# overview

## Compile Purpose

This document introduces the Linux POS SDK interface definition and programming guidance for some interfaces。

## User

The target audience of this article is application developers who are engaged in secondary development based on the Linux POS SDK.

# Return code and parameter description

## Return code classification

Explain the types of return codes and the range of values that appear in this document.

Table 2. 1 Return code classification

|  |  |  |
| --- | --- | --- |
| Type | Numeric value (decimal) | [explain](file:///C:\Program%20Files\baidu-translate-client\resources\app.asar\app.html) |
| Universal function return value | -1000~-1999 | As an API error common code |
| system function | -2200~-2299 |  |
| Power management | -2300~-2399 |  |
| Encryption and decryption | -2400~-2499 |  |
| Word bank | -2500~-2599 |  |
| LED display | -2600~-2699 |  |
| magnetic stripe reader | -2700~-2799 |  |
| Contact card reader | -2800~-2899 |  |
| Contactless card reader | -2900~-2999 |  |
| Communication port operation | -3000~-3099 |  |
| MODEM | -3100~-3299 | |
| **IP** config | -3300~-3399 | |
| PED | -3800~-3899 | |

## Universal return code

Explain the macro definitions, values, and corresponding explanations for the generic return codes that appear in this document。

Table 2.2 Universal return code

|  |  |  |
| --- | --- | --- |
| 宏 | 数值 | 说明 |
| RET\_OK | 0 | Success |
| ERR\_INVALID\_HANDLE | -1000 | invalid handle |
| ERR\_TIME\_OUT | -1001 | timeout |
| ERR\_APP\_NOT\_EXIST | -1002 | The application does not exist |
| ERR\_INVALID\_PARAM | -1003 | illegal parameter |
| ERR\_DEV\_NOT\_EXIST | -1004 | The device does not exist |
| ERR\_DEV\_BUSY | -1005 | Device occupied |
| ERR\_DEV\_NOT\_OPEN | -1006 | device is not open |
| ERR\_ACCESS\_DENY | -1007 | Unauthorized access |
| ERR\_FONT\_NOT\_EXIST | -1008 | No specified font |
| ERR\_FILE\_FORMAT | -1009 | File format error |
| ERR\_USER\_CANCEL | -1010 | User cancel |
| ERR\_NO\_PORT | -1011 | No communication port available error |
| ERR\_NOT\_CONNECT | -1012 | not connected |
| ERR\_MEM\_FAULT | -1013 | memory error |
| ERR\_SYS\_BAD | -1014 | There is an issue with the system configuration |
| ERR\_SYS\_NOT\_SUPPORT | -1015 | The system does not provide this function |
| ERR\_STR\_LEN | -1016 | string is too long |
| ERR\_TOO\_MANY\_HANDLE | -1017 | Too many handles |
| ERR\_APP\_MODE | -1018 | Mode error |
| ERR\_FILE\_NOT\_EXIST | -1019 | file does not exist |
| ERR\_TS\_DISABLE | -1020 | Touch screen not turned on |
| ERR\_FONT\_CODE | -1021 | Character encoding error |
| ERR\_VERSION\_TOO\_LOW | -1022 | Version too low |
| ERR\_BATTERY\_ABSENT | -1023 | Battery not present |
| ERR\_BATTERY\_VOLTAGE\_TOO\_LOW | -1024 | Battery voltage too low |

## Parameter Description

The parameters of the API are divided into input and output. Their types are identified in the detailed API description.

For input and output parameters of a string, use "\ x00" as the terminator. String type parameters must specify length limitations。

# Thread

Call the POSIX multi-threaded library ptthread. Ptthread is the abbreviation for POSIX threads, commonly abbreviated as Pthreads, and is a thread defined by the POSIX standard. Pthreads defines a set of C programming language types, functions, and constants, implemented as a ptthread. h header file and a thread library

#include <pthread.h>

static pthread\_tntid;

static void \*thread\_fn(void \*arg)

{

printf("This is child thread\n");

return ((void \*)0);

}

int main()

{

printf("This is main thread\n");

if(pthread\_create(&ntid,NULL,thread\_fn,NULL) != 0)

printf("can't create thread\n");

sleep(5);

return 0;

}

# System function

## List of function return values

Table 4. 1 List of function return values

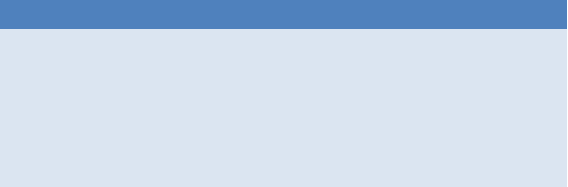
|  |  |  |
| --- | --- | --- |
| Macro | Value | Explain |
| ERR\_FILE\_NOT\_FOUND | -2201 | File not found |
| ERR\_VERIFY\_SIGN\_FAIL | -2204 | Verification signature failed |
| ERR\_NO\_SPACE | -2205 | Insufficient system space |
| ERR\_NEED\_ADMIN | -2207 | Need higher permissions |
| ERR\_PUK\_NOT\_EXIST | -2208 | PUK not exist |
| ERR\_OS\_VERSION\_TOO\_ LOW | -2209 | System version too low |
| ERR\_INVALID\_PARAM | -1003 | illegal parameter |
| ERR\_DEV\_NOT\_EXIST | -1004 | The device does not exist |
| ERR\_DEV\_BUSY | -1005 | The device is currently in use |
| ERR\_ACCESS\_DENY | -1007 | No access permission |
| ERR\_SYS\_NOT\_SUPPORT | -1015 | The system does not support |
| ERR\_FILE\_NOT\_EXIST | -1019 | file does not exist |

Installation file type definition

Table 4.2 Installation file type definition

|  |  |  |
| --- | --- | --- |
| 宏 | 数值 | 说明 |
| FILE\_TYPE\_APP | 1 | Application package |
| FILE\_TYPE\_APP\_PARAM | 2 | Application Data File |
| FILE\_TYPE\_SYS\_LIB | 3 | System dynamic library files |
| FILE\_TYPE\_PUB\_KEY | 4 | User public key file |
| FILE\_TYPE\_AUP | 5 | Application upgrade package |

## Data Definition

**LOG\_T** enum **LOG** Type

|  |  |
| --- | --- |
| LOG\_T：  typedefenum{  LOG\_DEBUG, LOG\_INFO,  LOG\_WARN,  LOG\_ERROR, } LOG\_T; | /\*debug info\*/  /\*prompt info\*/  /\*warning info\*/  /\*error info\*/ |
| **ST\_TIME** time struct | |
| ST\_TIME： | |
| typedef struct{  int Year; /\*year 1970– 2037\*/  int Month; /\*month 1 – 12\*/  int Day; /\*day 1 –31\*/  int Hour; /\*hour 0 – 23\*/  int Minute; /\*minute 0 – 59\*/  int Second; /\*second 0 – 59\*/  intDayOfWeek; /\* Week 1-7（Only valid for reading time）\*/  } ST\_TIME; | |

|  |
| --- |
| **ST\_TIMER** timer struct |
| ST\_TIMER： |
| typedef struct{  unsigned long Start;  unsigned long Stop;  unsigned long TimeLeft;  } ST TIMER; |
| **ST\_APP\_INFO** app info struct |
| ST\_APP\_INFO： |
| typedef struct{  charId[64];  char Name[64];  charBin[64];  char Artwork[64];  charDesc[64];  char Vender[32];  char Version[32] ;  } ST\_APP\_INFO; |
| **ST\_OPT\_INFO** Optional system component information structure |
| ST\_OPT\_INFO： |
| typedef struct {  char Name[64];  char Version[32];  } ST\_OPT\_INFO; |

## Set time

### OsSetTime

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | int OsSetTime(const ST\_TIME \*Time)； | | |
| 功能 | Set the current date and time of the system. | | |
| 参数 | Time【Input】 | [ST\_TIME](#bookmark467) Structure pointer，Time cannot be null | |
|  |  | ST\_TIME DayOfWeek,will be ignored | |
| 返回 | RET\_OK  ERR\_NEED\_ADMIN  ERR\_INVALID\_PARAM | | Success  Require system main application permissions  illegal parameter |
| 用法 | Only the main application can set the time, otherwise the setting will fail and return  ERR\_NEED\_ADMIN。 | | |

### OsGetTime

|  |  |  |
| --- | --- | --- |
| 原型 | void OsGetTime(ST\_TIME \*Time)； | |
| 功能 | 获取终端系统日期和时间。 | |
| 参数 | Time【Output】 | [ST\_TIME](#bookmark467) Structure pointer，Time cannot be null ST\_TIME DayOfWeek,will be ignored |
| 返回 | 无 | |
| 用法 |  | |

## Timer

### OsTimerSet

|  |  |  |
| --- | --- | --- |
| 原型 | int OsTimerSet(ST\_TIMER \*Timer,  unsigned long Ms); | |
| 功能 | Create a timer。 | |
| 参数 | Timer【Output】 | [ST\_TIMER](#bookmark469) Structure pointer，Timer cannot be null |
| Ms【Input】 | Timed time, in milliseconds |
| 返回 | RET\_OK Success  ERR\_INVALID\_PARAM illegal parameter | |
| 用法 | It is independent of system time, and the timer will stop after the system goes to sleep. | |

### OsTimerCheck

|  |  |  |
| --- | --- | --- |
| 原型 | unsigned long OsTimerCheck(ST\_TIMER \*Timer); | |
| 功能 | Detect the remaining value of the specified timer. | |
| 参数 | Timer【Input】 | Timer |
| 返回 | >=0 Timed time remaining value, in milliseconds. | |
|  | ERR\_INVALID\_PARAM illegal parameter | |
| 用法 |  | |

## 4.5 Delay

### OsSleep

|  |  |  |
| --- | --- | --- |
| 原型 | void OsSleep(unsigned int Ms); | |
| 功能 | Set the current process/thread to pause. | |
| 参数 | Ms 【Input】 | Delay time, in milliseconds. |
| 返回 |  | |
| 用法 |  | |

## Log

### OsLogSetTag

|  |  |  |
| --- | --- | --- |
| 原型 | void OsLogSetTag(const char \*Tag); | |
| 功能 | Set LOG information labels. | |
| 参数 | Tag【Input】 | LOG information labels. |
| 返回 |  | |
| 用法 | 1. Used to set the label for the LOG, the system default label is NULL;  2. Suggest setting Tag as the name of the application; Tag supports a maximum of 32 bytes. When it is greater than 32 bytes, only the first 32 bytes are used；  3. When Tag is null or an empty string, OsLog()No log information will be recorded. | |

### OsLog

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | int OsLog(LOG\_T Prio,const char \*fmt,…); | | |
| 功能 | Save the LOG. | | |
| 参数 | Prio【Input】 | [LOG\_T](#bookmark16) struct，LOG information type。 | |
| fmt 【输入】 | LOG info format | |
| 返回 | RET\_OK  ERR\_INVALID\_PARAM | | success  param invalid |
| 用法 | If OsLogSetTag() is not called to set the label before use, OsLog() will not record log information. | | |

## Get Count Values

### OsGetTickCount

|  |  |  |
| --- | --- | --- |
| 原型 | unsigned long OsGetTickCount(void); | |
| 功能 | Obtain the cumulative time from power on to current operation of the system, excluding sleep time, in milliseconds. | |
| 参数 |  |  |
| 返回 | >0 value | |
| 用法 |  | |

## Get application information

### OsGetAppInfo

|  |  |  |
| --- | --- | --- |
| 原型 | int OsGetAppInfo(ST\_APP\_INFO AppInfo[],  int InfoCnt); | |
| 功能 | Obtain information about installed application software. | |
| 参数 | AppInfo【Input】 | [ST\_APP\_INFO](#bookmark470) pointer,can not be null |
| InfoCnt【Output】 | AppInfo The number of applications that can be stored in an array |
| 返回 | >=0  ERR\_NEED\_ADMIN  ERR\_INVALID\_PARAM | the number of obtained application information |
| 用法 |  | |

### OsGetOptInfo

|  |  |  |
| --- | --- | --- |
| 原型 | int OsGetOptInfo(ST\_OPT\_INFO OptInfo[],  int InfoCnt); | |
| 功能 | Obtain information on installed optional system components. | |
| 参数 | OptInfo 【Input】 | Can not be NULL。 |
| InfoCnt 【Output】 | OptInfo The number of optional system components that can be stored |
| 返回 | >=0 Number of component information | |
| ERR\_FILE\_NOT\_FOUND | |
| ERR\_INVALID\_PARAM | |
| 用法 | When the number of optional system components is greater than InfoCnt, only the information of the first InfoCnt optional system components is obtained. | |

## Beep

### OsBeep

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | void OsBeep(int ToneType,  int DurationMs); | | |
| 功能 | Set the frequency and duration of the buzzer's beeping. | | |
| ToneType 【Input】  DurationMs 【Input】  参数 | | Beep type, range [1, 7] | |
| Duration, range [1010000] unit: milliseconds. If DurationMs<10, set the duration to 10; If DurationMs>10000, set the duration to 10000. | |
| 返回 |  | | |
| 用法 | The correspondence between the type of buzzing and the operating frequency of the buzzer is as follows:   |  |  | | --- | --- | | Beep Type | (Hz) | | ≤1 | 1680 | | 2 | 1850 | | 3 | 2020 | | 4 | 2130 | | 5 | 2380 | | 6 | 2700 | | ≥7 | 2750 | | |  |

## Key Tone

### OsSetKeyTone

|  |  |  |
| --- | --- | --- |
| 原型 | void OsSetKeyTone(int OnOff,  int DutyCycle); | |
| 功能 | Set the switch and duty cycle of the button sound. | |
| 参数 | OnOff【Input】 | 0 close， 1 open |
| DutyCycle【Input】 | range【1 ，99】。  Not yet effective. |
| 返回 |  | |
| 用法 |  | |

## Run App

### OsRunApp

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | int OsRunApp(char \*AppId,  char \*\*Argv,  void \*Data,  RUNAPP\_CB CbOut,  RUNAPP\_CB CbErr); | | |
| 功能 | Switch to the specified sub application. | | |
| 参数 | AppId【Input】 | The sub application ID that needs to be switched. | |
| Argv【Input】 | The parameter list for running sub applications has not yet taken effect. | |
| Data【Input】 | Custom data will be passed to the parameters CbOut and CbErr callbacks. Not yet effective | |
| CbOut【Input】 | Standard output information callback function. Not yet effective | |
| CbErr【Input】 | Standard error message callback function. Not yet effective | |
| 返回 | RET\_OK  ERR\_APP\_NOT\_EXIST  ERR\_ACCESS\_DENY  ERR\_APP\_MODE  ERR\_INVALID\_PARAM  ERR\_NEED\_ADMIN | | Success |

## File installation and uninstallation

### OsInstallFile

|  |  |  |
| --- | --- | --- |
| 原型 | int OsInstallFile(const char \*AppId,  const char \*FilePath,  int FileType); | |
| 功能 | Install or update application software, application data, OPT package, user public key, and peripheral firmware (FWP package). | |
| 参数 | AppId【Input】 | App id |
| FilePath【Input】 | The path to the Apk file to be installed cannot be nu |
| FileType | . FILE\_TYPE\_APP  . FILE\_TYPE\_APP\_PARAM  . FILE\_TYPE\_SYS\_LIB动  . FILE\_TYPE\_PUB\_KEY  . FILE\_TYPE\_AUP . FILE\_TYPE\_FWP |
| 返回 | RET\_OK  ERR\_PUK\_NOT\_EXIST |  |
| ERR\_FILE\_NOT\_FOUND |  |
| ERR\_FILE\_FORMAT |  |
| ERR\_INVALID\_PARAM |  |
| ERR\_VERIFY\_SIGN\_FAIL |  |
| ERR\_APP\_MODE |  |

### OsUninstallFile

|  |  |  |
| --- | --- | --- |
| 原型 | int OsUninstallFile(const char \*AppId,  int FileType); | |
| 功能 | Uninstall the specified application or OPT package. | |
| 参数 | AppId【Input】 | AppId |
| FileType | . FILE\_TYPE\_APP  . FILE\_TYPE\_SYS\_LIB |
| 返回 | RET\_OK  ERR\_APP\_NOT\_EXIST  ERR\_FONT\_NOT\_EXIST | |

## System firmware upgrade

### OsFirmwareGetVersion

|  |  |  |
| --- | --- | --- |
| 原型 | int OsFirmwareGetVersion(char \*Version，  int Size); | |
| 功能 | Obtain the current system firmware version. | |
| 参数 | Version【Output】 | Firmware version buffer |
| Size 【Input】 | Version buffer length. The recommended length is 64 bytes. |
| 返回 | RET\_OK  ERR\_INVALID\_PARAM | |
| 用法 |  | |

### OsFirmwareUpgrade

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | int OsFirmwareUpgrade(const char \*FwPath,int FwType); | | |
| 功能 | Upgrade system firmware. | | |
| 参数 | FwPath【Input】 | Firmware path | |
|  | FwType 【Input】 | Firmware type  FW\_TYPE\_SE\_APP SE app  FW\_TYPE\_SE\_FW\_CFG SE fm config  FW\_TYPE\_SE\_FW SE fm | |
| 返回 | RET\_OK  RR\_FILE\_NOT\_FOUND  ERR\_VERIFY\_SIGN\_FA IL  ERR\_APP\_MODE  ERR\_SYS\_NOT\_SUPPO RT | |  |

## Get system version

### OsGetSysVer

|  |  |  |
| --- | --- | --- |
| 原型 | void OsGetSysVer(int VerType,  char \*Version); | |
| 功能 | Obtain operating system and module firmware version information | |
| 参数 | VerType | Version type：  0x01: sys version  0x02: Middleware version number  0x03: Security CPU App version number  0x04: Secure CPU firmware version |
| Version【Output】 | The size of the version information buffer must be no less than 31 bytes |
| 返回 |  | |
| 用法 | 1. If Version [0] is equal to 0x00, it indicates that the corresponding module does not exist;  2. The length of each version information is less than or equal to 31 bytes, and ends with the "\ 0" character. | |

### OsGetTermSn

|  |  |  |
| --- | --- | --- |
| 原型 | int OsGetTermSn(char\* Sn); | |
| 功能 | Obtain the device serial number. | |
| 参数 | Sn【Output】 | The size of the SN information buffer must be no less than 31 bytes |
| 返回 | RET\_OK  ERR\_INVALID\_PARAM  ERR\_ACCESS\_DENY | |
| 用法 | 1. If Sn [0] equals 0x00, it indicates that SN does not exist;  2. The length of SN is less than or equal to 31 bytes and ends with the "\ 0" character. | |

# Encryption and Decryption

## List of function return values

Table 5. 1 List of function return values

|  |  |  |
| --- | --- | --- |
| 宏 | 数值 | 说明 |
| ERR\_DATA\_TOO\_BIG | -2400 | RSA 被加密数据大于模 |
| ERR\_GEN\_RANDOM | -2401 | 产生随机数失败 |
| ERR\_GEN\_FAIL | -2402 | 生成RSA 密钥对失败 |

## Random

### OsGetRandom

|  |  |  |
| --- | --- | --- |
| 原型 | void OsGetRandom(unsigned char \*Random,  int RandomLen); | |
| 功能 | Generate true random numbers | |
| 参数 | Random【Output】 | Pointer for storing random numbers |
| RandomLen【Input】 | The length of the random number to be read (less than or equal to 4096 bytes) |
| 返回 | 无 | |
| 用法 |  | |

## SHA

Support SHA- 1，SHA-2(SHA-256，SHA-512) SHA-2(SHA-224， SHA-384)

### OsSHA

|  |  |  |
| --- | --- | --- |
| 原型 | void OsSHA(int Mode,  const void \*Data,  int DataLen,  unsigned char\*ShaOut); | |
| 功能 |  | |
| 参数 | Mode | type  . SHA\_TYPE\_ 1  . SHA\_TYPE\_224  . SHA\_TYPE\_256  . SHA\_TYPE\_384  . SHA\_TYPE\_512 |
| Data【Input】 |  |
| DataLen 【Input】 |  |
| ShaOut【Output】 | The output SHA value should have a storage space of more than 64 bytes |
|  |  | Mode ShaOut Length  SHA\_TYPE\_ 1 20  SHA\_TYPE\_224 28  SHA\_TYPE\_256 32  SHA\_TYPE\_384 48  SHA\_TYPE\_512 64 |
| 返回 |  | |
| 用法 |  | |

## DES

### OsDES

|  |  |  |
| --- | --- | --- |
| 原型 | void OsDES(const unsigned char \*Input,  unsigned char \*Output,  const unsigned char \*DesKey,  int KeyLen,  int Mode); | |
| 功能 | DES/TDES Encryption and decryption operations | |
| 参数 | Input【Input】 | 8 bytes |
| Output【Output】 | 8 bytes |
| DesKey【Input】 | DES/TDES key |
| KeyLen 【Input】 | 8 、16 or 24(bytes) |
| Mode | 1. Decrypt   1-Encrypt |
| 返回 |  | |
| 用法 |  | |

## AES

Support [AES](http://en.wikipedia.org/wiki/Advanced_Encryption_Standard)， AES- 128 、AES- 192 、AES-256 .

