Prolin API Programming Guide

V 2.0.1

PAX Computer Technology(Shenzhen)Co., Ltd.

Revision History

Date	Version	Note	Author
2012-08-29	V1.0.0	Translated from Chinese standard version of 'Prolin API Programming Guide v1.0.0'.	Prolin Team
2012-11-19	V1.0.1	Modified the interfaces.	Prolin Team
2012-12-26	V1.0.2	1. Add the return code list in the System function; 2. Add a new function OsVerifySignExternal (); 3. Add the WIFI module; 4. Add the Register table in Appendix 3.	Prolin Team
2013-02-20	V1.0.3	1. Add descriptions in OsOpenFont ().	Prolin Team
2013-03-06	V1.0.4	1. Modify the description of OsModemOpen(); 2. Add two return values -3219 and -3220 in Modem; 3. Add the instruction of DetectDailTone; 4. Add a new function OsModemSwitchPower (); 5. Modify the parameter description of OsPrnOpen (), it does not support the png format; 6. Modify the sci_get_fd ().	Prolin Team
2013-04-17	V1.0.5	1. Modify the knots of the 14.4.1 open(); 2. Modify the "pow-hwclock -w" to "pow-hwclock -w -u" in the code of Timeset; 3. Update the instruction of the OsWlInit(); 4. Update the instruction of the OsCheckBattery ().	Prolin Team
2013-05-17	V1.0.6	1. Modify the description of 'Timer'. 2. Add description of 'Delay'. 3. Add description for registry table. 4. Modify the parameter description of Value in OsRegGetValue (). 5. Modify the parameter description of ShaOut in OsSHA (). 6. Update the description of GUI.	Prolin Team

		7. Add ‘1200, V22’ and ‘2400, FC’ to the parts of synchronous variable for MODEM.	
2013-08-09	V1.0.7	1. Modify the Register table. 2. Modify the data structure of Font. 3. Update the OsCheckBattery () . 4. Update the chapter 15 ICC Reader and chapter 16 RF Reader.	Prolin Team
2013-10-22	V1.0.8	1. Add 4.11 Save the crash report for system function. 2. Modify the brightness level to [0~10] in the chapter of LCD. 3. Update the key value definitions in the chapter of Keyboard.	Prolin Team
2013-10-31	V1.0.9	1. Update the System Function. 2. Modify the OsCodeConvert(). 3. Add new interfaces OsPortReset() and OsPortCheckTx(). 4. Update the Audio chapter, add a new interface OsPlayWave().	Prolin Team
2013-11-20	V2.0.0	1. Add the description about 0x02 in parameter KeyVarType. 2. Modify the value ranges in PED. 3. Modify the description of parameter ucATQ0 in OsPiccAntiSel (). 4. Delete OsPiccGetParam () and OsPiccSetParam (). 5. Update the instruction of OsRegGetValue (). 6. Add notes of OsPortOpen (). 7. Update instructions of OsSysSleep (). 8. Update the Return code list in chapter Network Configuration. 9. Add instruction of OsNetDns ().	Prolin Team
2014-1-13	V2.0.1	1. Modify the instruction of OsWlSelSim(). 2. Add a return value for OsMsrOpen (). 3. Add new interfaces OsWlLoginEx (), OsMount (), OsUmount (). 4. Add chapter 26 Barcode. 5. Add AES functions. 6. Update the Appendix 4. 7. Modify the parameter Name in the OsInstallFile(). 8. Add an error code “ERR_APP_MODE” for the OsInstallFile().	Prolin Team

{ This page intentionally left blank }

Contents

Table List

1 Introduction

1.1 Purpose

This document was previously named as Prolin 2.X API Programming Guide.

Prolin SDK supports necessary tools and resources to create Prolin applications, different from the system daemons. The Prolin applications, which are based on Prolin OS devices, work in Prolin internal and run as an independent executive program.

Whereas, Prolin application can access all system features of Prolin OS, can save data in the local file system and can even communicate with various installed programs through custom URL. It uses the default or customized GUI framework to develop the local GUI application program for Prolin OS.

1.2 Readers

This document is primarily targeted for Prolin OS developers, who are expected to create Prolin applications. The purpose is to introduce the framework of Prolin application program; demonstrate some of the key customization points in GUI and other significant frameworks; and provide programming aid to the API interfaces, which support driver control to the Prolin OS hardware platform. It will also provide guidance to design.

The interface which provide by Prolin 2.X is based on calling Linux system or POSIX interface.

Considering the compatibility requirements of the PaxME OS and applications, it will encapsulate a set of OSAL interface which begins with the prefix “Os”. The access of other

devices and system functions will provide the demo code to guide developers how to use the POSIX or system library to program.

