Terminal API Reference Manual

Version: V1.0.0

Version update record

No.	Version	Date	Modification record	Written by
1	V1.0.0	2025-12-5	Initial release, module framework setup	Li Yongliang

Contents

1 Introduction

DS50 system API interface document

2 Purposes

In order to facilitate application development, the API interface specification document has been written.

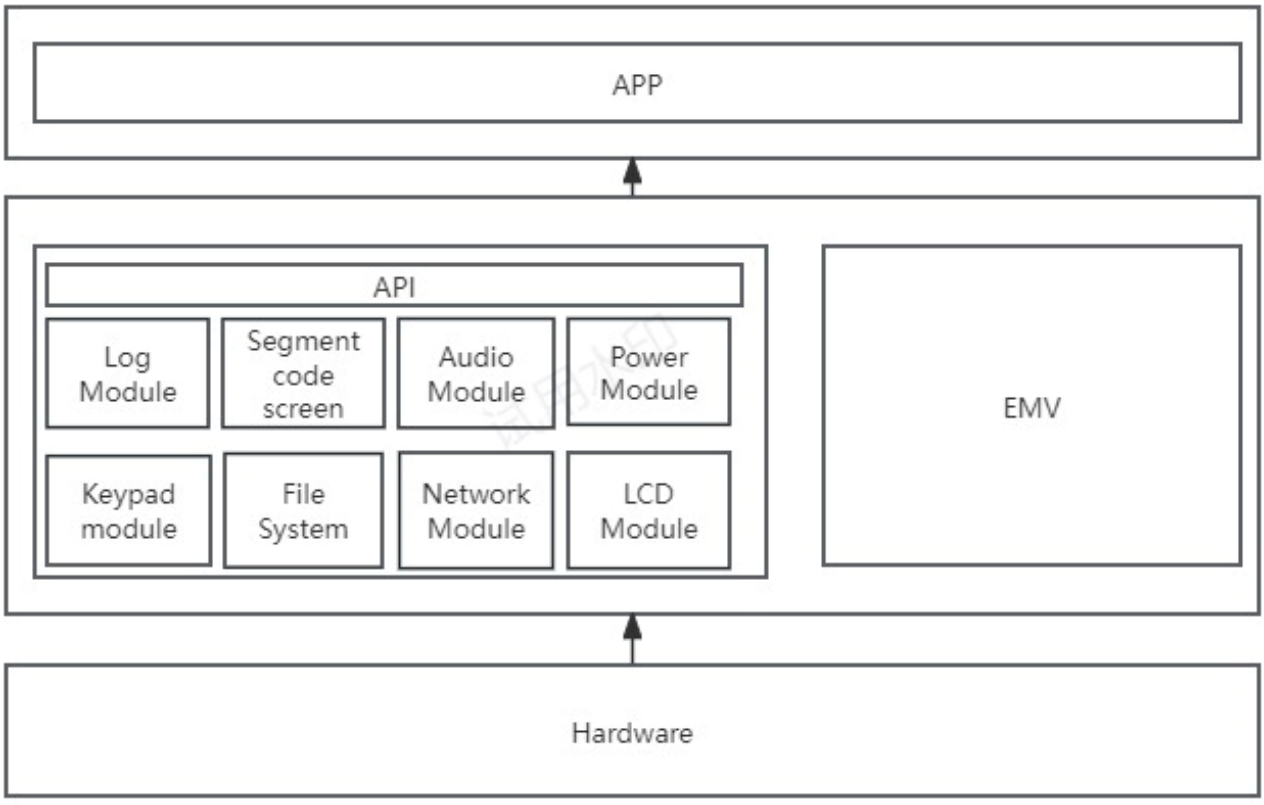
3 Background

4 Scope

4.1 Applicable Models

DS50

4.2 System Architecture



4.3 Data Type

The system interface has customized the following data types, as shown in the following table:

Table 1

No.	Name	Data Type	Type Description
1	s32	int	signed 4-byte integer

2	u32	unsigned int	unsigned 4-byte integer
3	s16	short	signed 2-byte integer
4	u16	unsigned short	unsigned 2-byte integer
5	s8	char	signed 1-byte integer
6	u8	unsigned char	unsigned 1-byte integer

4.4 Abbreviations

Table 2

No.	Name	Description
1	API	Application Programming Interface
2	BCD	Binary coded decimal numbers, 4 bits. The value ranges from 0 to 9, and the default value is BCD compression.
3	ms	milisecond
4	PCD	Proximity Coupling Device
5	PICC	Proximity IC Card

4.5 Function Name Conventions

The naming rules are similar to the Hungarian naming rules. "api" _ "module" _ "Function" operation. For the EMV module function, the rule is "EMV" _ "Function operation".

For example, the system module name is “sys” and the millisecond delay processing is “msleep”. Therefore, the final interface function form is OsSysMsleep().

The EMV module name is "EMV", the operation to obtain the version number is "GetModuleVer", so the final interface function form is: EMV_GetModuleVer().
The module full names and their corresponding abbreviations are listed as below:

Module Name	Function Name
System module	sys
Audio module	audio
Data coding module	base64
Timer module	timer
File system	file
Wireless communication	wireless
Wifi communication	wifi
Contact card module	ic
Contactless card module	rf
Keypad module	keypad
USB module	USB

Camera	camera
Segment code screen operation	seg
Buzzer module	beep
LCD module	lcd
SECURITY module	sec
LED module	led
MQTT module	mqtt
HTTP module	http
OTA module	ota
Power module	power
EMV module	EMV
LBS module	lbs

5 References

6 System Functions

6.1 Overview

System functions mainly provide functions to control system operation, including thread creation/destruction/synchronization mechanism, dynamic memory management, device restart and shutdown, queue, clock, version information, etc.

6.2 error code

API_THREADCREATE_ERROR -0x1001
API_GETTHREADID_ERROR -0x1002
API_MEMORY_ALLOC_ERROR -0x1003
API_MEMORY_FREE_ERROR -0x1004
API_MEMORY_REALLOC_ERROR -0x1005
API_REBOOT_ERROR -0x1006
API_POWEROFF_ERROR -0x1007
API_SEMCREATE_ERROR -0x1008
API_MUTEXCREATE_ERROR -0x1009
API_MUTEXLOCK_ERROR -0x1010
API_CREATEQUEUE_ERROR -0x1011
API_QUEUEPUT_ERROR -0x1012
API_GETHEAPINFO_ERROR -0x1013
API_QUEUEGET_ERROR -0x1014
API_GENERATERANDOM_ERROR -0x1015
API_SETRTC_ERROR -0x1016
API_GETRTC_ERROR -0x1017
API_GETSN_ERROR -0x1018
API_GETFIRMVERSION_ERROR -0x1019
API_GETHW_ERROR -0x1020
API_UPDATELOGO_ERROR -0x1021

6.3 OsSysMsleep

Prototype	int OsSysMsleep(unsigned int nMs);

Function		System blocking delay
Parameter	**Input**	nMs: Delay time unit: millisecond
	Output	N/A
Return		0: success Other: failure
Note		1. Set the hibernation time of the current thread. (The thread should not have looped code to prevent other threads from failing to run) 2. Pause the current thread task processing 3. The minimal granularity is 20ms.
Example		Correct example: while(1){ OsSysMsleep(1000); }

6.4 OsSysTick

Prototype		u32 OsSysTick();
Function		Return tick value from system startup to the current moment (in milliseconds)
Parameter	**Input**	N/A
	Output	N/A
Return		Tick value of the system running, in milliseconds
Note		The Return value of the function represents the total time from system startup to the current moment.
Example		U32 currTick = OsSysTick();

6.5 OsSysThreadCreate

Prototype		s32 OsSysThreadCreate(void *thread_fun,s8* thread_name,u32 thread_stack_size,void *params,u32 uxPriority,u32 *pThreadId);
Function		Create thread function
Parameter	**Input**	thread_fun: thread function entry thread_name: thread name thread_stack_size: thread stack size Params: thread parameter uxPriority: thread priority, see below

		typedef enum APIThreadPriority { API_PRIORITY_IDLE = 1, // reserved API_PRIORITY_LOW = 8, API_PRIORITY_BELOW_NORMAL = 16, API_PRIORITY_NORMAL = 24 } APIThreadPriority_t;
	Output	pThreadId: ID of the created thread
Return		0: success other: failure
Note		The size of the thread stack cannot be too large. The recommended value is from 10k to 100k. What needs to be noted is that the thread deletion function needs to be called explicitly when the thread function ends, otherwise the system will crash.Support multiple threads to use the same priority. When a thread exits, OsSysThreadDelete needs to be called.
Example		u32 threadID = 0; static void prvThreadEntry(void* param) { //Thread logic OsSysThreadDelete(threadID); } int ret=OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID);

6.6 OsSysThreadId

Prototype		u32 OsSysThreadId(void);
Function		Get the current running thread ID
Parameter	**Input**	N/A
	Output	N/A
Return		>0: thread ID other: failure
Note		N/A
Example		static void prvThreadEntry(void *param) { u32 thread_id=OsSysThreadId(); OsSysThreadDelete(thread_id); } u32 threadID = 0;

	int ret=OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID);

6.7 OsSysThreadSuspend

Prototype		void OsSysThreadSuspend(u32 uThread);
Function		Suspend a thread
Parameter	**Input**	uThread: thread ID to suspend
	Output	N/A
Return		N/A
Note		
Example		u32 threadID = 0; static void prvThreadEntry(void* param) { while (1) { OsSysMsleep(1000); } } void test_fun() { int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID); u8 key = 0; while (1) { key = OsKeypadGetKey(); if (key == DIGITAL1) { OsSysThreadSuspend(threadID); } } }

6.8 OsSysThreadResume

Prototype		void OsSysThreadResume(u32 uThread);
Function		Thread Function Recovery
Parameter	**Input**	uThread: thread ID to resume
	Output	N/A

Return	N/A
Note	N/A
Example	u32 threadID = 0; static void prvThreadEntry(void* param) { while (1) { OsSysMsleep(1000); } } void test_fun() { int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID); u8 key = 0; while (1) { key = OsKeypadGetKey(); if (key == DIGITAL1) { OsSysThreadSuspend(threadID); } if (key == DIGITAL2) { OsSysThreadResume(threadID); } } }

6.9 OsSysThreadDelete

Prototype		void OsSysThreadDelete(u32 uThread);
Function		Thread destruction
Parameter	**Input**	uThread: thread ID to delete
	Output	N/A
Return		N/A
Note		The thread deletion function needs to be called explicitly when the thread function ends, otherwise the system will crash.
		u32 threadID = 0; static void prvThreadEntry(void* param) {

Example	int i=0; for(i=0;i<1000;i++){ OsSysMsleep(1000); } OsSysThreadDelete(threadID); } void test_fun() { int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID); u8 key = 0; while (1) { key = OsKeypadGetKey(); if (key == DIGITAL1) { OsSysThreadSuspend(threadID); } } }

6.10 OsSysMalloc

Prototype		void* OsSysMalloc(u32 size);
Function		Heap memory request
Para meter	**Input**	The amount of heap memory to request
	Output	N/A
Return		Non-null: allocated heap memory object NULL: failure
Note		
Example		unsigned char *src=(unsigned char *)OsSysMalloc(32); if(src){ OsSysFree(src); src=NULL; }

6.11 OsSysFree

Prototype	s32 OsSysFree(void* buffer);
Function	Heap memory release

Parameter	**Input**	Objects to be released (NULL objects are supported)
	Output	N/A
Return		0: success other: failure
Note		
Example		unsigned char *src=(unsigned char *)OsSysMalloc(32); if(src){ OsSysFree(src); src=NULL; } OsSysFree(NULL);//acceptable

6.12 OsSysRealloc

Prototype		void* OsSysRealloc(void* buffer, unsigned int size);
Function		Heap memory reallocation
Parameter	**Input**	buffer: the object to be reallocated size: allocation size
	Output	N/A
Return		Non-null: success NULL: failure
Note		
Example		unsigned char *src=(unsigned char *)OsSysMalloc(32); if(src){ src=OsSysRealloc(src, 64); }

6.13 OsSysReboot

Prototype		s32 OsSysReboot();
Function		Restart device
Parameter	**Input**	N/A
	Output	N/A
Return		0: success Other: failure
Note		
Example		OsSysReboot();

6.14 OsSysPowerOff

Prototype		s32 OsSysPowerOff();
Function		Shutdown device
Parameter	**Input**	N/A
	Output	N/A
Return		0: success other: failure
Note		
Example		OsSysPowerOff();

6.15 OsSysSemNew

Prototype		u32 OsSysSemNew(int value);
Function		Create a semaphore
Parameter	**Input**	value: initial value of the semaphore (0 and 1 only)
	Output	N/A
Return		>0: return semaphore handle other: failure
Note		
Example		u32 threadID = 0; u32 sem_id; int num = 0; static void prvThreadEntry(void* param) { while (1) { OsSysSemWait(sem_id); num++; OsSysSemSignal(sem_id); OsSysMsleep(100); } } void test_fun() { sem_id = OsSysSemNew(1); OsLogOutput(0, ”API_sem_new id = % x\r\n”, sem_id); int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID);

	}

6.16 OsSysSemFree

Prototype		void OsSysSemFree(int sem_id);
Function		Release semaphore
Parameter	**Input**	sem_id: semaphore handle
	Output	N/A
Return		0: success other: failure
Note		
Example		u32 threadID = 0; u32 sem_id; int num = 0; static void prvThreadEntry(void* param) { int i=0; for(i=0;i<10;i++){ OsSysSemWait(sem_id); num++; OsSysSemSignal(sem_id); OsSysMsleep(100); } OsSysSemFree(sem_id); OsSysThreadDelete(threadID); } void test_fun() { sem_id = OsSysSemNew(1); OsLogOutput(0, ”API_sem_new id = % x\r\n”, sem_id); int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID); }

6.17 OsSysSemWait

Prototype		void OsSysSemWait(int sem_id);
Function		The semaphore waits all the time until a signal arrives
	Input	sem_id: semaphore handle

Parameter	**Output**	N/A
Return		N/A
Note		
Example		u32 threadID = 0; u32 sem_id; int num = 0; static void prvThreadEntry(void* param) { int i=0; for(i=0;i<10;i++){ OsSysSemWait(sem_id); num++; OsSysSemSignal(sem_id); OsSysMsleep(100); } OsSysSemFree(sem_id); OsSysThreadDelete(threadID); } void test_fun() { sem_id = OsSysSemNew(1); OsLogOutput(0, ”API_sem_new id = %x\r\n”, sem_id); int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID); }

6.18 OsSysSemWaitTimeout

Prototype		bool OsSysSemWaitTimeout(u32 sem_id, u32 timeout);
Function		Wait semaphore, and wait a signal for a certain amount of time
Parameter	**Input**	sem_id: semaphore handle timeout: timeout period (ms)
	Output	N/A
Return		true: get the signal successful false: timeout exit
Note		
Example		u32 threadID = 0; u32 sem_id; int num = 0;

```
static void prvThreadEntry(void* param) {
    int i=0;
     for(i=0;i<10;i++){
          OsSysSemWaitTimeout(sem_id,1000);
          num++;
          OsSysSemSignal(sem_id);
          OsSysMsleep(100);
    }
 OsSysSemFree(sem_id);
    OsSysThreadDelete(threadID);

}

void test_fun() {

    sem_id = OsSysSemNew(1);
    OsLogOutput(0, ”API_sem_new id = % x\r\n”, sem_id);
    int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL,
API_PRIORITY_BELOW_NORMAL, &threadID);
}
```

6.19 OsSysSemSignal

Prototype		void OsSysSemSignal(u32 sem_id)
Function		Semaphore notification
Parameter	**Input**	sem_id: semaphore handle
	Output	N/A
Return		N/A
Note		
Example		u32 threadID = 0; u32 sem_id; int num = 0; static void prvThreadEntry(void* param) { int i=0; for(i=0;i<10;i++){ OsSysSemWaitTimeout(sem_id,1000); num++; OsSysSemSignal(sem_id); OsSysMsleep(100); } OsSysSemFree(sem_id); OsSysThreadDelete(threadID);

	} void test_fun() { sem_id = OsSysSemNew(1); OsLogOutput(0, ”API_sem_new id = % x\r\n”, sem_id); int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID); }

6.20 OsSysMutexCreate

Prototype		u32 OsSysMutexCreate();
Function		Create a mutex
Parameter	**Input**	N/A
	Output	N/A
Return		>0: return a mutex handle other: failure
Note		This mutex is a reentrant lock. Relevant definitions of a reentrant lock can be found online.
Example		u32 threadID = 0; u32 mutex_id; int num = 0; static void prvThreadEntry(void* param) { int i=0; for(i=0;i<10;i++){ OsSysMutexLock(mutex_id); num++; OsSysMutexUnlock(mutex_id); OsSysMsleep(100); } OsSysMutexDelete(mutex_id); OsSysThreadDelete(threadID); } void test_fun() { mutex_id = OsSysMutexCreate(); OsLogOutput(0, ”OsSysMutexCreate id = % x\r\n”, mutex_id); int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID);

	}

6.21 OsSysMutexLock

Prototype		void OsSysMutexLock(u32 mutex_id);
Function		Lock a mutex
Parameter	**Input**	mutex_id: mutex id
	Output	N/A
Return		N/A
Note		This mutex is a reentrant lock.
Example		u32 threadID = 0; u32 mutex_id; int num = 0; static void prvThreadEntry(void* param) { int i=0; for(i=0;i<10;i++){ OsSysMutexLock(mutex_id); num++; OsSysMutexUnlock(mutex_id); OsSysMsleep(100); } OsSysMutexDelete(mutex_id); OsSysThreadDelete(threadID); } void test_fun() { mutex_id = OsSysMutexCreate(); OsLogOutput(0, ”OsSysMutexCreate id = % x\r\n”, mutex_id); int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID); }

6.22 OsSysMutexLockTimeout

Prototype		int OsSysMutexLockTimeout(unsigned int mutex_id, unsigned int timeout)
Function		Lock a mutex with timeout
	Input	Mutex_id：mutex id

Parameter		Timeout：timeout duration(ms)
	Output	N/A
Return		0: success <0: timeout exit other: failure
Note		This mutex is a reentrant lock. Relevant definitions of a reentrant lock can be found online.
Example		u32 threadID = 0; u32 mutex_id; int num = 0; static void prvThreadEntry(void* param) { int i=0; for(i=0;i<10;i++){ OsSysMutexLockTimeout(mutex_id,1000); num++; OsSysMutexUnlock(mutex_id); OsSysMsleep(100); } OsSysMutexDelete(mutex_id); OsSysThreadDelete(threadID); } void test_fun() { mutex_id = OsSysMutexCreate(); OsLogOutput(0, ”OsSysMutexCreate id = %x\r\n”, mutex_id); int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID); }

6.23 OsSysMutexUnlock

Prototype		void OsSysMutexUnlock(unsigned int mutex_id)
Function		Unlock a mutex
Parameter	**Input**	Mutex_id: mutex id
	Output	N/A
Return		N/A
Note		This mutex is a reentrant lock.
		u32 threadID = 0;

Example	u32 mutex_id; int num = 0; static void prvThreadEntry(void* param) { int i=0; for(i=0;i<10;i++){ OsSysMutexLockTimeout(mutex_id,1000); num++; OsSysMutexUnlock(mutex_id); OsSysMsleep(100); } OsSysMutexDelete(mutex_id); OsSysThreadDelete(threadID); } void test_fun() { mutex_id = OsSysMutexCreate(); OsLogOutput(0, ”OsSysMutexCreate id = %x\r\n”, mutex_id); int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID); }

6.24 OsSysMutexDelete

Prototype		void OsSysMutexDelete(unsigned int mutex_id)
Function		Destroy a mutex
Parameter	**Input**	mutex_id: mutex ID
	Output	N/A
Return		0: success
Note		other: failure
		u32 threadID = 0; u32 mutex_id; int num = 0; static void prvThreadEntry(void* param) { int i=0; for(i=0;i<10;i++){ OsSysMutexLockTimeout(mutex_id,1000); num++; OsSysMutexUnlock(mutex_id); OsSysMsleep(100); }

Example	OsSysMutexDelete(mutex_id); OsSysThreadDelete(threadID); } void test_fun() { mutex_id = OsSysMutexCreate(); OsLogOutput(0, ”OsSysMutexCreate id = %x\r\n”, mutex_id); int ret = OsSysThreadCreate(prvThreadEntry, "app_main", 1024 * 50, NULL, API_PRIORITY_BELOW_NORMAL, &threadID); }

6.25 OsSysGetHeapInfo

Prototype		int OsSysGetHeapInfo(unsigned int*memsize, unsigned int*avail_size, unsigned int*avail_block_size)
Function		Retrieve heap memory information
Parameter	**Input**	N/A
	Output	memsize: heap memory size avail_size: available size avail_block_size: available block size
Return		0: success other: failure
Note		
Example		u32 memsize = 0, avail_size=0, avail_block_size = 0; OsSysGetHeapInfo(&memsize,&avail_size,&avail_block_size); OsLogOutput(0,"app start memory:%x,%x,%x\r\n", memsize, avail_size, avail_block_size);

6.26 OsSysQueueCreate

Prototype		unsigned int OsSysQueueCreate(unsigned int count, unsigned int itemsize)
Function		Create a queue
Parameter	**Input**	count: number of queue ITEM itemsize: size of each ITEM
	Output	N/A
Return		>0: created queue id other: failure
Note		

Example	unsigned int queue_id=OsSysQueueCreate(10,1024);

6.27 OsSysQueuePut

Prototype		int OsSysQueuePut(int queue_id,void *item, unsigned int timeout)
Function		Insert data to queue
Parameter	**Input**	queue_id: queue id Item: data Timeout: timeout duration(0 - infinite wait)
	Output	N/A
Return		0: success other: failure
Note		
Example		unsigned int queue_id=OsSysQueueCreate(10,1024); OsSysQueuePut(queue_id,”123456”,0);

6.28 OsSysQueuePutFront

Prototype		int OsSysQueuePutFront(int queue_id, void* item, unsigned int timeout)
Function		Prepend data to queue
Parameter	**Input**	queue_id: queue id Item: data timeout: timeout duration(0 - infinite wait)
	Output	N/A
Return		0: success other: failure
Note		
Example		unsigned int queue_id=OsSysQueueCreate(10,1024); OsSysQueuePutFront(queue_id,”123456”,0);

6.29 OsSysQueueGet

Prototype		int OsSysQueueGet(int queue_id, void* item,unsigned int timeout)
Function		Retrieve data from queue
Parameter	**Input**	queue_id: queue id timeout: timeout duration
	Output	item: retrieved data buffer
Return		0: success

	other: failure
Note	
Example	unsigned char data[1024]={0}; unsigned int queue_id=OsSysQueueCreate(10,1024); OsSysQueuePutFront(queue_id,data,0);

6.30 OsSysQueueDelete

Prototype		void OsSysQueueDelete(int queue_id)
Function		Delete a existing queue
Parameter	**Input**	queue_id: queue id
	Output	N/A
Return		0: success other: failure
Note		The prerequisite is that the queue needs to be successfully created in advance.
Example		unsigned int queue_id=OsSysQueueCreate(10,1024); OsSysQueueDelete(queue_id);

