| 应用版本号 | b | 2 | 9F09 | 支付系统给应用分配的版本号 | 安装或调整时 | 交易应用 |
| TAC－缺省 | b | 5 | DF11 | 标识如果交易可以联机完成但终端没有联机交易能力时，拒绝交易的收单行条件 | 安装或调整时 | 交易应用 |
| TAC－联机 | b | 5 | DF12 | 标识联机交易的收单行条件 | 安装或调整时 | 交易应用 |
| TAC－拒绝 | b | 5 | DF13 | 标识不作联机尝试即拒绝交易的收单行条件 | 安装或调整时 | 交易应用 |
| 终端最低限额 | b | 4 | 9F1B | IC卡消费时终端允许的最低脱机限额 | 安装或调整时 | 交易应用 |
| 偏置随机选择的阈值 | b | 4 | DF15 | 在终端风险管理中用于随机交易选择的值 | 安装或调整时 | 交易应用 |
| 偏置随机选择的最大目标百分数 | cn（包含两位有效数字） | 1 | DF16 | 用于偏置随机选择的最大目标百分数 | 安装或调整时 | 交易应用 |
| 随机选择的目标百分数 | cn（包含两位有效数字） | 1 | DF17 | 用于随机选择的目标百分数 | 安装或调整时 | 交易应用 |
| 缺省DDOL | b | 变长 | DF14 | 卡片中无DDOL时用于构造内部认证命令的DDOL | 安装或调整时 | 交易应用 |
| 终端联机PIN支持能力 | b | 1 | DF18 | 指示终端在每个AID的要求下是否支持联机PIN的输入。 | 安装或调整时 | 交易应用  当值为00000001时表示支持联机PIN。当值为00000000时表示不支持联机PIN。 |
| 终端电子现金交易限额 | cn | 6 | 9F7B | 终端使用此数据元（如果存在的话）判断一个交易的处理方式，当授权金额小于该限额时允许电子现金交易，否则设置终端行为代码并根据判断确认交易方式（小额支付参数）。 | 安装或调整时 | 交易应用 |
| 非接触读写器脱机最低限额 | cn | 6 | DF19 | 在AID联合中，用来指示读写器中非接触交易的最低限额 | 安装或调整时 | 交易应用 |
| 非接触读写器交易限额 | cn | 6 | DF20 | 如果非接触交易的金额大于或等于此数值，则交易终止。允许在其他界面尝试此交易 | 安装或调整时 | 交易应用 |
| 读写器持卡人验证方法（CVM）所需限制 | cn | 6 | DF21 | 如果非接触交易超过此值，读写器要求一个持卡人验证方法（CVM）。 | 安装或调整时 | 交易应用 |

## 传入EMV参数

|  |  |  |
| --- | --- | --- |
| 原 型 | **int PdkEmv\_LoadPara(ushort usPackNum, uchar\* pucPara)** | |
| 功 能 | 获取并保存EMV参数信息 | |
| 参数 | **usPackNum**【输入】 | 请求包序号（若为0表示清除所有的已存参数） |
| **pucPara** | 包中的EMV参数信息（TLV），每个包一组参数信息-见“EMV参数信息  ” |
| 返回 | 0 | 成功 |
| 其它 | 见[返回码列表](#_返回码列表_1) |
| 用 法 | 装机调用 | |

EMV中心公钥信息

| 参数名称 | 参数属性 | 参数长度（byte） | 参数tag值 | 参数含义 | 参数下载时间 | 参数适应场合 |
| --- | --- | --- | --- | --- | --- | --- |
| RID | b | 5 | 9F06 | 与认证中心公钥索引一起标识认证中心的公钥 | 安装或调整时 | 交易应用 |
| 认证中心公钥索引 | b | 1 | 9F22 | 与RID一起标识认证中心的公钥 | 安装或调整时 | 交易应用 |
| 认证中心公钥有效期 | n8 | 8 | DF05 | 认证中心规定的有效期限 | 安装或调整时 | 交易应用 |
| 认证中心公钥哈什算法标识 | b | 1 | DF06 | 标识用于在数字签名方案中产生哈什结果的哈什算法 | 安装或调整时 | 交易应用 |
| 认证中心公钥算法标识 | b | 1 | DF07 | 标识使用在认证中心公钥上的数字签名算法 | 安装或调整时 | 交易应用 |
| 认证中心公钥模 | b | 变长，最大为248 | DF02 | 公钥模值 | 安装或调整时 | 交易应用 |
| 认证中心公钥指数 | b | 1或3 | DF04 | 公钥指数 | 安装或调整时 | 交易应用 |
| 认证中心公钥校验值 | b | 变长 | DF03 | 验证认证中心公钥用 | 安装或调整时 | 交易应用 |
| 1. 认证中心公钥校验值的计算内容为RID+认证中心公钥索引+认证中心公钥模+认证中心公钥指数；认证中心公钥校验值的计算方法为SHA-1。 | | | | | | |

## 传入EMV公钥

|  |  |  |
| --- | --- | --- |
| 原 型 | **int PdkEmv\_LoadCAPK(ushort usPackNum, uchar\* pucCAPK)** | |
| 功 能 | 获取并保存EMV参数信息 | |
| 参数 | **usPackNum**【输入】 | 请求包序号（若为0表示清除所有的已存公钥） |
| **pucCAPK** | 包中的EMV公钥信息（TLV），每个包一组公钥信息-见“EMV公钥信息  ” |
| 返回 | 0 | 成功 |
| 其它 | 见[返回码列表](#_返回码列表_1) |
| 用 法 | 装机调用 | |