For details, refer to the following document.

In this document, the interface that begins with the prefix “Os” has been defined in osal.h of SDK, unless otherwise specified.

1.3 Conditions

The prerequisites for this document are:

- Basic understanding of Linux
- Basic information about processes, threads and Linux system functions and their roles in application development
- Familiar with memory management and process management
- Familiar with Linux input subsystem and frame buffer.

For the Linux user, you should use Linux kernel version greater than 2.6.22 to run the Prolin SDK. For the Windows user, you should use Windows XP or higher to run the Prolin SDK. You can get the Prolin SDK from PAX support team.

1.4 Related Documents

- Prolin Terminal Manager Operating Guide
- Prolin Operating System Installation Guide
- Prolin SDK Tutorial
- Prolin GUI Programming Guide
- Prolin App Development Guide
- Prolin Operating System Introduction
- Prolin PCKit Operating Guide

1.5 Abbreviation

Table 1 Abbreviation

Abbreviation	Description
API	Application Programming Interface
CDPD	Cellular Digital Packet Data

CHAP	Challenge Handshake Authentication Protocol
DHCP	Dynamic Host Configuration Protocol
DUKPT	Derived unique key per transaction
EMV	Europay, MasterCard,Visa EMV is a global standard for credit and debit payment cards based on chip card technology
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
IC	Integrated circuit
IC Card	Integrated Circuit Card
IPCP	IP Control Protocol
KSN	Key Serial Number
KCV	Validation of the key plaintext.
LCP	Link Control Protocol
LRC	Longitude Redundancy Check
MODEM	Modulator-demodulator.
MSR	Magnetic Stripe Reader
NCP	Network Control Protocol
PAP	Password Authentication Protocol
PCD	Proximity Coupling Device
PED	PIN Entry Device
PICC	Proximity Integrated Circuit Card
POS	Point of Sale
PPP	Point-to-Point Protocol
PUK	Public Key
SAM	Security Authentication Module
SGSN	Serving GPRS Support Node
SIM	Subscriber Identity Module
SMS	Short Message Service
SSL	Secure Sockets Layer
TWK	Transaction working key

1.6 Document Conventions

【Used for labeling common notes.】

【Used for reminding the audience some place may have to pay attention to, which may lead to exceptions or errors.】

【Used for warning the audience some place must be very careful, which may lead to very serious errors or damages.】

2 Return Code and Parameter

2.1 Return code classification

List the types and values of the return code which appeared in this document.

Table 1 Return code classification List

Type	Value(Decimal)	Description
General return code	-1000~-1999	As general return code for API.
System function	-2200~-2299	
Power Management	-2300~-2399	
Encryption and Decryption	-2400~-2499	
Font library	-2500~-2599	
LED display	-2600~-2699	
MSR	-2700~-2799	
ICC Reader	-2800~-2899	
RF Reader	-2900~-2999	
Communication port	-3000~-3099	
MODEM	-3100~-3299	
IP Network Configuration	-3300~-3399	
PED	-3800~-3899	

2.2 General return code

Table 2 General return code list

Macro	Value	Description
RET_OK	0	Success
ERR_INVALID_HANDLE	-1000	Invalid handle
ERR_TIME_OUT	-1001	Timeout
ERR_APP_NOT_EXIST	-1002	Application not exists
ERR_INVALID_PARAM	-1003	Invalid Parameters
ERR_DEV_NOT_EXIST	-1004	Device not exists.
ERR_DEV_BUSY	-1005	Device is busy.
ERR_DEV_NOT_OPEN	-1006	Device is not open.
ERR_ACCESS_DENY	-1007	Access denied.
ERR_FONT_NOT_EXIST	-1008	Font not exists.
ERR_FILE_FORMAT	-1009	File format error.
ERR_USER_CANCEL	-1010	Users cancel.
ERR_NO_PORT	-1011	No communication port.
ERR_NOT_CONNECT	-1012	Not Connected
ERR_MEM_FAULT	-1013	Memory fault.
ERR_SYS_BAD	-1014	System configuration error.
ERR_SYS_NOT_SUPPORT	-1015	System does not support this function.
ERR_STR_LEN	-1016	Character string is too long.
ERR_TOO_MANY_HANDLE	-1017	Too many handle.
ERR_APP_MODE	-1018	Mode error.

ERR_FILE_NOT_EXIST	-1019	File does not exist.
ERR_TS_DISABLE	-1020	Touch screen is not open.

2.3 Parameter

API parameters are divided into input parameters and output parameters.

String input and output parameters are ending with "\x00". String parameters must indicate the length limit.