### OsAES

|  |  |  |
| --- | --- | --- |
| 原型 | void OsAES(const unsigned char \*Input,  unsigned char \*Output,  const unsigned char \*AesKey,  int KeyLen,  int Mode); | |
| 功能 |  | |
| 参数 | Input【Input】 | 16 bytes |
| Output【Output】 | 16 bytes |
| AesKey【Input】 | Key value |
| KeyLen 【输入】 | 16 、24 or 32(bytes) |
| Mode | 1-Encrypt  0-Decrypt |
| 返回 |  | |
| 用法 |  | |

## RSA

Support related algorithms for the RSA family, including public-private key pair generation, RSA encryption, and RSA decryption. The maximum supported module length is 4096 bits.

### OsRSA

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | int OsRSA(const unsigned char \*Modulus,  int ModulusLen, | | |
|  | const unsigned char \*Exp,  int ExpLen,  const unsigned char \*DataIn,  unsigned char \*DataOut  int Mode); | | |
| 功能 | Perform RSA encryption and decryption operations. | | |
| 参数 | Modulus 【Input】 | Store modular pointers for RSA operations (i.e. n=p \* q)  Store in order of high order before low order | |
| ModulusLen 【Input】 | Modulus length | |
| Exp【Input】 | Store exponential pointers for RSA operations. Store in order of high order before low order. | |
| ExpLen 【Input】 | Exponential length (in bytes) | |
| DataIn【Input】 | Input data buffer pointer, with the same length as the module length | |
| DataOut【Output】 | Output data buffer pointer, with the same length as the module length | |
| Mode 【Input】 | 1 Encrypt 0 Decrypt | |
| 返回 | RET\_OK  ERR\_INVALID\_PARAM  ERR\_DATA\_TOO\_BIG | |  |

### OsRSAKeyGen

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | int OsRSAKeyGen(unsigned char \*Modulus,  unsigned char \*PriExp,  int ModulusLen,  const unsigned char \* PubExp); | | |
| 功能 | Generate RSA public-private key pairs with specified indices and moduli. | | |
| 参数 | Modulus 【Output】 | The modulus of the key (stored in order of high bit first and low bit last) | |
|  | PriExp【Output】 | Private key index (stored in order of high order before low order) | |
| ModulusLen 【Input】 | The length of the module (in bytes, which can be 64, 128, 256, 512) | |
| PubExp【Input】 | Public key index.  Can only be: "\ x00 \ x01 \ x00 \ x01" | |
| 返回 | RET\_OK  ERR\_INVALID\_PARAM  ERR\_GEN\_RANDOM  ERR\_GEN\_FAIL | |  |
| 用法 |  | | |

# PED

SDK provides a series of PED interfaces, including built-in PED, MK/SK, DUKPT and other related interfaces.

PED adopts a three-layer key system, and the order from top to bottom is as follows:

. **TLK**－Terminal Key Loading Key

The private key of the acquiring bank or POS operator is directly written by the acquiring bank or POS operator in a secure environment.

Each PED terminal has only one key with an index number of 1.

. **TMK**－Terminal Master Key＝Acquirer Master Key

The terminal master key, also known as the acquiring bank master key.

There can be 100 keys of this type, with index numbers ranging from 1 to 100.

. **TWK**－Transaction working key = Transaction Pin Key + Transaction MAC Key +Terminal DES Key

Terminal working key, used for PIN ciphertext, MAC and other operations.

There can be 100 keys of this type, with index numbers ranging from 1 to 100.

**TPK：**Used to calculate the PIN Block after the application inputs the PIN.

**TAK**：Used for calculating MAC in application message communication.

**TDK：**Used for DES/TDES encrypted transmission or storage of sensitive data in applications.

Each key has its corresponding index number, length, purpose, and label. The label of the key is set through the API before writing the key to authorize its usage and ensure that the key is not abused.

. DUKPT：

DUKPT（Derived Unique Key Per Transaction）

This type of key can have 100 sets. When writing TIK, it is necessary to select the index number of the group, and when using the DUKPT key, select the corresponding group index.

## List of function return values

Table 6. 1PED return values

|  |  |  |
| --- | --- | --- |
| 宏 数值 说明 | | |
| ERR\_PED\_NO\_KEY -3801 Key does not exist  ERR\_PED\_KEY\_IDX\_ERR -3802 Key index error | | |
| ERR\_PED\_DERIVE\_ERR -3803 | When writing a key, the key hierarchy is incorrect, and the source key hierarchy is lower than the destination key hierarchy | |
| **ERR\_PED\_CHECK\_KEY\_FAIL** -3804 Key verification failed | | |
| ERR\_PED\_NO\_PIN\_INPUT -3805 No PIN input | | |
| ERR\_PED\_PIN\_INPUT\_CANCEL -3806 Cancel PIN input | | |
| ERR\_PED\_WAIT\_INTERVAL -3807 | The interval time between function calls is less than the set minimum time interval *(calc* *PIN block/MAC)* | |
| ERR\_PED\_KCV\_MODE\_ERR -3808 KCV mode error   |  |  |  | | --- | --- | --- | | ERR\_PED\_KEY\_TAG\_ERR | -3809 | Key label error, unable to use the key | | ERR\_PED\_KEY\_TYPE\_ERR | -3810 | The key type is not supported or the key type specified in the index does not match | | ERR\_PED\_PIN\_LEN\_ERR | -3811 | The length of the input PIN does not match the expected length | | ERR\_PED\_DSTKEY\_IDX\_ERR | -3812 | Destination key index error | | ERR\_PED\_SRCKEY\_IDX\_ERR | -3813 | Source key index error | | ERR\_PED\_KEY\_LEN\_ERR | -3814 | Key length error | | ERR\_PED\_INPUT\_PIN\_TIMEOUT | -3815 | PIN input timeout | | ERR\_PED\_NO\_ICC | -3816 | IC card does not exist | | ERR\_PED\_ICC\_INIT\_ERR | -3817 | IC card initialization error | | ERR\_PED\_GROUP\_IDX\_ERR | -3818 | DUKPT group index error | | ERR\_PED\_TAMPERED | -3819 | PED locked | | ERR\_PED\_NO\_MORE\_BUF | -3820 | No free buffer available | | ERR\_PED\_NORMAL\_ERR | -3821 | *PED* General error | | ERR\_PED\_NEED\_ADMIN | -3822 | The current operation requires advanced permissions | | ERR\_PED\_DUKPT\_KSN\_OVERFLOW | -3823 | Current DUKPT overflow | | ERR\_PED\_KCV\_CHECK\_FAIL | -3824 | KCV verification failed | | ERR\_PED\_SRCKEY\_TYPE\_ERR | -3825 | When writing the key, the type of the source key ID does not match the type of the source key | | ERR\_PED\_UNSPT\_CMD | -3826 | Command not supported | | ERR\_PED\_ADMIN\_ERR | -3827 | Failed to obtain system sensitive services | | **ERR\_PED\_DOWNLOAD\_INACTIVE** -3828 PED download function not activated | | | | | |
| ERR\_PED\_KCV\_ODD\_CHECK\_FAIL -3829 KCV odd check failed | | |
| **ERR\_PED\_PED\_DATA\_RW\_FAIL** -3830 Failed to read PED data | | |
| ERR\_PED\_ICC\_CMD\_ERR -3831 | | IC card operation error (offline plaintext, ciphertext PIN verification) |
| **ERR\_PED\_DUKPT\_NEED\_INC\_KSN** -3832 DUKPT KSN needs to add 1 first | | |
| **ERR\_PED\_DUKPT\_NO\_KCV** -3833 DUKPT key without KCV | | |
| **ERR\_PED\_NO\_FREE\_FLASH** -3834 Insufficient PED storage space | | |
| ERR\_PED\_INPUT\_CLEAR -3835 | | The user presses the CLEAR key to exit PIN input |
| ERR\_PED\_INPUT\_BYPASS\_BYFUNCTION -3836 | | The user presses the FN/ATM4 key to cancel PIN input |
| **ERR\_PED\_NO\_PIN\_MODE** -3837 PIN input mode not set | | |
| **ERR\_PED\_DATA\_MAC\_ERR** -3838 Data MAC verification error | | |
| **ERR\_PED\_DATA\_CRC\_ERR** -3839 Data CRC verification error | | |
| ERR\_PED\_KEY\_VALUE\_INVALID -3840 | | The work key value does not meet the requirements or already exists |

## Data Definition

### KEY TYPE

Table 6.2 key type

|  |  |  |
| --- | --- | --- |
| 宏  PED\_TLK | 数值  0x01 | 说明  Loading Key |
| PED\_TMK | 0x02 | Master Key |
| PED\_TPK | 0x03 | PIN Key |
| PED\_TAK | 0x04 | MAC Key |
| PED\_TDK | 0x05 | Data Key |
| PED\_TIK | 0x10 | DUKPT Initial Key |

## BASIC PED

### OsPedOpen

|  |  |  |
| --- | --- | --- |
|  | int OsPedOpen (void); | |
| func | Open ped service. | |
| param |  |  |
| return | RET\_OK  ERR\_DEV\_BUSY | success  Device occupied |
| usage | Before using PED, you need to call this function to turn on the device, otherwise subsequent related functions will not work. | |

### OsPedGetVer

|  |  |  |
| --- | --- | --- |
|  | int OsPedGetVer (unsigned char \* PedVer); | |
| func | Get ped version。 | |
| param | PedVer【output】 | PED version information buffer, 6 bytes. |
| return | RET\_OK  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM | success  PED not open.  Void param |
| usage |  | |

### OsPedEraseKeys

|  |  |  |
| --- | --- | --- |
|  | int OsPedEraseKeys (void); | |
| func | Erase all key information saved by PED service | |
| param |  |  |
| return | RET\_OK  ERR\_DEV\_NOT\_OPEN  OTHER | success  Ped not open  Please refer to the list of PED function return values |
| usage |  | |

### OsPedClose

|  |  |  |
| --- | --- | --- |
|  | void OsPedClose (void); | |
| func | Close ped. | |
| param |  |  |
| return |  | |
| usage | Call this function after using PED. | |

## PIN ENTER

### OsPedPinKeyNotify

|  |  |  |
| --- | --- | --- |
|  | int OsPedPinKeyNotify(int \*KeyCacheCnt,  uchar \*KeyCache); | |
| func | Monitor and obtain the number of PIN keys entered by the user in the current state, as well as the historical key sequence between the current monitoring and the last monitoring. | |
| param | KeyCacheCnt【output】 | The current number of historical key values obtained. |
| KeyCache【output】 | Not yet used |
| return | >= 0  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM  OTHER | Successfully obtained PIN input event, the size of the return value represents the number of "\*" on the PIN input interface at this time.  PED not open  Void param  Please refer to the list of PED function return values |
| usage |  | |

### OsPedSetAsteriskLayout

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPedSetAsteriskLayout(int Line,int Align); | | |
| func | When entering a PIN, set the layout properties of the interface to display asterisks. | | |
| param | Line | Which line is displayed in the password box, and the default display is the third line (starting from 0) | |
| Align | Alignment: default to left  PED\_ASTERISK\_ALIGN\_LEFT：  The starting position on the left is fixed, and asterisks are displayed in order from left to right; **PED\_ASTERISK\_ALIGN\_CENTER**：  Fixed in the middle position, displaying asterisks symmetrically on both sides; **PED\_ASTERISK\_ALIGN\_RIGHT**：  The starting position on the right is fixed, and asterisks are displayed in order from right to left | |
| return | RET\_OK  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM  OTHER | | success  PED not open  Void param  Please refer to the list of PED function return values |
| usage |  | | |

## MK/SK

### OsPedWriteKey

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPedWriteKey (const unsigned char \* KeyBlock); | | |
| func | Write a key, including TLK, TMK, TWK, SM4-TMK, and SM4-TWK for writing and diverging, and optionally use KCV to verify the correctness of the key. | | |
| param | KeyBlock【input】 | 1 byte | Format：0x03 |
| 1 byte | SrcKeyType：  . PED\_TLK  . PED\_TMK  . PED\_TPK/PED\_TAK/PED\_TD K |
| 1 byte | SrcKeyIdx：  . if SrcKeyType = PED\_TLK， SrcKeyIdx = 1；  . if SrcKeyType = PED\_TMK，SrcKeyIdx = [1~100]；  . if SrcKeyType =  PED\_TPK/PED\_TAK/PED\_TDKd，SrcKeyIdx = [1~100]；  If SrcKeyIdx = 0 Indicates plaintext writing |
| 1 byte | DstKeyIdx：  . if DstKeyType = PED\_TLK，DstKeyIdx = 1；  . if DstKeyType = PED\_TMK， DstKeyIdx = [1~100]；  . if DstKeyType = PED\_TPK or PED\_TAK or PED\_TDK，DstKeyIdx = [1~100]； |
| 7 bytes | Reserved field, random number |
| 1 byte | DstKeyType：  . PED\_TLK  . PED\_TMK  .PED\_TPK/PED\_TAK/PED\_TDK |
| 1 byte | DstKeyLen：8/16/24 |
| 24 bytes | DstKeyValue  Destination key plaintext/ciphertext |
| 1 byte | KcvMode：(Currently only supports 0x00 and 0x01)  0x00：No authentication  **0x01**：Calculate DES/TDES encryption for 0x00 of 8 bytes to obtain the first 3 bytes of the ciphertext as KCV.  **0x02**：Firstly, perform odd parity on the plaintext key, and then perform DES/TDES encryption on "\ x12 \ x34 \ x56 \ x78 \ x90 \ x12 \ x 34 \ x56" to obtain the first three bytes of the ciphertext as KCV.  **0x03**：Pass in a string of data KcvData, use the source key to perform MAC operations on [destination key ciphertext+KcvData] in the specified mode, and obtain an 8-byte MAC, which is KCV. |
| 128  bytes | KcvData：  . if KcvMode is 0x00/0x01/0x02，Fill in random data;  . if KcvMode is 0x03， The first byte of KcvData is the length of the KCV data participating in the calculation, followed by the KCV data, and the second byte after the KCV data represents the MAC operation mode |
| 8 bytes | . if KcvMode = 0x00，Fill in random data;  . if KcvMode =  0x01/0x02/0x03，KcvValue points to the value of KCV. |
| 10 bytes | Fill in random data; |
| return | 0  ERR\_DEV\_NOT\_OPEN  ERR\_INVALIDPARAM  ERR\_SYS\_BAD  ERR\_PED\_KEY\_IDX\_ERR ERR\_PED\_KEY\_TYPE\_ERRERR\_PED\_TAMPERED | | success  PED not open  Invalid param  System error  Key index error  Key type error  PED locked |
| ERR\_PED\_NO\_MORE\_BUF | | Insufficient system memory space |
| ERR\_PED\_NORMAL\_ERR | | PED General error (KeyBlock Format error) |
| ERR\_PED\_DERIVE\_ERR | | Key dispersion error |
| ERR\_PED\_KCV\_MODE\_ERR | | PED KCV Verification mode error |
| ERR\_PED\_KCV\_CHECK\_FAIL | | PED KCV Verification faild |
| OTHER | | Please refer to the list of PED function return values |
| usage | 1. When SrcKeyIdx=0, the system assumes that DstKeyValue is the plaintext of the key, and does not determine SrcKeyType or SrcKeyIdx. Instead, it directly writes DstKeyValue to the DstKeyIdx position in the DstKeyType area. | | |

### OsPedGetPinBlock

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPedGetPinBlock (int KeyIdx,  const unsigned char \*DataIn,  const char \*ExpPinLen,  int Mode,  unsigned long TimeoutMs,  unsigned char \*PinBlock); | | |
| func | Within the specified time limit, scan the keyboard and input a PIN of the specified length as ExpPinLenIn, and output the PIN block generated by the encryption algorithm specified by Mode. | | |
| Param | KeyIdx | The index number of TPK, with a value range of [1-100]。 | |
| DataIn【input】 | DataIn is pan data。 | |
| ExpPinLen【input】 | A valid password length string that can be entered, an enumeration set of 0 to 12.  The application program enumerates all the allowed password lengths and separates each length with a "," symbol. If it allows entering 4-digit or 6-digit passwords and allows pressing confirm without a password, the string should be set to "0,4,6". Enumerating 0 means that you can return directly by pressing the confirm key without entering any numbers.  When Mode=0x50, the enumeration set of valid password length strings that can be entered is 0~16. | |
| Mode | Choose the format of PIN BLOCK  The format of PinBlock encrypted using the DES-ECB (3DES-ECB) algorithm:   |  |  | | --- | --- | | 0x00 | ISO9564 format 0 | | 0x01 | ISO9564 format 1 | | 0x02 | ISO9564 format 3 | | |
|  |  |
| TimeoutMs | Time out for entering PIN  Unit: milliseconds  Maximum value is 200000 | |
| PinBlock【output】 | 8 bytes, pointing to the generated PIN block.  When Mode is 0x10, the length of PinBlock is 16 bytes. | |
| return | RET\_OK  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM  OTHER | | success  PED not open  Invalid param  Please refer to the list of PED function return values |
| usage | During the input process, you can press the cancel key to abort the operation. | | |

### OsPedGetMac

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | int OsPedGetMac(int KeyIdx,  const unsigned char \*DataIn, | | |
|  | int DataInLen,  unsigned char \*Mac,  int Mode); | | |
| func | Perform the Mode specified operation on DataIn using the MAC key specified by KeyIdx. | | |
| param | KeyIdx | The index number of TAK, with a range of values from [1 to 100]. | |
| DataIn【input】 | The data that requires MAC operation has a length of 2032 bytes or less. | |
| DataInLen | The data length of MAC operations is automatically filled with "\ x00" when the length cannot be divided by 8 bytes. | |
| Mac【output】 | 8 bytes, MAC data. | |
| Mode | Divide DataIn into blocks of 8 bytes, in order of Block 1, Block 2, Block 3, etc.  **0x00**：Use the MAC key to perform DES/TDES encryption on BLOCK1, perform bitwise XOR on the encryption result with BLOCK2, and then use TAK to perform DES/TDES encryption, sequentially obtaining an 8-byte encryption result.  **0x01**：Perform bitwise XOR on BLOCK1 and BLOCK2, and the XOR result is bitwise XOR on BLOCK3. Repeat this process in sequence to obtain an 8-byte XOR result. Use TAK to perform DES/TDES encryption on the result.  **0x02**：According to the ANSIX9.19 specification, BLOCK1 is encrypted using TAK for DES (only the first 8 bytes of the key are taken), and the encryption result is XOR bit by bit with BLOCK2. Then, TAK is used for DES encryption, and the 8-byte encryption result is obtained sequentially. Finally, DES/TDES encryption is used.  Other values, default to China UnionPay MAC algorithm | |
| return | RET\_OK  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM  OTHER | | success  PED not open  Invalid param  Please refer to the list of PED function return values |
| usage |  | | |