6.31 OsSysQueueSpaceAvail

Prototype		unsigned int OsSysQueueSpaceAvail(unsigned int queue_id)
Function		Get remained space in queue
Parameter	**Input**	queue_id: queue id
	Output	N/A
Return		Remaining available queue size
Note		The prerequisite is that the queue needs to be successfully created in advance.
Example		unsigned int queue_id=OsSysQueueCreate(10,1024); unsigned int free_size=OsSysQueueSpaceAvail(queue_id);

6.32 OsSysQueueClear

Prototype		void OsSysQueueClear(unsigned int queue_id)
Function		Clear a existing queue
Parameter	**Input**	queue_id: successfully created queue id
	Output	N/A
Return		N/A

Note	The prerequisite is that the queue needs to be successfully created in advance.
Example	unsigned int queue_id=OsSysQueueCreate(10,1024); OsSysQueueClear(queue_id);

6.33 OsSysGenerateRandom

Prototype		int OsSysGenerateRandom(void *buf,unsigned int len)
Function		Generate random numbers
Parameter	**Input**	Len: length
	Output	buf: buffer
Return		0: success other: failure
Note		Hardware-Generated True Random Numbers.
Example		char buf[16]={0}; OsSysGenerateRandom(buf,sizeof(buf));

6.34 OsSysGetSysVersion

Prototype		int OsSysGetSysVersion(char* version)
Function		Retrieve system version information
Parameter	**Input**	N/A
	Output	version: version information buffer
Return		0: success other: failure
Note		
Example		char version[32]; memset(version,0,sizeof(version)); OsSysGetSysVersion(version);

6.35 OsSysSetRtcTime

Prototype		s32 OsSysSetRtcTime(s8 *pTimestr);
Function		Set RTC time
Parameter	**Input**	pTimestr: time string
	Output	N/A
Return		0: success other: failure

Note	
Example	OsSysSetRtcTime(“20240102121212”);

6.36 OsSysGetRtcTime

Prototype		s32 OsSysGetRtcTime(s8* timeStr)
Function		Retrieve RTC time
Parameter	**Input**	N/A
	Output	timeStr: data buffer
Return		0: success other: failure
Note		
Example		s8 timeStr[15]; memset(timeStr,0,sizeof(timeStr)); OsSysGetRtcTime(timeStr); //“20240102121212”

6.37 OsSysReadSn

Prototype		s32 OsSysReadSn(u8* tsn, u32* tsnlen, u8* csn, u32* csnlen)
Function		Retrieve terminal serial number
Parameter	**Input**	N/A
	Output	tsn: local terminal serial number(the tsn Sunyard set) tsnlen: retrieved tsn length csn: Client Serial Number (Parameter preset, allowing customers to set their own terminal serial numbers) csnlen: retrieved csn length
Return		0: success other: failure
Note		
Example		u8 tsn[24]; u32 tsnlen; u8 csn[24]; u32 csnlen; s32 ret=OsSysReadSn(tsn, &tsnlen, csn, &csnlen);

6.38 OsSysGetFirmVersion

Prototype		s32 OsSysGetFirmVersion (u8 *lpOut, u32 nType);
Function		Retrieve SP firmware version information
Parameter	**Input**	nType: types of firmware See below: #define AP_BOOT_VER 0x00 #define AP_CORE_VER 0x01 #define AP_VSM_VER 0x02 #define ROM_VER 0x04 #define SP_BOOT_VER 0x10 #define SP_CORE_VER 0x11 #define SP_SECLIB_VER 0x12
	Output	lpOut: firmware version information buffer
Return		0: success other: failure
Note		
Example		unsigned char data[24]; s32 ret=OsSysGetFirmVersion (data, AP_BOOT_VER);

6.39 OsSysGetHwc

Prototype		s32 OsSysGetHwc(hwc_list_t *hwc_list);
Function		Retrieve the hardware configuration information
	Input	N/A
		hwc_list: retrieves the hardware configuration information structure information as follows: typedef struct __hwc_list { u32 crc; u8 ProjectItem[32];// Project type u8 HWboomVer[32]; // Hardware BOOM version u8 wireless; // N/A line module hwc_wireless_ts u8 bluetooth; // Bluetooth u8 modem; // modem hwc_modem_t u8 net; // Ethernet u8 wifi; // wifi u8 gps; // GPS

Parameter	**Output**	u8 scanner; // Barcode scanner hwc_scanner_t u8 camera; // Camera u8 fingerprint; // Fingerprint u8 icc; // IC card u8 nfc; // NFC card u8 mag; // Magnetic card u8 idcard; // ID card scanner module u8 touchscreen; // Touchscreen hwc_touchscreen_t u8 sdc; // SD card u8 printer; // Printer hwc_printer_t u8 audio; // Speaker u8 nes; // National1 Encryption Standard hwc_nes_t u8 psam; // psam card u8 beep; // Buzzer u8 esim; // esim card :0 N/Aesim 0x01 :USIM 0x02： u8 flash; // External flash u8 cameraled; // Camera light :0 N/A 0x02 Red 0x04 White 0x06 white + red u8 LSI; // LSI type: 0 External, 1 Internal, 2 Internal with RF-HSI calibrationu8 LCD; // LCD type: 0 RGB, 1 Monochrome 128*96 ST7571, 2 Monochrome 128*96 UC1617S, 3 Monochrome 128*64 ST7567A u8 SEG; // Segmented display u8 res1; // Reserved u8 res2; // Reserved u8 res3; // Reserved }hwc_list_t;
Return		0: success other: failure
Note		It is necessary to burn the hardware configuration table information file to the device in advance.
Example		hwc_list_t hwc_list; OsSysGetHwc(&hwc_list);

6.40 OsSysInit

Prototype		void* OsSysInit(u8 *ver);
Function		System initialization function (including initialization of tasks such as logs, flash, buttons, voice, etc.)
Parameter	**Input**	ver:app version info
	Output	N/A
Return		Return signal event handle
Note		The main entry function of the DS50 program is appimg_enter, so the application needs to call this interface in appimg_enter. This function is indispensable and contains the initialization logic of the system.

Example	Void * appimg_enter (void * param)//Main entry of the function { void *p = OsSysInit("V1.0.0");// System initialization //Application logic development return p; }

6.41 OsGetSleepState

Prototype		s32 OsGetSleepState(void);
Function		check current device sleep state
Parameter	**Input**	N/A
	Output	N/A
Return		Return 0 is unsleep 1 is sleep
Note		
Example		API_LOG_DEBUG("%d",OsGetSleepState());

6.42 OsSysAppLock

Prototype		void OsSysAppLock();
Function		Application locking
Parameter	**Input**	N/A
	Output	N/A
Return		N/A
Note		When the application is handling tasks, if you don't want the system to enter sleep mode, you can use this interface
Example		

6.43 OsSysAppUnLock

Prototype	void OsSysAppUnLock();
Function	Application unlocking

Parameter	**Input**	N/A
	Output	N/A
Return		N/A
Note		It is used in conjunction with the OsSysAppLock interface to prevent the system from failing to enter sleep mode.It is worth noting that this interface must appear in pairs with OsSysAppUnLock() to prevent failure to enter sleep mode.
Example		

6.44 OsSysGetAppLock

Prototype		int OsSysGetAppLock();
Function		get the count of application lock
Parameter	**Input**	N/A
	Output	N/A
Return		Obtain the current number of application locks.
Note		If the number of application locks is not zero, the system cannot enter sleep mode.
Example		API_LOG_DEBUG(“lock cnt:%d\r\n”,OsSysGetAppLock());

6.45 OsSysUpdateLogo

Prototype		int OsSysUpdateLogo(char* filename);
Function		Update the startup logo
Parameter	**Input**	filename:The file path to be updated (absolute filepath)
	Output	N/A
Return		0-success,other fail
Note		Currently, only 24-bit format bmp is supported
Example		OsSysUpdateLogo(“/ext/tmp.bmp”);

6.46 OsSysSetTmsHandle

Prototype		void OsSysSetTmsHandle(int (*func)());
Function		Update the startup logo
Parameter	**Input**	func:tms Function handle
	Output	N/A
Return		N/A
Note		This interface is designed to support the tms functions custom-developed by customers.It needs to be called before the OsSysInit interface.
Example		int tms_func(){ return 0; } OsSysSetTmsHandle(tms_func);

6.47 OsSysSetSecHandle

Prototype		void OsSysSetSecHandle(void (*func)(int, char*));
Function		set security function handle
Parameter	**Input**	func:sec Function handle The first parameter in the callback function is the length of the trigger code, and the second parameter is the content of the trigger code.
	Output	N/A
Return		N/A
Note		When the device is triggered, the relevant trigger information will be notified through this interface.It needs to be called before the OsSysInit interface
Example		int sec_func(int codelen,char *code){ API_DEBUG_LOG(“codelen:%d,%s\r\n”,codelen,code); return 0; } OsSysSetSecHandle(sec_func);

6.48 OsSecSmgtGetRunningMode

Prototype	int OsSecSmgtGetRunningMode(u8 *dataout, u8 *datalen)
Function	get device running mode

Parameter	**Input**	N/A
	Output	dataout: datalen:
Return		0-success other:fail
Note		
Example		unsigned char tmpbuf[16] = { 0 }; unsigned int tmplen = 0; int runmode = 0; OsSecSmgtGetRunningMode(tmpbuf, &tmplen); memcpy(&runmode, tmpbuf, tmplen); API_LOG_DEBUG("runmode:%d,SMGT_KM_PROD:%d\r\n", runmode, SMGT_KM_PROD); if (runmode == SMGT_KM_PROD) { }

7 Audio Interface

7.1 Summary

The audio module mainly provides functions for playing audio files and tts.

7.2 error code

#define API_AUDIOPLAY_ERROR -2001
#define API_AUDIOCLEAR_ERROR -2002
#define API_AUDIOSTOP_ERROR -2003
#define API_AUDIOSETVOLUME_ERROR -2004
#define API_AUDIOGETVOLUME_ERROR -2005
#define API_AUDIOSETTTSSPEED_ERROR -2006
#define API_AUDIOPLAYLIST_ERROR -2007

7.3 OsAudioPlay

Prototype		int OsAudioPlay(char* filename);
Function		Play audio files (supports MP3, WAV, PCM, AMR)
Parameter	**Input**	filename: file name (absolute path)
	Output	N/A
Return		0: success other: failure

Note	no prerequisite function, can be directly called. The file name must be all in lowercase or uppercase.
Example	OsAudioPlay(“/ext/audio/1.mp3”);

7.4 OsAudioTtsPlay

Prototype		int OsAudioTtsPlay(char* ttsbuf, int tts_type);
Function		TTS voice broadcasting
Parameter	**Input**	ttsbuf: the tts string to be broadcasted (in hexadecimal format) tts_type: 0-utf-8, 1-GB2312, 2-unicode
	Output	N/A
Return		0: success other: failure
Note		Only English supported
Example		OsAudioTtsPlay (“77656C636F6D65”,0); // Play “welcome” - utf8 encoding OsAudioTtsPlay (“77656C636F6D65”,1); // Play “welcome” - gb2312 encoding OsAudioTtsPlay (“770065006C0063006F006D006500”,2); // Play “welcome” - unicode encoding

7.5 OsAudioClear

Prototype		int OsAudioClear();
Function		Clear the remaining audio queue content
Parameter	**Input**	N/A
	Output	N/A
Return		0: success other: failure
Note		no prerequisite function, can be directly called.
Example		OsAudioClear();

7.6 OsAudioStop

Prototype		int OsAudioStop();
Function		Stop voice playback
Parameter	**Input**	N/A
	Output	N/A

Return	0: success other: failure
Note	no prerequisite function, can be directly called.
Example	OsAudioStop();

7.7 OsAudioSetVolume

Prototype		int OsAudioSetVolume(int mode, int volume);
Function		Set the volume of voice broadcast
Parameter	**Input**	mode: 0-voice, 1-play, 2-tone volume: 0-4
	Output	N/A
Return		0:success other: failure
Note		no prerequisite function, can be directly called.
Example		OsAudioSetVolume(1, 1);

7.8 OsAudioGetVolume

Prototype		int OsAudioGetVolume(int mode)
Function		Obtain the volume levels for each voice broadcast mode
Parameter	**Input**	mode: Audio mode 0-voice (call sound) 1-play (audio playback sound) 2-tone (prompt sound)
	Output	N/A
Return		0: success other: failure
Note		no prerequisite function, can be directly called.
Example		OsAudioGetVolume(1);

7.9 OsAudioGetState

Prototype		int OsAudioGetState();
Function		Get the current status of the voice broadcast
Parameter	**Input**	N/A
	Output	N/A

Return	0: not broadcasted 1: currently broadcasting
Note	no prerequisite function, can be directly called.
Example	OsAudioGetState();

7.10 OsAudioTtsGetState

Prototype		int OsAudioTtsGetState();
Function		Obtain TTS broadcast status
Parameter	**Input**	
	Output	N/A
Return		1- currently broadcasting 0- not broadcasted
Note		no prerequisite function, can be directly called.
Example		OsAudioTtsGetState();

7.11 OsAudioTtsSetSpeed

Prototype		int OsAudioTtsSetSpeed(int speed);
Function		Set TTS broadcast speed
Parameter	**Input**	speed：-32768~32767
	Output	N/A
Return		0: success other: failure
Note		The default is zero.
Example		OsAudioTtsSetSpeed(0);

7.12 OsAudioTtsSetPitch

Prototype		int OsAudioTtsSetPitch(int pitch);
Function		Set TTS broadcast tone
Parameter	**Input**	pitch:-32768~32767
	Output	N/A
Return		0: success other: failure
Note		The default is zero.

Example	OsAudioTtsSetPitch(0);

7.13 OsAudioPlaylist

Prototype		int OsAudioPlaylist(char* filenamelist[][128], int listSize);
Function		Play the list of audio files
Parameter	**Input**	filenamelist:List of file name arrays listSize:list size
	Output	N/A
Return		0: success other: failure
Note		
Example		char filenamelist[2][128]={"/ext/1.mp3","/ext/2.mp3"}; OsAudioPlaylist(filenamelist,2);

8 Data Encoding

8.1 Summary

This module provides functions for encoding and decoding data.

8.2 error code

#define API_BASE64ENCODE_ERROR -0x3001
#define API_BASE64DECODE_ERROR -0x3002

8.3 OsBase64Encode

Prototype		int OsBase64Encode(unsigned char* dst, unsigned int dlen, unsigned int* olen, const unsigned char* src, unsigned int slen)
Function		Base64 encods
Parameter	**Input**	src: source string slen: source string length
	Output	dst: encoded string buffer dlen: size of the encoded string buffer olen: actual length after encoding
Return		0: success other: failure
Note		

Example	unsigned char dst[12]; unsigned int olen; int ret=OsBase64Encode(dst, sizeof(dst),&olen,”123”,3);

8.4 Osbase64Decode

Prototype		int Osbase64Decode(unsigned char* dst, unsigned int dlen, unsigned int* olen, const unsigned char* src, unsigned int slen)
Function		Base64 decodes
Parameter	**Input**	src: source string slen: source string length
	Output	dst: decoded string buffer dlen: size of the decoded string buffer olen: actual length after decoding
Return		0: success other: failure
Note		
Example		unsigned char dst[12]; unsigned int olen; int ret=Osbase64Decode(dst,sizeof(dst),&olen,”EgM=”,4);

9 Log Interface

9.1 Summary

This module provides functions for log output, which includes both string and hexadecimal forms of output.

9.2 error code

9.3 OsLogSetLevel

Prototype		void OsLogSetLevel(unsigned char level)
Function		Set the current log level
Parameter	**Input**	level: The maximum is 4, as detailed below, enum { DEBUG_LEVEL=0, INFO_LEVEL=1, WARNING_LEVEL=2, ERROR_LEVEL=3, MAX_LEVEL=4,

		};
	Output	N/A
Return		N/A
Note		After setting the log level, any log level lower than the set level will be outputted, and the system defaults to MAX_LEVEL.
Example		OsLogSetLevel(ERROR_LEVEL);

9.4 OsLogSwitch

Prototype		void OsLogSwitch(int val);
Function		Set the log switch
Parameter	**Input**	val: -1- close the log 0 - output logs through the coolwatcher.
	Output	N/A
Return		N/A
Note		The system log is off by default. Currently, 0 is temporarily supported. It needs to be called before the OsSysInit() function.
Example		Void * appimg_enter (void * param)//Main entry of the function { OsLogSwitch(0); void *p = OsSysInit("V1.0.0");// System initialization //Application logic development return p; }

9.5 OsLogOutput

Prototype		void OsLogOutput(unsigned char level, const char* format, ...);
Function		Log outputs (in string format)
Parameter	**Input**	level: log level (any level lower than the current log level will output) format: format string ...: variable parameter
	Output	N/A
Return		N/A
Note		To output logs, the application needs to call OsLogSwitch (0)

Example	Void * appimd_enter (void * param)//Main entry of the function { OsLogSwitch(0);// Open the log and go to cooltracker log p = OsSysInit("V1.0.0"); return p; } //Application development OsLogOutput(DEBUG_LEVEL,"%s\r\n","error");

9.6 OsLogHexOutput

Prototype		void OsLogHexOutput(unsigned char level, unsigned char* data, int len)
Function		Hexadecimal prints log
Parameter	**Input**	level: log level data: string len: string length
	Output	N/A
Return		N/A
Note		To output logs, the application needs to call OsLogSwitch (0)
Example		Void * appimd_enter (void * param)//Main entry of the function { OsLogSwitch(0);// Open the log and go to cooltracker log p = OsSysInit("V1.0.0"); return p; } //Application development char *data="123"; OsLogHexOutput(DEBUG_LEVEL,data,3);

9.7 OsLogSetPause

Prototype		void OsLogSetPause();
Function		Stop log output
	Input	N/A

Parameter	**Output**	N/A
Return		N/A
Note		
Example		OsLogSetPause();

9.8 OsLogSetResume

Prototype		void OsLogSetResume();
Function		Log recovery output
Parameter	**Input**	N/A
	Output	N/A
Return		N/A
Note		
Example		OsLogSetResume();

10 Timer Interface

10.1 Summary

This module mainly provides functions for timer operations, including create, stop, and destroy timers.

10.2 error code

#define API_TIMERCREATE_ERROR -0x4001
#define API_TIMERFREE_ERROR -0x4002

10.3 OsTimerCreate

Prototype		unsigned int OsTimerCreate(unsigned int timer_ms, bool single, void (*fn)(void* arg), void* arg)
Function		Create a timer
Parameter	**Input**	timer_ms: timer time (in milliseconds) single: is it a single timer fn: timer function arg: parameter to be passed into the timer function
	Output	N/A
Return		>0: timer handle other: failure
Note		The N/A pre function can be called directly.

Example	void timer_callback(void *args){ OsLogOutput(0,"hello world\r\n"); } unsigned int id=OsTimerCreate(1000,false,timer_callback,NULL);

10.4 OsTimerStop

Prototype		bool OsTimerStop(unsigned int timerid)
Function		Stop timer
Parameter	**Input**	timerid: timer handle
	Output	N/A
Return		true: success false: failed
Note		The pre function is OsTimerCreate
Example		unsigned int id=OsTimerCreate(1000,false,timer_callback,NULL); OsTimerStop(id);

10.5 OsTimerFree

Prototype		bool OsTimerFree(unsigned int timerid)
Function		Destroy a timer
Parameter	**Input**	timerid: timer handle
	Output	N/A
Return		true: success false: failed
Note		The pre functions are OsTimerCreate and OsTimerStop
Example		unsigned int id=OsTimerCreate(1000,false,timer_callback,NULL); OsTimerStop(id); OsTimerFree(id);

11 File Interface

11.1 Summary

The file system mainly provides necessary data storage for applications, supporting both on-chip and off-chip file system operations. Due to limited on-chip file system resources, applications primarily use off-chip file systems to store data. The starting path of the on-chip file system is /, while the starting path of the off-chip file system is /ext.

11.2 error code

#define API_FILEOPEN_ERROR -0x5001
#define API_FILEREAD_ERROR -0x5002
#define API_FILEWRITE_ERROR -0x5003
#define API_FILECLOSE_ERROR -0x5004
#define API_FILESEEK_ERROR -0x5005
#define API_FILEREMOVE_ERROR -0x5006
#define API_FILEUNLINK_ERROR -0x5007
#define API_FILEGETSIZE_ERROR -0x5008
#define API_FILEGETFILESIZE_ERROR -0x5009
#define API_FILERENAME_ERROR -0x5010
#define API_FILEMKDIR_ERROR -0x5011
#define API_FILERMDIR_ERROR -0x5012
#define API_FILEUNZIP_ERROR -0x5013
#define API_FILELSTDIR_ERROR -0x5014
#define API_FILEFORMAT_ERROR -0x5015
#define API_FILECOPY_ERROR -0x5016

11.3 OsFileOpen

Prototype		int OsFileOpen(char* pchFileName, unsigned int ucMode)
Function		File opens
Parameter	**Input**	pchFileName: file name ucMode: open mode, with the following types: # DefiniteAPI_SDONLY //Open in read-only mode # DefiniteAPI_SRONLY //Open in write only mode # DefiniteAPI_SDWR //Open in read-write mode # DefiniteAPI_SPPEND //Open file as append # If this option is available when open a file, create a file if it does not exist # If the file exists, change its length to 0 # DefiniteAPI.exe CL //is mainly used in conjunction with API_CEAT. If the file exists, the creation will fail For more details, please refer to api_file.h
	Output	N/A
Return		>0: file handle other: failure
Note		The file name needs to be passed in an absolute path, with an on-chip starting path of/off chip starting path/ext Example of on-chip operation file: OsFileOpen(“/123.txt”,API_WRONLY|API_CREAT) Off chip file operation Example: OsFileOpen(“/ext/123.txt”,API_WRONLY|API_CREAT) Due to limited on-chip system space resources, it is recommended to operate the off chip file system.