{ This page intentionally left blank }

3Thread

PROLIN 2.X supports multithreaded development. The platform provides standard POSIX threading library (pthread) for developers to use.

The pthread header file is located under the installation directory of Prolin SDK.

toolchains\arm-linux\arm-softfloat-linux-gnu\include\pthread.h

{ This page intentionally left blank }

4System Function

4.1 Return code list

Table 3 System function return code list

Macro	Value	Description
ERR_FILE_NOT_FOUND	-2201	File has not found.
ERR_VERIFY_SIGN_FAIL	-2204	Verify signature failed.
ERR_NO_SPACE	-2205	No enough system space.
ERR_NEED_ADMIN	-2207	Permission denied. (Need higher permissions.)

Table 4 Macro definitions of file types

Macro	Value	Description
FILE_TYPE_APP	1	Application package
FILE_TYPE_APP_PARAM	2	Application data file
FILE_TYPE_SYS_LIB	3	System dynamic library file
FILE_TYPE_PUB_KEY	4	User public key file

FILE_TYPE_AUP	5	Application upgrade package

4.2 Data Definition

LOG_T (Enumerate LOG types)：

LOG_T：

```
typedef enum{
    LOG_DEBUG,          //Display debugging information
    LOG_INFO,           // Display prompt information
    LOG_WARN,           // Display warning information
    LOG_ERROR,          // Display error information
} LOG_T;
```

ST_TIME (structure of time)

ST_TIME：

```
typedef struct{
    int Year;           //year 1970 – 2037
    int Month;          //month 1 –12
    int Day;            //day 1 –31
    int Hour;           //hour 0 – 23
    int Minute;         //minute 0 – 59
    int Second;         //second 0 – 59
    int DayOfWeek;      //Monday – Sunday（Only effective for reading time）
} ST_TIME;
```

ST_TIMER (structure of timer)

ST_TIMER：

```
typedef struct{
    unsigned long Start;
    unsigned long Stop;
    unsigned long TimeLeft;
} ST_TIMER;
```

ST_APP_INFO (structure of application information)

ST_APP_INFO：

```
typedef struct{
    char Id[32];
    char Name[64];
    char Bin[64];
    char Artwork[64];
    char Desc[64];
    char Vender[32];
    char Version[32] ;
} ST_APP_INFO;
```

4.3 Timeset

4.3.1 OsSetTime

Prototype	**int OsSetTime(const ST_TIME *Time);**	
Function	Set the system time, the week value will not work.	
Parameters	Time【Input】	Time structure
Return	RET_OK	Success
	ERR_NEED_ADMIN	Need higher permissions.
	ERR_INVALID_PARAM	Invalid parameter
Instruction	Only the main application has permission to set the time, otherwise, it returns ERR_NEED_ADMIN.	

4.3.2 OsGetTime

Prototype	**void OsGetTime(ST_TIME *Time);**	
Function	Get the system time.	
Parameters	Time【Output】	Time structure
Return	None	
Instructio		

n	

4.4 Timer

4.4.1 OsTimerSet

Prototype	**int OsTimerSet(ST_TIMER *Timer, unsigned long Ms);**	
Function	Set the timer.	
Parameters	Timer【Output】	Timer
	Ms【Input】	Time【unit:ms】
Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid parameter
Instruction		

4.4.2 OsTimerCheck

Prototype	**unsigned long OsTimerCheck(ST_TIMER *Timer);**	
Function	Check the remaining time.	
Parameters	Timer【Input】	Timer
Return	>=0	The remaining time【unit:ms】
	ERR_INVALID_PARAM	Invalid parameter
Instruction		

4.5 Delay

4.5.1 OsSleep

Prototype	**void OsSleep(unsigned int Ms);**	
Function	System delay, the process will not be interrupted by signal.	
Parameters	Ms 【Input】	The delay time【unit:ms】
Return	None	

Instruction	

4.6 Thread dormancy

The dormancy is realized by thread management. Kindly refer to the following code.

Example:

```
#include <pthread.h>

static pthread_t ntid;

static void *thread_fn(void *arg)

{
   printf("This is child thread\n");
   return ((void *)0);
}


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

   if(pthread_create(&ntid,NULL,thread_fn,NULL) != 0)
        printf("can't create thread\n");

   sleep(5);
   return 0;
}
```

4.7 Log

4.7.1 OsLogSetTag

Prototype	**void OsLogSetTag(const char *Tag);**	
Function	Set the LOG tag.	
Paramete	Tag【Input】	LOG information tag

rs		
Return	None	
Instruction	Set the LOG tag, and the default tag is null. Suggest to set it as an application name. It supports up to 32 bytes, when more than it, just using the first 32 bytes. When the Tag is an empty string or NULL, the OsLog() will not work..	