### OsPedDes

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPedDes(int KeyIdx,  unsigned char \* InitVector,  const unsigned char \*DataIn,  int DataInLen,  unsigned char \*DataOut, | | |
|  | int Mode); | | |
| func | Use TDK to perform DES/TDES decryption or encryption operations on DataInLen length data, and output plaintext or ciphertext; A specified TDK can only be used for encryption or decryption operations. | | |
| param | KeyIdx | TDK index number, with a value range of [1~100]. | |
| InitVector【input】 | The initial vector for encryption and decryption operations, with a length of 8 bytes.  If *Mode*=0x02/0x03/0x04/0x05 , An initial vector is required, and when it is null, it defaults to  "\ x00 \ x00 \ x00 \ x00 \ x00 \ x00 \ x00 \ x00 \ x00";  If *Mode*=0x00/0x01，No initial vector is required, it can be NULL. | |
| DataIn【input】 | Point to the data that needs to be processed. | |
| DataInLen | The length of data to be calculated, in bytes, should be less than or equal to 1024;  If Mode is 0x00~0x03，Must be a multiple of 8; | |
| DataOut【output】 | Pointing to the data after operation. | |
| Mode | . 0x00：ECB decrypt  . 0x01：ECB encrypt  . 0x02：CBC decrypt  . 0x03：CBC encrypt | |
| return | RET\_OK  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM  OTHER | | success  PED not open  Invalid param  Please refer to the list of PED function return values |
| usage |  | | |

### OsPedGetKcv

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPedGetKcv(int KeyType,  int KeyIdx,  int KcvMode,  int KcvDataLen,  unsigned char \*KcvData,  unsigned char \*Kcv); | | |
| func | Obtain the KCV value of the key for key verification by both parties in the conversation: When the key type is not TIK, encrypt a segment of data with the specified key and algorithm, and return the first 3 bytes of the data ciphertext; When the key type is TIK, return an 8-byte KCV injected (calculated) together with TIK. | | |
| 参数 | KeyType | Key type：  . PED\_TLK  . PED\_TMK  . PED\_TAK  . PED\_TPK  . PED\_TDK  . PED\_TIK | |
| KeyIdx | Key index，：  . TLK ，only be 1。  . TMK can be 1~100。  . TWK can be 1~100。  . TIK can be 1~100。 | |
| KcvMode | KCV Verify Mode  0x00：Calculate the KCV of the key using the DES algorithm; | |
| KcvDataLen | The data length involved in calculating KCV shall not exceed 128 bytes and shall be divided by 8; When the key type is TIK, it can be 0; | |
| KcvData【input】 | The data involved in calculating KCV can be NULL when the key type is TIK. | |
| Kcv【output】 | KCV value, length 3. | |
| return | RET\_OK  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM  OTHER | | success  PED not open  Invalid param  Please refer to the list of PED function return values |
| usage |  | | |

### OsPedGetPinBlockEx

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPedGetPinBlockEx(int KeyIdx,  const unsigned char \*DataIn,  const char \*ExpPinLen,  int Mode,  unsigned long TimeoutMs,  unsigned char \*PinBlock,  PIN\_STAR\_DISPLAY pin\_star\_display); | | |
| func | Within the specified time limit, scan the keyboard and input a PIN of the specified length as ExpPinLenIn, and output the PIN block generated by the encryption algorithm specified by Mode. | | |
| 参数 | KeyIdx | The index number of TPK, with a value range of [1-100]. | |
| DataIn【input】 | DataIn is pan data。 | |
| ExpPinLen【input】 | A valid password length string that can be entered, an enumeration set of 0 to 12.  The application program enumerates all the allowed password lengths and separates each length with a "," symbol. If it allows entering 4-digit or 6-digit passwords and allows pressing confirm without a password, the string should be set to "0,4,6". Enumerating 0 means that you can return directly by pressing the confirm key without entering any numbers.  if Mode=0x50，An enumeration set of valid password length strings that can be entered, ranging from 0 to 16. | |
| Mode | Choose the format of PIN BLOCK  The format of PinBlock encrypted using the DES-ECB (3DES-ECB) algorithm: | |
|  |  | |  |  | | --- | --- | | . 0x00 | ISO9564 Format 0 | | . 0x01 | ISO9564 Format 1 | | . 0x02 | ISO9564 Format 3 | | |
| TimeoutMs | Time out for entering PIN  Unit: milliseconds  Maximum value is 200000 | |
| PinBlock【output】 | 8 bytes, pointing to the generated PIN block.  When Mode is 0x10, the length of PinBlock is 16 bytes. | |
| pin\_star\_display  【input】 | \* display callback function | |
| return | RET\_OK  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM  OTHER | | success  PED not open  Invalid param  Please refer to the list of PED function return values |
| suage | During the input process, you can press the cancel key to abort the operation. | | |

## DUKPT

### OsPedWriteTIK

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPedWriteTIK (const unsigned char \*KeyBlock); | | |
| func | Write a TIK key and choose to use KCV to verify the correctness of the key. | | |
| func | KeyBlock【input】 | 1 byte | Format： 0x03 |
| 1 byte | SrcKeyType：  . PED\_TLK |
| 1 byte | SrcKeyIdx：  . if SrcKeyType = PED\_TLK，SrcKeyIdx = 1；  . When writing plaintext，SrcKeyIdx = 0。 |
| 1 byte | DstKeyIdx。  DstKeyIdx = [1~100]。 |
| 7 bytes | Reserved field, random number |
| 1 byte | DstKeyType：  . PED\_TIK |
| 1 byte | DstKeyLen：8/16 |
| 24  bytes | DstKeyValue  Destination key plaintext/ciphertext |
| 1 byte | KcvMode：Currently only supports 0x00 and 0x01  **0x00**：No authentication  **0x01** ：Calculate DES/TDES encryption for 0x00 of 8 bytes to obtain the first 3 bytes of the ciphertext as KCV.  **0x02**：Firstly, perform odd parity on the plaintext key, and then perform DES/TDES encryption on "\ x12 \ x34 \ x56 \ x78 \ x90 \ x12 \ x 34 \ x56" to obtain the first three bytes of the ciphertext as KCV.  **0x03**：Pass in a string of data KcvData, use the source key to perform MAC operations on [destination key ciphertext+KcvData] in the specified mode, and obtain an 8-byte MAC, which is KCV. |
| 128  bytes | KcvData：  . if KcvMode is 0x00/0x01/0x02， random data；  .if KcvMode is 0x03，The first byte of KcvData is the length of the KCV data participating in the calculation, followed by the KCV data, and the second byte after the KCV data represents the MAC operation mode |
| 8 bytes | . if KcvMode = 0x00，random data。  . if KcvMode = 0x01/0x02/0x03，KcvValue points to the value of KCV. |
| 10  bytes | Initial KSN |
| return | ERR\_OK success  ERR\_DEV\_NOT\_OPEN ped not open  ERR\_INVALID\_PARAM Invalid param | | |
| OTHER Please refer to the list of PED function return values | | |
| usage | 1. If SrcKeyIdx = 0，The system assumes that DstKeyValue is the plaintext of the key, and does not determine SrcKeyType or SrcKeyIdx. Instead, it directly writes DstKeyValue to the DstKeyIdx position in the DstKeyType area. | | |

### OsPedGetPinDukpt

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPedGetPinDukpt (int GroupIdx,  const unsigned char \*DataIn,  const char \*ExpPinLen,  int Mode,  unsigned long TimeoutMs,  unsigned char \*Ksn,  unsigned char \*PinBlock); | | |
| func | Within the specified time limit, scan the PIN entered on the keyboard and output the PIN block generated using DUKPT's PIN key calculation. | | |
| param | GroupIdx | The DUKPT group index number has a range of values from [1 to 100]. | |
| DataIn【input】 | pan。 | |
| ExpPinLen【input】 | A valid password length string that can be entered, an enumeration set of 0 to 12.  The application program enumerates all the allowed password lengths and separates each length with a "," symbol. If it allows entering 4 or 6 digits of passwords and allows pressing confirm without a password, the string should be set to "0,4,6".  If the length of the enumeration is 0, it means that you can simply press the confirm key to return without entering any numbers. | |
| Mode | PIN block format：  . 0x20 ISO9564 format 0 ，KSN Not auto +[1](#bookmark541)  . 0x21 ISO9564 format 1 ，KSN Not auto +[1](#bookmark541)  . 0x22 ISO9564 format 3 ，KSN Not auto +[1](#bookmark541) | |
| TimeoutMs | The timeout time for entering a PIN, in milliseconds. Maximum value is 200000 | |
| Ksn【output】 | 10 bytes, pointing to the current KSN. | |
| PinBlock【output】 | 8 bytes, pointing to the generated PIN block. | |
| return | RET\_OK  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM  OTHER | | success  dev not open  invalid param  Please refer to the list of PED function return values |
| usage | When KSN does not increase by 1, a DUKPT PIN key can only calculate a PIN block once. | | |

### OsPedGetMacDukpt

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPedGetMacDukpt(int GroupIdx,  const unsigned char \*DataIn,  int DataInLen,unsigned char \*Mac,  unsigned char \*Ksn,int Mode); | | |
| func | Calculate MAC value using DUKPT key. | | |
| param | GroupIdx | The DUKPT group index number has a range of values from [1 to 100]. | |
| DataIn【input】 | Point to the data content that needs to calculate MAC. | |
| DataInLen | The length of the data, less than or equal to 2032 bytes, is automatically filled with "\ x00" when it is not an integer division of 8 bytes. | |
| Mac【output】 | Point to the obtained MAC. | |
| Ksn【output】 | Point to the current KSN. | |
| Mode | Divide DataIn into blocks of 8 bytes, in order of Block 1, Block 2, Block 3, etc.  **0x20**：Use MAC key to perform TDES encryption on BLOCK1, perform bitwise XOR on the encryption result with BLOCK2, and then use TAK to perform TDES encryption. Repeat this process to obtain an 8-byte encryption result.  **0x21**：Perform bitwise XOR on BLOCK1 and BLOCK2, and the XOR result is bitwise XOR on BLOCK3. Repeat this process in sequence to obtain an 8-byte XOR result. Use TAK to perform TDES encryption on the result.  **0x22**：According to the ANSIX9.19 specification, BLOCK1 is encrypted using TAK for DES (only the first 8 bytes of the key are taken), and the encryption result is XOR bit by bit with BLOCK2. Then, TAK is used for DES encryption, and 8 bytes of encryption result are obtained sequentially until the last time TDES encryption is used.  According to the ANSIX9.19 specification, BLOCK1 is encrypted using TAK for DES (only the first 8 bytes of the key are taken), and the encryption result is XOR bit by bit with BLOCK2. Then, TAK is used for DES encryption, and 8 bytes of encryption result are obtained sequentially until the last time TDES encryption is used.  Other values, default to China UnionPay MAC algorithm | |
| return | RET\_OK  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM  OTHER | | success  dev not open  invalid param  Please refer to the list of PED function return values |
| usage |  | | |

### OsPedDesDukpt

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPedDesDukpt(int GroupIdx,  int KeyVarType,  unsigned char \*InitVector,  int DataInLen,  unsigned char \*DataIn,  unsigned char \*DataOut,  unsigned char \*Ksn,  int Mode); | | |
| func | Use DUKPT key to encrypt and decrypt input data. | | |
| param | GroupIdx | The DUKPT group index number has a range of values from [1 to 100]. | |
| KeyVarType【input】 | **0x01**：Use DUKPT TDK key | |
| InitVector【input】 | Reserved. | |
| DataInLen | The length of data that needs to be encrypted and decrypted, less than or equal to 1024 | |
| DataIn【input】 | Point to the data that needs to be calculated | |
| DataOut【output】 | Pointing to the data after operation | |
| Ksn【output】 | Pointing to the current KSN, 10 bytes | |
|  | Mode | . 0x00：ECB DECRYPT  . 0x01：ECB ENCRYPT  . 0x02：CBC DECRYPT  . 0x03：CBC ENCRYPT | |
| return | RET\_OK  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM  OTHER | | SUCCESS  Dev not open.  Inavalid param.  Please refer to the list of PED function return values |
| usage |  | | |

### OsPedGetKsnDukpt

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPedGetKsnDukpt(int GroupIdx,  unsigned char \*Ksn); | | |
| func | Get the current KSN value. | | |
| param | GroupIdx | The DUKPT group index number has a range of values from [1 to 100]. | |
| Ksn【output】 | Pointing to the current KSN, 10 bytes | |
| return | RET\_OK  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM  OHTER | | Success  Dev not open  Invalid param  Please refer to the list of PED function return values |
| usage |  | | |

### OsPedIncreaseKsnDukpt

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPedIncreaseKsnDukpt (int GroupIdx); | | |
| func | Increase the KSN value for the specified DUKPT group. | | |
| param | GroupIdx | | The DUKPT group index number has a range of values from [1 to 100]. |
| return | RET\_OK SUCCESS | | |
| ERR\_DEV\_NOT\_OPEN dev not open | | |
| ERR\_INVALID\_PARAM | Invalid param | |
| OTHER | Please refer to the list of PED function return values | |
| usage |  |  | |

# LCD

## OsScrBrightness

|  |  |  |
| --- | --- | --- |
|  | void OsScrBrightness(int Brightness); | |
| func | Set screen brightness. | |
| param | Brightness | Brightness value, ranging from 0 to 10,  0: Turn off backlight.  10: Brightest.  The default value is 8, and other values have no action. |
| return |  | |
| usage | After the application exits, the settings of the application become invalid. | |

## OsScrGetSize

|  |  |  |
| --- | --- | --- |
|  | void OsScrGetSize(int \*Width,  int \*Height); | |
| func | Obtain the physical screen size of the LCD. | |
| param | Width【output】 | Width, measured in pixels. |
| Height【output】 | Height, measured in pixels. |
| return |  | |
| usage | The screen size is a read-only property, and this interface is only applicable to applications that do not use GUI. | |

# Keyboard

## Key Value Definition

|  |
| --- |
| EmvTermConfig\_t: |
| typedef enum KB\_KEYMAP\_e  {  EM\_KEY\_INVALID = -1,  EM\_KEY\_1,  EM\_KEY\_2,  EM\_KEY\_3,  EM\_KEY\_4,  EM\_KEY\_5,  EM\_KEY\_6,  EM\_KEY\_7,  EM\_KEY\_8,  EM\_KEY\_9,  EM\_KEY\_0,  EM\_KEY\_F1,  EM\_KEY\_F2,  EM\_KEY\_F3,  EM\_KEY\_F4,  EM\_KEY\_CLEAR,  EM\_KEY\_ENTER,  EM\_KEY\_UP,  EM\_KEY\_DOWN,  EM\_KEY\_POWER,  EM\_KEY\_CANCEL,  EM\_KEY\_MAX  }KB\_KEYMAP\_T; |
|  |

## Disp\_GetKey

|  |  |  |
| --- | --- | --- |
|  | int Disp\_GetKey(int ms); | |
| func | Get key value | |
| param | ms | Timeout for obtaining key values |
| return | KB\_KEYMAP\_T | |
| usage |  | |

## Disp\_ReleasKey

|  |  |  |
| --- | --- | --- |
|  | int Disp\_ReleasKey(); | |
| Func | Clear key value | |
| Param |  |  |
| Return | 0 success  OTHER fail | |
| usage |  | |

# Touch Screen

# Signature

暂未实现。

# Printer

The SDK supports physical printers and provides a unified printing interface.

## Return Value List

Table 11. 1 return value list

|  |  |  |
| --- | --- | --- |
|  | VALUE | EXPLAIN |
| ERR\_PRN\_BUSY | -3701 | Printer Busy |
| ERR\_PRN\_PAPEROUT | -3702 | Printer is out of paper. |
| ERR\_PRN\_WRONG\_PACKAGE | -3703 | Packet format error |
| ERR\_PRN\_OVERHEAT | -3704 | Printer overheating |
| ERR\_PRN\_OUTOFMEMORY | -3705 | To print data is to large. |
| ERR\_PRN\_OVERVOLTAGE | -3706 | Voltage too high. |

## Open and Close

### OsPrnOpen

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPrnOpen(unsigned int printertype,  const char\*targetname ); | | |
| func | Open the printer, including both physical and virtual printers. | | |
| param | printertype【input】 | PRN\_REAL：Physical printing equipment | |
| targetname【input】 | Reserved, fill in NULL | |
| return | RET\_OK  ERR\_DEV\_NOT\_EXIST  ERR\_INVALID\_PARAM  ERR\_DEV\_BUSY  ERR\_BATTERY\_ABSENT  ERR\_BATTERY\_VOLTAGE\_TOO\_LOW | | Success  The device does not exist.  Invalid param  Device occupied.  Battery not present.  Battery voltage too low. |
| usage |  | | |

### OsPrnReset

|  |  |  |
| --- | --- | --- |
|  | void OsPrnReset(void); | |
| func | Reset printing parameters and clear printing cache. | |
| param |  |  |
| return |  | |
| usage |  | |

### OsPrnClose

|  |  |  |
| --- | --- | --- |
|  | void OsPrnClose(void); | |
| Func | Close printer. | |
| Param |  |  |
| Return |  | |
| Usage | When the program exits, this function needs to be called to shut down the device. | |

## Printer Setting

### OsPrnSetGray

|  |  |  |
| --- | --- | --- |
|  | void OsPrnSetGray(int Level); | |
| Func | Set the printing grayscale. | |
| Param | Level | .Level=0, reserved.  . Level=1, default blackness, which is a regular print order.  . Level=2, reserved.  . Level=3, double-layer thermal printing.  . Level=4, double-layer thermal printing, higher blackness than 3.  Illegal values do not change the current settings. |
| Return |  | |
| Usage |  | |

## Layout Settings

### OsPrnSetSpace

|  |  |  |
| --- | --- | --- |
|  | void OsPrnSetSpace(int CharSpace,  int LineSpace); | |
| Func | Set the printing spacing. | |
| Param | CharSpace | Word spacing, in points |
| LineSpace | Row spacing, in points |
| Return |  | |
| Usage |  | |

### OsPrnSetReversal

|  |  |  |
| --- | --- | --- |
|  | void OsPrnSetReversal(int Attr); | |
| Func | Set font inversion properties to print normally by default. | |
| Param | Attr | 0: Normal  1: Reverse display |
| Return |  | |
| Usage | Invalid for reverse display of printed graphics | |

### OsPrnSetIndent

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPrnSetIndent (unsigned int Left,  unsigned int Right); | | |
| Func | Set the left and right margins | | |
| Param | Left【input】 | Left margin, valid range is [0100], default value is 0. | |
| Right【input】 | Right margin, valid range is [0100], default value is 0. | |
| Return | RET\_OK  ERR\_INVALID\_PARAM | | Success  Invalid param |
| Usage |  | | |

### OsPrnCheck

|  |  |  |
| --- | --- | --- |
|  | int OsPrnCheck(void); | |
| Func | Query the current status of the printer. | |
| Param |  |  |
| Return | RET\_OK  ERR\_PRN\_BUSY  ERR\_PRN\_PAPEROUT  ERR\_PRN\_OVERHEAT | The printer is working properly  Printer busy.  Printer is out of paper.  Printer overheating |
| Usage |  | |

### 11.4.5 OsPrnSetPrintParams

|  |  |  |
| --- | --- | --- |
|  | void OsPrnSetPrintParams(int FronSize,  int DoubleWidth,  int DoubleHeight,  int AlignType); | |
| Func | Set the font size for printing. | |
| Param | FronSize | Font size, can be set to 12, 16, 24 |
| DoubleWidth | Font width doubling, can be set from 1 to 4 |
| DoubleHeight | Font height doubled, can be set from 1 to 4 |
| AlignType | ALIGN\_TYPE\_LEFT 0  ALIGN\_TYPE\_CENTER 1  ALIGN\_TYPE\_RIGHT 2 |
| Return |  | |
| Usage |  | |