Example	int fp=OsFileOpen(“/ext/1.txt”,API_RDONLY); OsFileClose(fp);

11.4 OsFileRead

Prototype		int OsFileRead(int iFileId, unsigned char* pucOutData, unsigned int iReadLen);
Function		File reads
Parameter	**Input**	iFileId: open success file handle
	Output	pucOutData: buffer to be read iReadLen: length to be read
Return		>=0: actual length read other: failure
Note		It is best that the data read each time does not exceed 1024 bytes.
Example		char buf[12]; int fp=OsFileOpen(“/ext/1.txt”,API_RDONLY); OsFileRead(fp,buf,12); OsFileClose(fp);

11.5 OsFileWrite

Prototype		int OsFileWrite(int iFileId, unsigned char* pucInData, unsigned int iWriteLen);
Function		File writes
Parameter	**Input**	iFileId: file handle pucInData: data to be written iWriteLen: length to be written
	Output	N/A
Return		>=0: actual length of data written other: failure
Note		It is best that the data written each time does not exceed 1024 bytes.
Example		char buf[12]; int fp=OsFileOpen(“/ext/1.txt”,API_WRONLY); OsFileWrite(fp,buf,12); OsFileClose(fp);

11.6 OsFileClose

Prototype	int OsFileClose(int iFileId);
Function	File closes

Parameter	**Input**	iFileId: File Handle
	Output	N/A
Return		0: success other: failure
Note		
Example		int fp=OsFileOpen("/ext/1.txt",API_WRONLY); OsFileClose(fp);

11.7 OsFileSeek

Prototype		int OsFileSeek(int iFileId, int lOffset, unsigned char iMode);
Function		File offset handling
Parameter	**Input**	iFileId: file handle lOffset: the size to be offset iMode: there are three options for offsetting the starting position: API_SEEK_SET moves offset bytes from the beginning of the file header; API_SEEK_CUR moves offset bytes from the current position of the file pointer; API_SEEK_END moves offset bytes from the end position of the file. (It will only return to the end position of the file when it's greater than 0 while not enlarge the file.)
	Output	N/A
Return		>=0: return to the current file handle location other: failure
Note		The pre function is OsFileOpen
Example		int fp=OsFileOpen("/ext/1.txt",API_RDONLY); OsFileSeek(fp,0,API_FS_SEEK_SET); OsFileClose(fp);

11.8 OsFileRemove

Prototype		int OsFileRemove(char* pchFileName);
Function		File deletion
Parameter	**Input**	pchFileName: filename (absolute path)
	Output	N/A
Return		0: success other: failure
Note		To delete a file, you must close it first and ensure that no other threads opened the file.
Example		OsFileRemove("/ext/1.txt");

11.9 OsFileGetFileSize

Prototype		long OsFileGetFileSize(char* pchFileName);
Function		Get file size
Parameter	**Input**	pchFileName: filename (absolute path)
	Output	N/A
Return		>=0: file size other: failure
Note		N/A pre function
Example		long size=OsFileGetFileSize("/ext/1.txt");

11.10 OsFileExist

Prototype		int OsFileExist(char* pchFileName)
Function		Determine if the file exists
Parameter	**Input**	pchFileName: File Name
	Output	N/A
Return		1: existence -1: does not exist
Note		N/A pre function
Example		int ret=OsFileExist("/ext/1.txt");

11.11 OsFileReName

Prototype		int OsFileReName(char* pchOldFileName, char* pchNewFileName)
Function		File rename
Parameter	**Input**	pchOldFileName: Old file name pchNewFileName: New file name
	Output	N/A
Return		0: success other: failure
Note		N/A pre function
Example		int ret=OsFileReName("/ext/1.txt","/ext/2.txt");

11.12 OsFileMkdir

Prototype	int OsFileMkdir(char* pchDirName)

Function		Create a folder
Parameter	**Input**	pchDirName: folder name
	Output	N/A
Return		0: success other: failure
Note		N/A pre function
Example		int ret=OsFileMkdir("/ext/temp");

11.13 OsFileRmdir

Prototype		int OsFileRmdir(char* pchDirName)
Function		Remove folders
Parameter	**Input**	pchDirName: folder name
	Output	N/A
Return		0: success other: failure
Note		No prerequisite function, only supports deleting empty folders.
Example		int ret=OsFileRmdir("/ext/1");

11.14 OsFileLstdir

Prototype		int OsFileLstdir(char* pchDirName, fileinfo_t* fileinfo);
Function		Traverse folder
Parameter	**Input**	pchDirName: Folder Name (absolute path)
	Output	fileinfo: Obtained file (folder) information typedef struct fileinfo_t{ char dirname[64];// folder char filename[128];// file unsigned int filesize;// file size }fileinfo_t;
Return		>0: number of files (folders) obtained other: failure
Note		No prerequisite function, does not support recursive retrieval of all file information.
Example		fileinfo_t fileinfo[10]; OsFileLstdir("/ext/audio",fileinfo);

11.15 OsFileUnzip

Prototype		int OsFileUnzip(const char* file_path, const char* zip_path)
Function		Extract a zip file to a specified path
Parameter	**Input**	file_cath: zip package path (absolute path,/ext/temp. zip) zip_path: Unzip zip path (absolute path)
	Output	N/A
Return		0: success other: failure
Note		When extracting to the specified directory, it is necessary to create the folder in advance, otherwise the extraction will fail. For example, extracting/ext/tempzip to/ext/temp2 requires creating/ext/temp2 in advance
Example		int ret=OsFileUnzip(“/ext/temp.zip”, “/ext/temp2”);

11.16 OsFileTypeCheck

Prototype		int OsFileTypeCheck(char *filepath)
Function		File type determination
Parameter	**Input**	filepath: file path
	Output	N/A
Return		1: folder 0: file <0: the file does not exist
Note		No prerequisite function, file or directory needs to exist.
Example		int ret=OsFileTypeCheck(“/ext/1.txt”);

11.17 OsFileGetFileSysFreeSize

Prototype		long OsFileGetFileSysFreeSize(void);
Function		Retrieve the remaining size of the file system
Parameter	**Input**	N/A
	Output	N/A
Return		>=0 remaining size (bytes) <0 Failure
Note		N/A pre function.
Example		long freesize=OsFileGetFileSysFreeSize();

12 Wireless Interface

12.1 Summary

This module provides functions for applications to use SIM cards for network communication. It includes network registration, de-registration, network information query, and socket (SSL) operations.

12.2 error code

```
#define API_WIRELESSGETIP_ERROR -0x6001
#define API_WIRELESSPING_ERROR -0x6002
#define API_WIRELESSPDPACTIVE_ERROR -0x6003
#define API_WIRELESSPDPDEACTIVE_ERROR -0x6004
#define API_WIRELESSNTP_ERROR -0x6005
#define API_WIRELESSGETLAC_ERROR -0x6006
#define API_WIRELESSGETCELL_ERROR -0x6007
#define API_WIRELESSGETSIM_ERROR -0x6008
#define API_WIRELESSGETCCID_ERROR -0x6009
#define API_WIRELESSGETCSQ_ERROR -0x6010
#define API_WIRELESSGETIMEI_ERROR -0x6011
#define API_WIRELESSGETMNC_ERROR -0x6012
```

12.3 OsWlSetPdpConfig

Prototype		void OsWlSetPdpConfig(char* apn, char* usrname, char* pwd);
Function		Configure APN, username, and password before PDP activation
Parameter	**Input**	apn: apnName usrname: usrname pwd: password
	Output	N/A
Return		N/A
Note		Call this interface to set the APN information of the SIM card installed, and remember to call it before the OsSysInit function.
Example		void* appimg_enter(void* param)//Main entrance of the program { int* p = NULL; OsLogSwitch(0); OsWlSetPdpConfig(“CMNET”, NULL, NULL);//config apn Parameter p = **OsSysInit**(“V1.0.0”); return p; }

12.4 OsWlPdpActive

Prototype		int OsWlPdpActive(int simid);
Function		Pdp active
Parameter	**Input**	simid: SIM index(only support 0)
	Output	N/A
Return		0: success other: failure
Note		Automatically called in OsSysInit, no need to call it in application.
Example		OsWlPdpActive(0);

12.5 OsWlPdpDeactive

Prototype		int OsWlPdpDeactive(int simid);
Function		Pdp deactive Function
Parameter	**Input**	simid: SIM index(only support 0)
	Output	N/A
Return		0: success other: failure
Note		
Example		int ret=OsWlPdpDeactive(0);

12.6 OsWlPdpGetStatus

Prototype		int OsWlPdpGetStatus(int simid);
Function		Obtain PDP activation status
Parameter	**Input**	simid:SIM index(only support 0)
	Output	N/A
Return		1: success 0: fail
Note		Please insert the SIM card.
Example		int ret=OsWlPdpGetStatus(0);

12.7 OsWlGetNetState

Prototype	int OsWlGetNetState();

Function		Get wireless network status
Parameter	Input	N/A
	Output	N/A
Return		1- connected, 0- not connected
Note		Please insert the SIM card.
Example		int ret=OsWlGetNetState();

12.8 OsWlGetMncMcc

Prototype		int OsWlGetMncMcc(int* mcc, int* mnc);
Function		Wireless MCC/MNC value
Parameter	Input	N/A
	Output	mcc: obtained mcc value mnc: obtained mnc value
Return		0: success other: failure
Note		Please insert the SIM card.
Example		int mcc,mnc; int ret=OsWlGetMncMcc(&mcc, &mnc);

12.9 OsWlGetLacInfo

Prototype		int OsWlGetLacInfo(int* gsm_lac, int* lte_lac);
Function		Get wireless lac info
Parameter	Input	N/A
	Output	gsm_lac: gsm lac lte_lac: lte lac
Return		0: success other: failure
Note		Please insert the SIM card.
Example		int gsm_lac,lte_lac; int ret= OsWlGetLacInfo(&gsm_lac,&lte_lac);

12.10 OsWlGetCellId

Prototype	int OsWlGetCellId(int* gsm_cell_id, int* lte_cell_id);

Function		Get wireless cell id
Parameter	Input	N/A
	Output	gsm_cell_id: gsm cell id lte_cell_id: lte cell id
Return		0: success other: failure
Note		Please insert the SIM card.
Example		int gsm_cell_id, lte_cell_id; int ret=OsWlGetCellId(&gsm_cell_id, &lte_cell_id);

12.11 OsWlNtp

Prototype		int OsWlNtp(char* pucIP, unsigned int uiPort, unsigned int timeout);
Function		Synchronize NTP time based on NTP server
Parameter	Input	pucIP: ntp server IP uiPort: ntp server port timeout: timeout(ms)
	Output	N/A
Return		0: success other: failure
Note		The prerequisite is a successful network connection.
Example		int ret=OsWlNtp(“ntp.com”,123,10000);

12.12 OsWlPing

Prototype		int OsWlPing(int type, char* host, int ping_cnt, void(*fun)(int, char*))
Function		Wireless ping interface
Parameter	Input	type: 0-stop, 1-ipv4, 2-ipv6 host: host address ping_cnt: ping counts(1-255) fun: ping callback fun
	Output	N/A
Return		0: success other: failure
Note		The prerequisite is a successful network connection.
Example		static void pingRecvfun(int type,char* str) {

	//The ping test result information is in str } OsWlPing(1,"www.sunyard.com",3,pingRecvfun);

12.13 OsWlGetSimStatus

Prototype		int OsWlGetSimStatus(unsigned char* status);
Function		Determine if the SIM card is inserted
Parameter	**Input**	N/A
	Output	Status: 1-inserted, 0-not inserted
Return		0: success other: failure
Note		
Example		unsigned char status; int ret=OsWlGetSimStatus(&status);

12.14 OsWlGetCcid

Prototype		int OsWlGetCcid(unsigned char* ccid);
Function		Get wireless ccid
Parameter	**Input**	N/A
	Output	ccid: ccid value
Return		0: success other: failure
Note		Please insert the SIM card.
Example		unsigned char ccid; int ret=OsWlGetCcid(&ccid);

12.15 OsWlGetCsq

Prototype		int OsWlGetCsq(int* rssi, int* ber);
Function		Get wireless csq
Parameter	**Input**	N/A
	Output	rssi: RSSI signal value (0~31, 99 abnormal) ber: Error rate value
Return		0: success other: failure

Note	Please insert the SIM card.
Example	int rssi,ber; int ret=OsWlGetCsq(&rssi, &ber);

12.16 OsWlGetImeiImsi

Prototype		int OsWlGetImeiImsi(unsigned char* imei, unsigned char* imsi);
Function		Get wireless module imei and imsi info
Parameter	**Input**	N/A
	Output	imei: imei number imsi: imsi number
Return		0: success other: failure
Note		
Example		unsigned char imei[32],imsi[32]; int ret=OsWlGetImeiImsi(imei,imsi);

12.17 OsWlGetOperatoName

Prototype		int OsWlGetOperatoName(char* operatorname, int bufsize);
Function		Obtain SIM card operator name
Parameter	**Input**	N/A
	Output	operatorname: operator name bufsize: buffer size
Return		0: success other: failure
Note		Please insert the SIM card.
Example		char operatorname[24]; int bufsize=24; int ret=OsWlGetOperatoName(operatorname,bufsize);

12.18 OsWlSocketCreate

Prototype		int OsWlSocketCreate(int net_type);
Function		Create socket
Parameter	**Input**	net_type: 0-ipv4, 1-ipv6
	Output	N/A

Return	>=0: socket handle other: failure
Note	The prerequisite is a successful network connection.
Example	int fd=OsWlSocketCreate(0);

12.19 OsWlSocketClose

Prototype		int OsWlSocketClose(int sockid);
Function		Close socket
Parameter	**Input**	sockid: socket handle
	Output	
Return		0: success other: failure
Note		
Example		int fd=OsWlSocketCreate(0); OsWlSocketClose(fd);

12.20 OsWlSocketConnect

Prototype		int OsWlSocketConnect(int nettype, int sockid, char* pIP, char* pPort)
Function		Socket connects
Parameter	**Input**	nettype: 0-ipv4 1-ipv6 sockid: socket handle pIP: server ip pPort: server port
	Output	N/A
Return		0: success other: failure
Note		The prerequisite is a successful network connection.
Example		int fd=OsWlSocketCreate(0); OsWlSocketConnect(0,fd,”www.baidu.com”,80); OsWlSocketClose(fd);

12.21 OsWlSocketSend

Prototype	int OsWlSocketSend(int sockid, unsigned char* pucdata, unsigned int iLen);
Function	Socket sends data

Parameter	**Input**	sockid: socket handle pucdata: data iLen: datalen
	Output	N/A
Return		>=0: actual length of data sent <0: sending failed
Note		The prerequisite is a successful network connection.It is best that the data sent each time does not exceed 1024 bytes.
Example		int fd=OsWlSocketCreate(0); OsWlSocketConnect(0,fd,"www.baidu.com",80); OsWlSocketSend(fd,"123",3); OsWlSocketClose(fd);

12.22 OsWlSocketRecv

Prototype		int OsWlSocketRecv(int sockid, unsigned char* pucdata, unsigned short iLen);
Function		Socket recv data
Parameter	**Input**	sockid: socket handle
	Output	pucdata: receive buffer iLen: receive buffer size
Return		>=0: Actual received data length <0: Reception failed
Note		The prerequisite is a successful network connection.It is best that the data received each time does not exceed 1024 bytes.
Example		char buf[12]; int fd=OsWlSocketCreate(0); OsWlSocketConnect (0,fd,"www.baidu.com",80); OsWlSocketRecv(fd,buf,12); OsWlSocketClose(fd);

12.23 OsWlSocketRecvTimeout

Prototype		int OsWlSocketRecvTimeout(int sockid, unsigned char* pucdata, unsigned short iLen, unsigned int uiTimeOut);
Function		Wireless socket receives data with time out
Parameter	**Input**	sockid: socket handle uiTimeOut: timeout(ms)
	Output	pucdata: recvbuf iLen: recvbufsize
Return		>=0: actual received data length <0: reception failed
Note		The prerequisite is a successful network connection.It is best that the data received each time does not exceed 1024 bytes
Example		char buf[12];

	int fd=OsWlSocketCreate(0); OsWlSocketConnect (0,fd,"www.baidu.com",80); OsWlSocketRecvTimeout(fd,buf,12,3000); OsWlSocketClose(fd);

12.24 OsWlGetHostName

Prototype		int OsWlGetHostName(char* Domainname, char* ipv4_addr, char* ipv6_addr);
Function		Wireless domain name resolution
Parameter	**Input**	Domainname: domain
	Output	ipv4_addr: ipv4 buf ipv6_addr: ipv6 buf
Return		0: success other: failure
Note		The prerequisite is a successful network connection.
Example		char ipv4_addr[32], char ipv6_addr[32]; int ret=OsWlGetHostName("www.baidu.com",ipv4_addr,ipv6_addr);

12.25 OsWlSslSetTlsVer

Prototype		int OsWlSslSetTlsVer(int ver);
Function		Version when set up SSL connection
Parameter	**Input**	ver: 1 SSL3.0 2 TLS 1.0 3 TLS 1.1 4 TLS 1.2
	Output	N/A
Return		0: success other: failure
Note		
Example		int ret=OsWlSslSetTlsVer(4);

12.26 OsWlSslSetFile

Prototype		int OsWlSslSetFile(char* cacert, char* clientcert, char* clientkey);
Function		Set ssl certificate
Parameter	**Input**	Cacert: ca certificate(Absolute Path) Clientcert: client certificate(Absolute Path) Clientkey: client key(Absolute Path)

	Output	N/A
Return		0: success other: failure
Note		The certificate file needs to be imported into the device via PC tools.
Example		int ret=OsWlSslSetFile("/ext/ca.crt", "/ext/cli.crt", "/ext/key.key");

12.27 OsWlSslSocketSetCheckMode

Prototype		void OsWlSslSocketSetCheckMode(int mode);
Function		Set whether SSL verifies certificates
Parameter	**Input**	mode: 0-not verify, 1-verify
	Output	N/A
Return		N/A
Note		
Example		OsWlSslSocketSetCheckMode (0);

12.28 OsWlSslSocketCreate

Prototype		int OsWlSslSocketCreate();
Function		Create ssl socket
Parameter	**Input**	N/A
	Output	N/A
Return		-1: create fail >0: ssl socket handle
Note		
Example		int fd=OsWlSslSocketCreate(); OsWlSslSocketClose(fd);

12.29 OsWlSslSocketConnect

Prototype		int OsWlSslSocketConnect(int ssl_fd, char* pIp, char *pPort)
Function		Wireless ssl socket connects
Parameter	**Input**	ssl_fd: ssl socket handle pIp: server ip pPort: server port

	Output	N/A
Return		0: success other: failure
Note		OsWlSslSocketCreate must be return valid ssl socket handle
Example		int fd=OsWlSslSocketCreate(); OsWlSslSocketConnect(fd, “www.baidu.com”, 443); OsWlSslSocketClose(fd);

12.30 OsWlSslSocketSend

Prototype		int OsWlSslSocketSend(int ssl_fd, char* senddata, unsigned short sendlen);
Function		Wireless ssl socket sends
Parameter	**Input**	ssl_fd: ssl socket handle Senddata: senddata Sendlen: senddatalen
	Output	N/A
Return		>=0: actual length of data sent <0: sending failed
Note		Remember to close the ssl socket when things done.It is best that the data sent each time does not exceed 1024 bytes.
Example		int fd=OsWlSslSocketCreate(); OsWlSslSocketConnect(fd, “www.baidu.com”, 443); OsWlSslSocketSend(fd, “123”, 3); OsWlSslSocketClose(fd);

12.31 OsWlSslSocketRecv

Prototype		int OsWlSslSocketRecv(int ssl_fd, char* recvdata, unsigned short recvlen);
Function		Wireless ssl socket recvdata
Parameter	**Input**	ssl_fd: ssl socket handle recvlen: recvbufsize
	Output	recvdata: recvbuf
Return		>=0: actual received data length <0: reception failed
Note		Remember to close the ssl socket when things done.It is best that the data received each time does not exceed 1024 bytes.
Example		char buf[12]; int fd=OsWlSslSocketCreate(); OsWlSslSocketConnect(fd, “www.baidu.com”, 443); OsWlSslSocketRecv(fd, buf, 12);

	OsWlSslSocketClose(fd);

12.32 OsWlSslSocketRecvTimeout

Prototype		int OsWlSslSocketRecvTimeout(int ssl_fd, char* recvdata, unsigned short recvlen, unsigned int timeout);
Function		Wireless ssl socket recvtimeout
Parameter	**Input**	ssl_fd: ssl socket handle recvlen: recvbufsize timeout: timeout(ms)
	Output	recvdata: recvbuf
Return		>=0: actual received data length <0: reception failed
Note		Remember to close the ssl socket when things done.It is best that the data received each time does not exceed 1024 bytes.
Example		char buf[12]; int fd=OsWlSslSocketCreate(); OsWlSslSocketConnect(fd, “www.baidu.com”, 443); OsWlSslSocketRecvTimeout(fd, buf, 12,10000); OsWlSslSocketClose(fd);

12.33 OsWlSslSocketClose

Prototype		int OsWlSslSocketClose(int ssl_fd);
Function		Wireless ssl closes
Parameter	**Input**	Ssl_fd: ssl socket handling
	Output	N/A
Return		0: success other: failure
Note		
Example		int fd=OsWlSslSocketCreate(); api_wireless_sslScocketClose(fd);

13 WIFI Module

13.1 Summary

This module provides functions for applications to use WIFI for network communication. It includes network registration, de-registration, network information query, and socket (SSL) operations.

13.2 error code

#define API_WIFIINIT_ERROR -0x7001
#define API_WIFIOPEN_ERROR -0x7002
#define API_WIFICLOSE_ERROR -0x7003
#define API_WIFIWAKE_ERROR -0x7004
#define API_WIFISTATUS_ERROR -0x7005
#define API_WIFICHECKAP_ERROR -0x7006
#define API_WIFILOADCRT_ERROR -0x7007
#define API_WIFISCAN_ERROR -0x7008
#define API_WIFICONNECT_ERROR -0x7009
#define API_WIFIDISCONNECT_ERROR -0x7010
#define API_WIFINTP_ERROR -0x7011
#define API_WIFITCPCONNECT_ERROR -0x7012
#define API_WIFITCPCLOSE_ERROR -0x7013
#define API_WIFISOCKETCREATE_ERROR -0x7014
#define API_WIFISOCKETCLOSE_ERROR -0x7015
#define API_WIFISOCKETSEND_ERROR -0x7016
#define API_WIFISOCKETRECV_ERROR -0x7017
#define API_WIFISSLCONNECT_ERROR -0x7018
#define API_WIFISSLCONFIG_ERROR -0x7019
#define API_WIFISSLCLOSE_ERROR -0x7020
#define API_WIFISSLCREATE_ERROR -0x7021
#define API_WIFISSLSEND_ERROE -0x7022
#define API_WIFISSLRECV_ERROR -0x7023
#define API_WIFIWEBSTATUS_ERROR -0x7024
#define API_WIFIMAC_ERROR -0x7025
#define API_WIFIMACSET_ERROR -0x7026
#define API_WIFIGETAPLIST_ERROR -0x7027
#define API_WIFISNTPCFG_ERROR -0x7028
#define API_WIFIUPTIME_ERROR -0x7029
#define API_WIFIWEBNETCONNECT_ERROR -0x7030
#define API_WIFIOTA_ERROR -0x7031
#define API_WIFIVERSION_ERROR -0x7032
#define API_WIFIWEBCLOSE_ERROR -0x7033

13.3 OsWifiOpen

Prototype		int OsWifiOpen(void)
Function		WIFI module opens
Parameter	**Input**	N/A
	Output	N/A
Return		0: success other: failure
Note		Automatically called in OsSysInit, no need to call it in application.