4.7.2 OsLog

Prototype	**int OsLog(LOG_T Prio, const char *fmt, …);**	
Function	Record the LOG information.	
Parameters	Prio【Input】	LOG type
	fmt 【Input】	Format of log information
Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid parameter
Instruction		

4.8 Get the count value

4.8.1 OsGetTickCount

Prototype	**unsigned long OsGetTickCount(void);**	
Function	Get the system count value.	
Parameters	None	
Return	>0	Count value【unit:ms】
Instruction	The value represents the time from the boot to the present time. 【unit:ms】	

4.9 Get Appliaction information

4.9.1 OsGetAppInfo

Prototype	**int OsGetAppInfo(ST_APP_INFO AppInfo[], int InfoCnt);**
Function	Get the application information.

Parameters	AppInfo【Output】	Array of AppInfo structure
	InfoCnt【Input】	The number of Apps that can be stored in the array
Return	>=0	Returns the number of obtained App information
	ERR_NEED_ADMIN	Need higher permissions.
	ERR_INVALID_PARAM	Invalid parameter
Instruction	Only the main application has permission to get the information. When the number of existing applications are more than InfoCnt, the InfoCnt shall prevail.	

4.10 Buzzer

4.10.1 OsBeep

Prototype	**void OsBeep(int ToneType, int DurationMs);**	
Function	The buzzer.	
Parameters	ToneType 【Input】	Tone type. The value ranges from 1 to 7.
	DurationMs 【Input】	Duration【unit:ms】The value ranges from 10 to 10000.
Return	None	
Instruction	If ToneType<1, set it as 1, if ToneType>7, set it as 7. If DurationMs<10, set it as 10, if DurationMs>10000, set it as 10000.	

4.11 Run Application

4.11.1 OsRunApp

Prototype	**int OsRunApp(char *AppId, char **Argv, void *Data, RUNAPP_CB CbOut, RUNAPP_CB CbErr);**	
Function	Switch to a specified sub-application.	
Parameters	AppId【Input】	sub-application ID
	Argv【Input】	Argument list, it can be NULL
	Data【Input】	Custom data, it will be passed to CbOut() and CbErrk() to

		call back.
	CbOut【Input】	Callback function of the standard output information
	CbErr【Input】	Callback function of the standard error information
Return	RET_OK	Success
	ERR_ACCESS_DENY	Access denied
	ERR_APP_MODE	Mode Error
	ERR_INVALID_PARAM	Invalid parameter
	ERR_NEED_ADMIN	Need system main application permissions
Instruction	Only the main application has permission to switch application, Otherwise, it returns ERR_NEED_ADMIN. Switch to a specified sub-application. This will output the standard output information and standard error information of the sub-application to CbOut() and CbErr(), respectively. For the multi-line standard output and standard error, the callback function will be called multiple times. The callback function is defined as: **typedef void (*RUNAPP_CB)(char *appid, char *str, void *data)**.	

4.12 Set and Read the registry table

OsGetTerminalInfo () and OsReadSn () have been obsolete, and the related functions can be realized by calling OsRegGetValue ().

4.12.1 OsRegSetValue

Prototype	**int OsRegSetValue(const char *Key, const char *Value);**	
Function	Set system parameters.	
Parameters	Key 【Input】	System configuration name, it needs ending with'\0'.
	Value 【Input】	The parameter value cannot be null and should be less than 64 bytes. It needs ending with '\0'.
Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid Parameters

	ERR_NEED_ADMIN	Need system main application permissions
Instruction	The system configuration name can only be set begin with “persist.sys.” For example,’ persist.sys.app0.pic’.	

4.12.2 OsRegGetValue

Prototype	**int OsRegGetValue(const char *Key, char *Value);**	
Function	Read system parameters.	
Parameters	Key 【Input】	System configuration name, it needs ending with'\0'.
	Value 【Output】	The parameter value cannot be null and it must be more than 64 bytes.
Return	>=0	Read the string length
	ERR_INVALID_PARAM	Invalid Parameters
Instruction	The system configuration name can only be set begin with “ro.fac.” or “persist.sys.” About the "ro.fac.", users can refer to register table. If the query parameter does not exist, or the parameter value is empty, and then returns 0, the output parameter Value is “ ”.	