### 11.4.6 OsPrnFeed

|  |  |  |
| --- | --- | --- |
|  | void OsPrnFeed(int Pixel); | |
| Func | Within the print buffer, execute several dot lines for paper feeding. | |
| Param | Pixel | Number of dots and lines to execute paper feeding:  >0: Paper feed  =0: No action |
| Return |  | |
| Usage |  | |

### 11.4.6 OsPrnPrintf

|  |  |  |
| --- | --- | --- |
|  | void OsPrnPrintf(const char \*Str, ...); | |
| Func | Format the output string into the print buffer. | |
| Param | Str【input】 | Pointer to the string to be printed |
| Return |  | |
| Usage | Support variable parameters; | |

### OsPrnRawData

|  |  |  |
| --- | --- | --- |
|  | int OsPrnRawData (const char \*data, int len); | |
| Func | Print the formatted data directly. | |
| Param | data【input】 | Pointer to the string to be printed. |
| len【input】 | The length of the string pointer to be printed. |
| Return | RET\_OK  ERR\_INVALID\_PARAM  ERR\_DEV\_NOT\_OPEN  ERR\_PRN\_PAPEROUT  ERR\_PRN\_OVERHEAT  ERR\_PRN\_OVERVOLTAGE  ERR\_PRN\_BUSY | The printer is working properly  Invalid param  Open fail.  Printer is out of paper.  Printer overheating  Voltage too high.  Printer busy. |
| Usage | 1. After calling this function, the printer will print and will not return until the printing is completed;  2. After the printing is completed, this function will return the printing status, so there is no need to query the printing status again; | |

### OsPrnPutImage

|  |  |  |
| --- | --- | --- |
|  | void OsPrnPutImage(const unsigned char \*LogoPath); | |
| Func | Output graphics to the print buffer. | |
| Param | LogoPath【input】 | The path to the logo file to be printed. The image needs to be a monochromatic BMP image, with a width not exceeding 384. |
| Return |  | |
| Usage | The steps for generating graphic data are as follows:  1. Draw logo: You can use the "Start/Program/Accessories/Draw" tool in Windows to draw graphics, and the graphics should be saved in "monochrome, BMP format"; | |

### OsPrnStart

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPrnStart(void); | | |
| Func | Start the printer and print out the data in the buffer. | | |
| Param |  |  | |
| Return | RET\_OK  ERR\_PRN\_BUSY  ERR\_PRN\_PAPEROUT  ERR\_PRN\_WRONG\_PACKAGE  ERR\_PRN\_OVERHEAT  ERR\_PRN\_OUTOFMEMORY | | The printer is working properly  Printer busy.  Printer is out of paper.  Voltage too high  Printer overheating |
| Usage | 1. After calling this function, the printer will print and will not return until the printing is completed;  2. After the printing is completed, this function will return the printing status, so there is no need to query the printing status again; | | |

### OsPrnClrBuf

|  |  |  |
| --- | --- | --- |
|  | void OsPrnClrBuf(void); | |
| Func | Clear print buffer | |
| Param |  |  |
| Return |  | |
| Usage |  | |

### OsPrnSetParam

|  |  |  |
| --- | --- | --- |
|  | int OsPrnSetParam(unsigned int cmd); | |
| Func | Set whether the printer pre feeds paper | |
| Param | cmd 【input】 | 1: Printing without pre feeding paper  2: Printing with pre fed paper |
| Return | RET\_OK  ERR\_INVALID\_PARAM | Success  Invalid param |
| Usage | The function only affects the printer's pre feeding function | |

# Word bank

The system currently only supports default font libraries.

# Code

The encoding uses UTF-8 as the system default encoding and provides an encoding conversion interface.

## Encoding conversion

### OsCodeConvert

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsCodeConvert (const char \*FromCharset,  const char \*ToCharset,  const char \*InBuf,  char \*OutBuf,  unsigned int LenOut); | | |
| Func | Character encoding conversion | | |
| Return | FromCharset 【input】 | Original character encoding | |
| ToCharset 【input】 | Target character encoding | |
| InBuf 【input】 | The original encoded string ends with '\ 0'; Unicode encoding needs to end with '\ 0 \ 0'. | |
| OutBuf 【output】 | The converted target encoding string | |
| LenOut 【input】 | OutBuf array size, at least 1.5 times the size of InBuf array | |
| Return | >=0  ERR\_INVALID\_PARAM | | Conversion successful, return the length of the converted string  Invalid param |
| Usage | Supports conversion between codes as follows:  ISO-8859-(1,2,3,4,5,6,7,8,9,10,11, 13,14,15,16)  cp(850,874,932,  1250,1251,1252,1253,1254,1255,1256,1257,1258)  GBK/GB18030(2 bytes part)  BIG5  SHIFT\_JIS  EUC-KR  UNICODE  UTF-8  Notes:  1. It is only recommended to use the above encoding for conversion to UTF-8 encoding, as conversion between other encodings is likely to fail; | | |

# Magnetic stripe reader

The SDK provides the function of magnetic stripe readers to read magnetic stripe data and provides a unified reading interface.

## Return value list

Table 14. 1 return value list

|  |  |  |
| --- | --- | --- |
|  | Value | Explain |
| ERR\_MSR\_FAILED | -2701 | Swipe card failed |
| ERR\_MSR\_HEADERR | -2702 | No starting character found |
| ERR\_MSR\_ENDERR | -2703 | No ending character found |
| ERR\_MSR\_LRCERR | -2704 | LRC verification error |
| ERR\_MSR\_PARERR | -2705 | Data verification error in a certain bit of the magnetic track |
| ERR\_MSR\_NOT\_SWIPED | -2706 | No card swiping |
| ERR\_MSR\_NO\_DATA | -2707 | No data |
| ERR\_MSR\_END\_ZEROERR | -2708 | Data format error |
| ERR\_MSR\_PED\_DECRYPTERR | -2709 | PED decrypt error |
| ERR\_MSR\_NO\_TRACK\_ERR | -2710 | No corresponding magnetic track detected on the magnetic card |

## Data Struct

Track information structure: records the information and status of each track read. 

ST\_MSR\_DATA:

typedef struct

{

unsigned char TrackData[256];/\* Decoded bitstream \*/

int DataLen;/\* Track data length \*/

int Status;/\* Track data status, equal to 0 indicates successful data reading; Other values indicate read failure.\*/

}ST\_MSR\_DATA;



## Magnetic reader api

### OsMsrOpen

|  |  |  |
| --- | --- | --- |
|  | int OsMsrOpen(void); | |
| Func | Open the magnetic stripe reader device. | |
| Param |  |  |
| Return | |  |  | | --- | --- | | RET\_OK | Success | | ERR\_DEV\_NOT\_EXIST | Dev not exist. | | ERR\_DEV\_BUSY | Dev is busy. | | |
|  | ERR\_DEV\_NOT\_OPEN Dev not open. | |
| Usage | Before operating the magnetic stripe card reader, it is necessary to first call OsMsrOpen() to turn on the device, otherwise the subsequent related functions will not work. | |

### OsMsrClose

|  |  |  |
| --- | --- | --- |
|  | void OsMsrClose(void); | |
| Func | Close the magnetic stripe reader device. | |
| Param |  |  |
| Return |  | |
| Usage | When the program exits, this function needs to be called to shut down the device. | |

### OsMsrReset

|  |  |  |
| --- | --- | --- |
|  | void OsMsrReset(void); | |
| Func | Soft reset the magnetic stripe card reader and clear the read magnetic stripe card data. | |
| Param |  |  |
| Return |  | |
| Usage |  | |

### OsMsrSwiped

|  |  |  |
| --- | --- | --- |
|  | int OsMsrSwiped(void); | |
| Func | Check for card swiping. | |
| Param |  |  |
| Return | TRUE Card swiped  FALSE Not swiped card  ERR\_DEV\_NOT\_OPEN Dev not open. | |
| Usage | 1.Regardless of whether the magnetic stripe is brushed or not, this function will immediately return;  2.After calling OsMsrOpen(), OsMsrRead(), or OsMsrReset(), the status of the brushed magnetic stripe will be cleared. | |

### OsMsrRead

|  |  |  |
| --- | --- | --- |
|  | int OsMsrRead(ST\_MSR\_DATA \*Track1,  ST\_MSR\_DATA \*Track2,  ST\_MSR\_DATA \*Track3); | |
| Func | Read magnetic stripe card data. | |
| Param | Track1【output】 | Track 1 data |
| Track2【output】 | Track 2 data |
| Track3【output】 | Track 3 data |
| Return | RET\_OK Read Success | |
| ERR\_MSR\_NOT\_SWIPED Not Swiped | |
| ERR\_INVALID\_PARAM Invalid param | |
| ERR\_DEV\_NOT\_OPEN Dev not open | |
| Usage | If a certain track data is not needed, the pointer corresponding to that track can be set to NULL, and the data for that track will not be output. | |

# IC Card Reader

## Api Return List

Table 15. 1 Api Return List

|  |  |  |
| --- | --- | --- |
|  | VALUE | EXPLAIN |
| ERR\_SCI\_HW\_NOCARD | -2800 | No Card |
| ERR\_SCI\_HW\_STEP | -2801 | Not initialized during data exchange, not powered on during hot reset. |
| ERR\_SCI\_HW\_PARITY | -2802 | Parity check failed |
| ERR\_SCI\_HW\_TIMEOU T | -2803 | Timeout |

## Data Struct

### 15.2.1 APDU Request Struct



ST\_APDU\_REQ:

typedef struct

{

Unsigned char Cmd[4]; /\*CLA, INS, P1, P2\*/

int LC; /\* DataIn length \*/

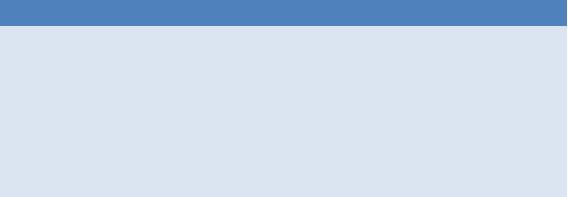
unsigned char DataIn[512]; /\*data\*/

int LE; /\* Expected length of returned data \*/

}ST\_APDU\_REQ;



### 15.2.2 APDU Response Struct





ST\_APDU\_RSP:

typedef struct

{

Int LenOut;

unsigned char DataOut[512];

unsigned char SWA;

unsigned char SWB;

}ST\_APDU\_RSP;



/\* Data pointer returned from IC card \*/

/\*IC card status word 1\*/

/\* IC card status word 2\*/

/\* The length of the returned data \*/

## API

### OsIccOpen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | int OsIccOpen(int Slot); | | | |
| Func | Open the IC card reader | | | |
| Param | Slot | IC Slot No.：  ICC\_USER\_SLOT ICC\_SAM1\_SLOT ICC\_SAM2\_SLOT ICC\_SAM3\_SLOT  ICC\_SAM4\_SLOT  Currently supports user cards | | User Card  SAM Slot 1  SAM Slot 2  SAM Slot 3  SAM Slot 4 |
| Param | RET\_OK  ERR\_DEV\_NOT\_EXIST  ERR\_DEV\_BUSY | | Open Success  Dev not exist  Dev is busy | |
| Usage |  | | | |

### OsIccDetect

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsIccDetect(int Slot); | | |
| Func | Check if the specified card holder has a card inserted. | | |
| Param | Slot | IC Slot No.：  ICC\_USER\_SLOT ICC\_SAM1\_SLOT ICC\_SAM2\_SLOT ICC\_SAM3\_SLOT  ICC\_SAM4\_SLOT | User Card  SAM Slot 1  SAM Slot 2  SAM Slot 3  SAM Slot 4 |
| Return | RET\_OK  OTHER | Card insertion  Please refer to the list of function return list. | |
| Usage | Regardless of whether there is a card in the card holder, this function immediately returns. | | |

### OsIccInit

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsIccInit(int Slot,  unsigned long Option,  unsigned char \*Atr); | | |
| Func | Init IC card reader. | | |
| Param | Slot | IC Slot No.:  ICC\_USER\_SLOT ICC\_SAM1\_SLOT ICC\_SAM2\_SLOT ICC\_SAM3\_SLOT  ICC\_SAM4\_SLOT | User Card  SAM Card 1 Slot  SAM Card 2 Slot  SAM Card 3 Slot  SAM Card 4 Slot |
| Option | (Bit 0~1) Card voltage selection:  00-5V  01-1.8V  10-3V  (Bit 2) Reserve 0  (bits 3-4) Reserve 0  (Bit 5) indicates the supported specification types:  0- EMV specification  1- ISO7816 specification  If this mark indicates that the specification is EMV mode, then the power on rate mark is invalid and the standard rate is used by default.  (Bit 6~31) Reserve 0  Option is set to 0 by default (i.e. 5V, following EMVx) | |
|  |
| Atr【output】 | 1. Answer To Reset, the card can return up to  Return 34 bytes of response data.  2. Its content is:  Reset response content length (1 byte)+reset response content | |
| Return | RET\_OK  OHTER | Success  Please refer to the list of function return list. | |
| Usage | The Atr output buffer should allocate at least 34 bytes of space; | | |

### OsIccExchange

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsIccExchange(int Slot,  int CtrlFlag,  const ST\_APDU\_REQ \*ApduReq,  ST\_APDU\_RSP \*ApduRsp); | | |
| Func | Interact commands with the IC card. | | |
| Param | Slot | IC Slot No.：  ICC\_USER\_SLOT ICC\_SAM1\_SLOT ICC\_SAM2\_SLOT ICC\_SAM3\_SLOT  ICC\_SAM4\_SLOT | User Card  SAM Card 1 Slot  SAM Card 2 Slot  SAM Card 3 Slot  SAM Card 4 Slot |
| CtrlFlag | Reserve | |
| ApduReq【input】 | Send command data structure to IC card | |
| ApduRsp【output】 | Receive data structure from IC card response | |
| Return | RET\_OK  OHTER | Success  Please refer to the list of function return list. | |
| Usage |  | | |

### 15.3.5 OsIccTransfer

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsIccTransfer(int Slot,  int CtrlFlag, const unsigned char,  \*pucTxBuff,  int iTxLen,  unsigned char \*pucRxBuff,  int \*piRxLen); | | |
| Func | Interact with the IC card using the Apdu command. | | |
| Param | Slot | IC Slot No.：  ICC\_USER\_SLOT ICC\_SAM1\_SLOT ICC\_SAM2\_SLOT ICC\_SAM3\_SLOT  ICC\_SAM4\_SLOT | User Card  SAM Card 1 Slot  SAM Card 2 Slot  SAM Card 3 Slot  SAM Card 4 Slot |
| CtrlFlag | Reserve | |
| pucTxBuff【input】 | Data buffer to be transmitted | |
| iTxLen 【input】 | The length of data to be transmitted, in bytes | |
| pucRxBuff【output】 | Buffer for receiving card response data | |
| piRxLen【output】 | Received card data length | |
| Return | RET\_OK Success  OTHER Please refer to the list of function return list. | | |

### OsIccClose

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsIccClose(int Slot); | | |
| Func | Close the IC card reader | | |
| Param | Slot | IC Slot No.：  ICC\_USER\_SLOT ICC\_SAM1\_SLOT ICC\_SAM2\_SLOT ICC\_SAM3\_SLOT  ICC\_SAM4\_SLOT | User Card  SAM Card 1 Slot  SAM Card 2 Slot  SAM Card 3 Slot  SAM Card 4 Slot |
| Param | RET\_OK  OHTER | Success  Please refer to the list of function return list. | |
| Usage |  | | |

# NFC card reader

This chapter mainly describes the application programming interface of non-contact IC card readers that comply with ISO14443 and EMV Contactless Book D V2.1 specifications.

## Return Value List

|  |  |  |  |
| --- | --- | --- | --- |
|  | Value | | Explain |
| PCD\_ERR\_PAR\_FLAG | -2901 | | Verification error |
| PCD\_ERR\_CRC\_FLAG | -2902 | | CRC Error |
| PCD\_ERR\_WTO\_FLAG | -2903 | | Time out or no card |
| PCD\_ERR\_COLL\_FLAG | -2904 | | Multi card conflict |
| PCD\_ERR\_ECD\_FLAG | -2905 | | Frame format error |
| PCD\_ERR\_EMD\_FLAG | -2906 | | [Disturb](file:///C:\Program%20Files\baidu-translate-client\resources\app.asar\app.html) |
| PCD\_ERR\_COM\_FLAG | -2907 | | Chip malfunction, unable to communicate correctly |
| PCD\_ERR\_AUT\_FLAG | -2908 | | M1 authentication error |
| PCD\_ERR\_TRANSMIT\_ FLAG | -2909 | | transmission error |
| PCD\_ERR\_PROTOCOL\_ FLAG | -2910 | | protocol error |
| PCD\_ERR\_PARAMFILE \_FLAG | | -2911 The configuration file does not exist | |
| PCD\_ERR\_USER\_CANCEL-2912 User cancel | | | |
| PCD\_ERR\_CARRIER\_O BTAIN\_FLAG | | -2913 Carrier not obtained | |
| PCD\_ERR\_CONFIG\_FL AG | | -2914 | Failed to configure register |
| PCD\_ERR\_RXLEN\_EXC EED\_FLAG | | -2915 | The length of the card's returned data exceeds the set receiving length |
| PCD\_ERR\_NOT\_ALLOW ED\_FLAG | | -2951 Parameter error or value not allowed | |
| PCD\_CHIP\_ABNORMAL -2952 Chip not present or abnormal | | | |
| PCD\_CHIP\_NOT\_OPEN -2953 Module not opened | | | |
| PCD\_CHIP\_CARDEXIST -2954 Card not removed | | | |
| PCD\_ERR\_NOT\_IDLE\_FLAG -2955 The card is not in idle state | | | |
| PCD\_ERR\_NOT\_POLLI  NG\_FLAG | | -2956 Card not Polling | |
| PCD\_ERR\_NOT\_WAKE  UP\_FLAG | | -2957 Card not awakened | |
| PCD\_ERR\_NOT\_ACTIV E\_FLAG | | -2958 Card not activated | |
| PCD\_ERR\_NOT\_SUPPORT -2959 Chip not supported | | | |

## Data Struct

Please refer to 15.2.1 and 15.2.2 in the IC card reader chapter for the request data structure and response data structure.