Example	

13.4 OsWifiClose

Prototype		int OsWifiClose(void)
Function		WIFI module closes
Parameter	Input	N/A
	Output	N/A
Return		0: success other: failure
Note		
Example		

13.5 OsWifiGetScanResults

Prototype		int OsWifiGetScanResults(ST_AP_LIST *pstApList, int iApCount)
Function		WIFI module scans hot spots
Parameter	Input	iApCount: expected number of scans
	Output	pstApList: wifi ap info, sorted by rssi in forward order
Return		0: success other: failure
Note		ap info： typedef struct { int iEcn; /*enc mode*/ char cSsid[64]; /*ap name*/ int iRssi; /*rssi*/ char cBssid[20]; /*bssid*/ int iChannel; /*channel*/ int iFreqOffset; }ST_AP_LIST;
Example		ST_AP_LIST ap[30]={0}; S32 ap_num = 32; int Scan_num = 0; int i,j; s8 show_tmp[512]={0}; OsWifiOpen(); Scan_num = OsWifiGetScanResults(&ap, ap_num);

13.6 OsWifiConnectHotSpot

Prototype		int OsWifiConnectHotSpot(unsigned char *pucSsid, unsigned char *pucPassword)
Function		WIFI module connects AP
Parameter	**Input**	pucSsid: AP name pucPassword: AP password, coded in utf-8
	Output	
Return		0: success other: failure
Note		
Example		

13.7 OsWifiGetConnectStatus

Prototype		int OsWifiGetConnectStatus(void)
Function		WIFI module state check
Parameter	**Input**	N/A
	Output	N/A
Return		<0: Communication error 2: Connected to AP and obtained IP address 3: TCP or UDP link established 4: Disconnect network connection 5: Not connected to AP
Note		
Example		

13.8 OsEspCheckApAsync

Prototype		int OsEspCheckApAsync(ST_AP_INFO *pstAPInfo)
Function		Query connected AP information
Parameter	**Input**	N/A
	Output	ST_AP_INFO *pstAPInfo：ap info
Return		0: success other: failure
Note		typedef struct { char cSsid[64]; /*ap name*/ char cBssid[20]; /*ap bssid*/

	int iChannel; /*channel*/ int iRssi; /*rssi*/ }ST_AP_INFO;
Example	

13.9 OsWifiGetMac

Prototype		int OsWifiGetMac(unsigned char *pcMacBuf)
Function		Get current wifi MAC information
Parameter	**Input**	N/A
	Output	pcMacBuf: mac buf
Return		0: success other: failure
Note		
Example		

13.10 OsWifiConnectParamConfig

Prototype		int OsWifiConnectParamConfig(ST_WIFI_PARAM *pstWifiParam)
Function		WIFI configure parameter
Parameter	**Input**	pstWifiParam: typedef struct { int iDHCPEnable; /*DHCP enabled, 0- off 1- on*/ char cIp[20]; /*Static IP*/ char cNetMask[20]; /*mask*/ char cGateWay[20]; /*gateway*/ }ST_WIFI_PARAM;
	Output	N/A
Return		0: success other: failure
Note		Can only be called when not connected to an AP
Example		

13.11 OsWifiConnectParamGet

Prototype	s32 OsWifiConnectParamGet(ST_WIFI_PARAM *pstWifiParam);

Function		WIFI get connection parameters
Parameter	Input	pstWifiParam
	Output	N/A
Return		0: success other: failure
Note		typedef struct { int iDHCPEnable; /*DHCP enable selection: 0- off, 1- on*/ char cIp[20]; /*Static IP*/ char cNetMask[20]; /*mask*/ char cGateWay[20]; /*gateway*/ }ST_WIFI_PARAM;
Example		

13.12 OsWifiSocketCreate

Prototype		int OsWifiSocketCreate(int type)
Function		WIFI creates socket
Parameter	Input	type: 0-TCP, 1-UDP
	Output	N/A
Return		>0: success-socket ID other: failure
Note		
Example		

13.13 OsWifiSocketClose

Prototype		int OsWifiSocketClose(int sockid)
Function		WIFI closes socket
Parameter	Input	sockid: sock id
	Output	N/A
Return		0: success other: failure
Note		
Example		

13.14 OsWifiSslSocketCreate

Prototype		int OsWifiSslSocketCreate(void);
Function		WIFI creates ssl socket
Parameter	**Input**	N/A
	Output	N/A
Return		>0: success, socket id other: failure
Note		
Example		

13.15 OsWifiSslSocketClose

Prototype		int OsWifiSslSocketClose(int sockid)
Function		WIFI close SSL socket
Parameter	**Input**	Sockid: ssl socket id
	Output	N/A
Return		0: success other: failure
Note		
Example		

13.16 OsWifiTcpConnect

Prototype		int OsWifiTcpConnect(int sockid, char *pcTcpUdp,char *pcServeraddr,char *port,int timeout)
Function		WIFI-TCP connects
Parameter	**Input**	sockid: socket ID pcTcpUdp: TCP or UDP pcServeraddr: server ip Port: serverport Timeout: timeout(ms)
	Output	N/A
Return		0: success other: failure
Note		
Example		

13.17 OsWifiTcpClose

Prototype		int OsWifiTcpClose(void)
Function		TCP disconnects
Parameter	**Input**	sockid-socket id
	Output	N/A
Return		0: success other: failure
Note		
Example		

13.18 OsWifiSend

Prototype		int OsWifiSend(int sockid,char *data, int datalen, int timeout)
Function		WIFI sends data
Parameter	**Input**	sockid-socket id; data-senddata datalen-datalen; timeout-timeout(ms)
	Output	N/A
Return		0: success other: failure
Note		
Example		

13.19 OsWifiRecv

Prototype		int OsWifiRecv(int sockid, unsigned char *pucdata, unsigned short iLen, unsigned int uiTimeOut)
Function		WIFI tcp recvdata
Parameter	**Input**	sockid-socket id; uiTimeOut-timeout(ms)
	Output	pucdata-recvbuf iLen-recvsize;
Return		>=0: actual received data length other: failure
Note		
Example		

13.20 OsWifiSslConnect

Prototype		int OsWifiSslConnect(int sockid,char *serveraddr,char *port,int timeout);
Function		WIFI ssl server connect
Parameter	**Input**	sockid: ssl socket id serveraddr: server ip port: server port timeout: timeout(ms)
	Output	N/A
Return		>=0: socket id other: failure
Note		
Example		

13.21 OsWifiSslClose

Prototype		int OsWifiSslClose(int sockid);
Function		WIFI ssl server close
Parameter	**Input**	sockid: ssl socket id
	Output	N/A
Return		0: success other: failure
Note		
Example		

13.22 OsWifiSslSend

Prototype		int OsWifiSslSend(int sockid, char *data, int datalen, int timeout);
Function		WIFI ssl sends data to server
Parameter	**Input**	sockid: ssl socket id data: senddata datalen: datalen timeout: timeout(ms)
	Output	N/A
Return		0: success other: failure
Note		

Example	

13.23 OsWifiSslRecv

Prototype		int OsWifiSslRecv(int sockid, unsigned char *pucdata, unsigned short iLen, unsigned int uiTimeOut);
Function		WIFI ssl socket recv data from server with timeout
Parameter	Input	sockid: ssl socket id timeout: timeout(ms)
	Output	data: recvbuf datalen: recvbufsize
Return		>=0: actual received data length other: failure
Note		
Example		

13.24 OsWifiSntpCfg

Prototype		int OsWifiSntpCfg(int enable,int timezone, unsigned char* SNTP1,unsigned char* SNTP2,unsigned char* SNTP3);
Function		wifi set ntp server address
Parameter	Input	enable: timezone: SNTP1: SNTP2: SNTP3:
	Output	
Return		0:success other: failure
Note		
Example		OsWifiSntpCfg(1,8, "cn.ntp.org.cn", "ntp.sjtu.edu.cn","");

13.25 OsWifiUptime

Prototype		int OsWifiUptime(void);
Function		Set the terminal time via wifi
Parameter	Input	N/A
	Output	N/A

Return	0:success other: failure
Note	OsWifiSntpCfg must be called first
Example	

13.26 OsWifiWebNetConnect

Prototype		int OsWifiWebNetConnect(unsigned char* pui8Ssid, unsigned char* pui8Password, unsigned int ui32Timeout);
Function		Turn on the wifi distribution network
Parameter	**Input**	pui8Ssid:ssid pui8Password:password ui32Timeout:timeout(ms)
	Output	N/A
Return		0:success other: failure
Note		
Example		

13.27 OsWifiWebNetworkClose

Prototype		int OsWifiWebNetworkClose(void);
Function		Turn off the wifi distribution network
Parameter	**Input**	pui8Ssid:ssid pui8Password:password ui32Timeout:timeout(ms)
	Output	N/A
Return		0:success other: failure
Note		
Example		

13.28 OsWifiUserRota

Prototype		int OsWifiUserRota(char* URL, int URLlen, int i32Timeout);
Function		Update the wifi firmware in url format
Param	**Input**	URL:url URLlen:URLlen

eter		ui32Timeout:timeout(ms)
	Output	N/A
Return		0:success other: failure
Note		
Example		

13.29 OsWifiGetVersion

Prototype		int OsWifiGetVersion(char* buf, int buflen);
Function		get the wifi version
Parameter	**Input**	buflen:buflen
	Output	buf:output bufsize
Return		0:success other: failure
Note		
Example		

14 ICC/PSAM Card

14.1 Summary

IC (Integrated Circuit Card) card often refers to a contact type synchronous/asynchronous integrated circuit card that complies with or is based on the ISO7816 specification.
IC cards have diverse application environments and a wide range of fields, with many different names depending on the application area, such as bank cards, bus cards, social security cards, telephone SIM cards, gas cards, access cards, and so on.
Asynchronous IC cards, also known as CPU cards, smart cards, are mainly used for functions such as transaction authentication, security control, and small payments, including other fields with high security requirements, such as finance, mobile phones, public transportation, etc. The implementation and application of such IC cards must strictly follow standards, and each industry has relevant industry standards and certifications, such as PSAM card certification from the Ministry of Construction, EMV certification from the financial industry, PBOC certification, etc.
There is no standard definition for synchronous IC cards in the specifications, and each manufacturer's product has its own characteristics. Corresponding drivers must be developed for cards from different manufacturers. Mainly used for storing data information, the most widely used ones on the market include SIMENS' SLE4442 \ SLE4428, ATMEL's AT24Cxx series, and AT88SCxx series. Mainly used in access control, schools, water and electricity meters, highway toll collection and other fields.

14.2 error code

#define API_ICCOPEN_ERROR -0x8001
#define API_ICCCLOSE_ERROR -0x8002
#define API_ICCPOWERON_ERROR -0x8003
#define API_ICCPOWEROFF_ERROR -0x8004
#define API_ICCGETSTATUS_ERROR -0x8005
#define API_ICCAPDU_ERROR -0x8006
#define API_ICCIOCTL_ERROR -0x8007

14.3 OsIcCardOpen

<table>
<tr><td colspan="2">Prototype</td><td>S32 OsIcCardOpen(int slot);</td></tr>
<tr><td colspan="2">Function</td><td>open ICC/PSAM device</td></tr>
<tr><td rowspan="2">Parameter</td><td>Input</td><td>slot: slot number
0: ICC slot
>=1: PSAM card holder number</td></tr>
<tr><td>Output</td><td>N/A</td></tr>
<tr><td colspan="2">Return</td><td>success
other: failure</td></tr>
<tr><td colspan="2">Note</td><td>1. The PSAM card holder number is consistent with the silk screen label on the card holder.
2. Please close when process ends.</td></tr>
<tr><td colspan="2">Example</td><td>
<pre>
void BcdToAsc(u8 *Dest,u8 *Src,u32 Len)
{
  u32 i;
  for(i=0;i<Len;i++)
  {

    if(((*(Src + i) & 0xF0) >> 4) <= 9)
    {
      *(Dest + 2*i) = ((*(Src + i) & 0xF0) >> 4) + 0x30;
    }
    else
    {
      *(Dest + 2*i) = ((*(Src + i) & 0xF0) >> 4) + 0x37;
    }

    if((*(Src + i) & 0x0F) <= 9)
    {
      *(Dest + 2*i + 1) = (*(Src + i) & 0x0F) + 0x30;
    }
    else
    {
      *(Dest + 2*i + 1) = (*(Src + i) & 0x0F) + 0x37;
</pre>
</td></tr>
</table>

```
      }
   }
}

void test_ic_fun(){
    int ret=0;
    int nslot=0;
    u8 ATR[100]={0};
    u8 buf[280]={0},buf2[32]={0},buf3[8] = {0},buf4[32]={0};
    APDU_SEND ApduSend;
    APDU_RESP ApduResp;
    ret = OsIcCardOpen(nslot);
    while(1){
        ret = OsIcCardGetStatus(nslot);
        if(ret == 0)
        {
            ret = 0;
            break;
        }
        OsSysMsleep(10);
    }
    ret = OsIcCardGetStatus(nslot);
    if(ret)
    {

        goto EXIT;
    }

    memset(buf,0,sizeof(buf));
    memset(ATR,0,sizeof(ATR));
    ret=OsIcCardPowerOn(nslot, buf);
    if(ret)
    {

        goto EXIT;
    }
    BcdToAsc(ATR, buf+1, buf[0]);


    ApduSend.Command[0]=0x00;
ApduSend.Command[1]=0xa4;
    ApduSend.Command[2]=0x04;
    ApduSend.Command[3]=0x00;
    ApduSend.Lc=0x0e;
    ApduSend.Le=256;
    memcpy(ApduSend.DataIn,"1PAY.SYS.DDF01",ApduSend.Lc);
```

```
    ret=OsIcCardExchangeApdu(nslot, &ApduSend,&ApduResp);
    if(ret==0)
    {
        memcpy(buf2, ApduResp.DataOut, 6);
        memcpy(buf4, ApduResp.DataOut+ApduResp.LenOut-6, 6);
        memcpy(buf,"data:",5);
        BcdToAsc(buf+5, buf2, 6);
        memcpy(buf+5+12, "**********",13);
        BcdToAsc(buf+5+12+13, buf4, 6);
        memcpy(buf+5+12+13+12,"SWA:",4);
        BcdToAsc(buf+5+12+13+12+4, (u8*)&ApduResp.SWA,1);
        memcpy(buf+5+12+13+12+4+2," SWB:",5);
        BcdToAsc(buf+5+12+13+12+4+2+5, (u8*)&ApduResp.SWB,1);
        OsSysMsleep(1000);
    }



EXIT:
    OsIcCardPowerOff(nslot);
    OsIcCardlose(nslot);

    return;
}
```

14.4 OsIcCardlose

Prototype		S32 OsIcCardlose(int slot);
Function		Close ICC/PSAM device
Parameter	**Input**	slot: slot number 0: ICC slot >=1: PSAM card holder number
	Output	N/A
Return		0: success other: failure
Note		The PSAM card holder number is consistent with the silk screen label on the card holder If the card is not powered off, the power off operation will be forcibly executed
Example		void BcdToAsc(u8 *Dest,u8 *Src,u32 Len) { u32 i; for(i=0;i<Len;i++) {

```
        if(((*(Src + i) & 0xF0) >> 4) <= 9)
        {
            *(Dest + 2*i) = ((*(Src + i) & 0xF0) >> 4) + 0x30;
        }
        else
        {
            *(Dest + 2*i)  = ((*(Src + i) & 0xF0) >> 4) + 0x37;
        }

        if((*(Src + i) & 0x0F) <= 9)
        {
            *(Dest + 2*i + 1) = (*(Src + i) & 0x0F) + 0x30;
        }
        else
        {
            *(Dest + 2*i + 1) = (*(Src + i) & 0x0F) + 0x37;
        }
    }
}
void test_ic_fun(){
    int ret=0;
    int nslot=0;
    u8 ATR[100]={0};
    u8 buf[280]={0},buf2[32]={0},buf3[8] = {0},buf4[32]={0};
    APDU_SEND ApduSend;
    APDU_RESP ApduResp;
    ret = OsIcCardOpen(nslot);
    while(1){
        ret = OsIcCardGetStatus(nslot);
        if(ret == 0)
        {
            ret = 0;
            break;
        }
        OsSysMsleep(10);
    }
    ret = OsIcCardGetStatus(nslot);
    if(ret)
    {

        goto EXIT;
    }

    memset(buf,0,sizeof(buf));
    memset(ATR,0,sizeof(ATR));
    ret=OsIcCardPowerOn(nslot, buf);
```

```
    if(ret)
    {

        goto EXIT;
    }
    BcdToAsc(ATR, buf+1, buf[0]);


    ApduSend.Command[0]=0x00;
ApduSend.Command[1]=0xa4;
    ApduSend.Command[2]=0x04;
    ApduSend.Command[3]=0x00;
    ApduSend.Lc=0x0e;
    ApduSend.Le=256;
    memcpy(ApduSend.DataIn,"1PAY.SYS.DDF01",ApduSend.Lc);
    ret=OsIcCardExchangeApdu(nslot, &ApduSend,&ApduResp);
    if(ret==0)
    {
        memcpy(buf2, ApduResp.DataOut, 6);
        memcpy(buf4, ApduResp.DataOut+ApduResp.LenOut-6, 6);
        memcpy(buf,"data:",5);
        BcdToAsc(buf+5, buf2, 6);
        memcpy(buf+5+12, "**********",13);
        BcdToAsc(buf+5+12+13, buf4, 6);
        memcpy(buf+5+12+13+12,"SWA:",4);
        BcdToAsc(buf+5+12+13+12+4, (u8*)&ApduResp.SWA,1);
        memcpy(buf+5+12+13+12+4+2," SWB:",5);
        BcdToAsc(buf+5+12+13+12+4+2+5, (u8*)&ApduResp.SWB,1);
        OsSysMsleep(1000);
    }



EXIT:
    OsIcCardPowerOff(nslot);
    OsIcCardlose(nslot);

    return;
}
```

14.5 OsIcCardPowerOn

Prototype	S32 OsIcCardPowerOn(int slot,u8 *lpAtr);
Function	Power on ICC/PSAM equipment

Parameter	**Input**	slot: slot number 0: ICC slot >=1: PSAM card holder number
	Output	lpAtr: Reset response data, format: B..LLVAR， Maximum length N=32
Return		0: success other: failure
Note		1. The PSAM card holder number is consistent with the silk screen label on the card holder 2. The lpAtr reset response buffer must be no less than the maximum length (N)+1
Example		void BcdToAsc(u8 *Dest,u8 *Src,u32 Len) { u32 i; for(i=0;i<Len;i++) { if(((*(Src + i) & 0xF0) >> 4) <= 9) { *(Dest + 2*i) = ((*(Src + i) & 0xF0) >> 4) + 0x30; } else { *(Dest + 2*i) = ((*(Src + i) & 0xF0) >> 4) + 0x37; } if((*(Src + i) & 0x0F) <= 9) { *(Dest + 2*i + 1) = (*(Src + i) & 0x0F) + 0x30; } else { *(Dest + 2*i + 1) = (*(Src + i) & 0x0F) + 0x37; } } } void test_ic_fun(){ int ret=0; int nslot=0; u8 ATR[100]={0}; u8 buf[280]={0},buf2[32]={0},buf3[8] = {0},buf4[32]={0}; APDU_SEND ApduSend; APDU_RESP ApduResp; ret = OsIcCardOpen(nslot); while(1){ ret = OsIcCardGetStatus(nslot); if(ret == 0) {

```
            ret = 0;
            break;
        }
        OsSysMsleep(10);
    }
    ret = OsIcCardGetStatus(nslot);
    if(ret)
    {

        goto EXIT;
    }

    memset(buf,0,sizeof(buf));
    memset(ATR,0,sizeof(ATR));
    ret=OsIcCardPowerOn(nslot, buf);
    if(ret)
    {

        goto EXIT;
    }
    BcdToAsc(ATR, buf+1, buf[0]);


    ApduSend.Command[0]=0x00;
    ApduSend.Command[1]=0xa4;
    ApduSend.Command[2]=0x04;
    ApduSend.Command[3]=0x00;
    ApduSend.Lc=0x0e;
    ApduSend.Le=256;
    memcpy(ApduSend.DataIn,"1PAY.SYS.DDF01",ApduSend.Lc);
    ret=OsIcCardExchangeApdu(nslot, &ApduSend,&ApduResp);
    if(ret==0)
    {
        memcpy(buf2, ApduResp.DataOut, 6);
        memcpy(buf4, ApduResp.DataOut+ApduResp.LenOut-6, 6);
        memcpy(buf,"data:",5);
        BcdToAsc(buf+5, buf2, 6);
        memcpy(buf+5+12, "**************",13);
        BcdToAsc(buf+5+12+13, buf4, 6);
        memcpy(buf+5+12+13+12,"SWA:",4);
        BcdToAsc(buf+5+12+13+12+4, (u8*)&ApduResp.SWA,1);
        memcpy(buf+5+12+13+12+4+2," SWB:",5);
        BcdToAsc(buf+5+12+13+12+4+2+5, (u8*)&ApduResp.SWB,1);
        OsSysMsleep(1000);
    }
```

```
EXIT:
    OsIcCardPowerOff(nslot);
    OsIcCardlose(nslot);
    return;
}
```

14.6 OsIcCardPowerOff

Prototype		S32 OsIcCardPowerOff(int slot);
Function		Power off ICC/PSAM equipment
Parameter	**Input**	slot: deck number 0: ICC slot >=1: PSAM card holder number
	Output	
Return		0: success other: failure
Note		The PSAM card holder number is consistent with the silk screen label on the card holder
Example		

```
void BcdToAsc(u8 *Dest,u8 *Src,u32 Len)
{
  u32 i;
  for(i=0;i<Len;i++)
  {

    if(((*(Src + i) & 0xF0) >> 4) <= 9)
    {
      *(Dest + 2*i) = ((*(Src + i) & 0xF0) >> 4) + 0x30;
    }
    else
    {
      *(Dest + 2*i)  = ((*(Src + i) & 0xF0) >> 4) + 0x37;
    }

    if((*(Src + i) & 0x0F) <= 9)
    {
      *(Dest + 2*i + 1) = (*(Src + i) & 0x0F) + 0x30;
    }
    else
    {
      *(Dest + 2*i + 1) = (*(Src + i) & 0x0F) + 0x37;
    }
  }
```

```
}
void test_ic_fun(){
    int ret=0;
    int nslot=0;
    u8 ATR[100]={0};
    u8 buf[280]={0},buf2[32]={0},buf3[8] = {0},buf4[32]={0};
    APDU_SEND ApduSend;
    APDU_RESP ApduResp;
    ret = OsIcCardOpen(nslot);
    while(1){
        ret = OsIcCardGetStatus(nslot);
        if(ret == 0)
        {
            ret = 0;
            break;
        }
        OsSysMsleep(10);
    }
    ret = OsIcCardGetStatus(nslot);
    if(ret)
    {

        goto EXIT;
    }

    memset(buf,0,sizeof(buf));
    memset(ATR,0,sizeof(ATR));
    ret=OsIcCardPowerOn(nslot, buf);
    if(ret)
    {

        goto EXIT;
    }
    BcdToAsc(ATR, buf+1, buf[0]);


    ApduSend.Command[0]=0x00;
ApduSend.Command[1]=0xa4;
    ApduSend.Command[2]=0x04;
    ApduSend.Command[3]=0x00;
    ApduSend.Lc=0x0e;
    ApduSend.Le=256;
    memcpy(ApduSend.DataIn,"1PAY.SYS.DDF01",ApduSend.Lc);
    ret=OsIcCardExchangeApdu(nslot, &ApduSend,&ApduResp);
    if(ret==0)
    {
```

```
            memcpy(buf2, ApduResp.DataOut, 6);
            memcpy(buf4, ApduResp.DataOut+ApduResp.LenOut-6, 6);
            memcpy(buf,"data:",5);
            BcdToAsc(buf+5, buf2, 6);
            memcpy(buf+5+12, "************",13);
            BcdToAsc(buf+5+12+13, buf4, 6);
            memcpy(buf+5+12+13+12,"SWA:",4);
            BcdToAsc(buf+5+12+13+12+4, (u8*)&ApduResp.SWA,1);
            memcpy(buf+5+12+13+12+4+2," SWB:",5);
            BcdToAsc(buf+5+12+13+12+4+2+5, (u8*)&ApduResp.SWB,1);
            OsSysMsleep(1000);
    }
EXIT:
    OsIcCardPowerOff(nslot);
    OsIcCardlose(nslot);

    return;
}
```