4.13 Install and uninstall files

4.13.1 OsInstallFile

Prototype	**int OsInstallFile(const char *Name, const char *FileName, int FileType);**	
Function	Install application and system files.	
Parameters	Name【Input】	When FileType is FILE_TYPE_PUB_KEY, Name gets the value from "uspuk0" to "uspuk8", and it represents the user public key ranged from 0 to 8. When FileType is FILE_TYPE_APP_PARAM and Name is the Application ID of parameter file. When FileType is the other type, Name is invalid, and it can be NULL.
	FileName 【Input】	The filename which needs to be installed.
	FileType	• FILE_TYPE_APP (the application package) • FILE_TYPE_APP_PARAM (the application data file) • FILE_TYPE_SYS_LIB (the dynamic library file) • FILE_TYPE_PUB_KEY (the user public key file)

		• FILE_TYPE_AUP (the application update package)
Return	RET_OK	Success
	ERR_PUK_NOT_EXIST	The specified user public key does not exist.
	ERR_FILE_NOT_FOUND	FileName does not exist.
	ERR_FILE_FORMAT	FileName format error.
	ERR_INVALID_PARAM	Invalid Parameters
	ERR_VERIFY_SIGN_FAIL	Signature verification failed.
	ERR_APP_MODE	Mode error
Instruction	Name will be valid only when the FileType is FILE_TYPE_PUB_KEY or FILE_TYPE_APP_PARAM, other types are invalid.	

4.13.2 OsUninstallFile

Prototype	**int OsUninstallFile(const char *AppName, int FileType);**	
Function	Uninstall applications and system files.	
Parameters	AppName 【Input】	1. When the FileType is FILE_TYPE_APP, it means AppName is the Application ID which needs to be deleted. 2. When the FileType is FILE_TYPE_ SYS_LIB, AppName is the name of system library.
	FileType	• FILE_TYPE_APP (Application package, the application has been installed all the files.) • FILE_TYPE_SYS_LIB (System library file)
Return	RET_OK	Success
	ERR_APP_NOT_EXIST	The application specified by AppName does not exist.
	ERR_FONT_NOT_EXIST	The font library does not exist.
Instruction	This function is only used for unloading application and parts of system files.	

After calling this function to uninstall all files that need to uninstall, the application should prompt the user to restart the terminal to complete the uninstallation.

4.14 Verify signature

The PROLIN 2.X provides a function to verify the file signature. It should use this interface to verify the signature before using the files.

4.14.1 OsVerifySign

Prototype	**int OsVerifySign(const char *FileName, int PUKType);**	
Function	Verify the file signature specified by FileName is legal or not, the signature data is included in the file.	
Parameters	FileName 【Input】	The file name which contains the path.
	PUKType 【Input】	▪ PUK_TYPE_M The public key of manufacturers. It is used to do the signature verification for firmware, released by manufacturer. ▪ PUK_TYPE_US_PUK The public key of user signature certificate, it is used to do the signature verification for the public key certificate. ▪ PUK_TYPE_USER0~ PUK_TYPE_USER8 The public key of users, it is used to do the signature verification for user application.
Return	RET_OK	Success
	ERR_VERIFY_SIGN_FAIL	Illegal signature.
	ERR_FILE_NOT_EXIST	The file does not exist.
	ERR_INVALID_PARAM	Invalid parameter
Instruction	1. This function is only used for the application to verify the application parameter file, for example, to verify the root certificate defined by the application. 2. In order to avoid validation repetition, it should not use this function to verify the legitimacy of the file until the file is installed. (System will verify automatically in OsInstallFile ()).	

4.14.2 OsVerifySignExternal

Prototype	**int OsVerifySignExternal(const char *FileName, const void *SignData, int PUKType);**
Function	Verify the file signature specified by FileName is legal or not. The file does not include the signature data.

Parameters	FileName 【Input】	The file name which contains the path.
	SignData 【Input】	Signature data, it has 284 bytes.
	PUKType 【Input】	▪ PUK_TYPE_M The public key of Manufacturers. It is used to do the signature verification for firmware, released by manufacturer. ▪ PUK_TYPE_US_PUK The public key of user signature certificate, it is used to do the signature verification for the public key certificate. ▪ PUK_TYPE_USER0~ PUK_TYPE_USER8 The public key of users, it is used to do the signature verification for user application.
Return	RET_OK	Success
	ERR_VERIFY_SIGN_FAIL	Illegal signature.
	ERR_FILE_NOT_EXIST	The file does not exist.
	ERR_INVALID_PARAM	Invalid parameter
Instruction	1. This function is only used for the application to verify the application parameter file, for example, to verify the root certificate defined by the application. 2. In order to avoid validation repetition, it should not use this function to verify the legitimacy of the file until the file is installed. (System will verify automatically in OsInstallFile ()).	

4.15 Get system version

This interface is reserved for future used.