## API

### OsPiccOpen

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsPiccOpen(void); | | |
| Func | Power on the PCD module to put it into a ready to work state. | | |
| Param |  |  | |
| Return | 0  ERR\_BATTERY\_VOLTAG E\_TOO\_LOW  ERR\_BATTERY\_ABSENT  OTHER | | Open Success  Battery voltage too low  No battery  Please refer to the list of function return list. |
| Usage |  | | |

### OsPiccClose

|  |  |  |
| --- | --- | --- |
|  | int OsPiccClose(void); | |
| Func | PCD module power-off processing. | |
| Param |  |  |
| Return | 0 Close success  OTHER Please refer to the list of function return list. | |
| Usage |  | |

### OsPiccResetCarrier

|  |  |  |
| --- | --- | --- |
|  | int OsPiccResetCarrier(void); | |
| Func | Carrier reset. | |
| Param |  |  |
| Return | 0 Carrier reset success  OTHER Please refer to the list of function return list. | |
| Usage | After performing a carrier reset operation on the non-contact IC card reader and calling this interface, the status of the card in the RF field will change to idle state. | |

### OsPiccPoll

|  |  |  |
| --- | --- | --- |
|  | int OsPiccPoll(char\*pcPiccType,  unsigned char\*pucATQx); | |
| Func | For the time being, there are only two types of rotating cards, "A" and "B", for card searching and power on. | |
| param | pcPiccType【output】 | Card Type：  . “A ” - A card  . “B ” - B card |
| pucATQx 【output】 | Card response：  . A: 2 bytes  . B: 12 bytes |
| Return | 0 Searching success  OTHER Please refer to the list of function return list. | |
| Usage |  | |

### 16.3.5 OsPiccTransfer

|  |  |  |
| --- | --- | --- |
|  | int OsPiccTransfer(const unsigned char\*pucTxBuff,  int iTxLen,  unsigned char\*pucRxBuff,  int \*piRxLen); | |
| Func | Transparent transmission/reception function. | |
| Param | pucTxBuff【input】 | Data buffer to be transmitted |
|  | iTxLen 【input】 | The length of data to be transmitted, in bytes |
| pucRxBuff【output】 | Buffer for receiving card response data |
| piRxLen【output】 | Number of received card data bytes |
| Return | 0 success  OTHER Please refer to the list of function return list. | |
| Usage |  | |

### OsPiccRemove

|  |  |  |
| --- | --- | --- |
|  | int OsPiccRemove (void); | |
| Func | Perform card transfer operation in EMV mode. | |
| Param |  |  |
| Return | 0 success  OTHER Please refer to the list of function return list. | |
| Usage |  | |

### OsPiccIsoCommand

|  |  |  |
| --- | --- | --- |
|  | int OsPiccIsoCommand(int cid,  ST\_APDU\_REQ\*ApduReq,  ST\_APDU\_RSP\*ApduRsp); | |
| Func | Send APDU format data to the card on the specified channel and receive a response. | |
| Param | cid 【input】 | Used to specify the logical channel number of the card. The values are all 0. |
| ApduReq 【input】 | Send command data structure to PICC card  [ST\_APDU\_REQ](#bookmark223) |
| ApduRsp 【output】 | Data structure returned from PICC card  [ST\_APDU\_RSP](#bookmark224) |
| Return | 0 success  OTHER Please refer to the list of function return list. | |
| Usage |  | |

### OsPiccOffCarrier

|  |  |
| --- | --- |
|  | int OsPiccOffCarrier(void); |
| Func | Turn off the carrier. |
| Param |  |
| Return | 0 success  OTHER Please refer to the list of function return list. |
| Usage | Performing a carrier shutdown operation on a non-contact IC card reader will change the status of the card in the RF field to a powered off state. |

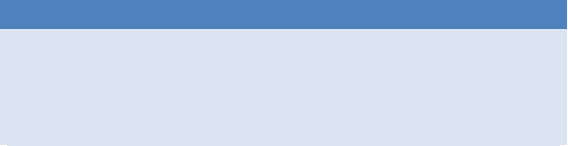
# Communication Port

POSIX interface that supports serial ports。

## POSIX API

### Open

Open the communication port.





int fd;

fd = open(“/dev/ttyAMA1”, O\_RDWR);

if(- 1 == fd){

perror(“Open uart error”);

}



### 17.1.2 Read

Read data from the communication port.



char buff[1024];

int Len = 1024;

intreadByte = read(fd, buff, Len);

### Write

Write data to the communication port (send).



char buffer[1024];

int Length = 1024;

intnByte;

nByte = write(fd, buffer, Length);

### Close

Close he communication port serial port.

close(fd);

### Query communication port buffer data



int remain;

int count;

ioctl(fd, TIOCOUTQ, &remain); /\* Query the remaining bytes in the send buffer.\*/

ioctl(fd, TIOCINQ, &count);/\* Query the remaining bytes in the receive buffer.\*/

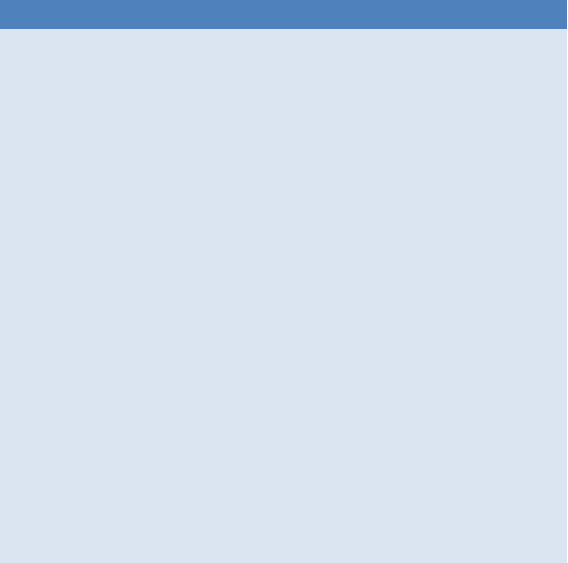


### Clear communication port buffer data



tcflush(fd, TCIOFLUSH);

### Set serial port configuration parameters



/\* Set the baud rate, data bits, checksum bits, and stop bits for UART。\*/

int SetTermios (int fd, intnSpeed, intnBits, char cEvent, intnStop) {

struct termios newtio, oldtio;

/\* Obtain UART configuration information \*/

if (tcgetattr (fd, &oldtio) != 0)

{

printf("Get serial error\n");

return - 1;

}

/\* Initialize configuration information \*/

bzero (&newtio, sizeof (newtio));

newtio.c\_cflag |= CLOCAL | CREAD;

newtio.c\_cflag &= ~CSIZE;

/\* Close soft flow control \*/

newtio.c\_iflag &= ~(ICRNL | IXON | IXOFF);

/\* Close hard flow control \*/

newtio.c\_cflag &= ~CRTSCTS;

/\* set data bit \*/

switch (nBits)

{

case 7:

newtio.c\_cflag |= CS7;

break;

case 8:

newtio.c\_cflag |= CS8;

break;

}

/\* Set checksum \*/

switch (cEvent)

{

case 'o':

newtio.c\_cflag |= PARENB;

newtio.c\_cflag |= PARODD;

newtio.c\_iflag |= (INPCK | ISTRIP);

break;

case 'e':

newtio.c\_iflag |= (INPCK | ISTRIP);

newtio.c\_cflag |= PARENB;

newtio.c\_cflag &= ~PARODD;

break;

case 'n':

newtio.c\_cflag &= ~PARENB;

break;

}

/\* set baud rate \*/

switch (nSpeed)

{

case 1200:

cfsetispeed (&newtio, B1200);

cfsetospeed (&newtio, B1200);

case 2400:

cfsetispeed (&newtio, B2400);

cfsetospeed (&newtio, B2400);

break;

case 4800:

cfsetispeed (&newtio, B4800);

cfsetospeed (&newtio, B4800);

break;

case 9600:

cfsetispeed (&newtio, B9600);

cfsetospeed (&newtio, B9600);

break;

case 19200:

cfsetispeed (&newtio, B19200);

cfsetospeed (&newtio, B19200);

break;

case 38400:

cfsetispeed (&newtio, B38400);

cfsetospeed (&newtio, B38400);

break;

case 57600:

cfsetispeed (&newtio, B57600);

cfsetospeed (&newtio, B57600);

break;

case 115200:

cfsetispeed (&newtio, B115200);

cfsetospeed (&newtio, B115200);

/\* Set stop bit \*/

if (nStop == 1)

newtio.c\_cflag &= ~CSTOPB;

else if (nStop == 2)

newtio.c\_cflag |= CSTOPB;

/\* 设置停止位\*/

if (nStop == 1)

newtio.c\_cflag &= ~CSTOPB;

else if (nStop == 2)

newtio.c\_cflag |= CSTOPB;

/\* 设置等待时间和最小字节数\*/

newtio.c\_cc[VTIME] = 0;

newtio.c\_cc[VMIN] = 0;

/\* 清空发送缓冲区\*/

tcflush (fd, TCIFLUSH);

/\* 设置新的配置信息\*/

if ((tcsetattr (fd, TCSANOW, &newtio)) != 0)

{

printf("Set serial error\n");

return - 1;

}

return 0;

}

default:

printf ("Not support the speed %d\n", nSpeed);

cfsetispeed (&newtio, B9600);

cfsetospeed (&newtio, B9600);

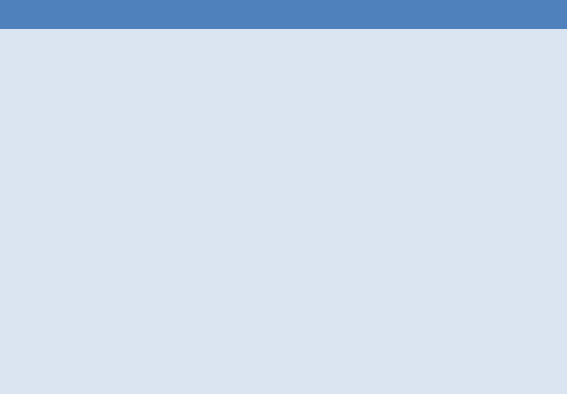
return - 1;

}

# Network communication

For network communication programming, support standard socket programming.

## Programming examples



/\* C language server-side code \*/

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

int main(void)

{

struct sockaddr\_in stSockAddr;

int SocketFD = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP); if(- 1 == SocketFD)

{

perror("can not create socket");

exit(EXIT\_FAILURE);

}

memset(&stSockAddr, 0, sizeof(stSockAddr));

stSockAddr.sin\_family = AF\_INET;

stSockAddr.sin\_port = htons(1100);

stSockAddr.sin\_addr.s\_addr = INADDR\_ANY;

if(- 1 == bind(SocketFD,(struct sockaddr \*)&stSockAddr,

sizeof(stSockAddr)))

{

perror("error bind failed");

close(SocketFD);

exit(EXIT\_FAILURE);

}

if(- 1 == listen(SocketFD, 10))

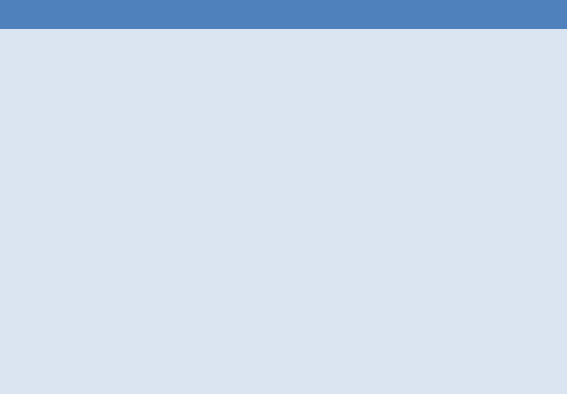
{

perror("error listen failed");

close(SocketFD);

exit(EXIT\_FAILURE);

}



for(;;)

{

int ConnectFD = accept(SocketFD, NULL, NULL);

if(0 > ConnectFD)

{

perror("error accept failed");

close(SocketFD);

exit(EXIT\_FAILURE);

}

/\* 执行读写操作 \*/

read(ConnectFD,buff,size)

if (- 1 == shutdown(ConnectFD, SHUT\_RDWR))

{

perror("can not shutdown socket");

close(ConnectFD);

exit(EXIT\_FAILURE);

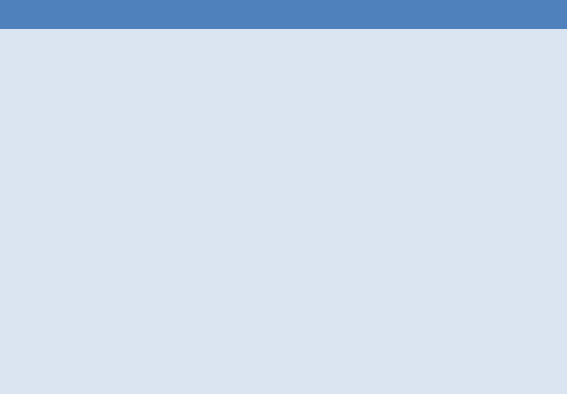
}

close(ConnectFD);

}

return EXIT\_SUCCESS;

}



/\* C language client code \*/

# include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

int main(void)

{

struct sockaddr\_in stSockAddr;

int Res;

int KeepAlive = 1;

int SocketFD = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP); if (- 1 == SocketFD)

{

perror("cannot create socket");

exit(EXIT\_FAILURE);

}

Res = setsockopt(SocketFD, SOL\_SOCKET, SO\_KEEPALIVE, (void \*)&KeepAlive, sizeof(KeepAlive));

if (- 1 == Res)

{

perror("cannot setkeepalive");

exit(EXIT\_FAILURE);

}

memset(&stSockAddr, 0, sizeof(stSockAddr));

stSockAddr.sin\_family = AF\_INET;

stSockAddr.sin\_port = htons(1100);

Res = inet\_pton(AF\_INET, "<192.168.1.3> ", &stSockAddr.sin\_addr); if (0 > Res)

{

perror("error: first parameter is not a valid address family"); close(SocketFD);

exit(EXIT\_FAILURE);

}

else if (0 == Res)

{

perror("char string (second parameter does not contain valid ipaddress)");

close(SocketFD);

exit(EXIT\_FAILURE);

}

if (- 1 == connect(SocketFD, (struct sockaddr \*)&stSockAddr, sizeof(stSockAddr)))

{

perror("connect failed");

close(SocketFD);

exit(EXIT\_FAILURE);

}

/\* Perform read and write operations \*/

shutdown(SocketFD, SHUT\_RDWR);

close(SocketFD);

return EXIT\_SUCCESS;

}

## UDP Program

UDP example:

#include <stdio.h>

#include <errno.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <unistd.h> /\*

#include <stdlib.h>

int main(void)

{

int sock = socket(PF\_INET, SOCK\_DGRAM, IPPROTO\_UDP); struct sockaddr\_in sa;

char buffer[1024];

ssize\_trecsize;

socklen\_t fromlen;

memset(&sa, 0, sizeof sa);

sa.sin\_family = AF\_INET;

sa.sin\_addr.s\_addr = INADDR\_ANY;

sa.sin\_port = htons(7654);

fromlen = sizeof(sa);

if (- 1 == bind(sock,(struct sockaddr \*)&sa, sizeof(sa)))

{

perror("error bind failed");

close(sock);

exit(EXIT\_FAILURE);

}

for (;;)

{

printf ("recv test....\n");

recsize = recvfrom(sock, (void \*)buffer, sizeof(buffer), 0, (struct sockaddr \*)&sa, &fromlen);

if (recsize < 0) {

fprintf(stderr, "%s\n", strerror(errno));

exit(EXIT\_FAILURE);

}

printf("recsize: %z\n ", recsize);

sleep(1);

printf("datagram: %.\*s\n", (int)recsize, buffer);

}

}

#include <stdlib.h>

#include <stdio.h>

#include <errno.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <unistd.h>

#include <arpa/inet.h>

int main(int argc, char \*argv[])

{

int sock;

struct sockaddr\_in sa;

intbytes\_sent;

char buffer[200];

strcpy(buffer, "hello world!");

sock = socket(PF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

if (- 1 == sock)

{

printf("Error Creating Socket");

exit(EXIT\_FAILURE);

}

memset(&sa, 0, sizeof sa);

sa.sin\_family = AF\_INET;

sa.sin\_addr.s\_addr = inet\_addr("<127.0.0.1> ");

sa.sin\_port = htons(7654);

bytes\_sent = sendto(sock, buffer, strlen(buffer), 0,(struct sockaddr\*)&sa, sizeof sa);

if (bytes\_sent < 0) {

printf("Errorsending packet: %s\n", strerror(errno));

exit(EXIT\_FAILURE);

}

close(sock);

return 0;

}



# GPRS

The SDK supports GPRS and CDMA wireless networks, and provides a series of APIs for developers to use in enabling and configuring these wireless modules.

When the highest network standard supported by the SIM card used is different from the highest network standard supported by the wireless module, the module initialization time may become longer.

## Return Values List

Table19.1 Return Valuse List

|  |  |  |
| --- | --- | --- |
|  | Value | Explain |
| PPP\_LOGINING | 1 | PPP Logging in |
| PPP\_LOGOUTING | 2 | PPP Logging out |
| WL\_CSD\_READY | 3 | CSD dial-up service ready |
| WL\_GPRS\_CSD\_READY | 4 | GPRS and CSD dial-up services ready |
| ERR\_WL\_POWER\_ONING | -3501 | Wireless module powered on |
| ERR\_WL\_POWER\_OFF | -3502 | Wireless module not powered on |
| ERR\_WL\_NOT\_INIT | -3503 | Module not initialized |
| ERR\_WL\_NEEDPIN -3504 SIM card requires a PIN   |  |  |  | | --- | --- | --- | | ERR\_WL\_RSPERR | -3505 | Module response error | | ERR\_WL\_NORSP | -3506 | Module not responding | | ERR\_WL\_NEEDPUK | -3507 | SIM card requires PUK | | ERR\_WL\_WRONG\_PIN | -3508 | SIM card PIN error | | ERR\_WL\_NOSIM | -3509 | No SIM card inserted | | ERR\_WL\_NOREG | -3510 | Unable to register to the network | | ERR\_WL\_AUTO\_RST | -3511 | Module automatic reset | | ERR\_WL\_BUF | -3512 | Wireless module memory error | | ERR\_WL\_GET\_SIGNAL | -3513 | Obtaining signal, please wait for 3 seconds | | ERR\_WL\_NOTYPE | -3514 | Module not recognized | | ERR\_WL\_PPP\_ONLINE | -3515 | Module PPP online, unable to sleep | | ERR\_WL\_ERR\_BUSY | -3516 | Module busy | | ERR\_WL\_SLEEP\_ONING | -3517 | The wireless module is entering sleep mode | | ERR\_WL\_SLEEP\_FAIL | -3518 | Wireless module failed to enter sleep mode | | ERR\_WL\_SIM\_FAILURE | -3519 | Wireless module failed to operate SIM card | | | |

## GPRS API

### OsWlLock

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsWlLock(void); | | |
| Func | Open the wireless module to obtain usage permissions for the wireless device/module. | | |
| Param |  | |  |
| Return | RET\_OK  ERR\_DEV\_BUSY | | Open success  Dev busy |
| ERR\_DEV\_NOT\_EXIST | Module/device not present | |
| ERR\_BATTERY\_VOLTA GE\_TOO\_LOW | Battery voltage too low | |
| ERR\_BATTERY\_ABSENT | Terminal battery not present | |
| Usage | Before calling OsWlInit(), OsWlLogin(), or OsWlLogout(), OsWlLock() must be called first;  After the operation is completed, OsWlUnLock() should be called to turn off the wireless module; | | |

### OsWlUnLock

|  |  |  |
| --- | --- | --- |
|  | void OsWlUnLock(void); | |
| Func | Release the usage rights of the wireless module. | |
| Param | \ |  |
| Return | \ | |
| Usage |  | |

### OsWlInit

|  |  |  |
| --- | --- | --- |
|  | int OsWlInit(const char \*SimPin); | |
| Func | Initialize wireless module | |
| Param | SimPin【input】 | The password string pointer of the SIM card, with a length of less than 50 bytes.  Can be NULL, indicating that no password is required. |
| Return | RET\_OK  ERR\_DEV\_NOT\_OPEN  ERR\_DEV\_NOT\_EXIST  ERR\_NO\_PORT    ERR\_WL\_NEEDPIN  ERR\_WL\_RSPERR  ERR\_WL\_NORSP  ERR\_WL\_NEEDPUK  ERR\_WL\_WRONG\_PIN  ERR\_WL\_NOSIM  ERR\_WL\_NOTYPE  ERR\_WL\_NOREG | Init Success  Module not open  Module not exist  Insufficient physical serial port on the termina  SIM need PIN  Module response error  Module no response  SIM need PUK  PIN error  No SIM  Module type not recognized  Unable to register to GPRS network |
| Usage | Before using this function, it is necessary to successfully call OsWlLock(); | |

### OsWlGetSignal

|  |  |  |
| --- | --- | --- |
|  | int OsWlGetSignal(void); | |
| Func | Obtain wireless signal strength. | |
| Param |  |  |
| Return | 0~5  ERR\_DEV\_NOT\_EXIST  ERR\_WL\_RSPERR  ERR\_WL\_POWER\_ONING | Signal strength level: 0 indicates no signal; 5 represents the strongest signal.  Wireless module not present  Module response error  The wireless module is in a power-off state |
| Usage | There is no need to call OsWlLock() before using this function;  When no wireless link is established, the signal value is directly obtained from the module through AT commands; | |