14.7 OsIcCardGetStatus

Prototype		S32 OsIcCardGetStatus(int slot);
Function		Get ICC/PSAM device status
Parameter	**Input**	slot: slot number 0: ICC slot >=1: PSAM card holder number
	Output	
Return		0: card is inserted other: failure
Note		
Example		

```
void BcdToAsc(u8 *Dest,u8 *Src,u32 Len)
{
   u32 i;
   for(i=0;i<Len;i++)
   {

      if(((*(Src + i) & 0xF0) >> 4) <= 9)
      {
         *(Dest + 2*i) = ((*(Src + i) & 0xF0) >> 4) + 0x30;
      }
      else
      {
         *(Dest + 2*i)  = ((*(Src + i) & 0xF0) >> 4) + 0x37;
```

```
        }

        if((*(Src + i) & 0x0F) <= 9)
        {
            *(Dest + 2*i + 1) = (*(Src + i) & 0x0F) + 0x30;
        }
        else
        {
            *(Dest + 2*i + 1) = (*(Src + i) & 0x0F) + 0x37;
        }
    }
}
void test_ic_fun(){
    int ret=0;
    int nslot=0;
    u8 ATR[100]={0};
    u8 buf[280]={0},buf2[32]={0},buf3[8] = {0},buf4[32]={0};
    APDU_SEND ApduSend;
    APDU_RESP ApduResp;
    ret = OsIcCardOpen(nslot);
    while(1){
        ret = OsIcCardGetStatus(nslot);
        if(ret == 0)
        {
            ret = 0;
            break;
        }
        OsSysMsleep(10);
    }
    ret = OsIcCardGetStatus(nslot);
    if(ret)
    {

        goto EXIT;
    }

    memset(buf,0,sizeof(buf));
    memset(ATR,0,sizeof(ATR));
    ret=OsIcCardPowerOn(nslot, buf);
    if(ret)
    {

        goto EXIT;
    }
    BcdToAsc(ATR, buf+1, buf[0]);
```

```
    ApduSend.Command[0]=0x00;
ApduSend.Command[1]=0xa4;
    ApduSend.Command[2]=0x04;
    ApduSend.Command[3]=0x00;
    ApduSend.Lc=0x0e;
    ApduSend.Le=256;
    memcpy(ApduSend.DataIn,"1PAY.SYS.DDF01",ApduSend.Lc);
    ret=OsIcCardExchangeApdu(nslot, &ApduSend,&ApduResp);
    if(ret==0)
    {
        memcpy(buf2, ApduResp.DataOut, 6);
        memcpy(buf4, ApduResp.DataOut+ApduResp.LenOut-6, 6);
        memcpy(buf,"data:",5);
        BcdToAsc(buf+5, buf2, 6);
        memcpy(buf+5+12, "**********",13);
        BcdToAsc(buf+5+12+13, buf4, 6);
        memcpy(buf+5+12+13+12,"SWA:",4);
        BcdToAsc(buf+5+12+13+12+4, (u8*)&ApduResp.SWA,1);
        memcpy(buf+5+12+13+12+4+2," SWB:",5);
        BcdToAsc(buf+5+12+13+12+4+2+5, (u8*)&ApduResp.SWB,1);
        OsSysMsleep(1000);
    }



EXIT:
    OsIcCardPowerOff(nslot);
    OsIcCardlose(nslot);

    return;
}
```

14.8 OsIcCardExchangeApdu

Prototype		S32 OsIcCardExchangeApdu(int slot,APDU_SEND *ApduSend, APDU_RESP *ApduResp);
Function		Read and write ICC/PSAM
Parameter	**Input**	slot: slot number 0: ICC slot >=1: PSAM card holder number ApduSendParameter: typedef struct { unsigned char Command[4]; // CLA INS P1 P2

		int Lc; // P3 unsigned char DataIn[512];//512 int Le; } APDU_SEND;
	Output	ApduRespParameter: typedef struct { int LenOut; // length of dataout unsigned char DataOut[512]; unsigned char SWA; unsigned char SWB; } APDU_RESP;
Return		0: success other: failure
Note		For CASE 2 and CASE 4 in CAPDU, Le represents the expected length of the Return data. If Le=256, it indicates that the expected length of the Return is unknown. Example： CASE1：CLA INS P1 P2，ApduSend need set Lc=0,Le=0 CASE2: CLA INS P1 P2 Le，ApduSend need set Lc=0,Le=xx(Le=256 represents 0, other values are specific values) Example: APDU_SEND.Le=256, CLA INS P1 P2 0 APDU_SEND.Le=0x10, CLA INS P1 P2 10 CASE3: CLA INS P1 P2 Lc Data, ApduSend need set Lc=xx,Le=0 CASE4: CLA INS P1 P2 Lc Data Le, ApduSend need set Lc=xx,Le=xx(Le=256 represents 0, other values are specific values)
Example		void BcdToAsc(u8 *Dest,u8 *Src,u32 Len) { u32 i; for(i=0;i<Len;i++) { if(((*(Src + i) & 0xF0) >> 4) <= 9) { *(Dest + 2*i) = ((*(Src + i) & 0xF0) >> 4) + 0x30; } else { *(Dest + 2*i) = ((*(Src + i) & 0xF0) >> 4) + 0x37; } if((*(Src + i) & 0x0F) <= 9)

```
        {
            *(Dest + 2*i + 1) = (*(Src + i) & 0x0F) + 0x30;
        }
        else
        {
            *(Dest + 2*i + 1) = (*(Src + i) & 0x0F) + 0x37;
        }
    }
}
void test_ic_fun(){
    int ret=0;
    int nslot=0;
    u8 ATR[100]={0};
    u8 buf[280]={0},buf2[32]={0},buf3[8] = {0},buf4[32]={0};
    APDU_SEND ApduSend;
    APDU_RESP ApduResp;
    ret = OsIcCardOpen(nslot);
    while(1){
        ret = OsIcCardGetStatus(nslot);
        if(ret == 0)
        {
            ret = 0;
            break;
        }
        OsSysMsleep(10);
    }
    ret = OsIcCardGetStatus(nslot);
    if(ret)
    {

        goto EXIT;
    }

    memset(buf,0,sizeof(buf));
    memset(ATR,0,sizeof(ATR));
    ret=OsIcCardPowerOn(nslot, buf);
    if(ret)
    {

        goto EXIT;
    }
    BcdToAsc(ATR, buf+1, buf[0]);


    ApduSend.Command[0]=0x00;
    ApduSend.Command[1]=0xa4;
```

```
    ApduSend.Command[2]=0x04;
    ApduSend.Command[3]=0x00;
    ApduSend.Lc=0x0e;
    ApduSend.Le=256;
    memcpy(ApduSend.DataIn,"1PAY.SYS.DDF01",ApduSend.Lc);
    ret=OsIcCardExchangeApdu(nslot, &ApduSend,&ApduResp);
    if(ret==0)
    {
        memcpy(buf2, ApduResp.DataOut, 6);
        memcpy(buf4, ApduResp.DataOut+ApduResp.LenOut-6, 6);
        memcpy(buf,"data:",5);
        BcdToAsc(buf+5, buf2, 6);
        memcpy(buf+5+12, "************",13);
        BcdToAsc(buf+5+12+13, buf4, 6);
        memcpy(buf+5+12+13+12,"SWA:",4);
        BcdToAsc(buf+5+12+13+12+4, (u8*)&ApduResp.SWA,1);
        memcpy(buf+5+12+13+12+4+2," SWB:",5);
        BcdToAsc(buf+5+12+13+12+4+2+5, (u8*)&ApduResp.SWB,1);
        OsSysMsleep(1000);
    }



EXIT:
    OsIcCardPowerOff(nslot);
    OsIcCardlose(nslot);

    return;
}
```

15 Contactless Card

15.1 Summary

Contactless IC card, also known as RF card, consists of an IC chip and an induction antenna, packaged in a standard PVC card, with the chip and antenna witout exposure of any part. It successfully combines radio frequency identification technology with IC card technology, ending the problem of Passive (no power supply in the card) and Contactless operation, which is a major breakthrough in the field of electronic devices. The card approaches the surface of the reader/writer within a certain distance range (maximum 5-10mm) and completes data read and write operations through the transmission of wireless radio waves.
PCD: Proximity Coupling Device
PICC: Proximity IC Card
Contactless cards have the following advantages:
High reliability: no mechanical contact between the IC card and the reader/writer, avoids various faults caused by contact read and write.
Easy to operate: The card can be operated within a 10cm range, so there is no need to insert or remove the card, which is very convenient for users to use.

Fast transaction speed: The time for the terminal to send commands and process card response data is less than 100ms
Product classification of Contactless cards:
It can be divided into 13.56M and 2.4G according to frequency, etc
13.56M: including A card/B card/Mifare card, etc
2.4G: mainly for mobile/telecom mobile payment
The card types supported by the company's products:
Compliant with ISO14443 and EMV Contactless for Type A and Type B.

15.2 error code

#define API_RFOPEN_ERROR -0x9001
#define API_RFCLOSE_ERROR -0x9002
#define API_RFPOWERON_ERROR -0x9003
#define API_RFPOWEROFF_ERROR -0x9004
#define API_RFGETSTATUS_ERROR -0x9005
#define API_RFACTIVE_ERROR -0x9006
#define API_RFAPDU_ERROR -0x9007
#define API_RFIOCTL_ERROR -0x9008

15.3 OsRfOpen

Prototype		s32 OsRfOpen (void)
Function		Open rf device
Parameter	**Input**	N/A
	Output	N/A
Return		API_OK: success API_ERR: fail API_EBUSY: the device is busy or occupied
Note		
Example		int cRet = 0; APDU_SEND ApduSend; APDU_RESP ApduResp; ApduSend.Command[0] = 0x00; ApduSend.Command[1] = 0xa4; ApduSend.Command[2] = 0x04; ApduSend.Command[3] = 0x00; ApduSend.Lc = 0x0e; ApduSend.Le = 256; memcpy(ApduSend.DataIn, "2PAY.SYS.DDF01", ApduSend.Lc); OsRfOpen(); OsRfPownOn(PICC_TYPE_A); cRet = OsRfGetStatus(); if (cRet == 3) {

```
    if (!OsRfActivate())
    {
        if (!OsRfExchangeApdu(&ApduSend, &ApduResp)) {

            while (1) {
                cRet = OsRfRemove();
                if (cRet == 0)
                    break;
                OsSysMsleep(100);
            }
        }
    }
}
OsRfPowerOff();
OsRfClose();
```

15.4 OsRfClose

Prototype		s32 OsRfClose(void);
Function		Close rf device
Parameter	**Input**	N/A
	Output	N/A
Return		API_OK: success API_ERR: fail API_EBUSY: the device is busy or occupied
Note		
Example		int cRet = 0; APDU_SEND ApduSend; APDU_RESP ApduResp; ApduSend.Command[0] = 0x00; ApduSend.Command[1] = 0xa4; ApduSend.Command[2] = 0x04; ApduSend.Command[3] = 0x00; ApduSend.Lc = 0x0e; ApduSend.Le = 256; memcpy(ApduSend.DataIn, "2PAY.SYS.DDF01", ApduSend.Lc); OsRfOpen(); OsRfPownOn(PICC_TYPE_A); cRet = OsRfGetStatus(); if (cRet == 3) { if (!OsRfActivate())

```
    {
        if (!OsRfExchangeApdu(&ApduSend, &ApduResp)) {

            while (1) {
                cRet = OsRfRemove();
                if (cRet == 0)
                    break;
                OsSysMsleep(100);
            }
        }
    }
}
OsRfPowerOff();
OsRfClose();
```

15.5 OsRfPownOn

Prototype		s32 OsRfPownOn (u32 nType)
Function		Power on the Contactless card reader and start card search
Parameter	**Input**	nType： #define PICC_TYPE_A 0x01//type a card #define PICC_TYPE_B 0x02//type b card #define PICC_TYPE_AB 0x03//type a card and type b card simultaneously #define PICC_TYPE_M1 0x04//M1 card
	Output	N/A
Return		API_OK: success API_ERR: fail API_EINVAL: parameter error API_EIO: device not open
Note		
Example		int cRet = 0; APDU_SEND ApduSend; APDU_RESP ApduResp; ApduSend.Command[0] = 0x00; ApduSend.Command[1] = 0xa4; ApduSend.Command[2] = 0x04; ApduSend.Command[3] = 0x00; ApduSend.Lc = 0x0e; ApduSend.Le = 256; memcpy(ApduSend.DataIn, "2PAY.SYS.DDF01", ApduSend.Lc); OsRfOpen();

```
OsRfPownOn(PICC_TYPE_A);
cRet = OsRfGetStatus();
if (cRet == 3) {
    if (!OsRfActivate())
    {
        if (!OsRfExchangeApdu(&ApduSend, &ApduResp)) {

            while (1) {
                cRet = OsRfRemove();
                if (cRet == 0)
                    break;
                OsSysMsleep(100);
            }
        }
    }
}
OsRfPowerOff();
OsRfClose();
```

15.6 OsRfPowerOff

Prototype		s32 OsRfPowerOff ()
Function		Power off rf device
Parameter	**Input**	N/A
	Output	N/A
Return		API_OK: success API_ERR: fail API_EINVAL: parameter error API_EIO: device not open
Note		
Example		int cRet = 0; APDU_SEND ApduSend; APDU_RESP ApduResp; ApduSend.Command[0] = 0x00; ApduSend.Command[1] = 0xa4; ApduSend.Command[2] = 0x04; ApduSend.Command[3] = 0x00; ApduSend.Lc = 0x0e;

```
ApduSend.Le = 256;
memcpy(ApduSend.DataIn, "2PAY.SYS.DDF01", ApduSend.Lc);
OsRfOpen();
OsRfPownOn(PICC_TYPE_A);
cRet = OsRfGetStatus();
if (cRet == 3) {
    if (!OsRfActivate())
    {
        if (!OsRfExchangeApdu(&ApduSend, &ApduResp)) {

            while (1) {
                cRet = OsRfRemove();
                if (cRet == 0)
                    break;
                OsSysMsleep(100);
            }
        }
    }
}
OsRfPowerOff();
OsRfClose();
```

15.7 OsRfGetStatus

Prototype		s32 OsRfGetStatus (void)
Function		Check rf status
Parameter	**Input**	N/A
	Output	N/A
Return		>0: Contactless card reader status value 1: no card or card has been removed 2: found multiple cards 3: found a single Type A CPU card 4: found a single Type B CPU card 5: found a single Type A Memory card 6: found a single Type B Memory card 7: found a single FeliCa card other: API_SRR: failed API_SIO: device not turned on or powered on
Note		1. Further information needs to be obtained through SAK to determine whether Type A CPU and Memory 2-in-1 card can return to "Single type A CPU card" status.

Example	int cRet = 0; APDU_SEND ApduSend; APDU_RESP ApduResp; ApduSend.Command[0] = 0x00; ApduSend.Command[1] = 0xa4; ApduSend.Command[2] = 0x04; ApduSend.Command[3] = 0x00; ApduSend.Lc = 0x0e; ApduSend.Le = 256; memcpy(ApduSend.DataIn, "2PAY.SYS.DDF01", ApduSend.Lc); OsRfOpen(); OsRfPownOn(PICC_TYPE_A); cRet = OsRfGetStatus(); if (cRet == 3) { if (!OsRfActivate()) { if (!OsRfExchangeApdu(&ApduSend, &ApduResp)) { while (1) { cRet = OsRfRemove(); if (cRet == 0) break; OsSysMsleep(100); } } } } OsRfPowerOff(); OsRfClose();

15.8 OsRfActivate

Prototype		s32 OsRfActivate (void)
Function		Activate rf device
Parameter	**Input**	N/A
	Output	N/A
		API_OK: success API_ERR: fail API_EIO: device not open

Return	
Note	
Example	int cRet = 0; APDU_SEND ApduSend; APDU_RESP ApduResp; ApduSend.Command[0] = 0x00; ApduSend.Command[1] = 0xa4; ApduSend.Command[2] = 0x04; ApduSend.Command[3] = 0x00; ApduSend.Lc = 0x0e; ApduSend.Le = 256; memcpy(ApduSend.DataIn, "2PAY.SYS.DDF01", ApduSend.Lc); OsRfOpen(); OsRfPownOn(PICC_TYPE_A); cRet = OsRfGetStatus(); if (cRet == 3) { if (!OsRfActivate()) { if (!OsRfExchangeApdu(&ApduSend, &ApduResp)) { while (1) { cRet = OsRfRemove(); if (cRet == 0) break; OsSysMsleep(100); } } } } OsRfPowerOff(); OsRfClose();

15.9 OsRfExchangeApdu

Prototype		s32 OsRfExchangeApdu(APDU_SEND *ApduSend, APDU_RESP *ApduResp);
Function		Read and write Contactless card operations
Parame	**Input**	lpCApdu： Send command APDU in the following format: CLA INS P1 P2 [Lc Data] [Le] nCApduLen：

ter		LPCApdu content length. nRApduSize: response data lpRApdu cache length
	Output	lpRApdu： Response Data (RAPDU) Format: [Data] sw1 sw2 lpRApduLen： Response Data (RAPDU) data length, including 2 bytes of SW.
Return		API_OK：success API_SRR: Failed API_CIMEOUT: timeout API_SINVAL: Parameter error API_SPROTOCOL: Protocol Error API_STRANSPORT: Transmission error API_SIO: Device not turned on
Note		1. LpRApduLen should not exceed lpRApdu, such as the status word Return 61XX, and the GET RESPENSE command must be sent to extract data; 2. The underlying communication of Contactless cards (CPU) is a block structure, similar to T1 card communication in IC cards, and the bottom does not care about CASE1, 2, 3, and 4.
Example		int cRet = 0; APDU_SEND ApduSend; APDU_RESP ApduResp; ApduSend.Command[0] = 0x00; ApduSend.Command[1] = 0xa4; ApduSend.Command[2] = 0x04; ApduSend.Command[3] = 0x00; ApduSend.Lc = 0x0e; ApduSend.Le = 256; memcpy(ApduSend.DataIn, "2PAY.SYS.DDF01", ApduSend.Lc); OsRfOpen(); OsRfPownOn(PICC_TYPE_A); cRet = OsRfGetStatus(); if (cRet == 3) { if (!OsRfActivate()) { if (!OsRfExchangeApdu(&ApduSend, &ApduResp)) { while (1) { cRet = OsRfRemove(); if (cRet == 0) break;

```
                    OsSysMsleep(100);
                }
            }
        }
    }
OsRfPowerOff();
OsRfClose();
```

15.10 OsRfRemove

Prototype		s32 OsRfRemove (void)
Function		Enable the removal of Contactless card detection
Parameter	**Input**	N/A
	Output	N/A
Return		API_OK: success API_ERR: fail API_EIO: device not open
Note		Once the removal of Contactless cards detection is initiated, the final result needs to be obtained through OsRfGetStatus ()
Example		

```
int cRet = 0;
APDU_SEND ApduSend;
APDU_RESP ApduResp;
ApduSend.Command[0] = 0x00;
ApduSend.Command[1] = 0xa4;
ApduSend.Command[2] = 0x04;
ApduSend.Command[3] = 0x00;
ApduSend.Lc = 0x0e;
ApduSend.Le = 256;
memcpy(ApduSend.DataIn, "2PAY.SYS.DDF01", ApduSend.Lc);
OsRfOpen();
OsRfPownOn(PICC_TYPE_A);
cRet = OsRfGetStatus();
if (cRet == 3) {
    if (!OsRfActivate())
    {
        if (!OsRfExchangeApdu(&ApduSend, &ApduResp)) {

            while (1) {
                cRet = OsRfRemove();
                if (cRet == 0)
                    break;
```

```
                    OsSysMsleep(100);
                }
            }
        }
    }
OsRfPowerOff();
OsRfClose();
```

15.11 OsRfIoctl

Prototype		s32 OsRfIoctl(u32 nCmd, u32 lParam, u32 wParam)
Function		Operation of the contactless card reader
Parameter	**Input**	nCmd: command strings

Serial number	Name of Command	**CMD Code**	Code Instructions
1	API_RF_CTL_VER	**0**	Get the version of the Contactless card reader
2	API_RF_CTL_SAK	**1**	Get the SAK value of selection responding
3	API_RF_CTL_UID	**2**	Card ID
4	API_RF_CTL_MF_AUTH	**3**	Mifare card certification
5	API_RF_CTL_MF_READ_RAW	**4**	Read data of the origin binary
6	API_RF_CTL_MF_WRITE_RAW	**5**	Write data of the origin binary
7	API_RF_CTL_MF_READ_VALUE	**6**	Read value
8	API_RF_CTL_MF_WRITE_VALUE	**7**	Write value
9	API_RF_CTL_MF_INC_VALUE	**8**	Operation of increasing value
10	API_RF_CTL_MF_DEC_VALUE	**9**	Operation of decreasing value
11	API_RF_CTL_MF_BACKUP_VALUE	**10**	Value backup
12	API_RF_CTL_SET_PARAM	**11**	Set parameter of the contactless chip
13	API_RF_CTL_GET_PARAM	**12**	Get parameter of the contactless chip

Command descriptions：

1) API_RF_CTL_VER lParam：

wParam：Transfer format(u8 *），Version（the strings ending with '\0'， available length of the string fixed at 17Bytes）

Return：API_OK：Success API_ERR

		：Failure API_EIO： device is not open or there is a wrong read 2) API_RF_CTL_SAK lParam： wParam：SAK Value（1Byte） Return：API_OK：Success API_ERR：Failure API_EINVAL： Parameter error API_EIO：device not open or read error Remark：1、SAK is only applied to tpyeA Card 3) API_RF_CTL_UID lParam： wParam：UID data，transfer format(u8 *），B.. LLVAR，the max length N=10; Return：API_OK：Success API_ERR：Failure API_EINVAL： Parameter error API_EIO：device not open or read error Remark：1、The UID data cache must be no less than the maximum length of the UID data (N) + 1 4) API_RF_CTL_MF_AUTH lParam：Mifare card certification construction，transfer format(strMfAuth *） typedef struct _strMfAuth { u8 m_authmode;//certifying method，0x60：A certifying，0x61：B certifying u8 m_key[6];//certifying key，Fixed at 6Bytes u8m_uid[10];//UID, executed using 4Bytes u8m_block; } strMfAuth; wParam： Return：API_OK：Success API_ERR：Failure API_EINVAL： Parameter error API_EIO：device not open or read error 5) API_RF_CTL_MF_READ_RAW lParam：Block code， wParam：Block data，tranfer format(u8 *），B16 ; Return：API_OK：Success API_ERR：Failure API_EINVAL： Parameter error API_EIO：device not open or read error 6) API_RF_CTL_MF_WRITE_RAW lParam： wParam：Block data，Transfer format(u8 *）， B16 Return：API_OK：Success API_ERR：Failure API_EINVAL： Parameter error API_EIO：device not open or read error 7) API_RF_CTL_MF_READ_VALUE lParam：Block code wParam：(u32*)Block value; Return：API_OK：Success API_ERR：Failure

		API_EINVAL： Parameter error API_EIO：device not open or read error 8) API_RF_CTL_MF_WRITE_VALUE lParam：Block code wParam：Block value Return：API_OK：Success API_ERR：Failure API_EINVAL： Parameter error API_EIO：device not open or read error 9) API_RF_CTL_MF_INC_VALUE lParam：Block code wParam：Increasing value Return：API_OK：Success API_ERR：Failure API_EINVAL： Parameter error API_EIO：device not open or read error 10) API_RF_CTL_MF_DEC_VALUE lParam：Block code wParam：decreasing value Return：API_OK：Success API_ERR：Failure API_EINVAL： Parameter error API_EIO：device not open or read error 11) API_RF_CTL_MF_BACKUP_VALUE lParam：Source block code wParam： target block code; Return：API_OK：Success API_ERR：Failure API_EINVAL： Parameter error API_EIO：device not open or read error 12) API_RF_CTL_SET_PARAM lParam ： the construction of the contactless chip parameter ， tranfer format(strRfChipParam*） typedef struct _strRfChipParam { u8 m_modindex;//debug depth u8 m_impedance;//output impedance u8 m_typeagain;//TypeA gain u8 m_typebgain;//TypeB gain u8 m_felicagain;//FeliCa gain u8 m_rfu[28]; //reserve }strRfChipParam; wParam： Return：API_OK：Success API_ERR：Failure API_EINVAL： Parameter error API_EIO ： device not open or read error Remark：1、Internal reserved interface 13) API_RF_CTL_GET_PARAM lParam： wParam：Output the construction of the contactless chip parameter,transfer format

		(strRfChipParam*） typedef struct _strRfChipParam { u8 m_modindex;//debug depth u8 m_impedance;//Output impedance u8 m_typeagain;//TypeA gain u8 m_typebgain;//TypeB gain u8 m_felicagain;//FeliCa gain u8 m_rfu[28]; //reserve } strRfChipParam; Return：API_OK：Success API_ERR：Failure API_EIO ： device not open or read error Remark：1、internal reserved interface
Return		API_OK：Success API_ERR：Failure API_EIO：device not open
Remark		1. Once start removing contactless card checking, need to get the removing result through ddi rf get_status
Example		Referred to the controlling of the contactless cases.