4.15.1 OsGetSysVer

Prototype	**void OsGetSysVer(int VerType, char *Version);**	
Function	Read information of the system version.	
Parameters	VerType	Version types: • TYPE_OS_VER Operating system version • TYPE_OSAL_VERAPI Library version • TYPE_DRIVER_VER Driver version • TYPE_PED_VER built-in PED version • TYPE_MSR_VER MSR version • TYPE_ICC_VER ICC Reader version

		• TYPE_PCD_VER PCD Reader version • TYPE_EMVL1_VER EMV Level1 version • TYPE_PRINTER_VER Printer version • TYPE _MODEM_VER Modem version • TYPE_ETH_VER Netcard version • TYPE_GPRS_VER GPRS version • TYPE_CDMA_VER CDMA version • TYPE_TD_VER TD version • TYPE_WIFI_VER WIFI version • TYPE_BT_VER Bluetooth version
	Version 【Output】	Version information. (Ending with "\0", and <=31 bytes)
Return	None	
Instruction	1. If Version[0] is equal to 0, it means the module does not exist, 2. The buffer space of version must be >= 31 bytes.	

4.16 Determine whether on the base

PROLIN 2.X supports a function to determine whether the handset is on the base or not.

4.16.1 OsOnBase

Prototype	**int OsOnBase(void) x ;**	
Function	Determines whether the handset is on the base.	
Parameters	None	
Return	1	yes
	0	no
Instruction	This function is effective only to handsets which on the base.	

4.17 Save the crash report

PROLIN supports monitoring program state, once the program crashes, it will generate crash report in the directory '/usr/tombstones' after calling this function.

4.17.1 OsSaveCrashReport

Prototype	**void OsSaveCrashReport(ing sig);**	
Function	Save the crash report.	
Parameters	Sig	Signal value
Return	None	
Instruction	Method one: Through the function signal(SIG_XXX, OsSaveCrashReport);	

	OsSaveCrashReport will be registered as signal handler, for example:
	signal(SIGILL, OsSaveCrashReport);
	signal(SIGABRT, OsSaveCrashReport);
	signal(SIGBUS, OsSaveCrashReport);
	signal(SIGFPE, OsSaveCrashReport);
	signal(SIGSEGV, OsSaveCrashReport);
	signal(SIGSTKFLT, OsSaveCrashReport);
	Method two: During the signal handler process, calling OsSaveCrashReport(sig) to save the error message. For example:
	int mysighandler(int sig)
	{
	do_something();
	OsSaveCrashReport(sig);
	}
	The recommended signals are SIGILL, SIGABRT, SIGBUS, SIGFPE, SIGSEGV and SIGSTKFLT.
	After calling this function, it will ignore the signal which is corresponding to sig. That is, it calls the signal(sig, SIG_IGN) in function OsSaveCrashReport().
	In Terminal Manager(TM), it can export the report to U disk.

4.18 Mount

4.18.1 OsMount

Prototype	**int OsMount(const char *Source, const char *Target, const char *FileSystemType, unsigned long MountFlags, const void *Data);**	
Function	Mount the source file system to the target file system.	
Parameters	Source【Input】	The file system that needs to be mounted, it is usually a device that located in /dev/block/, and the path length cannot exceed 128 bytes.
	Target【Input】	The target file directory that will mount to, it must be in the /mnt/ directory, and the path length cannot exceed 128 bytes.
	FileSystemType【Input】	File system type that needs to be mounted, it can value as 'vfat'.
	MountFlags【Input】	Mount flag, it can be the combinations of the following flags: MS_DIRSYNC: Synchronize the directory updates MS_MANDLOCK: Allow the mandatory locks on files MS_MOVE: Move the subdirectory tree MS_NOATIME: Need not to update the access time

		MS_NODEV: Don't allow access to device file. MS_NODIRATIME: Don't allow update the access time on directory MS_NOEXEC: Don't allow execute programs on the mounted file system. MS_NOSUID: When executing program, do not follow to the set-user-ID and set-group-ID. MS_RDONLY: Specify the file system as read-only. MS_RELATIME: When a file is accessed, if the last access time (atime) is less than or equal to the last modification time (mtime) or last status change time (ctime), then update the last access time (atime) values. MS_SILENT: Stop writing warning information to the system kernel log MS_STRICTATIME: Always updating the last access time (atime) MS_SYNCHRONOUS: Synchronize the file updates
	Data【Input】	The user-defined additional data
Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid parameter
	ERR_STR_LEN	The stringR_LENameis overlength.
	ERR_NEED_ADMIN	Permission denied.
Instruction	Only the main application can mount, otherwise it fails to mount and returns ERR_NEED_ADMIN.	