### OsWlCheck

|  |  |  |
| --- | --- | --- |
|  | int OsWlCheck(void); | |
| Func | Query the wireless link status. | |
| Param |  |  |
| Return | PPP\_LOGOUTING  PPP\_LOGINING  RET\_OK  ERR\_DEV\_NOT\_EXIST  ERR\_WL\_POWER\_ONING  ERR\_WL\_POWER\_OFF  ERR\_WL\_NOT\_INIT  ERR\_NET\_PASSWD  ERR\_NET\_LOGOUT  ERR\_NET\_IF | Breaking link  Creating link  Link created successfully  Wireless module not present  Wireless module powered on  Wireless module power-off  Module not initialized  Password error  Active disconnection of applications  Link unavailable, indicating that the link has not been established or disconnected |
| Usage |  | |

### OsWlLogin

|  |  |  |
| --- | --- | --- |
|  | int OsWlLogin(const char \*APN, | |
|  | const char \*Name,  const char \*Password,  long Auth,  int TimeOutMs,  int KeepAlive,  int ReserParam); | |
| Func | Log in to the wireless network and establish a wireless link. | |
| Param | APN【input】 | Wireless access point name, represented by an empty string "" or NULL when there is no APN |
| Name【input】 | User name, length cannot exceed 50 bytes;  When there is no username, use an empty string or NULL to represent it. |
| Password【input】 | Password, length cannot exceed 50 bytes; When there is no password, use an empty string or NULL to represent it. |
| Auth | Authentication algorithm, default to 0 |
| TimeOutMs | Time out, in milliseconds;  The value range is 0-3600000. |
| KeepAlive | Reserved, fill in 0 |
| ReserParam | Reserve parameters for extended use. |
| Return | PPP\_LOGINING  PPP\_LOGOUTING  RET\_OK  ERR\_DEV\_NOT\_EXIST  ERR\_DEV\_NOT\_OPEN  ERR\_INVALID\_PARAM  ERR\_WL\_POWER\_ONING  ERR\_WL\_POWER\_OFF  ERR\_WL\_NOT\_INIT  ERR\_NET\_PASSWD  ERR\_NET\_SERVER\_BUSY  ERR\_NET\_AUTH | Processing  Wireless module is logging out  Link established successfully  No wireless module  OsWlLock did not execute successfully  Invalid param  Wireless module powered on  Wireless module not powered on  Failed to initialize successfully  Passwork Error  Server busy, communication failed  Unable to connect to the radius server |
| Usage | 1.Before calling this function, please ensure that OsWlLock() is successfully called;  2. Before calling this function, please ensure that OsWlInit() is successfully called;  3.When TimeOutMs is 0, the system does not wait and returns immediately; | |

### OsWlLogout

|  |  |  |
| --- | --- | --- |
|  | int OsWlLogout(void); | |
| Func | Exit the wireless network and disconnect the wireless link. | |
| Param |  |  |
| Return | PPP\_LOGOUTING  ERR\_DEV\_NOT\_OPEN | Link disconnected  Module/device not open |
| Usage |  | |

### OsWlGetImei

|  |  |  |
| --- | --- | --- |
|  | int OsWlGetImei(char\* Imei); | |
| Func | Obtain IMEI. | |
| Param | Imei【output】 | The size of the Imei information buffer must be no less than 16 bytes |
| Return | RET\_OK Success  ERR\_INVALID\_PARAM Invalid param  ERR\_ACCESS\_DENY No access | |
| Usage |  | |

### OsWlGetImsi

|  |  |  |
| --- | --- | --- |
|  | int OsWlGetImsi(char\* Imsi); | |
| Func | Obtain IMSI. | |
| Param | Imsi【output】 | The size of the Imsi information buffer must be no less than 16 bytes |
| Param | RET\_OK Success  ERR\_INVALID\_PARAM Invalid param  ERR\_ACCESS\_DENY No access | |
| Usage |  | |

### OsWlGetIccid

|  |  |  |
| --- | --- | --- |
|  | int OsWlGetIccid(char\* Iccid); | |
| Func | Obtain ICCID。 | |
| Param | Iccid【output】 | The size of the Iccid information buffer must be no less than 16 bytes |
| Return | RET\_OK Success  ERR\_INVALID\_PARAM Invalid param  ERR\_ACCESS\_DENY No access | |
| Usage |  | |

### OsWlGetSimStatus

|  |  |  |
| --- | --- | --- |
|  | int OsWlGetSimStatus(int\* simStatus); | |
| Func | Obtain SIM card status. | |
| Param | simStatus【output】 | Sim card status, simStatus=1 card in place, other values not in place |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

### OsWlGetSimOperator

|  |  |  |
| --- | --- | --- |
|  | int OsWlGetSimOperator(char\* simOperator); | |
| Func | Obtain the operator. | |
| Parm | simOperator【output】 | Operator Name |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

### OsWlGetSignalStrength

|  |  |  |
| --- | --- | --- |
|  | int OsWlGetSignalStrength(void); | |
| Func | Obtain signal strength. | |
| Param |  |  |
| Return | Signal strength value | |
| Usage |  | |

### OsWlGetModuleREG

|  |  |  |
| --- | --- | --- |
|  | int OsWlGetModuleREG(int \*state, char\* lac, char\* cid); | |
| Func | Obtain the module's network registration status. | |
| Param | State【output】 | network registration status：  0 unregistered  1 Register local network  2 Registering  3 Registration rejected  4 Unknown  5 Roam |
| lac【output】 | location area code |
| cid【output】 | Base station number |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

### OsWlGetNetworkIp

|  |  |  |
| --- | --- | --- |
|  | int OsWlGetNetworkIp(char\* ipaddr); | |
| Func | Obtain IP. | |
| Param | ipaddr【output】 | IP assigned by the operator |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

### OsWlGetAutoConnectStatus

|  |  |  |
| --- | --- | --- |
|  | int OsWlGetAutoConnectStatus(int \*status); | |
| Func | Get automatic connection status. | |
| Param | status【output】 | Connection status, 1 system automatically connects to the network 0 system does not automatically connect to the network |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

### OsWlSetAutoConnectStatus

|  |  |  |
| --- | --- | --- |
|  | int OsWlSetAutoConnectStatus(int onoff); | |
| Func | Set automatic connection status. | |
| Param | onoff【input】 | Connection status, 1 system automatically connects to the network 0 system does not automatically connect to the network |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

### OsGetAirplaneMode

|  |  |  |
| --- | --- | --- |
|  | int OsGetAirplaneMode(int \*onoff); | |
| Func | Obtain the Airplane mode status. | |
| Param | onoff【output】 | Airplane mode status,  1 Airplane mode on  0 Airplane mode off |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

### OsSetAirplaneMode

|  |  |  |
| --- | --- | --- |
|  | int OsSetAirplaneMode(int onoff); | |
| Func | Set the Airplane mode status. | |
| Param | onoff【input】 | Airplane mode status,  1 Airplane mode on  0 Airplane mode off |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

### OsGetApnParams

|  |  |  |
| --- | --- | --- |
|  | int OsGetApnParams(char\* apn, char\* user,char\* password); | |
| Func | Get the Apn parameters. | |
| Param | apn【output】 | Apn，Cannot be NULL |
| user【output】 | UserName，Cannot be NULL |
| password【output】 | password，Cannot be NULL |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

### OsSetApnParams

|  |  |  |
| --- | --- | --- |
|  | int OsSetApnParams(char\* apn, char\* user,char\* password); | |
| Func | Set Apn parameters. | |
| Param | apn【input】 | Apn， Can be NULL |
| user【input】 | UserName， Can be NULL |
| password【input】 | Password， Can be NULL |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

# WiFi

Wi Fi supports Station mode of operation.

Station mode: Terminal connects to wireless access points such as wireless routers;

## Return Values List

Table 20. 1 Return Values list

|  |  |  |
| --- | --- | --- |
|  | Value | Explain |
| ERR\_MODE\_NOT\_SUPPORT | -3350 | Mode setting error |
| ERR\_WIFI\_POWER\_OFF | -3351 | Module not powered on |
| ERR\_NOT\_FOUND\_AP | -3352 | No AP found |
| ERR\_AUTH\_SEC\_MODE | -3353 | Authentication mode or encryption mode error |
| ERR\_WIFI\_BAD\_SIGNAL | -3354 | WiFi signal difference |
| RET\_CONNECTING | 1 | connecting |
| ERR\_EAP\_ID | -3359 | Certificate chain error or certificate verification failure |

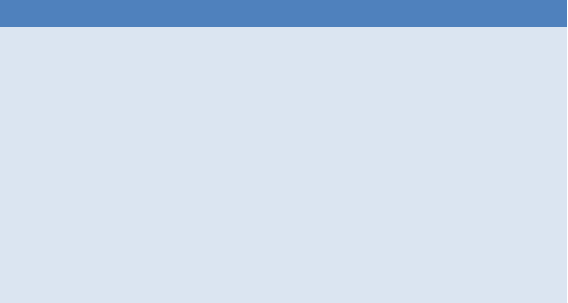
## List of encryption types

Table 20.2 List of encryption types

|  |  |  |
| --- | --- | --- |
|  | Value | Explain |
| PARE\_CIPHERS\_NONE | 0x00000000 | no encryption |
| PARE\_CIPHERS\_WEP64 | 0x00000001 | WEP 40bit Key |
| PARE\_CIPHERS\_WEP128 | 0x00000002 | WEP 104bit Key |
| PARE\_CIPHERS\_WEPX | 0x00000004 | WEP encryption, unknown key bit |
| PARE\_CIPHERS\_CCMP | 0x00000010 | AES encryption method |
| PARE\_CIPHERS\_TKIP | 0x00000020 | TKIP encryption method |

## Data Struct

Certification mode:



WIFI\_AUTH\_MODE

enum WIFI\_AUTH\_MODE{

AUTH\_NONE\_OPEN=1,

AUTH\_NONE\_WEP,

AUTH\_NONE\_WEP\_SHARED,

AUTH\_IEEE8021X,

AUTH\_WPA\_PSK,

AUTH\_WPA\_EAP,

AUTH\_WPA\_WPA2\_PSK,

AUTH\_WPA\_WPA2\_EAP,

AUTH\_WPA2\_PSK,

AUTH\_WPA2\_EAP

};

WEP64,WEP128 extensions:

WEP\_SEC\_KEY

typedef struct \_WepSecKey{

char Key[4][40]; /\* WEP Key data \*/

int KeyLen; /\* WEP Key length \*/

int Idx; /\* WEP Key Index [0,3] \*/

} WEP\_SEC\_KEY;

WPA/WPA2-PSK extension:

WPA\_PSK\_KEY

typedef struct \_WpaPskKey{

char Key[4][40]; /\* PSK Key data \*/

int KeyLen; /\* PSK Key length \*/

} WPA\_PSK\_KEY;

EAP extension:

WPA\_EAP\_KEY

typedef struct \_WpaEapKey{

int EapType; /\* EAP type \*/

charPwd[132]; /\* Password \*/

charId[132]; /\* 特性 \*/

char CaCert[132]; /\* CA 证书的路径和文件名 \*/

char CliCert[132];/\* 客户端证书的路径和文件名\*/

char PriKey[132]; /\* 文件路径到客户端的私钥文件 \*/

char PriKeyPwd[132]; /\* 密码私钥文件\*/

} WPA\_EAP\_KEY;

Scan hotspot information:

ST\_WifiApInfo {

charEssid[33]; /\* AP Name \*/

char Bssid[20]; /\* MAC\*/

int Channel; /\* Channel \*/

int Mode; /\* Connection method, 0:Station; 1:IBSS \*/

int Rssi; /\* Signal value, range [-99, 0]\*/

intAuthMode; /\*Authentication mode \*/

int SecMode; /\* Encryption modes, NONE, WEP, TKIP, CCMP \*/

} ST\_WifiApInfo;

Connection hotspot settings:

ST\_WifiApSet

typedef struct \_WifiApSet{

charEssid[33]; /\* AP name, supports a maximum of 32 bytes, ending with '\ 0' \*/

char Bssid[20]; /\*MAC address, ending with '\ 0'; If there is no AP with the same ESSID, Bssid can be’\0’\*/

int Channel; /\* Channel, only valid in IBSS mode, 0: default setting\*/

int Mode; /\* Connection method, 0: Station; 1: IBSS \*/

intAuthMode; /\* authentication mode \*/

int SecMode /\* Encryption modes, NONE, WEP, TKIP, CCMP \*/

union KEY\_UNION{ /\* Key settings \*/

WEP\_SEC\_KEY WepKey; /\* WEP Mode\*/

WPA\_PSK\_KEY PskKey; /\* wpa,wpa2-psk Mode \*/

WPA\_EAP\_KEY EapKey; /\* wpa,wpa2-eap Mode \*/

} KeyUnion;

}ST\_WifiApSet;

## WIFI API

### OsWifiOpen

|  |  |  |
| --- | --- | --- |
|  | int OsWifiOpen(void); | |
| Func | Connect to WiFi service and obtain usage rights. | |
| Param |  |  |
| Return | RET\_OK  ERR\_DEV\_NOT\_EXIST  ERR\_DEV\_BUSY  ERR\_BATTERY\_VOLTAG E\_TOO\_LOW ERR\_BATTERY\_ABSENT | Success  WiFi module driver loading abnormal or module error  WiFi has been used  Battery voltage too low  Battery not present |
| Usage |  | |

### OsWifiClose

|  |  |  |
| --- | --- | --- |
|  | void OsWifiClose(void); | |
| Func | Release the usage rights of the WiFi module, and after calling this function, WiFi communication is not affected. | |
| Param |  |  |
| Return |  | |
| Usage |  | |

### 20.6 OsWifiScan

|  |  |  |
| --- | --- | --- |
|  | int OsWifiScan (ST\_WifiApInfo \*\*Aps); | |
| Func | Scan existing networks. | |
| Param | Aps【output】 | Scan the hotspot information structure ST\_WifiApInfo, which stores the information of the scanned network |
| Return | >=0 Number of APs found in search  ERR\_MEM\_FAULT memory error  ERR\_INVALID\_PARAM parameter error  ERR WIFI\_POWER\_OFF WiFi module not powered on  ERR\_DEV\_NOT\_OPEN Not obtaining WiFi device usage rights | |
| Usage | int i, num;  ST\_WifiApInfo \* Aps;  num = OsWifiScan (&Aps);  if(num <= 0)  return -1;  for(i=0; i<num; i++)  printf("[%d] AP name: %s\n", i, Aps[i].Essid); | |

### OsWifiConnect

|  |  |  |
| --- | --- | --- |
|  | int OsWifiConnect(const ST\_WifiApSet \*Ap,  int TimeOutMs); | |
| Func | Connect to the specified wireless network. | |
| Param | Ap【input】 | Connect the hotspot setting structure ST\_WifiApSet, which stores the specified wireless network properties. |
| TimeOutMs【input】 | Time out, in milliseconds;  The value range is 0-3600000. |
| Return | RET\_OK Connection successful  RET\_CONNECTING Connecting  ERR\_NOT\_CONNECT Connect failed  ERR\_WIFI\_BAD\_SIGNAL Signal difference  ERR\_NOT\_FOUND\_AP AP not found  ERR\_ NET\_PASSWD Password error  ERR\_AUTH\_SEC\_MODE Authentication mode or encryption mode error  ERR\_WIFI\_POWER\_OFF WiFi Module not powered on  ERR\_DEV\_NOT\_OPEN Not obtaining WiFi device usage rights  ERR\_INVALID\_PARAM Parameter error | |
| Usage |  | |

### OsWifiDisconnect

|  |  |  |
| --- | --- | --- |
|  | int OsWifiDisconnect(void); | |
| Func | Disconnect from the current network. | |
| Param |  |  |
| Return | RET\_OK  ERR\_DEV\_NOT\_OPEN | Success  Not obtaining WiFi device usage rights |
| Usage |  | |

### OsWifiCheck

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsWifiCheck(char \*Essid,  char \*Bssid,  int \*Rssi); | | |
| Func | Obtain the current network status of the terminal. | | |
| Param | Essid【output】 | The current ESSID that is connected to the network cannot be null and has a length of 33 bytes. | |
| Bssid【output】 | The current BSSID that is connected to the network cannot be null and has a length of 20 bytes. | |
| Rssi【output】 | Signal strength cannot be null.  The range of values is [-99, 0], where 0 represents the strongest signal. | |
| Return | RET\_OK  RET\_CONNECTING  ERR\_NOT\_CONNECT  ERR\_WIFI\_BAD\_SIGNAL  ERR\_NOT\_FOUND\_AP  ERR\_NET\_PASSWD  ERR\_AUTH\_SEC\_MODE  ERR\_INVALID\_PARAM | | Connection successful  Connecting  Not connected to the network  WiFi signal difference  AP not found  Password error  Authentication mode or encryption mode error  Invalid param |
| Usage |  | | |

### OsWifiCmd

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsWifiCmd (const char \*Argv[],  int Argc,  char \*Result,  int Len); | | |
| Func | Send a command to the WPA\_Supplicant backend service and obtain the return result. | | |
| Param | Argv【input】 | The commands supported by WPA\_Supplicant cannot be NULL. | |
| Argc【input】 | The Argv two-dimensional array stores the number of commands or parameters. | |
| Result【output】 | WPA\_Supplicant returns a result that cannot be null and has a length greater than 2048 bytes. | |
| Return | Len【input】 | Result array length | |
| RET\_OK  ERR\_INVALID\_PARAM  ERR\_WIFI\_POWER\_OFF  ERR\_DEV\_NOT\_OPEN | | Send success  Invalid param  WiFi module not powered on  Not obtaining WiFi device usage rights |
| Usage |  | | |

### OsGetWifiStatus

|  |  |  |  |
| --- | --- | --- | --- |
|  | int OsGetWifiStatus(char \*Essid,char \*Bssid,char\* Ip,int \*Rssi)； | | |
| Func | Obtain WIFI status. | | |
| Param | Essid【output】 | The current ESSID that is connected to the network cannot be null and has a length of 33 bytes. | |
| Bssid【output】 | The BSSID of the currently connected network cannot be null and has a length of 20 bytes. | |
| Ip【output】 | The current IP address cannot be null and has a length greater than 15 bytes. | |
| Rssi【output】 | Signal strength cannot be null.  The range of values is [-99, 0], where 0 represents the strongest signal. | |
| Return | RET\_OK  OTHER | | Success  Fail |
| Usage |  | | |

### OsWifiGetAutoConnectStatus

|  |  |  |
| --- | --- | --- |
|  | int OsWifiGetAutoConnectStatus(int \*status); | |
| Func | Obtain WIFI automatic connection status. | |
| Param | status【output】 | Connection status:  1 system automatically connects to the network  0 system does not automatically connect to the network |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

### OsWifiSetAutoConnectStatus

|  |  |  |
| --- | --- | --- |
|  | int OsWifiSetAutoConnectStatus(int onoff); | |
| Func | Set the WIFI automatic connection status. | |
| Param | onoff【input】 | Connection status,  1 system automatically connects to the network  0 system does not automatically connect to the network |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

# File System

The file system uses standard ANSI.c Access the file operation API.