16 Keypad

16.1 Summary

The keypad module mainly provides functions for read keypad value and set processing modes of keypad.

16.2 error code

16.3 OsKeypadGetKey

Prototype		uint8_t OsKeypadGetKey(void);
Function		Read the physical keyboard of keypad equipment
Parameter	**Input**	N/A
	Output	N/A
Return		>0 ： the obtained key values are defined as follows #define KEYPAD_NO_KEY 0x00 #define KEYPAD_VOLUME_INC 0x81 #define KEYPAD_VOLUME_DEC 0x82 #define POWER 0x80 // power key #define KEYPAD_POWER_LONG 0x84 // power key long #define PGUP 0x2a // up #define PGDOWN 0x23 // down #define ENTER 0x18 // enter #define FUNCTION 0x70 // Function

	#define CLEAR 0x17 // clear #define CANCEL 0x1b // cancel #define PLUS 0x68 // plus #define DIGITAL0 0x30 #define DIGITAL1 0x31 #define DIGITAL2 0x32 #define DIGITAL3 0x33 #define DIGITAL4 0x34 #define DIGITAL5 0x35 #define DIGITAL6 0x36 #define DIGITAL7 0x37 #define DIGITAL8 0x38 #define DIGITAL9 0x39 0:N/A
Note	The application needs to handle key messages by itself. When this interface is not called, the system will only handle the power and volume keys by default. If the application needs to control the volume and power keys, please use OsKeypadSetMode(1);
Example	while(1){ u32 key = 0; key = OsKeypadGetKey(); if(key!=0) { if(key == POWER) { } else if((key == VOLUME_INC)||(key == VOLUME_DEC)) { OsLogOutput(0,"volume key press\r\n"); } else { } } OsSysMsleep(10); }

16.4 OsKeypadSetMode

Prototype	void OsKeypadSetMode(unsigned char mode);
Function	Set keypad process mode

Parameter	**Input**	Mode: 1: shielding system key handling Mode: 0: allow system key processing
	Output	N/A
Return		N/A
Note		The default key processing mode is 0, indicating that the system layer will default to processing the power key and volume up/down keys. When the key processing mode is set to 1, it means that the system layer no longer controls the power key and volume up/down keys. If the application wants to control them, it needs to be processed by the application layer itself.
Example		OsKeypadSetMode(1);

17 USB Module

17.1 Summary

The USB module is mainly provided for communication process between applications and host computers. At present, it includes open, close, read and write, and inserting.

17.2 error code

#define API_USBOPEN_ERROR -0x1101
#define API_USBCLOSE_ERROR -0x1102
#define API_USBREAD_ERROR -0x1103
#define API_USBWRITE_ERROR -0x1104

17.3 OsUsbOpen

Prototype		s32 OsUsbOpen(void);
Function		USB Opens
Parameter	**Input**	N/A
	Output	N/A
Return		0: success other: failure
Note		Only support virtual PORT6 port

Example	s32 ret=OsUsbOpen();

17.4 OsUsbClose

Prototype		s32 OsUsbClose(void);
Function		USB closes
Parameter	**Input**	N/A
	Output	N/A
Return		0: success other: failure
Note		The prerequisite is OsUsbOpen.
Example		s32 ret=OsUsbClose();

17.5 OsUsbRead

Prototype		s32 OsUsbRead(u8 *lpOut, u32 nLe);
Function		USB reads data
Parameter	**Input**	N/A
	Output	lpOut: recvbuf nLe: bufsize
Return		>=0: read data length other: failure
Note		The host computer needs to open the virtual serial port PORT6 to receive successfully.
Example		unsigned char buf[32]; OsUsbRead(buf, 32);

17.6 OsUsbWrite

Prototype		s32 OsUsbWrite(u8 *lpIn, u32 nLe);
Function		USB sends data
Parameter	**Input**	lpOut: recvbuf nLe: bufsize

	Output	N/A
Return		>=0: actual length of sent data other: failure
Note		The host computer needs to open the virtual serial port PORT6 to receive successfully.
Example		unsigned char buf[32]; OsUsbWrite(buf, 32);

17.7 OsUsbIsConnected

Prototype		s32 OsUsbIsConnected()
Function		Evaluate whether the USB is connected
Parameter	**Input**	N/A
	Output	N/A
Return		1: connected 0: disconnected
Note		
Example		s32 ret= OsUsbIsConnected();

18 Camera Module

18.1 Summary

This module is used for passive scanning transactions. When the customer displays a QR code, the device can activate the preview mode and then process the scanned QR code to complete the transaction.

18.2 error code

#define API_CAMERASTOP_ERROR -0x1201
#define API_CAMERASTART_ERROR -0x1202
#define API_CAMERADECODE_ERROR -0x1203
#define API_CAMERAINIT_ERROR -0x1204
#define API_CAMERADEINIT_ERROR -0x1205

18.3 OsCameraInit

Prototype	int OsCameraInit();
Function	Camera initialization

Parameter	**Input**	N/A
	Output	N/A
Return		0: success other: failure
Note		This module consumes a large amount of memory, make sure the thread invoking it has sufficient stack space.
Example		int ret=OsCameraInit();

18.4 OsCameraStartPreview

Prototype		int OsCameraStartPreview();
Function		Start camera preview
Parameter	**Input**	N/A
	Output	N/A
Return		0: success other: failure
Note		The prerequisite is OsCameraInit
Example		int ret=OsCameraStartPreview();

18.5 OsCameraStopPreview

Prototype		int OsCameraStopPreview();
Function		Stop camera preview
Parameter	**Input**	N/A
	Output	N/A
Return		0: success other: failure

Note	The prerequisite is OsCameraInit、OsCameraStartPreview
Example	int ret=OsCameraStopPreview();

18.6 OsCameraDecode

Prototype		int OsCameraDecode(uint8_t* sweep_code_buf, uint32_t* sweep_code_len, int* sweep_code_type, int timeout);
Function		The camera decodes barcodes and QR codes
Parameter	**Input**	Timeout: timeout(ms)
	Output	sweet_comde-buf: decoded buffer sweet_comde_1en: length after decoding sweet_comde_type: the code types after decoding include the following: (UPCA, 1, C39, 2, C128, 3, ITF25, 4, C93, 5, PDF417, 6, QR, 7, DATAMATRIX, 8, CBAR, 9, UPCE, 10, EAN8, 11, EAN13, 12)
Return		0: success -0xff: timeout other: fail
Note		The prerequisite is OsCameraInit and the decoded string buffer needs to be large enough
Example		uint8_t sweep_code_buf[128]; uint32_t sweep_code_len; int sweep_code_type; int OsCameraDecode(sweep_code_buf, &sweep_code_len, &sweep_code_type, 5000);

18.7 OsCameraGetIsPreview

Prototype		int OsCameraGetIsPreview();
Function		Determine whether the camera has enabled preview function
	Input	N/A

Parameter	**Output**	N/A
Return		0: on 1: off
Note		The prerequisite is OsCameraInit();
Example		int ret=OsCameraGetIsPreview();

18.8 OsCameraDeinit

Prototype		int OsCameraDeinit();
Function		camera deinit
Parameter	**Input**	N/A
	Output	N/A
Return		0: success other:fail
Note		
Example		OsCameraDeinit();

18.9 OsCameraSetPreviewShow

Prototype		void OsCameraSetPreviewShow(int flag);
Function		Whether to enable preview
Parameter	**Input**	flag：1-enable 0-disable
	Output	N/A
Return		0: success other:fail
Note		
Example		OsCameraSetPreviewShow(1);

19 Segmented Display Module

19.1 Summary

This module is used to display information including communication signals strength, battery level, and amount.

19.2 error code

#define API_SEGCLEAR_ERROR -0x1301
#define API_SEGSHOWDOT_ERROR -0x1302
#define API_SHOWCSQ_ERROR -0x1303
#define API_SEGSHOWSOC_ERROR -0x1304
#define API_SEGSHOWMONEY_ERROR -0x1305
#define API_SEGSHOWCHAR_ERROR -0x1306

19.3 OsSegClear

Prototype		int OsSegClear(void);
Function		Clear seg display content
Parameter	**Input**	N/A
	Output	N/A
Return		0: success
Remark		
Example		

19.4 OsSegShowDot

Prototype		int OsSegShowDot(bool isShow);
Function		The segmented display screen shows the decimal point
Parameter	**Input**	isShow: true show false not show
	Output	NA
Return		0: success

Remark	
Example	

19.5 OsSegShowMoney

Prototype		int OsSegShowMoney(char* str);
Function		Segmented display shows amount (The DS50 has 8 digits)
Parameter	**Input**	Str: amount strings
	Output	N/A
Return		0: success
Remark		
Example		

20 Buzzer Module

20.1 Summary

This module is used to handle buzzer related operations, providing prompts when transaction succeeded or failed.

20.2 error code

#define API_BEEPPLAY_ERROR -0x1401

20.3 OsBeepPlaying

Prototype		s32 OsBeepPlaying(int frequency);
Function		Buzzer Play
Parameter	**Input**	frequency:The parameters are retained and are not currently effective
	Output	N/A
Return		0:success other: failure
Note		

Example	OsBeepPlaying(0);

21 SECURITY Module

21.1 Summary

This module is mainly responsible for the communication between AP and security chip (SP), including random data acquisition, algorithm processing, key processing, PINBLOCK processing, security trigger processing and etc.

21.2 error code

21.3 OsSecGetRandNum

Prototype		S32 OsSecGetRandNum(u8 *pData, u32 dataLen);
Function		Obtaining Soft Random Data
Parameter	**Input**	N/A
	Output	pData: buffer dataLen: buffer size
Return		0 – success; other – failure
Note		
Example		u8 data[12]; OsSecGetRandNum(data,12);

21.4 OsSecCalcDes

Prototype		S32 OsSecCalcDes(s32 mode, u8 *data, u8 inlen,u8 *key, u32 key_len, u8* dataout, u32 *dataoutlen);
Function		DES encryption and decryption
Parameter	**Input**	mode: Mode 0x00: Decryption 0x01: Encryption data: Data inlen: Data length (multiple of 8 bytes) key: Key key_len: Key length (8, 16, 24 bytes)

	Output	Dataoutlen: length of data encrypted and decrypted dataout: encrypted and decrypted data
Return		0 - success; other - failure
Note		
Example		unsigned char data[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12 ,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x19}; unsigned char key[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x13 }; u8 out[16]; u32 outlen = 0; ret = OsSecCalcDes(0x00, data, 16,key, 16, out, &outlen);

21.5 OsSecCalcDesCbc

Prototype		S32 OsSecCalcDesCbc(s32 mode, u8* data, u32 data_len, u8* key, u32 key_len, u8* iv, u8* out, u32* outlen);
Function		DES encryption and decryption with CBC mode
Parameter	**Input**	Mode: Mode 0x00: Decryption 0x01: Encryption data: Data inlen: Data length (multiple of 8 bytes) key_1en: Key length (8, 16, 24 bytes) key: Key Iv: Initial vector (8 bytes)
	Output	dataoutlen: length of data encrypted and decrypted dataout: encrypted and decrypted data
Return		0 – success; other – failure
Note		
Example		unsigned char data[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12 ,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x19}; unsigned char iv[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12 }; unsigned char key[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x13 }; u8 out[16]; u32 outlen = 0; ret = OsSecCalcDesCbc(0x00, data, 16,key, 16, iv, out, &outlen);

21.6 OsSecCalcAes

Prototype		S32 OsSecCalcAes(s32 mode, u8* datain, u32 inlen, u32 keylen, u8 *keydata, u8* dataout, u32 *outlen);
Function		AES Encryption and Decryption (ECB)
Parameter	**Input**	Mode: Mode 0x00: Decryption 0x01: Encryption inlen: Data length (multiples of 16 bytes) datain: Data keylen: Key length(16 bytes) Keydata: Key
	Output	outlen: length of data encrypted and decrypted dataout: encrypted and decrypted data
Return		0 – success; other – failure
Note		
Example		unsigned char data[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x19}; unsigned char key[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x13 }; u8 out[16]; u32 outlen = 0; int ret = OsSecCalcAes(0x00, data, 16, key, 16,out, &outlen);

21.7 OsSecCalcAesCbc

Prototype		S32 OsSecCalcAesCbc(s32 mode, u8* data, u32 data_len, u8* key, u32 key_len, u8* iv, u8* out, u32* outlen);
Function		AES Encryption and Decryption (CBC)
Parameter	**Input**	Mode: Mode 0x00: Decryption 0x01: Encryption inlen: Data length (multiples of 16 bytes) datain: Data keylen: Key length keydata: Key iv: Initial vector (16 bytes)
	Output	outlen: length of data encrypted and decrypted out: encrypted and decrypted data

Return	0 – success; other – failure
Note	
Example	unsigned char data[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x19}; unsigned char iv[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12 }; unsigned char key[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x13 }; u8 out[16]; u32 outlen = 0; int ret = OsSecCalcAesCbc(0x00, data, 16, key, 16, iv, out, &outlen);

21.8 OsSecCalcEccEnc

Prototype		s32 OsSecCalcEccEnc(u32 idx, u8 *pData, u32 datalen, u8 *pCipher, u32 *pCipherLen);
Function		ECC encryption
Parameter	**Input**	idx: Index pData: Data datalen: Data length
	Output	pCipher: encrypted data PCipherLen: length of data encrypted
Return		0 – success; other – failure
Note		
Example		static int bn_get_digit(uint32_t* d, char c) { *d = 255; if (c >= 0x30 && c <= 0x39) *d = c - 0x30; if (c >= 0x41 && c <= 0x46) *d = c - 0x37; if (c >= 0x61 && c <= 0x66) *d = c - 0x57; if (*d >= (uint32_t)16) return(-1); return(0); } s32 bn_read_string_from_head(uint8_t* r, uint32_t len, const char* s) {

```
    int32_t i, j, slen;
    uint32_t d;

    if ((slen = strlen(s)) < 0)
    {
        return -1;
    }

    if ((len * 2) < slen)
        return(-1);

    if (slen & 1)
    {
        slen += 1;
    }

    memset(r, 0, len);

    len = slen >> 1;

    for (i = slen, j = 0; i > 0; i--, j++)
    {
        if (-1 == bn_get_digit(&d, s[i - 1]))
            return (-1);
        r[len - 1 - j / 2] |= d << ((j % 2) << 2);
    }

    return(len);
}


#define    KEY_LEN_MAX    ((521+7)>>3)
#define                                                                    ECC_PRIB_256r1
"E4E8118F5ED6BF84D3188391F147A25ADF3993C9AB59A4713968F73F066905AE"
#define                                                                   ECC_PUBXB_256r1
"A932895FC829657D0156F7A05829354DF18C74B66AD23985D8F2B5A4DD2AB817"
#define                                                                   ECC_PUBYB_256r1
"25BB6882ABB9C880172568047E0A85D392B75A73BDE2487C4AC525DADF763593"



int ret=-1;
u8 cipher_data[256] = {0};
u8 plain_data[256] = {0};

uint8_t au8Puk[KEY_LEN_MAX+KEY_LEN_MAX];
uint8_t au8Pri[KEY_LEN_MAX],au8PubX[KEY_LEN_MAX],au8PubY[KEY_LEN_MAX];
```

	bn_read_string_from_head(au8PubX, sizeof(au8PubX),ECC_PUBXB_256r1); bn_read_string_from_head(au8PubY, sizeof(au8PubY),ECC_PUBYB_256r1); bn_read_string_from_head(au8Pri, sizeof(au8Pri),ECC_PRIB_256r1); ret = OsSecUpdateEccKey(0,0,au8Puk,au8Pri); ret = OsSecCalcEccEnc (0,plain_data,16,cipher_data,&cipher_data_len); for(i=0;i<cipher_data_len;i++) { SEC_TEST_DEBUG("%02x ",cipher_data[i]); } memset(plain_data,0x00,sizeof(plain_data)); ret = OsSecCalcEccDec (0,cipher_data,cipher_data_len,plain_data,&plain_data_len);

21.9 OsSecCalcEccDec

Prototype		s32 OsSecCalcEccDec(u32 idx, u8 *pData, u32 datalen, u8 *pPlaintext, u32 *pPlaintextLen);
Function		ECC decryption
Parameter	**Input**	idx: Index pData: Data datalen: Data length
	Output	pPlaintext: decrypted data pPlaintextLen: length of data decrypted
Return		0 – success; other – failure
Note		
Example		static int bn_get_digit(uint32_t* d, char c) { *d = 255; if (c >= 0x30 && c <= 0x39) *d = c - 0x30; if (c >= 0x41 && c <= 0x46) *d = c - 0x37; if (c >= 0x61 && c <= 0x66) *d = c - 0x57; if (*d >= (uint32_t)16) return(-1); return(0); } s32 bn_read_string_from_head(uint8_t* r, uint32_t len, const char* s)

```
{
    int32_t i, j, slen;
    uint32_t d;

    if ((slen = strlen(s)) < 0)
    {
        return -1;
    }

    if ((len * 2) < slen)
        return(-1);

    if (slen & 1)
    {
        slen += 1;
    }

    memset(r, 0, len);

    len = slen >> 1;

    for (i = slen, j = 0; i > 0; i--, j++)
    {
        if (-1 == bn_get_digit(&d, s[i - 1]))
            return (-1);
        r[len - 1 - j / 2] |= d << ((j % 2) << 2);
    }

    return(len);
}



#define    KEY_LEN_MAX    ((521+7)>>3)
#define                                                                    ECC_PRIB_256r1
"E4E8118F5ED6BF84D3188391F147A25ADF3993C9AB59A4713968F73F066905AE"
#define                                                                   ECC_PUBXB_256r1
"A932895FC829657D0156F7A05829354DF18C74B66AD23985D8F2B5A4DD2AB817"
#define                                                                   ECC_PUBYB_256r1
"25BB6882ABB9C880172568047E0A85D392B75A73BDE2487C4AC525DADF763593"



int ret=-1;
u8 cipher_data[256] = {0};
u8 plain_data[256] = {0};

uint8_t au8Puk[KEY_LEN_MAX+KEY_LEN_MAX];
uint8_t au8Pri[KEY_LEN_MAX],au8PubX[KEY_LEN_MAX],au8PubY[KEY_LEN_MAX];
```

	bn_read_string_from_head(au8PubX, sizeof(au8PubX),ECC_PUBXB_256r1); bn_read_string_from_head(au8PubY, sizeof(au8PubY),ECC_PUBYB_256r1); bn_read_string_from_head(au8Pri, sizeof(au8Pri),ECC_PRIB_256r1); ret = OsSecUpdateEccKey(0,0,au8Puk,au8Pri); ret = OsSecCalcEccEnc (0,plain_data,16,cipher_data,&cipher_data_len); for(i=0;i<cipher_data_len;i++) { OsLogOutput("%02x ",cipher_data[i]); } memset(plain_data,0x00,sizeof(plain_data)); ret = OsSecCalcEccDec (0,cipher_data,cipher_data_len,plain_data,&plain_data_len);

21.10 OsSecCalcMd5

Prototype		s32 OsSecCalcMd5(u8* pData, u32 dataLen, u8* pHashValue, u32* outlen);
Function		MD5 Algorithm
Parameter	**Input**	pData: data Datalen: datalen
	Output	pHashValue: data after summarization outlen: Data length after summarization
Return		0 – success; other – failure
Note		
Example		unsigned char data[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x19 }; int ret=0; u8 out[128]; u32 outlen = 0; ret = OsSecCalcMd5(data, sizeof(data) / sizeof(data[0]), out, &outlen); if (outlen > 0) { api_log_DEBUG_HEX(out, outlen); }

21.11 OsSecCalcSha1

Prototype	s32 OsSecCalcSha1(u8 *pData, u32 dataLen, u8 *pHashValue, u32* outlen)

Function		SHA1 Algorithm
Parameter	**Input**	pData: data dataLen: data length
	Output	pHashValue: data after summarization outlen: Data length after summarization
Return		0 – success; other – failur
Note		
Example		unsigned char data[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12 ,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x19 }; u8 out[128]; u32 outlen = 0; OsSecCalcSha1(data, sizeof(data) / sizeof(data[0]), out, &outlen);

21.12 OsSecCalcSha256

Prototype		s32 OsSecCalcSha256(u8 *pData, u32 dataLen, u8 *pHashValue， u32* outlen);
Function		SHA256 Algorithm
Parameter	**Input**	pData: data dataLen: data length
	Output	pHashValue: Data after summarization outlen: Data length after summarization
Return		0 – success; other – failure
Note		
Example		unsigned char data[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12 ,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x19 }; u8 out[128]; u32 outlen = 0; OsSecCalcSha256(data, sizeof(data) / sizeof(data[0]), out, &outlen);

21.13 OsSecCalcSha512

Prototype	s32 OsSecCalcSha512(u8* pData, u32 dataLen, u8* pHashValue, u32* outlen)
Function	SHA512 Algorithm

Parameter	**Input**	pData: data dataLen: data length
	Output	pHashValue: Data after summarization outlen: Data length after summarization
Return		0 – success; other – failure
Note		
Example		unsigned char data[] = { 0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x19 }; u8 out[128]; u32 outlen = 0; OsSecCalcSha512(data, sizeof(data) / sizeof(data[0]), out, &outlen);

21.14 OsSecCalcMac

Prototype		s32 OsSecCalcMac(u32 mode, u8 alg, u32 idx, u8 *pData, u32 dataLen, u8 *pMac)
Function		MAC Algorithm
Parameter	**Input**	mode: 0-MAC_ALG_XOR 、1-MAC_ALG_99 2-MAC_ALG_919 3-MAC_ALG_PBOC
		alg: 0-des 1-tdes 3-aes
		idx: Key index
		pData: Data to be calculated
		dataLen: Data length
	output	pMac: Mac value
Return		0 – success; other – failure
Note		
Example		

21.15 OsSecSetPinblockParam

Prototype	s32 OsSecSetPinblockParam(s32 mode,u8 alg, u8 idx, u8 minLen, u8 maxLen,u8 timeOut,u8 lenCard, u8 *pCardNo,u8 mark,u8 lenAmount, u8 *amount);