# Power Management

## Return Values List

|  |  |  |
| --- | --- | --- |
|  | Value | Explain |
| BATTERY\_LEVEL\_0 | 0 | Battery level 0 grid |
| BATTERY\_LEVEL\_1 | 1 | Battery level 0 grid |
| BATTERY\_LEVEL\_2 | 2 | Battery level 0 grid |
| BATTERY\_LEVEL\_3 | 3 | Battery level 0 grid |
| BATTERY\_LEVEL\_4 | 4 | Battery level 0 grid |
| BATTERY\_LEVEL\_CHARGE | 5 | battery charging |
| BATTERY\_LEVEL\_COMPLETE | 6 | The battery is fully charged |
| BATTERY\_LEVEL\_ABSENT | 7 | Battery not present |

## OsCheckBattery

|  |  |  |
| --- | --- | --- |
|  | int OsCheckBattery(void); | |
| Func | Check the battery level. | |
| Param |  |  |
| Return | BATTERY\_LEVEL\_0  BATTERY\_LEVEL\_ 1  BATTERY\_LEVEL\_2  BATTERY\_LEVEL\_3  BATTERY\_LEVEL\_4  BATTERY\_LEVEL\_CHARGE  BATTERY\_LEVEL\_COMPLETE  BATTERY\_LEVEL\_ABSENT | 0~5% battery level, low power supply, requires immediate charging. It is recommended not to engage in transactions, printing, wireless communication, and other operations. When the battery level is too low, the system will automatically shut down  5%~15%  15%~40%  40%~70%  70%~100%  battery charging  Battery fully charged, externally powered  Battery not present, external power supply |
| ERR\_SYS\_NOT\_SUPPORT | The system does not support detecting battery level, and terminals without batteries return this value. |
| Usage |  | |

## OsCheckPowerSupply

|  |  |  |
| --- | --- | --- |
|  | int OsCheckPowerSupply(void); | |
| Func | Detect the power supply type. | |
| Param |  |  |
| Return | POWER\_BATTERY POWER\_ADAPTER POWER\_USB  POWER\_WPC | Built in battery power supply  Power adapter power supply  USB peripheral power supply, such as a PC.  Wireless base power supply |
| Usage |  | |

## OsReboot

|  |  |  |
| --- | --- | --- |
|  | int OsReboot(void); | |
| Func | Restart the machine. | |
| Param |  |  |
| Return | RET\_OK Success | |
| Usage | This function is in blocking mode, and if executed successfully, it will restart directly without returning. | |

## OsPowerOff

|  |  |  |
| --- | --- | --- |
|  | int OsPowerOff (void); | |
| Func | Turn off the machine | |
| Param |  |  |
| Return | RET\_OK Success | |
| Usage | This function is in blocking mode, and if executed successfully, it will shut down without returning. | |

## OsGetBatCap

|  |  |  |
| --- | --- | --- |
|  | int OsGetBatCap(void); | |
| Func | Obtain battery level | |
| Param |  |  |
| Return | Greater than 0: Battery level percentage, other acquisition failed | |
| Usage |  | |

## OsGetBatVol

|  |  |  |
| --- | --- | --- |
|  | int OsGetBatVol(void); | |
| Func | Obtain battery voltage | |
| Param |  |  |
| Return | Greater than 0: Battery voltage (microvolts μ V)  Other acquisition fail | |
| Usage |  | |

## OsGetBatChgState

|  |  |  |
| --- | --- | --- |
|  | int OsGetBatChgState(void); | |
| Func | Get charging status | |
| Param |  |  |
| Return | Charging  state; Other discharge states | |
| Usage |  | |

## OsSysSleepTime

|  |  |  |
| --- | --- | --- |
|  | int OsSysSleepTime(int Time); | |
| Func | Set sleep time | |
| Param | Time 【input】 | Sleep time, in seconds, recommended range (10-3600) |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

## OsGetSysSleepTime

|  |  |  |
| --- | --- | --- |
|  | int OsGetSysSleepTime(int \*Time); | |
| Func | Get Sleep Time | |
| Param | Time 【output】 | Sleep time, in seconds, the system defaults to 300s and cannot be null |
| Return | RET\_OK Success  OTHER Fail | |
| Usage |  | |

## OsSetSysSleepStatus

|  |  |  |
| --- | --- | --- |
|  | int OsSetSysSleepStatus(int status); | |
| Func | Set sleep mode | |
| Param | status【input】 | 0 prohibits sleep;  1 Allow sleep.  The system allows hibernation by default, and the device restarts to allow hibernation. |
| Return | RET\_OK Success  OTHER Fail | |
| Usage | Device restart allows hibernation to resume | |

## OsGetSysSleepStatus

|  |  |  |
| --- | --- | --- |
|  | int OsGetSysSleepStatus(int \*status); | |
| Func | Get Sleep State | |
| Param | status【output】 | 0 prohibits sleep;  1 Allow sleep.  Cannot be null, the system allows sleep by default, and the device restarts to allow sleep. |
| Return | RET\_OK Success  OTHER Fail | |
| Usage | Device restart allows hibernation to resume | |

# Database

The SDK provides database operation functions, and the database interface is implemented by calling the sqlite interface, which is encapsulated into a unified interface and provided externally

## DB\_bInit

|  |  |  |
| --- | --- | --- |
|  | int DB\_bInit(char \*szAppID); | |
| Func | Init Database. | |
| Param | szAppID【input】 | Pass in the current APP ID |
| Return | RET\_OK Success  OTHER Fail | |
| Usage | Application startup requires calling interface initialization | |

## DB\_hOpen

|  |  |  |
| --- | --- | --- |
|  | HANDLE DB\_hOpen(const char\* pszName, int nMaxLenOfRec); | |
| Func | Open the specified record table | |
| Param | szTableName【input】 | Table name |
| nMaxLenOfRec【input】 | Maximum length of a single record |
| Return | Success： Return Table Handle；  Fail： NULL； | |
| Usage |  | |

## DB\_vClose

|  |  |  |
| --- | --- | --- |
|  | void DB\_vClose(HANDLE hTable); | |
| Func | Close the specified table | |
| Param | hTable【input】 | Corresponding table handle. |
| Return |  | |
| Usage |  | |

## DB\_bClear

|  |  |  |
| --- | --- | --- |
|  | int DB\_bClear(HANDLE hTable); | |
| Func | Clear table (do not delete table) | |
| Param | hTable【input】 | Corresponding table handle. |
| Return | RET\_OK Success  OTHER Fail | |
| Usage | DB\_hOpen is required before calling | |

## 23.5 DB\_bRemove

|  |  |  |
| --- | --- | --- |
|  | int DB\_bRemove(const char\* pszName, int nAccess); | |
| Func | Delete Table | |
| Param | pszName【input】 | Table name |
| nAccess【input】 | Reserved, not used |
| Return | RET\_OK Success  OTHER Fail | |
| Usage | You don't need DB\_hOpen to call it | |

## DB\_nReadRec

|  |  |  |
| --- | --- | --- |
|  | int DB\_nReadRec(HANDLE hTable, int nIndexOfRec, int nLenOfBuffer, char\*pRecordBuf); | |
| Func | Reading records through index numbers | |
| Param | hTable【input】 | table handle. |
| nIndexOfRec【input】 | Record index value (starting from 0) |
| nLenOfBuffer【input】 | The length of the buffer pointed to by pRecordBuff |
| pRecordBuf【output】 | Return Record Content |
| Return | Success：Return the actual length value of the record  Fail：-1 | |
| Usage | DB\_hOpen is required before calling | |

## DB\_nReadRecByKey

|  |  |  |
| --- | --- | --- |
|  | int DB\_nReadRecByKey(HANDLE hTable, char\* pKey, int nLenOfBuffer, char\*pRecordBuf); | |
| Func | Reading records through index numbers | |
| Param | hTable【input】 | Table handle |
| pKey【input】 | Query keywords must be unique |
| nLenOfBuffer【input】 | The length of the buffer pointed to by pRecordBuff |
| pRecordBuf【output】 | Return Record Content |
| Return | Success: Returns the actual length value of the record  Failure: -1 | |
| Usage | DB\_hOpen is required before calling | |

## DB\_nAppendRecByKey

|  |  |  |
| --- | --- | --- |
|  | int DB\_nAppendRecByKey(HANDLE hTable, char\* pKey, char\* pRecordBuf, int nRecLen); | |
| Func | Add record (with indexed keywords) | |
| Param | hTable【input】 | Table handle |
| pKey【input】 | Query keywords must be unique  (For a string such as "REC:0000000 1") |
| pRecordBuf【input】 | Add a record data pointer (pRecordBuf can be a structure or a string) |
| nRecLen【input】 | Add record length |
| Return | Success: Returns the actual length value of the written record  Failure: -1 | |
| Usage | DB\_hOpen is required before calling | |

## DB\_nReplaceRec

|  |  |  |
| --- | --- | --- |
|  | int DB\_nReplaceRec(HANDLE hTable, int nIndexOfRec, char\* pRecord, int nRecordLen); | |
| Func | Update records, update through index | |
| Param | hTable【input】 | Table handle |
| nIndexOfRec【input】 | Table Record Index (starting from 0) |
| pRecord【input】 | Records that need to be updated |
| nRecordLen【input】 | Record length |
| Return | Success: Returns the actual length value of the written record  Failure: -1 | |
| Usage | DB\_hOpen is required before calling | |

## DB\_nReplaceRecByKey

|  |  |  |
| --- | --- | --- |
|  | int DB\_nReplaceRecByKey(HANDLE hTable, char\* pKey,int nRecordLen,char\* pRecord); | |
| Func | Update records by keywords | |
| Param | hTable【input】 | Table handle |
| pKey【input】 | Query Keywords |
| pRecord【input】 | Records that need to be updated |
| nRecordLen【input】 | Record length |
| Return | Success: Returns the actual length value of the written record  Failure: -1 | |
| Usage | DB\_hOpen is required before calling | |

## DB\_bDeleteRec

|  |  |  |
| --- | --- | --- |
|  | int DB\_bDeleteRec(HANDLE hTable, int nIndex); | |
| Func | Delete records through record index | |
| Param | hTable【input】 | Table handle |
| nIndex【input】 | Table Record Index (starting from 0) |
| Return | Success: 0  Failure: Other | |
| Usage | DB\_hOpen is required before calling | |

## DB\_bDeleteRecByKey

|  |  |  |
| --- | --- | --- |
|  | int DB\_bDeleteRecByKey(HANDLE hTable, char\* pKey); | |
| Func | Delete records through keywords | |
| Param | hTable【input】 | Table handle |
| pKey【input】 | Query Keywords |
| Return | Success: 0  Failure: Other | |
| Usage | DB\_hOpen is required before calling | |

## DB\_bIsExsit

|  |  |  |
| --- | --- | --- |
|  | int DB\_bIsExsit(const char\* pszTableName, int nAccess); | |
| Func | Check if the table name exists | |
| Param | pszTableName【input】 | Table handle |
| nAccess【input】 | Access mode (reserved) |
| Return | 0: Exists  Other: Not present | |
| Usage | You don't need DB\_hOpen to call it | |

## DB\_nGetRecordSUM

|  |  |  |
| --- | --- | --- |
|  | int DB\_nGetRecordSUM(HANDLE hTable); | |
| Func | Obtain the number of valid records | |
| Param | hTable【input】 | Table handle |
| Return | Success: Record the total number of records; 0  Failure: -1 | |
| Usage | DB\_hOpen is required before calling | |

## DB\_nGetLastError

|  |  |  |
| --- | --- | --- |
|  | int DB\_nGetLastError(HANDLE hTable); | |
| Func | Obtain current error information | |
| Param | handle【input】 | Table handle |
| Return | Return error value:  0 Normal  -1 Operation failed  -2 Parameter error  -3 Length error | |
| Usage | DB\_hOpen needs to be called before it can be called | |

## DB\_nGetFirstRecord

|  |  |  |
| --- | --- | --- |
|  | int DB\_nGetFirstRecord(HANDLE hTable, int \*nIndexOfRec, void\* pRecordBuf, int nRecLen); | |
| Func | Get the first record | |
| Param | hTable【input】 | Table handle |
| nIndexOfRec【input】 | Index pointer (unused) |
| pRecordBuf【output】 | The returned data content |
| nRecLen【output】 | The length of the buffer pointed to by pRecordBuff |
| Return | Success: Returns the actual length value of the record  Failure: -1 | |
| Usage | DB\_hOpen needs to be called before it can be called | |

# UI Display(Not LVGL)

This module provides interfaces and operations for points, lines, images, display controls, and input controls.

## Data Struct

|  |
| --- |
| 数据结构定义： |
| DEFINE DPORT\_STYLE\_T{  // Default Style  DPORT\_DEFAULT\_STYLE = (1 << 0),  // Prohibit numeric keypad selection  DPORT\_DISABLE\_NUMBERKEY = (1 << 1),  // Automatically add index numbers and index each page independently  DPORT\_AUTO\_INDEX\_SEPARATELY = (1 << 2),  // Automatically add index numbers, all items are indexed consecutively  DPORT\_AUTO\_INDEX\_CONTINUITY = (1 << 3),  // Enable BYPASS function  DPORT\_ENABLE\_BYPASS = (1 << 4),  DPORT\_LIST\_ALWAYS\_RETURN\_NUMBERKEY = (1 << 5),   // Display English  DPORT\_ENABLE\_LANGEN = (1 << 6),  }  DEFINE DPORT\_DTYPE\_T{  EM\_DTYPE\_NORMAL, // Positive display  EM\_DTYPE\_REVERT, // reverse display  EM\_DTYPE\_WITH\_UNDERLINE, // Underlined positive display  EM\_DTYPE\_WITH\_TOPLINE, // Upright display with underline  EM\_DTYPE\_WITH\_RECT // frame  }  DEFINE DPORT\_LINETYPE\_T{  EM\_LINETYPE\_SOLID = 0, // Solid line  EM\_LINETYPE\_DOTTED = 1, // Dashed line  }  //Color  DEFINE DPORT\_COLOR\_T{  EM\_COLOR\_BLACK,  EM\_COLOR\_WHITE,  EM\_COLOR\_GREY,  EM\_COLOR\_RED,  EM\_COLOR\_MAX,  }  // inputting method  DEFINE DPORT\_IME\_T{  DPORT\_IME\_PY = (1<<0),  DPORT\_IME\_EN = (1<<1),  DPORT\_IME\_NUM =(1<<2),  DPORT\_IME\_SYMBOL = (1<<3)  }  //MessageBox Display confirmation key and cancel key  DEFINE DPORT\_TIPPAGE\_KEY\_TYPE\_T{  EM\_STYLE\_OK = (1 <<0), //confirm  EM\_STYLE\_CANCEL= (1 <<1) //cancel  }  typedef enum tagPOINT\_RANGE  {  POINT\_X\_BEGIN = 22,  POINT\_X\_END = 320-20,  POINT\_Y\_BEIGN = 0,  POINT\_Y\_END = 240,  }POINT\_RANGE\_T;  // Widget return value  DEFINE DPORT\_RET\_T{  RET\_DPORT\_TIMEOUT = -3, //timeout  RET\_DRPOT\_CANCEL = -2, //cancel  RET\_DPORT\_FAILD = -1, //fail  RET\_DPORT\_NORMAL = 0, //success  RET\_DPORT\_OK = RET\_DPORT\_NORMAL,  RET\_DPORT\_BYPASS = RET\_DPORT\_NORMAL,  }  DEFINE InputPinPara{  Int8 szPan[20];  Int32 nPanLen;  Int32 nKeyIndex;  Int32 nPinMode;  Int8 szHint[64];  Int32 nMinPinLen;  Int32 nMaxPinLen;  }  //Font Size  typedef enum DPORT\_FONT\_SIZE  {  DPORT\_DEFAULT = 0,  DPORT\_FONT\_48 = 1,  DPORT\_FONT\_32 = 2,  DPORT\_FONT\_24 = 3,  DPORT\_FONT\_16 = 4,  DPORT\_FONT\_MAX  }DPORT\_FONT\_SIZE\_T;  DEFINE DPORT\_LANGEUGE\_T;  {  DPORT\_CHINESE = 0,  DPORT\_ENGLISH = 1  }  typedef enum DPORT\_STYLE  {  DPORT\_LEFT = 0,  DPORT\_RIGHT,  DPORT\_CENTER  }DPORT\_TEXTSTYLE\_T; |
|  |

## Display API

### Disp\_vInit

|  |  |  |
| --- | --- | --- |
|  | void Disp\_vInit(void) | |
| Func | Disp module initialization | |
| Param |  |  |
| Return |  | |
| Usage | Must be called once during system initialization | |

### Disp\_vClearLine

|  |  |  |
| --- | --- | --- |
|  | void Disp\_vClearLine(Uint8 nAtLine) | |
| Func | Clear the display content of the specified row | |
| Param | nAtLine【input】 | Designated row position |
| Return |  | |
| Usage |  | |

### Disp\_vShowStrAt

|  |  |  |
| --- | --- | --- |
|  | void Disp\_vShowStrAt(Uint8 nAtLine, Uint8 nAtColumn, Int8\* fmt, ...) | |
| Func | Display string content in specified rows and columns | |
| Param | nAtLine【input】 | Specify line number |
| nAtColumn【input】 | Specify column position (currently not implemented, default to 0) |
| fmt【input】 | Displayed data, input using variable parameter methods |
| Return |  | |
| Usage | Line number range:   1. Large font subscript is 0-5, a total of 6 lines, 2. small font subscript is 0-6, a total of 7 lines   Disp\_vShowStrAt (1, 2, "% s", "Show here") | |

### Disp\_vShowStr

|  |  |  |
| --- | --- | --- |
|  | void Disp\_vShowStr(Uint8 nAtLine, Uint8 nType, Uint8 nPattern, Int8\* fmt, ...) | |
| Func | Display string content in specified rows and columns | |
| Param | nAtLine【input】 | Specify line number |
| nType【input】 | Display type as defined in  DPORT-DTYPE-T |
| nPattern【input】 | The alignment method can be found in the ALIGN-DEF-T definition |
| fmt【input】 | Displayed data, input using variable parameter methods |
| Return |  | |
| Usage | Line number range:  Large font subscript is 0-5, a total of 6 lines,  small font subscript is 0-6, a total of 7 lines | |

### Disp\_vShowImage

|  |  |  |
| --- | --- | --- |
|  | void Disp\_vShowImage(  Uint8\* pcBmpData,  Uint32 LenofBufferSize,  POINT\_T tStartPoint,  Uint32 uBmpWidth,  Uint32 uBmpHeight) | |
| Func | Display the image at the specified pixel point as the starting position | |
| Param | pcBmpData【input】 | Image data buffering |
| LenofBufferSize【input】 | The length of the buffer pointed to by pBmpData |
| nStartPoint【input】 | Specify the starting X and Y points for screen display, and their positions (the range limit of X and Y can be obtained by calling the Disp\_vGetUserDisplayRange interface) |
| uBmpWidth【input】 | Image width |
| uBmpHeight【input】 | Image height |
| Reurn |  | |
| Usage |  | |

### Disp\_vDrawLineAt

|  |  |  |
| --- | --- | --- |
|  | void Disp\_vDrawLineAt(POINT\_T tBegin, POINT\_T tEnd, DPORT\_LINETYPE\_T nType, DPORT\_COLOR\_T nColor) | |
| Func | Draw a line from the specified point (nBeginX, nBeginY) to (nEndX, nEndY) | |
| Param | tBegin【input】 | The specified starting point X and Y coordinates (the range limit of X and Y can be obtained by calling the Disp\_vGetUserDisplayRange interface) |
| tEnd【input】 | The specified starting point X and Y coordinates (the range limit of X and Y can be obtained by calling the Disp\_vGetUserDisplayRange interface) |
| nType【input】 | Specify line drawing type (supports solid and dashed lines), diagonal lines can only draw solid lines |
| nColor【input】 | Specify the color of the depiction |
| Return |  | |
| Usage |  | |

### Disp\_vClearPort

|  |  |
| --- | --- |
|  | void Disp\_vClearPort(void) |
| Func | Clear all content displayed by the application |
| Param |  |
| Return |  |
| Usage |  |