Function		Set Pinblock Parameter
Parameter	Input	Mode: Mode (not used yet) Alg: Algorithm Idx: Key index MinLen: Minimum length MaxLen: Maximum length TimeOut: Timeout period LenCard: Card length PCardNo: Card number Mark: Not used yet LenAmount: Amount length Amount: Amount
	Output	
Return		0 – success; other – failure
Note		
Example		

21.16 OsSecSetPinpadOfflineModeParam

Prototype		s32 OsSecSetPinpadOfflineModeParam(s32 mode, u8 cardType, u8 minLen, u8 maxLen, u8 timeOut,u32 lenMod,u8 *mod, u32 lenE, u8 *e, u32 lenRand, u8 *rand);
Function		Set Pinpad Offline Parameter
Parameter	Input	Mode: Mode Offline type 0 - plaintext PIN, 1- ciphertext PIN (ciphertext requires the transmission of entering public key information, and plaintext is not required) CardType: Card type 0 - contactless IC card, 1 - contact IC card MinLen: Minimum length MaxLen: Maximum length TimeOut: Timeout
	Output	LenMod: Length of public key module Mod: Public key module LenE: Length of public key index E: Public key index LenRand: Length of public key random number Rand: Public key random number
Return		0 – success; other – failure

Note	
Example	

21.17 OsSecGetPinblockStatus

Prototype		s32 OsSecGetPinblockStatus(u8 *data, u8 *datalen);
Function		Get Pinblock
Parameter	**Input**	
	Output	Datalen: Data length (M bytes) Data: Data M=1： '*': numeric key value #: Other buttons 0x18: Confirm key (password free transaction) 0x17: Clear key 0x1B: Cancel key 0x02: Error M=16: PINBLOCK data
Return		0 – success; other – failure
Note		
Example		

21.18 OsSecCalcCrc16

Prototype		s32 OsSecCalcCrc16(u8 *pData, u32 dataLen);
Function		CRC16 Algorithm
Parameter	**Input**	pData: Data datalen: Data length
	Output	
Return		CRC16 value
Note		
Example		

21.19 OsSecCalcRsaGen

Prototype		s32 OsSecCalcRsaGen(s32 mode, u32 modlen, s32 exp, u32* pPkeylen, u8 *pPkey, u32* pSkeylen, u8 *pSkey);
Function		RSA Key Generation
Parameter	**Input**	mode: Mode (not used yet) model: Model length (supports 1024 and 2048 only) exp: Index 0 x10001
	Output	PPkeylen: Public key length PPkey: Public key PSkeylen: Private key length PSkey: Private key
Return		0 – success; other – failure
Note		
Example		

21.20 OsSecCalcRsaPk

Prototype		S32 OsSecCalcRsaPk(u32 mode, u32 idx, u8 *pData, u32 datalen, u8 *pCipher, u32 *pCipherLen);
Function		RSA Public Key Operation
Parameter	**Input**	Mode: Mode 0x00: Public key 1024 bits decryption 0x01: Public key 1024 bits encryption 0x02: Public key 2048 bits decryption 0x03: Public key 2048 bits encryption Idx: Index PData: Data Datalen: Data length
	Output	PCipher: Data processed PCipherLen: Length of data processed
Return		0 – success; other – failure
Note		
Example		

21.21 OsSecCalcRsaSk

Prototype		S32 OsSecCalcRsaSk(u32 mode, u32 idx, u8 *pData, u32 datalen, u8 *pCipher, u32 *pCipherLen);
Function		RSA Private Key Operation
Parameter	**Input**	Mode: Mode 0x00: Private key 1024 bits decryption 0x01: Private key 1024 bits encryption 0x02: Private key 2048 bits decryption 0x03: Private key 2048 bits encryption Idx: Index PData: Data Datalen: Data length
	Output	PCipher: Data processed PCipherLen: Length of data processed
Return		0 – success; other – failure
Note		
Example		

21.22 OsSecCalcDukptMac

Prototype		s32 OsSecCalcDukptMac(s32 macMode,u8 alg,u32 keyIndex,u8* pData,u32 dataLen,u8*pMac,u32 *pMacLen,u8* pKsn,u32* pKsnLen);
Function		MAC Calculation By DUKPT
Parameter	**Input**	macMode: Xor ANSI9.9 PBOC ANSI9.19 Alg: 0 - Supports 3des only keyIndex: 0 - 9 pData: Data dataLen: Data length
	Output	pMac: Calculated MAC data pMacLen: Data length of calculated MAC pKSn: KSN data pKsnLen: KSN data length

Return	0 – success; other – failure
Note	
Example	

21.23 OsSecCalcDukptData

Prototype		s32 OsSecCalcDukptData(s32 mode,u8 alg,u8 key_type,u32 keyIndex,u8* pdataIn,u32 dataInLen,u8* pDataOut,u32* pOutLen,u8* pKsn,u32* pKsnLen);
Function		Data Calculation By DUKPT
Parameter	**Input**	Mode: 0x00: ECB Decryption 0x01: ECB Encryption 0x10: CBC decryption 0x11: CBC encryption Alg: 0 - Supports 3des only key_type:0-pinkey 2-dataRequestKey 4-dataResponseKey keyIndex:0 - 9 pData: Data dataLen: Data length
	Output	pMac: Calculated data pMacLen: Length of calculated data pKSn: KSN data pKsnLen: KSN data length
Return		0 – success; other – failure
Note		
Example		u8 ipek[16] = { 0x13,0x00,0x73,0x1D,0x1E,0xB9,0xAF,0xF9,0xC5,0x59,0x19,0x16,0xC3,0x7C,0x9A,0x88 }; u8 ksn[10] = { 0x29,0x00,0x08,0x01,0x24,0x00,0x21,0xe0,0x00,0x01}; u8 outdata[256]={0},ksndata[24] = {0X62,0X22,0X83,0X01,0X53,0X27,0X80,0X55,0XD2,0X70,0X32,0X06,0X04,0X90,0X82,0X44,0X00,0X0F,0X06,0X06,0X06,0X06,0X06,0X06}; u32 outlen = 0, ksnlen = 0; unsigned char kcv[4] = { 0x5f, 0x49, 0x9e, 0xF8 }; OsSecUpdateDukpt(0,0,0,16,ipek, ksn,kcv); OsSecCalcDukptData(0x11, 0, 0, 0,ksndata, 16, outdata, &outlen, ksn, &ksnlen); //res:77 33 59 5E 5D B3 89 CA 65 50 3B 9E 7B 24 A8 E7 60 B4 17 33 13 A2 D2 D9

21.24 OsSecUpdatePlainKey

Prototype		s32 OsSecUpdatePlainKey(u8 type, u8 alg, u32 idx, u8 *pMkey, u32 mkeyLen);
Function		Update Plaintext Key
Parameter	**Input**	Type: Type 0x00- Master key 0x01- Work key 0x02- Hardware serial number key(unsupport) 0x03- Customer customized key(unsupport) 0x04- Terminal authentication key(unsupport) 0x05- Transmission key 0x06- TR31 KBPK key Alg: Algorithm 0x00 – DES 0x01 – 3DES 0x02 – SM4(unsupport) 0x03 – AES Idx: Index PMkey: Key MkeyLen: Key length
	Output	
Return		0 – success; other – failure
Note		
Example		

21.25 OsSecUpdateCipherKey

Prototype		s32 OsSecUpdateCipherKey(u8 type, u8 mk_alg, u8 wk_alg, u32 mk_idx, u32 wk_idx, u8 *pCipher, u32 cipherLen, u8 chvmod, u8 *chkval, u32 chkvalLen);
Function		Update Ciphertext Key
		Type: Type 0x00- Master key 0x01- Work key 0x02- Hardware serial number key 0x03- Customer customized key 0x04- Terminal authentication key 0x11- TPK (used to calculate PINBLOCK) 0x21- TAK (used to calculate MAC) 0x31- TDK (used for track data encryption)

Parameter	**Input**	0x41- TFK (used for facial data encryption) Mk_alg: Master key algorithm Wk_alg: Work key Mk_idx: Master key index Wk_idx: Work key index PCipher: Key CipherLen: Key length Chvmod: Verified or not Chkval: Verification value ChkvalLen: Verification length
	Output	
Return		0 – success; other – failure
Note		
Example		

21.26 OsSecUpdateDukpt

Prototype		s32 OsSecUpdateDukpt(u8 group, u8 type, u8 idx, u8 sec_len, u8 *ipek, u8 *ksn, u8* check_crc);
Function		Update DUKPT Key
Parameter	**Input**	Group: Group number (currently supports 0 only) Type: Type (0 - IPEK) Idx: Index (starting from 0, up to 4) Sec_1en: Key length Ipek: ipek 24 bytes Ksn: KSN (DES-10bytes AES-12bytes) Check_crc: Verification value (IPEK encrypts 8 zeros using DES, take the first 4 Bytes), 4 bytes
	Output	
Return		0 – success; other – failure
Note		
Example		u8 ipek[16] = { 0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF,0xFE,0xDC,0xBA,0x98,0x76,0x54,0x32,0x10 };

	u8 ksn[10] = { 0xFF,0xFF,0x98,0x76,0x54,0x32,0x10,0xe0,0x00,0x00 }; u8 pinblock[8] = { 0xFB,0x48,0xBB,0x73,0xD7,0x75,0xD8,0xF3 }; u8 pin[4] = { 0x31,0x32,0x33,0x34 }; u8 cardno[16] = { 0x34,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x39 }; u8 outdata[256] = { 0 }, ksndata[16] = { 0 }; u8 flashdata[1024] = { 0 }; u32 outlen = 0, ksnlen = 0; unsigned char kcv[4] = { 0x08, 0xD7, 0xB4, 0xFB }; OsSecUpdateDukpt(0,0,0,16,ipek, ksn,kcv); OsSecCalcDukptData(1, 0, 0, ksndata, 16, outdata, &outlen, ksn, &ksnlen);

21.27 OsSecUpdateEccKey

Prototype		s32 OsSecUpdateEccKey(u32 mode, u32 idx, u8 *pPkey, u8 *pSkey);
Function		Update ECC Key
Parameter	**Input**	Mode: Mode (not used yet) Idx: Index PPkey: Public key (64 bytes) PSkey: Private key (32 bytes)
	Output	
Return		0 – success; other – failure
Note		
Example		static int bn_get_digit(uint32_t* d, char c) { *d = 255; if (c >= 0x30 && c <= 0x39) *d = c - 0x30; if (c >= 0x41 && c <= 0x46) *d = c - 0x37; if (c >= 0x61 && c <= 0x66) *d = c - 0x57; if (*d >= (uint32_t)16) return(-1); return(0); } s32 bn_read_string_from_head(uint8_t* r, uint32_t len, const char* s) {

```
    int32_t i, j, slen;
    uint32_t d;

    if ((slen = strlen(s)) < 0)
    {
        return -1;
    }

    if ((len * 2) < slen)
        return(-1);

    if (slen & 1)
    {
        slen += 1;
    }

    memset(r, 0, len);

    len = slen >> 1;

    for (i = slen, j = 0; i > 0; i--, j++)
    {
        if (-1 == bn_get_digit(&d, s[i - 1]))
            return (-1);
        r[len - 1 - j / 2] |= d << ((j % 2) << 2);
    }

    return(len);
}


#define    KEY_LEN_MAX    ((521+7)>>3)
#define                                                                    ECC_PRIB_256r1
"E4E8118F5ED6BF84D3188391F147A25ADF3993C9AB59A4713968F73F066905AE"
#define                                                                  ECC_PUBXB_256r1
"A932895FC829657D0156F7A05829354DF18C74B66AD23985D8F2B5A4DD2AB817"
#define                                                                  ECC_PUBYB_256r1
"25BB6882ABB9C880172568047E0A85D392B75A73BDE2487C4AC525DADF763593"



int ret=-1;
u8 cipher_data[256] = {0};
u8 plain_data[256] = {0};

uint8_t au8Puk[KEY_LEN_MAX+KEY_LEN_MAX];
uint8_t au8Pri[KEY_LEN_MAX],au8PubX[KEY_LEN_MAX],au8PubY[KEY_LEN_MAX];
```

	bn_read_string_from_head(au8PubX, sizeof(au8PubX),ECC_PUBXB_256r1); bn_read_string_from_head(au8PubY, sizeof(au8PubY),ECC_PUBYB_256r1); bn_read_string_from_head(au8Pri, sizeof(au8Pri),ECC_PRIB_256r1); ret = OsSecUpdateEccKey(0,0,au8Puk,au8Pri); ret = OsSecCalcEccEnc(0,plain_data,16,cipher_data,&cipher_data_len); for(i=0;i<cipher_data_len;i++) { SEC_TEST_DEBUG("%02x ",cipher_data[i]); } memset(plain_data,0x00,sizeof(plain_data)); ret = OsSecCalcEccDec (0,cipher_data,cipher_data_len,plain_data,&plain_data_len);

21.28 OsSecUpdateRsaKey

Prototype		s32 OsSecUpdateRsaKey(u32 mode, u32 idx, u32 modlen, u32 exp, u32 pPkeylen, u8 *pPkey, u32 pSkeylen, u8 *pSkey);
Function		Update RSA Key
Parameter	**Input**	Mode: Mode (not used yet) Idx: Index Model length: Model length Exp: Index PPkeylen: Public key length PPkey: Public key PSkeylen: Private key length PSkey: Private key
	Output	
Return		0 – success; other – failure
Note		
Example		

21.29 OsSecCheckKeyStatus

Prototype		s32 OsSecCheckKeyStatus(u8 type, u8 alg, u32 idx, u8 *crc16);
Function		Check Key Status
Parameter	**Input**	Type: Type Alg: Algorithm Idx: Index
	Output	crc16: Verification value
Return		0 – success; other – failure
Note		
Example		

21.30 OsSecDelKey

Prototype		s32 OsSecDelKey(u8 type, u8 alg, u32 idx);
Function		Delete Key
Parameter	**Input**	Type: Key usage Alg: Algorithm type Idx: Key Index
	Output	
Return		0 – success; other – failure
Note		
Example		

21.31 OsSecSmgtGetTamperStatus

Prototype	int OsSecSmgtGetTamperStatus(u8 *dataout, int *len);
Function	Obtain Security Trigger Status

Parameter	**Input**	
	Output	Dataout: Tamper data Len: Data length
Return		0 – success; other – failure
Note		
Example		

21.32 OsSecGetTamperAuthCode

Prototype		int OsSecGetTamperAuthCode(u8* outCodelen, u8* outCodedata)
Function		Obtain the clear trigger request code.
Parameter	**Input**	N/A
	Output	outCodelen: codelen outCodedata: code data
Return		0: Success -1: Failure. -2: The request code length exceeds 8. -3: No attack was detected.
Note		
Example		int codelen=0; u8 code[6]={0}; OsSecGetTamperAuthCode(&codelen, code);

21.33 OsSecTamperAuthRun

Prototype		int OsSecTamperAuthRun(char codeLen, u8* codedata)
Function		Execution clearance trigger.
Parameter	**Input**	codelen: codelen codedata: code data
	Output	N/A

Return	0: Success -1: The authorization code is invalid and the number of retries has reached the maximum allowed number. -2: The authorization code is invalid, but the number of retries has not reached the maximum allowed number. -3: Authorization was successful but clearing failed.
Note	1) The authorization code is invalid. Retries are allowed. If the number of retries is not exceeded, the request code will not be automatically changed. Only when the number of retries exceeds the limit will a new request code be generated. 2) The authorization code is valid, but the clearing fails. The request code remains unchanged and continuous retries are allowed.
Example	int codelen=6; u8 code[6]={0x31,0x32,0x33,0x34,0x35,0x36}; OsSecTamperAuthRun(codelen, code);

22 MQTT Module (4G Mode)

22.1 Summary

MQTT is an asynchronous communication message protocol based on the TCP/IP protocol stack, which is a lightweight publish / subscribe information transmission protocol. The currently encapsulated functions include MQTT account and password setting, SSL mode setting, linking, disconnecting, subscribing, unsubscribing, publishing, will message, and status query function. The supported version is 3.1.1.

22.2 error code

#define API_MQTTSETUSERNAMEPWD_ERROR -0x1801
#define API_MQTTSETPSK_ERROR -0x1802
#define API_MQTTSETTLSVER_ERROR -0x1803
#define API_MQTTSETWILL_ERROR -0x1804
#define API_MQTTCONNECT_ERROR -0x1805
#define API_MQTTCLOSE_ERROR -0x1806
#define API_MQTTSUB_ERROR -0x1807
#define API_MQTTUBSUB_ERROR -0x1808
#define API_MQTTPUB_ERROR -0x1809

22.3 OsMqttSetUserNamePwd

Prototype		int OsMqttSetUserNamePwd(const char* usrname, const char* pwd);
Function		Set MQTT account and password
Parameter	**Input**	User name: account PWD: password
	Output	nothing
Return		0 means success and others mean failure
Notes		The precondition is that the network connection is successful
Example		OsMqttSetUserNamePwd(“test”, “123456”);

22.4 OsMqttSetSslMode

Prototype		void OsMqttSetSslMode(int mode);
Function		Set MQTT SSL verification mode
Parameter	**Input**	mode: 0 means not verified and 1 means verified
	Output	nothing
Return		0 means success and others are failure
Notes		
Example		OsMqttSetSslMode(1);

22.5 OsMqttSetPsk

Prototype		int OsMqttSetPsk(const char* psk_id, const char* psk_key);
Function		Set MQTT shared key
Parameter	**Input**	psk_id: shared Key id psk_key: shared Key
	Output	nothing
Return		0 means success, and others are failure
Notes		
Example		

22.6 OsMqttSetWill

Prototype		int OsMqttSetWill(char* topic, unsigned char qos, bool retain, char* message);
Function		Set MQTT testament message
Parameter	**Input**	topic: theme qos: the QoS level; values are 0, 1 and 2 retain: Whether to keep it or not? The values are 0 and 1 message: Testament message
	Output	nothing
Return		0 means success and others are failure
Notes		Must be done before OsMqttConnect and the settings work only
Example		OsMqttSetWill(“/test”,1,1,”test”);

22.7 OsMqttConnect

Prototype		int OsMqttConnect(char* clinetId, char* addr, unsigned short port, unsigned char cleansession, unsigned short keepalive, unsigned char usetls);
Function		MQTT link
Parameter	**Input**	clinetId: id addr: server address port: server ip cleansession: Do you want to clear the session keepalive: Survival time (Unit milliseconds) usetls: Whether to use ssl or not
	Output	nothing
Return		0 means success and others are failure
Notes		
Example		OsMqttConnect(“clientid”, “mqtt.org.com”, “1883”,1, 60000,1);

22.8 OsMqttClose

Prototype		int OsMqttClose();
Function		MQTT close
Parameter	**Input**	nothing
	Output	nothing
Return		0 means success and others are failure
Notes		
Example		OsMqttClose();

22.9 OsMqttSub

Prototype		int OsMqttSub(char* topic, unsigned char qos, void (*message)(void* args));
Function		MQTT subscribe
Parameter	**Input**	topic: theme qos: QoS level and the values are 0, 1, 2 message: Message callback function

	Output	nothing
Return		0 means success and others are failure
Notes		Message parameter refers to the structure as follows typedef struct mqtt_message_t { char* tpoic; int qos; char* message; int message_len; }mqtt_message_t;
Example		void messagefun(void *args) { mqtt_message_t *message=(mqtt_message_t*)args; } OsMqttSub("/topics/test", 1, void (*message)(void* args));

22.10 OsMqttUnSub

Prototype		int OsMqttUnSub(char* topic);
Function		MQTT Unsubscribe
Parameter	**Input**	topic: theme
	Output	nothing
Return		0 means success and others are failure
Notes		The precondition is that the MQTT connection is successful
Example		OsMqttUnSub("/topics/test");

22.11 OsMqttPub

Prototype	int OsMqttPub(char* topic, unsigned char qos, bool retain, char* message, unsigned short messlen);

Function		MQTT publishes a message
Parameter	Input	topic: theme qos: qos level and the values are 0,1,2 message: message content messlen: message length
	Output	nothing
Return		0 means success and others are failure
Notes		
Example		OsMqttPub("/topic/test",1, 0,"123456", 6);

22.12 OsMqttSetStateHandle

Prototype		int OsMqttSetStateHandle(void (*message)(void* args))
Function		MQTT status registration
Parameter	Input	message: function handle
	Output	N/A
Return		0 means success and others are failure
Note		typedef struct mqtt_state_t { int type;//msg type int ret;//1-success 0-fail }mqtt_state_t;
Example		static void state_fun(void *args){ mqtt_state_t *state=(mqtt_state_t *)args; api_log_DEBUG("%d:%d\r\n",state->type,state->ret); } OsMqttSetStateHandle (state_fun);

23 MQTT Module(WIFI Mode)

23.1 Summary

MQTT is an asynchronous communication message protocol based on the TCP/IP protocol stack, which is a lightweight publish / subscribe information transmission protocol. The currently encapsulated functions

include MQTT account and password setting, SSL mode setting, linking, disconnecting, subscribing, unsubscribing, publishing, will message, and status query function. The supported version is 3.1.1.

23.2 error code

#define API_MQTTWIFICFG_ERROR -0x1810
#define API_MQTTWIFICONNECT_ERROR -0x1811
#define API_MQTTWIFISUB_ERROR -0x1812
#define API_MQTTWIFIPUB_ERROR -0x1813
#define API_MQTTWIFICLOSE_ERROR -0x1814

23.3 OsMqttWifiSetUserCfg

Prototype		int OsMqttWifiSetUserCfg(int connect_type, char* clientId, char* usrname, char* pwd);
Function		MQTT config
Parameter	**Input**	connect_type:1-tcp\2-tls(not verify cert)\3-tls(verify server cert)\4-tls(supply client cert)\5-tls(verify server cert and supply client cert)
	Output	N/A
Return		0 means success and others are failure
Notes		
Example		OsMqttWifiSetUserCfg(1, “123”, "123", "123");

23.4 OsMqttWifiConn

Prototype		int OsMqttWifiConn(char* host, int port, unsigned char cleansession, char* willTopic, char* willMsg, char willQos, int keepalive, void (*cb)(void* args));
Function		MQTT connect
Parameter	**Input**	host: port: cleansession:0-csave 1-clean willTopic:will topic willMsg:will msg willQos:support 0\1\2 keepalive:Unit second cb:connect callback function
	Output	N/A
Return		0 means success and others are failure
Notes		

Example	OsMqttWifiConn(“123”, 1883, 0, "offline", “wiil msg”,1,60,mqtt_state_fun); void mqtt_state_fun(void* args) { mqtt_state_t* state = (mqtt_state_t*)args; }

23.5 OsMqttWifiSub

Prototype		int OsMqttWifiSub(char* topic, char qos, void (*message)(void* args));
Function		MQTT subscribe
Parameter	**Input**	topic:subscribe topic qos:support 0\1\2 message:subscribe callback function
	Output	N/A
Return		0 means success and others are failure
Notes		
Example		void issuedCallback(void* args) { mqtt_message_t* msg = (mqtt_message_t*)args; } OsMqttWifiSub(topics, 1,issuedCallback);

23.6 OsMqttWifiPub

Prototype		int OsMqttWifiPub(char* topic, char* msg, char qos, char retain);
Function		MQTT publish
Parameter	**Input**	topic:subscribe topic msg:publish message qos:support 0\1\2 retain:0-not reatin 1-retain
	Output	N/A
Return		0 means success and others are failure
Notes		
Example		OsMqttWifiPub("online", “online”, 1, 0);

23.7 OsMqttWifiClose

Prototype		int OsMqttWifiClose();
Function		MQTT close
Parameter	**Input**	N/A
	Output	N/A
Return		0 means success and others are failure
Notes		When an mqtt disconnection status is received, this interface needs to be called; otherwise, the next connection will fail
Example		OsMqttWifiClose();

24 HTTP Module

24.1 Summary

This module is used for HTTP protocol interaction and currently supports GET, POST requests and file download functions.