### Disp\_nUserInputView

|  |  |  |
| --- | --- | --- |
|  | Int32 Disp\_nUserInputView(int X, int Y, int nWidth,int nHeight, Uint32 nLimtMin, Uint32 nLimtMax,Uint32 nInputMethod, Uint32 nOutBufferLen, Uint8 \*pOutBuffer, Int32 nTimeOut, Uint32 nStyle) | |
| Func | User Input Box Widget | |
| Param | X Y【input】 | Display starting position |
| nWidth nHeight【input】 | Display width and height |
| nLimtMin【input】 | Enter minimum length limit |
| nLimtMax【input】 | Enter maximum length limit |
| nInputMethod【input】 | Input Method Type |
| nOutBufferLen【input】 | Output cache length |
| nTimeOut【input】 | Time out (in seconds) |
| nStyle【input】 | Number key response and automatic indexing function (reserved, fill in 0) |
| pOutBuffer【output】 | output caching |
| Return | Success -- Enter length  Failed -- RET-DPORT-FAILD  Time out -- RET-DPORT-TIMEOUT  User Cancellation -- RET-DRPOT-CANCEL | |
| Usage | The maximum number of characters displayed on a single line varies depending on the font size | |

### Disp\_nUserInputViewEx

|  |  |  |
| --- | --- | --- |
|  | Int32 Disp\_nUserInputViewEx(int X, int Y, int nWidth, int nHeight,  Uint32 nLimtMin,  Uint32 nLimtMax,  Uint32\* pInputMethod,  Uint32 nLenOfInputMethod,  Uint32 nOutBufferLen,  Uint8 \*pOutBuffer,  Int32 nTimeOut,  Uint32 nStyle) | |
| Func | User Input Box Control (can set the order of input method types) | |
| Param | X Y【input】 | Display starting position |
| nWidth nHeight【input】 | Display width and height |
| nLimtMin【input】 | Enter minimum length limit |
| nLimtMax【input】 | Enter maximum length limit |
| pInputMethod【input】 | Input method type pointer |
| nLenOfInputMethod【input】 | Number of input methods |
| nOutBufferLen【input】 | Output cache length |
| nTimeOut【input】 | Time out (in seconds) |
| nStyle【input】 | Number key response and automatic indexing function (reserved, fill in 0) |
| pOutBuffer【output】 | output caching |
| Return | Success -- Enter length  Failed -- RET-DPORT-FAILD  Time out -- RET-DPORT-TIMEOUT  User Cancellation -- RET-DRPOT-CANCEL | |
| Usage |  | |

### Disp\_nSetMsgBoxLanguage

|  |  |  |
| --- | --- | --- |
|  | void Disp\_nSetMsgBoxLanguage(Uint8 language); | |
| Func | Set messgeBox display language | |
| Param | Language【input】 | Set display language (DPORT-CHINESE or DPORT-ENGLISH) |
| Return |  | |
| Usage | Default DPORT-ENGLISH | |

### Disp\_nMessageBox

|  |  |  |
| --- | --- | --- |
|  | Int32 Disp\_nMessageBox(Int8 \*szTitle, Int32 theStype, Int32 nTimeoutSec, Uint32 nStyle, Int8 \*args,...) | |
| Func | Prompt Box Widget | |
| Param | szTitle【input】 | Header Information |
| theStype【input】 | Interface style reference  DPORT-TIPPAGE-KEY-TYPE-T |
| nTimeoutSec【input】 | Time out (in seconds) |
| nStyle【input】 | Number key response and automatic indexing function (reserved, fill in 0) |
| args【input】 | Format string |
| Return | RET\_DPORT\_OK --- Confirm key return value  RET\_DRPOT\_CANCEL --- Cancel key returns value  RET\_DPORT\_TIMEOUT --Timeout | |
| Usage | Interface type:  Currently, there are only single confirm and confirm cancel keys available  Supports up to 100 Chinese characters and displays 200 characters | |

### Disp\_nMessageBoxEx

|  |  |  |
| --- | --- | --- |
|  | int Disp\_nMessageBoxEx(int X, int Y, int nWidth, int nHeight, char\* szTitle, int theStype, int nTimeoutSec, Uint32 nStyle, char \*args,...) | |
| Func | Message box | |
| Param | X Y【input】 | Display starting position |
| nWidth nHeight【input】 | Display width and height |
| szTitle【input】 | Header Information |
| theStype【input】 | Interface style reference DPORT-TIPPAGE-KEY-TYPE-T |
| nTimeoutSec【input】 | Time out (in seconds) |
| nStyle【input】 | Number key response and automatic indexing function (reserved, fill in 0) |
| args【input】 | format string |
| Return | RET\_DPORT\_OK --- Confirm  RET\_DRPOT\_CANCEL --- Cancel  RET\_DPORT\_TIMEOUT --Timeout | |
| Usage | Interface type:  Currently, there are only single confirm and confirm cancel keys available  Supports up to 100 Chinese characters and displays 200 characters | |

### Disp\_nShowEditBox

|  |  |  |
| --- | --- | --- |
|  | int Disp\_nShowEditBox(int X, int Y, int nWidth, int nHeight, int nTimeoutSec, unsigned int nStyle, char\* fmt, ...) | |
| Func | Display string control (content can be flipped) | |
| Param | X Y【input】 | Display starting position |
| nWidth nHeight【input】 | Control display width and height |
| nTimeoutSec【input】 | Time out (in seconds) |
| nStyle【input】 | Number key response and automatic indexing function (reserved, fill in 0) |
| fmt【input】 | format string |
| Return | RET\_DPORT\_OK --- Confirm  RET\_DRPOT\_CANCEL --- Cancel  RET\_DPORT\_TIMEOUT --Timeout | |
| Usage | Supports up to 100 Chinese characters and displays 200 characters | |

### Disp\_nShowEditBoxEx

|  |  |  |
| --- | --- | --- |
|  | int Disp\_nShowEditBoxEx(int X, int Y, int nWidth, int nHeight, int nTimeoutSec, unsigned int nStyle, char\* string) | |
| Func | Display string control (content can be flipped) | |
| Param | X Y【input】 | Display starting position |
| nWidth nHeight【input】 | Control display width and height |
| nTimeoutSec【input】 | Time out (in seconds) |
| nStyle【input】 | Number key response and automatic indexing function (reserved, fill in 0) |
| string【input】 | Displayed string |
| Return | RET\_DPORT\_OK --- Confirm  RET\_DRPOT\_CANCEL --- Cancel  RET\_DPORT\_TIMEOUT --Timeout | |
| Usage |  | |

### Disp\_vShowLogoView

|  |  |  |
| --- | --- | --- |
|  | void Disp\_vShowLogoView(Int8 \*szLogoPath, Int8\* pszScreenTip) | |
| Func | Display standby screen | |
| Param | szLogoPath【input】 | The standby screen image path needs to be in 565 format |
| pszScreenTip【input】 | Content of standby screen prompt information |
| Return |  | |
| Usage |  | |

### Disp\_nShowAmountInputView

|  |  |  |
| --- | --- | --- |
|  | int Disp\_nShowAmountInputView(int nX, int nY,  int nWidth, int nHeight,  char\* pszCurrencyFlag,  int nMinLen,  int nMaxLen,  int nLenOfBuffer,  char\* pszAmount,  int nTimeoutSec,  unsigned int nStyle) | |
| Func | Display the amount input interface | |
| Param | nX nY【input】 | Display starting position |
| nWidth nHeight【input】 | Display width and height |
| pszCurrencyFlag【input】 | Currency symbol string |
| nMinLen【input】 | Minimum length limit |
| nMaxLen【input】 | Maximum length limit |
| nLenOfBuffer【input】 | The length of the buffer pointed to by pszAmount |
| nTimeoutSec【input】 | Time out (in seconds) |
| nStyle【input】 | Number key response and automatic indexing function (reserved, fill in 0) |
| pszAmount【output】 | Enter amount buffer |
| Return | Success -- Amount length  Failed -- RET-DPORT-FAILD  User Cancellation -- RET-DRPOT-CANCEL | |
| Suage |  | |

### Disp\_nShowPasswdInputView

|  |  |  |
| --- | --- | --- |
|  | int Disp\_nShowPasswdInputView(int nX, int nY, int nWidth, int nHeight,  int nMinLen,  int nMaxLen,  int nLenOfBuffer,  char\* pszPasswd,  int nTimeoutSec,  unsigned int nStyle) | |
| Func | Display user login password input box | |
| Param | nX nY【input】 | Display starting position |
| nWidth nHeight【input】 | Display width and height |
| nMinLen【input】 | Currency symbol string |
| nMaxLen【input】 | Minimum length limit |
| nLenOfBuffer【input】 | The length of the buffer pointed to by pszPasswd |
| nTimeoutSec【input】 | Time out (in seconds) |
| nStyle【input】 | Number key response and automatic indexing function (reserved, fill in 0) |
| pszPasswd【output】 | Enter password buffer |
| Return | Success -- returns the length of the entered password  Failed -- RET-DPORT-FAILD (including timeout)  User Cancellation -- RET-DRPOT-CANCEL | |
| Usage |  | |

### Disp\_nShowPINPADInputView

|  |  |  |
| --- | --- | --- |
|  | int Disp\_nShowPINPADInputView(int nX,int nY,  int nWidth,int nHeight,  InputPinPara \*pszInputPin,  int nLenOfPinBuffer,  int ntimeoutsec,  char\* pszPin) | |
| Func | Built in password keyboard offline pin input box | |
| Param | nX nY【input】 | Input box starting coordinates |
| nWidth nHeight【input】 | Input box size |
| pszInputPin【input】 | Enter PIN attribute |
| nLenOfPinBuffer【input】 | The length of the buffer pointed to by pszPin |
| ntimeoutsec【input】 | Time out, in seconds, with a maximum of 60 seconds |
| pszPin【output】 | PIN code returned by pszPin |
| Return | Success - Fixed return of the character length entered by the user  Failed - RET-DPORT-FAILD  User Cancellation - RET-DRPOT-CANCEL  Timed out - RET-DPORT-TIMEOUT | |
| Usage |  | |

### Disp\_nShowIPInputLine

|  |  |  |
| --- | --- | --- |
|  | int Disp\_nShowIPInputLine(int X, int Y,  int nWidth, int nHeight,  int nMinLen,  int nMaxLen,  int nLenOfBuffer,  char \*pszIP,  int nTimeoutSec,  unsigned int nStyle) | |
| Func | Display IP input box | |
| Param | X Y【input】 | Input box starting coordinates |
| nWidth nHeight【input】 | Input box size |
| nMinLen【input】 | Enter minimum length limit |
| nMaxLen【input】 | Enter maximum length limit |
| nLenOfBuffer【input】 | The size of the memory pointed to by pszIP |
| nTimeoutSec【input】 | Time out, in seconds, with a maximum of 60 seconds |
| nStyle【input】 | Number key response and automatic indexing function (reserved, fill in 0) |
| pszIP【output】 | Return IP string |
| Return | Success - Fixed return of the character length entered by the user  Failed - RET-DPORT-FAILD  User Cancellation - RET-DRPOT-CANCEL  Timed out - RET-DPORT-TIMEOUT | |
| Usage |  | |

### Disp\_vGetUserDisplayRange

|  |  |  |
| --- | --- | --- |
|  | void Disp\_vGetUserDisplayRange(int \*pnStartX, int \*pnEndX, int \*pnStartY, int \*pnEndY) | |
| Func | Return to the screen to display the range of the display area (start and end parameters of the X and Y axes) | |
| Param | pnStartX【output】 | Starting X coordinate |
| pnEndX【output】 | Starting Y coordinate |
| pnStartY【output】 | End X coordinate |
| pnEndY【output】 | End Y coordinate |
| Return |  | |
| Usage | int nStartX, nEndX, nStartY, nEndY;  Disp\_vGetUserDisplayRange(&nStartX, nEndX, &nStartY, &nEndY); | |

### Disp\_nShowListView

|  |  |  |
| --- | --- | --- |
|  | int Disp\_nShowListView(int iX, int iY, unsigned int uWidth, unsigned int uHeight, char\* pszTitle, char\*\* pszListItem, int nItemCount, int nItemsPerPage,int nSelectItem, int nTimeoutSec, int\* pnIndexOfSelect,unsigned int nStyle) | |
| Func | List display window, used to display a single list window | |
| Param | iXiY【input】 | List box coordinates |
| uWidthuHeight【input】 | List size |
| pszTitle【input】 | List Title |
| pszListItem【input】 | Show list titles |
| nItemCount【input】 | Display the number of items |
| nItemsPerPage【input】 | Display items per page |
| nSelectItem【input】 | Initial selected item index |
| nTimeoutSec【input】 | Time out: in seconds |
| nStyle【input】 | Number key response and automatic indexing function |
| pnIndexOfSelect【output】 | Selected project index (starting from 0) |
| Return | SUCCESS:RET\_DPORT\_NORMAL  TIMEOUT:RET\_DPORT\_TIMEOUT  CANCEL:RET\_DRPOT\_CANCEL | |
| Usage |  | |

### Disp\_nShowListViewEx

|  |  |  |
| --- | --- | --- |
|  | int Disp\_nShowListViewEx(char\* pszTitle, char\*\* pszListItem, int nItemCount, int nItemsPerPage, int nSelectItem, int nTimeoutSec, int\* pnIndexOfSelect, unsigned int nStyle) | |
| Func | List display window, used to display a single list window | |
| Param | pszTitle【input】 | List Title |
| pszListItem【input】 | show list |
| nItemCount【input】 | Display the number of items |
| nItemsPerPage【input】 | Display items per page |
| nSelectItem【input】 | Initial selected item index |
| nTimeoutSec【input】 | Time out: in seconds |
| nStyle【input】 | Number key response and automatic indexing function |
| pnIndexOfSelect【output】 | Selected project index (starting from 0) |
| Return | SUCCESS:RET\_DPORT\_NORMAL  TIMEOUT:RET\_DPORT\_TIMEOUT  CANCEL:RET\_DRPOT\_CANCEL | |
| Usage |  | |

### Disp\_nShowTableView

|  |  |  |
| --- | --- | --- |
|  | int Disp\_nShowTableView(int iX, int iY, unsigned int uWidth, unsigned int uHeight, char\* pszTitle, char\*\* pszListItem, int nItemCount, int nItemsPerPage, int nSelectItem,int nTimeoutSec, int\* pnIndexOfSelect, unsigned int nStyle) | |
| Func | Dual list display UI. Used for dual column display | |
| Param | iXiY【input】 | List box coordinates |
| uWidthuHeight【input】 | List size |
| pszTitle【input】 | List Title |
| pszListItem【input】 | show list |
| nItemCount【input】 | Display the number of items |
| nItemsPerPage【input】 | Display items per page |
| nSelectItem【input】 | Initial selected item index |
| nTimeoutSec【input】 | Time out: in seconds |
| nStyle【input】 | Number key response and automatic indexing function |
| pnIndexOfSelect【output】 | Selected project index (starting from 0) |
| Return | Success: RET-DPORT-NORMAL  Time out: RET-DPORT-TIMEOUT  Cancel: RET-DRPOT-CANCEL  Failed: RET-DPORT-FAILD | |
| Usage |  | |

### Disp\_nShowTableViewEx

|  |  |  |
| --- | --- | --- |
|  | int Disp\_nShowTableViewEx(char\* pszTitle, char\*\* pszListItem, int nItemCount,int nItemsPerPage, int nSelectItem,int nTimeoutSec, int\* pnIndexOfSelect, unsigned int nStyle) | |
| Func | Double list display UI. Used for dual column display | |
| Param | pszTitle【input】 | List Title |
| pszListItem【input】 | show list |
| nItemCount【input】 | Display the number of items |
| nItemsPerPage【input】 | Display items per page |
| nSelectItem【input】 | Initial selected item index |
| nTimeoutSec【input】 | Time out: in seconds |
| nStyle【input】 | Number key response and automatic indexing function |
| pnIndexOfSelect【output】 | Selected project index (starting from 0) |
| Return | Success: RET-DPORT-NORMAL  Time out: RET-DPORT-TIMEOUT  Cancel: RET-DRPOT-CANCEL  Failed: RET-DPORT-FAILD | |
| Usage |  | |

### Disp\_vCleanArea

|  |  |  |
| --- | --- | --- |
|  | void Disp\_vCleanArea(int X, int Y, int nWidth, int nHeight) | |
| Func | Clear (fixed size area of X, Y starting pixels) | |
| Param | XY【input】 | Clear the coordinates of the area |
| nWidthnHeight【input】 | Clear the size of the area |
| Return |  | |
| Usage |  | |

### Disp\_nShowDateInputView

|  |  |  |
| --- | --- | --- |
|  | int Disp\_nShowDateInputView(SetLocalTimeType\* pDate, int nTimeOutSec) | |
| Func | Display date input box | |
| Param | nTimeOutSec【input】 | Time out (in seconds) |
| pDate【input】 | Output date string |
| Return | Success: RET-DPORT-NORMAL  Time out: RET-DPORT-TIMEOUT  Cancel: RET-DRPOT-CANCEL  Failed: RET-DPORT-FAILD | |
| usage |  | |

### Disp\_vSetSysFont

|  |  |  |
| --- | --- | --- |
|  | void Disp\_vSetSysFont(DPORT\_FONT\_SIZE\_T FontType) | |
| Func | Set system font (currently only supports font size 24) | |
| Param | FontType【input】 | Font type |
| Return |  | |
| Usage |  | |

### Disp\_vDrawTextA

|  |  |  |
| --- | --- | --- |
|  | void Disp\_vDrawTextA(int x, int y, int w, int h, const char \* pStr, int len, DPORT\_TEXTSTYLE\_T nStyle) | |
| Func | Set system font | |
| Param | x, y【input】 | The x and y coordinates displayed in the font |
| w, h【input】 | The width and height of the display area |
| pStr【input】 | Displayed string |
| Len【input】 | The length of a string |
| nStyle【input】 | Font display method |
| Return |  | |
| Usage |  | |

### Disp\_GetKey

|  |  |  |
| --- | --- | --- |
|  | int Disp\_GetKey(int timeoutMs) | |
| Func | Get key value | |
| Param | timeoutMs【input】 | Timeout for obtaining key values |
| Return | The key value is defined as follows:  typedef enum KB\_KEYMAP\_e  {  EM\_KEY\_INVALID = -1,  EM\_KEY\_1,  EM\_KEY\_2,  EM\_KEY\_3,  EM\_KEY\_4,  EM\_KEY\_5,  EM\_KEY\_6,  EM\_KEY\_7,  EM\_KEY\_8,  EM\_KEY\_9,  EM\_KEY\_0,  EM\_KEY\_F1,  EM\_KEY\_F2,  EM\_KEY\_F3,  EM\_KEY\_F4,  EM\_KEY\_CLEAR,  EM\_KEY\_ENTER,  EM\_KEY\_UP,  EM\_KEY\_DOWN,  EM\_KEY\_POWER,  EM\_KEY\_CANCEL,  EM\_KEY\_MAX  }KB\_KEYMAP\_T; | |
| Usage |  | |

### Disp\_ReleasKey

|  |  |  |
| --- | --- | --- |
|  | int Disp\_ReleasKey() | |
| Func | Release button cache | |
| Param |  |  |
| Return | Success: RET-DPORT-NORMAL  Other: Failed | |
| Param |  | |

### Disp\_GetTouchPadAxis

|  |  |  |
| --- | --- | --- |
|  | int Disp\_GetTouchPadAxis(int \*piX, int \*piY, int timeoutMs) | |
| Func | Obtain the coordinates of the touch screen | |
| param | piX【output】 | The value of the x-axis cannot be null |
|  | piY【output】 | The value of the y-axis cannot be null |
|  | timeoutMs【input】 | Time out |
| Return | Success: RET-DPORT-NORMAL  Other: Failed | |
| Usage |  | |

### Disp\_vSetStatusBarStatus

|  |  |  |
| --- | --- | --- |
|  | void Disp\_vSetStatusBarStatus(int nState) | |
| Func | Set status bar status | |
| Param | nState【input】 | 0 Hide status bar  1 Show status bar |
| Return |  | |
| Usage |  | |