24.2 error code

#define API_HTTP_ERROR -0x1901

24.3 OsHttpHandle

Prototype		int OsHttpHandle(char* url, char* method,char *headers, const char* params, const char* body, int bodylen, char** resp_content, int* resp_content_len);
Function		http request
Parameter	**Input**	url:url link method: only support GET POST headers: request headers info params: url parameter body: request parameter data bodylen: request data len
	Output	resp_content: received content resp_content_len: received content length
Return		0 means success and other are failure
Note		
Example		char* url = (char*)"https://www.baidu.com/test / "; char* body = (char*)"{\"data\":\"40100716240001\"}";

	unsigned char* resp_content = NULL; int resp_content_len = 0; char headers[1024] = { 0 }; sprintf(headers, (char*)"Content-Type: application/json\r\nContent-Length: %zu\r\n",strlen(body)); ret = OsHttpHandle(url, (char*)"POST", headers,NULL , body, strlen(body), &resp_content, &resp_content_len); api_log_DEBUG("OsHttpHandle ret=%d\r\n", ret); if (resp_content) { api_log_DEBUG("resp_content(%d):\n%s\n", resp_content_len, resp_content); OsSysFree(resp_content); }

24.4 OsHttpDownloadFile

Prototype		int OsHttpDownloadFile(char* url, char* savefilename, void (*pProgressfun)(int cur, int total));
Function		http file download
Parameter	**Input**	url:url path savefilename: filename (absloute path) pProgressfun:progress callback
	Output	
Return		0 means success and others are failure
Note		
Example		int ret=OsHttpDownloadFile(“https://www.baidu.com”,”/ext/1.tmp”);

24.5 OsHttpSetTimeout

Prototype		void OsHttpSetTimeout(unsigned char val);
Function		set http timeout
Parameter	**Input**	val:timeout(s)
	Output	N/A
Return		N/A
Note		
Example		OsHttpSetTimeout(60);

25 OTA Module

25.1 Summary

This module is used to upgrade applications and firmware.

25.2 error code

#define API_OTAUPDATEAP_ERROR -0x2001
#define API_OTAUPDATEFIRM_ERROR -0x2002
#define API_OTAUPDATESP_ERROR -0x2003
#define API_OTACHECK_ERROR -0x2004

25.3 OsOtaAppUpdate

Prototype		int OsOtaAppUpdate(const char *filename);
Function		app updating function
Parameter	**Input**	filename: app file path (absolute file path)
	Output	N/A
Return		0 means success and others are ailure
Note		It must be a file with the extension. img after application compilation, and the interface call 'success' will automatically restart the application
Example		OsOtaAppUpdate(“/ext/tmp.img”);

25.4 OsOtaFirmwareUpdate

Prototype		int OsOtaFirmwareUpdate(const char* filename);
Function		Firmware upgrading function
Parameter	**Input**	filename: Absolute path of firmware file with upgrade
	Output	N/A
Return		0 means success and others are failure
Note		
Example		OsOtaFirmwareUpdate(“/ext/tmp.bin”);

25.5 OsOtaSpwareUpdate

Prototype		int OsOtaSpwareUpdate(char *file,char *filename);
Function		sp firmware upgrading function
Param	**Input**	filename: Absolute path of firmware file with upgrade filename:sp firmware filename

eter	**Output**	N/A
Return		0 means success and others are failure
Note		
Example		int ret = OsOtaSpwareUpdate("/ext/tmp.tmp", "DS50_core.bin");

25.6 OsOtaFirmwareUpdateNoreboot

Prototype		int OsOtaFirmwareUpdateNoreboot(const char* filename);
Function		ap firmware upgrade without reboot
Parameter	**Input**	filename: Absolute path of firmware file with upgrade
	Output	N/A
Return		0 means success and others are failure
Note		If the interface call is successful, the firmware will be updated automatically when restarting manually.
Example		int ret = OsOtaFirmwareUpdateNoreboot("/ext/tmp.tmp");

25.7 OsOtaAppCheck

Prototype		int OsOtaAppCheck(char* data, unsigned int datalen);
Function		app img data check
Parameter	**Input**	data:img data datalen:img data len
	Output	N/A
Return		0 means success and others are failure
Note		This function is often used to verify whether the data is legal when upgrading applications via ota
Example		

26 Power Module

26.1 Summary

This module is used to obtain information about the power module, including charging status, battery voltage and percentage of battery charge.

26.2 error code

#define API_BATTERYSTATE_ERROR -0x2101
#define API_BATTERYVOLTAGE_ERROR -0x2102
#define API_BATTERYPERCENT_ERROR -0x2103

26.3 OsBattGetChargeState

Prototype		int OsBattGetChargeState(void);
Function		Obtain charging status
Parameter	**Input**	N/A
	Output	N/A
Return		0- No charging or charging turned off 1- Charging in progress 2- Fully charged
Note		
Example		int ret=OsBattGetChargeState();

26.4 OsBattGetVoltage

Prototype		int OsBattGetVoltage(void);
Function		Obtain battery voltage
Parameter	**Input**	N/A
	Output	N/A
Return		Current battery voltage (in millivolts)
Note		Voltage value range (3333-4130mv)
Example		int ret=OsBattGetVoltage();

26.5 OsBattGetPercent

Prototype		u32 OsBattGetPercent (void);
Function		Obtain the percentage of battery consumption
Parameter	**Input**	N/A
	Output	N/A
Return		Current percentage of the battery
Note		
Example		u32 ret=OsBattGetPercent();

27 Font Module

27.1 Summary

In Chapter 22, LCD displays text using a dot matrix font library. Some customers want to display vector font libraries, so we have packaged some API interfaces that support displaying vector font libraries, using the open-source library lvgl v7 version.

27.2 error code

28 DUKPT-AES

28.1 Summary

This section introduces the interfaces to manupilate DUKPT-AES key.
Since DUKPT-AES keys are stored in a sperate area, so you can have 5 DUKPT keys introduced in section 23, as well as 5 DUKPT-AES keys introduced here.
When key usage is _2TDEA and KSN length is 10, the key injected is a DUKPT ISO 9797 through and through, but it is more efficient and cost-effective.
If you are a newcomer to this terminal, and you are still using DUKPT ISO 9797, this new method is recommended. Section 23 is retained for legacy reasons only.

28.2 error code

28.3 OsSecUpdateDukptAes

Prototype		s32 OsSecUpdateDukptAes (u8 group, u8 type, u8 idx, u8 usage, u8 *ipek, u8 ksn_len, u8 *ksn, u8 kcv_len, u8 kcv_mode, u8* kcv);
Function		Update DUKPT-AES Key
Parameter	**Input**	Group: Group number (currently supports 0 only) Type: Type (0 - IPEK; 1 – BDK, not recommended) Idx: key index (starting from 0, up to 4) usage: Key Usage，can be 0 (_2TDEA) or 2 (_AES128) ipek: IPEK ksn_len: KSN length, supports 10 or 12。 ksn： KSN kcv_len: kcv length, 0 means no KCV is provided kcv_mode: 0 is supported for now, when usage=_2TDEA, encrypts 8 0x00's with IPEK using 2TDEA; when usage=_AES128, encrypts 16 0x00's with IPEK using AES128 kcv: KCV
	Output	
Return		0 – success; other – failure

Note	
Example	u8 ipek[16] = { 0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF,0xFE,0xDC,0xBA,0x98,0x76,0x54,0x32,0x10 }; u8 ksn[12] = { 0xFF,0xFF,0x98,0x76,0x54,0x32,0x10,0xe0,0x00,0x00 ,0x00,0x00 }; u8 pinblock[8] = { 0xFB,0x48,0xBB,0x73,0xD7,0x75,0xD8,0xF3 }; u8 pin[4] = { 0x31,0x32,0x33,0x34 }; u8 cardno[16] = { 0x34,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x39 }; u8 outdata[256] = { 0 }, ksndata[16] = { 0 }; u8 flashdata[1024] = { 0 }; u32 outlen = 0, ksnlen = 0; unsigned char kcv[4] = { 0x08, 0xD7, 0xB4, 0xFB }; // update DUKPT-AES key 0, initial key type is IPEK, keyusage=AES128, no KCV is required, ksn length is 12 OsSecUpdateDukptAes(0,0,0,2,ipek, 12, ksn,0, 0, NULL); // update DUKPT-AES key 1, initial key type is IPEK, keyusage=2TDEA, no KCV is required, ksn length is 10 OsSecUpdateDukptAes(0,0,1,0,ipek, 10, ksn,0, 0, NULL);

28.4 OsSecCalcDukptAesMac

Prototype		s32 OsSecCalcDukptAesMac(s32 macMode,u8 alg,u32 keyIndex,u8* pData,u32 dataLen,u8*pMac,u32 *pMacLen,u8* pKsn,u32* pKsnLen);
Function		MAC Calculation By DUKPT-AES
Parameter	**Input**	macMode: Xor ANSI9.9 PBOC ANSI9.19 Alg: 0 keyIndex: 0 - 4 pData: Data dataLen: Data length
	Output	pMac: Calculated MAC data pMacLen: Data length of calculated MAC pKSn: KSN data pKsnLen: KSN data length

Return	0 – success; other – failure
Note	
Example	ret = OsSecCalcDukptAesMac(0, 0, keyid, data, sizeof(data), outdata, &outlen, ksndata, &ksnlen);

28.5 OsSecCalcDukptAesData

Prototype		s32 OsSecCalcDukptAesData(s32 mode,u8 alg,u32 keyIndex,u8* pdataIn,u32 dataInLen,u8* pDataOut,u32* pOutLen,u8* pKsn,u32* pKsnLen);
Function		Data Calculation By DUKPT-AES
Parameter	**Input**	Mode: 0x00: ECB Decryption 0x01: ECB Encryption 0x10: CBC decryption 0x11: CBC encryption Alg: 0 keyIndex:0 - 9 pData: Data dataLen: Data length
	Output	pMac: Calculated data pMacLen: Length of calculated data pKSn: KSN data pKsnLen: KSN data length
Return		0 – success; other – failure
Note		
Example		u8 outdata[256]={0},indata[24] = {0X62,0X22,0X83,0X01,0X53,0X27,0X80,0X55,0XD2,0X70,0X32,0X06,0X04,0X90,0X82,0X44,0X00,0X0F,0X06,0X06,0X06,0X06,0X06,0X06}; u32 outlen = 0, ksnlen = 0; u8 ksn[12]; OsSecCalcDukptAesData(0x11, 0, 0, indata, sizeof(indata), outdata, &outlen, ksn, &ksnlen);

28.6 Pinblock Calculation By DUKPT-AES

Pinblock Calculation By DUKPT-AES can be implement with legacy interfaces OsSecSetPinblockParam and OsSecGetPinblockStatus, with key management system set to DUKPT-AES (2).
Here is an example:

```
s32 ret;
u32 i;
u8  pinbuf[16];
```

```
    u32 pinlen;
    u32 tickS, tickE;
    u8 cardno[16]={0x34,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x39, 0};
    u8 amount[]="1234567.89";
    u8 outdata[256]={0},ksndata[16] = {0};
    u32 outlen = 0,ksnlen = 0;

                api_lcd_clear_screen();
                sprintf(disp, "Amount: %s", amount);
                api_lcd_fill_row_ram(1, 0, disp, FDISP|CDISP);
                api_lcd_fill_row_ram(2, 0, "Password please:", FDISP|LDISP);
                api_lcd_brush_screen();
                pinbuf[0] = 0;
                pinlen = 0;
                tickS = api_utils_stimer_get();
                ret = OsSecSetPinblockParam((2 << 24), 0, keyid, 4, 6, 60, strlen(cardno), cardno, 0,
strlen(amount), amount);
                tickE = api_utils_stimer_get();
                if(ret == 0)
                {
                    while(1)
                    {
                        ret = OsSecGetPinblockStatus(outdata, &outlen);
                        if(ret < 0)
                        {
                            sprintf(disp, "Fail! getPin=%d", ret);
                            api_lcd_fill_row_ram(5, 0, disp, NOFDISP|CDISP);
                            break;
                        }

                        if(outlen == 1)
                        {
                            if((outdata[0] == CANCEL) || (outdata[0] == ENTER))
                            {
                                break;
                            }
                            switch(outdata[0])
                            {
                            case '*':
                                pinbuf[pinlen] = '*';
                                pinlen++;
                                pinbuf[pinlen] = 0;
                                api_lcd_fill_row_ram(3, 0, pinbuf, FDISP|CDISP);
                                api_lcd_brush_screen();
                                break;
                            case BACKSPACE:
                                if(pinlen > 0)
                                {
                                    pinlen--;
                                    pinbuf[pinlen] = 0;
```

```
                                api_lcd_fill_row_ram(3, 0, pinbuf, FDISP|CDISP);
                                api_lcd_brush_screen();
                            }
                            break;
                        }
                    }
                    else if(outlen == 16)
                    {
                        sprintf(disp, "PINBLOCK:");
                        api_lcd_fill_row_ram(5, 0, disp, FDISP|CDISP);
                        disp[0] = 0;
                        for(i=0;i<outlen;i++)
                        {
                            sprintf(disp+strlen(disp), "%02X",outdata[i]);
                        }
                        api_lcd_fill_multi_row_ram(6, 0, disp, FDISP|LDISP);
                        break;
                    }
                    OsSysMsleep(20);
                }
            }

            if(ret!=0)
            {
                sprintf(disp, "Fail! calcMac=%d", ret);
                api_lcd_fill_row_ram(1, 0, disp, NOFDISP|CDISP);
            }
            api_lcd_brush_screen();
            getch(20000);
```

getch is implemented as following:

```
#define FIBO_TaskSleepGranularity 40
u8 getch(u32 wait_ms)
{
    u32 i;
    u8 keyin;
    u32 loopcnt;

    loopcnt = (wait_ms+FIBO_TaskSleepGranularity-1) / FIBO_TaskSleepGranularity;

    for(i=0;i<loopcnt;i++)
    {
        keyin = GetKeyValue(TRUE);
        if(keyin != 0) break;
        OsSysMsleep(1);
    }
    return keyin;
}
```

29 SYS PARAMS

29.1 Summary

This module is used to read or set relevant parameters of the system

29.2 error code

#define API_PARAMSSETNETMODE_ERROR -0x2401
#define API_PARAMSGETNETMODE_ERROR -0x2402
#define API_PARAMSGETSECMODE_ERROR -0x2403

29.3 OsParamsSetNetMode

Prototype		int OsParamsSetNetMode(int value);
Function		set network mode
Parameter	**Input**	value:0-wifi 1-4g
	Output	N/A
Return		
Note		
Example		OsParamsSetNetMode(1);

29.4 OsParamsGetNetMode

Prototype		int OsParamsGetNetMode();
Function		get network mode
Parameter	**Input**	N/A
	Output	N/A
Return		0-wifi 1-4g
Note		
Example		OsParamsGetNetMode();

29.5 OsParamsGetSecMode

Prototype		int OsParamsGetSecMode();
Function		Determine whether the current device has been triggered.
Parameter	**Input**	N/A
	Output	N/A

Return	return 0-no trigger otherwise 1-trigger
Note	
Example	OsParamsGetSecMode();

30 LBS

30.1 Summary

This module provides relevant positioning interfaces.

30.2 error code

#define API_LBSQUERY_ERROR -0x2501

30.3 OsLbsQuery

Prototype		int OsLbsQuery(char* key, int keyType, double* lat, double* lng);
Function		Latitude and longitude query
Parameter	**Input**	key:lbs key keyType:5-google 6-gaode
	Output	lat:latitude lng:longitude
Return		0-success other:fail
Note		4g network registration is required to be successful.
Example		

31 EMV

31.1 Summary

This module provides an interface for the EMV kernel.

31.2 Emv return code

```
typedef enum
{
    APP_RC_START = -1,
    APP_RC_COMPLETED = 0,          // Transaction success
    APP_RC_TERMINAL,               //Transaction Termination
    APP_RC_CANCEL,             // Transaction cancel
    APP_RC_EMV_DENAIL,             // Transaction denail (fail)
    APP_RC_EMV_GAC2_DENAIL,        // Transaction gac2 denail (fail)
    APP_RC_NFC_NOT_ALLOW,          //NFC transactions are not allowed
    APP_RC_EMV_APP_BLOCK,          //Card application locked
```

```
    APP_RC_EMV_APP_SEE_PHONE,      //Please check your phone and authorize
    APP_RC_EMV_TRANS_TRY_ANOTHER_INTERFACE,  //Please use contact to initiate the transaction
again
    APP_RC_EMV_TRANS_GPO_NOT_SUPPORT,    // gpo not support
    APP_RC_FALL_BACK,                    // Chip card reading failed, please swipe the card
    APP_RC_EMV_CARD_BLOCK,          // card block
    APP_RC_CARD_NOT_SUPPORT,       //Unsupported card

    APP_RC_NFC_RETAP_TIMEOUT,     //Card re pasting timeout
    APP_RC_NFC_RETAP_CANCEL,      //When prompted to reattach the card, timeout occurred
    APP_RC_NFC_TERMINAL,       //NFC transaction termination
    APP_RC_NFC_DOUBLETAP_DENAIL, //Second tap card denail
    APP_RC_NFC_MULTI_CARD,     //Multiple cards detected
    APP_RC_NFC_TRY_AGAIN,      //NFC card tap, please try again
    APP_RC_TRANS_REVERSEL,     //The server approved, but the second GAC of the card failed and a
reorganization needs to be initiated

    APP_RC_NUMS,
}EMV_L2_Return;
```

31.3 Emv_KernelInit

Prototype		int Emv_KernelInit(EmvCallBack_t t_callbackfun);
Function		Register EMV callback function
Parameter	**Input**	Callbackfun: Emv callback function pointer
	Output	NULL
Return		0-success other:fail
Note		
Example		

31.4 Emv_Process

Prototype		EMV_L2_Return Emv_Process(EmvTransParams_t emvTransParams);
Function		Initiate EMV trading
Parameter	**Input**	emvTransParams: Transaction Information Structure
	Output	NULL
Return		APP_RC_COMPLETED-success other:fail

Note	Call after finding IC card
Example	

31.5 Emv_GetCoreData

Prototype		unsigned char* Emv_GetCoreData(unsigned int tagname, int *pvallen);
Function		Obtain the tag value
Parameter	**Input**	tagname: The TAG value to be read, see EMV TAG macro definition
	Output	Pvallen: The length of the returned data
Return		NOT NULL: Pointer to TAG value NULL: No TAG value obtained
Note		
Example		

31.6 Emv_SetCoreData

Prototype		int Emv_SetCoreData(unsigned int tagname, unsigned char *pvalue, int valuelen)
Function		Set tag value
Parameter	**Input**	tagname: The TAG value to be read, see EMV TAG macro definition
	Input	pvalue: The address of the set tag value
	Input	Valuelen: The length of tag value
Return		0: Success Ohter:Fail
Note		This function only takes effect in the following callback functions. *//affter select app callback* *void* (*EMV_AfterSelectApp)(); *//affter read record callback* *void* (*EMV_AfterReadRecord)();
Example		

31.7 Emv_FetchData

Prototype		int Emv_FetchData(unsigned int* tagname, int count, unsigned char* obuf, int olen)
Function		Retrieve a series of TLV data from tagname [], and return the data format as TLV
Param	**Input**	tagname: The number of TLV data to be obtained
	Input	Count: tag counts

eter	**Input**	Olen: The size of obuf
	Output	obuf: Output the obtained TLV data pointer
Return		0 No data -1 Fail > 0 Data length
Note		
Example		u8 buf[512]; u32 totalLength; u32 tags*[]* = {EMVTAG_AC, EMVTAG_CID, EMVTAG_IAD, EMVTAG_RND_NUM, EMVTAG_ATC, EMVTAG_TVR, EMVTAG_TXN_DATE, EMVTAG_TXN_TYPE, EMVTAG_AMOUNT, EMVTAG_CURRENCY, EMVTAG_AIP, EMVTAG_COUNTRY_CODE, EMVTAG_OTHER_AMOUNT, EMVTAG_TERM_CAP, EMVTAG_CVM, EMVTAG_TERM_TYPE, EMVTAG_IFD, EMVTAG_DF, EMVTAG_APP_VER, EMVTAG_TXN_SN, EMVTAG_CARD_ID, EMVTAG_ARC, EMVTAG_EC_AUTH_CODE}; totalLength = Emv_FetchData(tags, ARRAY_SIZE(tags), buf, ARRAY_SIZE(buf)); nBcd2Asc(buf,totalLength*2,*pEmvData*,0); *//memcpy(pEmvData,buf,totalLength);* **pEmvDataLen* = totalLength*2; return 0;

31.8 Emv_ClrCAPKFile

Prototype		int Emv_ClrCAPKFile()
Function		Clear emv capk file
Param eter	**Input**	
	Output	
Return		0: Success Other: Fail
Note		
Example		

31.9 Emv_ClrAIDFile

Prototype		int Emv_ClrAIDFile ()
Function		Clear emv aid pram file.
	Input	

Parameter	**Output**	
Return		0: Success Other: Fail
Note		
Example		

31.10 Emv_PARAM_InputCAPKData

Prototype		int Emv_PARAM_InputCAPKData(unsigned char *ptlvstrin, int tlvlenin)
Function		Set Emv capk
Parameter	**Input**	Ptlvstrin: Capk data
	Input	Tlvlenin: Capk data length
Return		0: Success Other: Fail
Note		The data format is BCD format, with a maximum length of 384
Example		

31.11 Emv_PARAM_InputAIDData

Prototype		int Emv_PARAM_InputAIDData(unsigned char *ptlvstrin, int tlvlenin)
Function		Set Emv aid param
Parameter	**Input**	Ptlvstrin: Aid param data
	Input	Tlvlenin: Aid param data length
Return		0: Success Other: Fail
Note		The data format is BCD format, with a maximum length of 384
Example		

31.12 Emv_GetAidTotalNum

Prototype		int Emv_GetCapkTotalNum();
Function		Obtain the number of capk
Parameter	**Input**	
	Input	

Return	>= 0: Capk counts < 0: Fail
Note	
Example	

31.13 Emv_GetCapkTotalNum

Prototype		int Emv_GetCapkTotalNum();
Function		Obtain the number of aid param
Parameter	**Input**	
	Input	
Return		>= 0: Aid counts < 0: Fail
Note		
Example		

31.14 EMV_L2_GetLastError

Prototype		EMV_L2_Error EMV_L2_GetLastError();
Function		Obtain the emv kernel error code
Parameter	**Input**	
	Input	
Return		EMV_L2_Error
Note		
Example		

31.15 Emv_GetKernelVersion

Prototype		int Emv_GetKernelVersion(EmvKernelType kernelType,unsigned char *ver)
Function		Obtain EMV kernel version
Parameter	**Input**	kernelType: KERNEL to be queried
	Output	Return kernel version number address
Return		Version data length
Note		
Example		