4.18.2 OsUmount

Prototype	**int OsUmount(const char *Target, int Flags);**	
Function	Unmount the file system file system.	
Parameters	Target【Input】	The file system that needs to unmount, it must be in the /mnt/ directory, and the path length cannot exceed 128 bytes.
	Flags【Input】	Unmount flag, it can be combination of the following flags: MNT_DETACH: Lazy umount, the mount point is inaccessible after execution; it will unmount only when

		the mount point is not busy. MNT_EXPIRE:The mount point is marked as expired UMOUNT_NOFOLLOW: If the target is a symbolic link, do not reduce the reference count.
Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid parameter
	ERR_STR_LEN	The string length is overlength.
	ERR_NEED_ADMIN	Permission denied.
Instruction	Only the main application can unmount, otherwise it fails to unmount and returns ERR_NEED_ADMIN.	

5Encryption and Decryption

5.1 Return code list

Table 5 Encryption and decryption return code list

Macro	Value	Description
ERR_DATA_TOO_BIG	-2400	The encrypted data of RSA is greater than module.
ERR_GEN_RANDOM	-2401	Fail to generate random numbers.
ERR_GEN_FAIL	-2402	Fail to generate RSA key pairs.

5.2 Random number

PROLIN 2.X supports true random number, and provides the application interface to generate true random number.

5.2.1 OsGetRandom

Prototype	**void OsGetRandom(unsigned char *Random, int RandomLen);**
Function	Read the true random number.

Parameters	Random 【Output】	Storing the pointer of random number.
	RandomLen	Length of random number which needs to be read. (<=4096bytes)
Return	None	
Instruction		

5.3 SHA algorithm

As compared to the original PROLIN1.X, PROLIN 2.X provides comprehensive supports to the SHA algorithms, such as SHA-1, SHA-2 (SHA-256, SHA-512) and truncates form of the SHA-2 (SHA-224,SHA-384).

5.3.1 OsSHA

Prototype	**void OsSHA(int Mode, const void *Data, int DataLen, unsigned char* ShaOut);**	
Function	Calculate the Secure Hash value.	
Parameters	Mode	SHA_TYPE_1 SHA_TYPE_224 SHA_TYPE_256 SHA_TYPE_384 SHA_TYPE_512
	Data 【Input】	the input data buffer
	DataLen	the input data length
	ShaOut	Output value of SHA, the array should be equal to or more than 64 bytes.
Return	None	
Instruction	Calculate the hash values of SHA family.	

5.4 DES algorithm

PROLIN 2.X supports DES & TDES algorithms, and provide corresponding interface for the application.

5.4.1 OsDES

Prototype	**void OsDES(const unsigned char *Input,**

	unsigned char *Output, **const unsigned char *DesKey,** **int KeyLen,** **int Mode);**	
Function	Do DES / TDES encryption and decryption of the 8 bytes.	
Parameters	Input 【Input】	8 bytes input data
	Output【Output】	8 bytes output data
	DesKey【Input】	DES/TDES key
	KeyLen	8, 16 or 24 (bytes)
	Mode	0-decryption; 1-encryption.
Return	None	
Instruction	The encryption or decryption should be according to the mode selection. If the parameters are invalid, there will be no any operations.	

5.5 AES algorithm

PROLIN 2.X supports AES algorithm, including AES-128, AES-192, AES-256.

5.5.1 OsAES

Prototype	**void OsAES(const unsigned char *Input,** **unsigned char *Output,** **const unsigned char *AesKey,** **int KeyLen,** **int Mode);**	
Function	Perform AES encryption and decryption operation.	
Parameters	Input 【Input】	16 bytes input data
	Output【Output】	16 bytes output data
	AesKey【Input】	Key

	KeyLen	16, 24 or 32 (bytes)
	Mode	0-decryption; 1-encryption.
Return	None	
Instruction	This function supports 128, 192 or 256 (bits) AES encryption and decryption. If the parameter is invalid, there will be no any operations.	

5.6 RSA algorithm

PROLIN 2.X supports RSA algorithm, including public/private key-pair generation, RSA encryption and RSA decryption and also supports corresponding interface for the application. Currently PROLIN 2.X supports the modulus length up to 2048 bits.

5.6.1 OsRSA

Prototype	**int OsRSA(const unsigned char * Modulus, int ModulusLen, const unsigned char *Exp, int ExpLen, const unsigned char *DataIn, unsigned char *DataOut);**	
Function	Perform RSA encryption and decryption operation.	
Parameters	Modulus【Input】	Pointer that used to store the RSA algorithm modulus buffer (n=p*q). High byte first.
	ModulusLen	Modulus length(byte)
	Exp【Input】	Pointer to the exponent buffer in RSA operation. High byte first.
	ExpLen	Exponent length.(byte)
	DataIn【Input】	Pointer to input data buffer, length is the same as module.
	DataOut【Output】	Pointer to output data buffer, length is the same as module.

Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_DATA_TOO_BIG	ExpLen is bigger than ModulusLen.
Instruction	1. This function performs RSA encryption / decryption operations; encryption and decryption are performed by selecting different keys. If select a private key, such as Modulus, Exp, it will do encryption; for public key, does decryption. 2. This function can perform RSA operation with the length of no more than 2048 bits.	

5.6.2 OsRSAKeyGen

Prototype	**Int OsRSAKeyGen(unsigned char *Modulus, unsigned char *PriExp, int ModulusLen, const unsigned char * PubExp);**	
Function	Generate RSA Key pair.	
Parameters	Modulus【Output】	The key modulus. (High byte first)
	PriExp【Output】	Private key exponent. (High byte first)
	ModulusLen	Modulus length. (It can be 64, 128, 256 (bytes)).
	PubExp【Input】	Public key exponent. It only can be:"\x00\x00\x00\x03" or "\x00\x01\x00\x01"
Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid Parameters
	ERR_GEN_RANDOM	Fail to generate random data.
	ERR_GEN_FAIL	Fail to generate.

Instruction	By calling this interface, it will randomly generate a RSA Key pair with a specifying exponent and modulus.

6PED

PROLIN 2.X provides a series of PED interface, including built-in pinpad, MK / SK, DUKPT, RSA and other related interfaces.

6.1 Return code list

Table 6 PED Return code list

Macro	Value	Description
ERR_PED_NO_KEY	-3801	Key does not exist.
ERR_PED_KEY_IDX_ERR	-3802	Key index error.
ERR_PED_DERIVE_ERR	-3803	When key is written, the source key level is lower than the destination level.
ERR_PED_CHECK_KEY_FAIL	-3804	Key verification failed.
ERR_PED_NO_PIN_INPUT	-3805	No PIN input.
ERR_PED_INPUT_CANCEL	-3806	Cancel to enter PIN.
ERR_PED_WAIT_INTERVAL	-3807	Calling function interval is less than minimum interval time.(CalculatePINBLOCK/MAC)
ERR_PED_KCV_MODE_ERR	-3808	KCV mode error.
ERR_PED_KEY_TAG_ERR	-3809	Key tag error, the key can't be used.
ERR_PED_KEY_TYPE_ERR	-3810	Key type error.
ERR_PED_PIN_LEN_ERR	-3811	The input PIN length is not equal to the expected PIN length.

ERR_PED_DSTKEY_IDX_ERR	-3812	Destination key index error.
ERR_PED_SRCKEY_IDX_ERR	-3813	Source key index error.
ERR_PED_KEY_LEN_ERR	-3814	Key length error.
ERR_PED_INPUT_PIN_TIMEOUT	-3815	PIN input timeout.
ERR_PED_NO_ICC	-3816	IC card does not exist.
ERR_PED_ICC_INIT_ERR	-3817	IC card is not initialized.
ERR_PED_GROUP_IDX_ERR	-3818	DUKPT group index error.
ERR_PED_LOCKED	-3819	PED locked.
ERR_PED_NOMORE_BUF	-3820	No free buffer.
ERR_PED_NORMAL_ERR	-3821	PED general error.
ERR_PED_NEED_ADMIN	-3822	Not administration.
ERR_PED_DUKPT_KSN_OVERFLOW	-3823	DUKPT overflow.
ERR_PED_KCV_CHECK_FAIL	-3824	KCV check error.
ERR_PED_SRCKEY_TYPE_ERR	-3825	Source key type error.
ERR_PED_UNSPT_CMD	-3826	Command not support.
ERR_PED_ADMIN_ERR	-3827	Administration error
ERR_PED_DOWNLOAD_INACTIVE	-3828	PED download inactive.
ERR_PED_KCV_ODD_CHECK_FAIL	-3829	KCV parity check failed.
ERR_PED_PED_DATA_RW_FAIL	-3830	Read PED data failed.
ERR_PED_ICC_CMD_ERR	-3831	ICC operation failed.
ERR_PED_DUKPT_NEED_INC_KSN	-3832	DUKPT KSN needs to plus 1 first.
ERR_PED_DUKPT_NO_KCV	-3833	NO KCV.
ERR_PED_NO_FREE_FLASH	-3834	PED has not enough space.
ERR_PED_INPUT_CLEAR	-3835	Press [CLEAR] key to exit PIN input.
ERR_PED_INPUT_BYPASS_BYFUNCTION	-3836	Press FN/ATM4 to cancel PIN input.
ERR_PED_NO_PIN_MODE	-3837	PIN input mode is not set.
ERR_PED_DATA_MAC_ERR	-3838	Data MAC check error.
ERR_PED_DATA_CRC_ERR	-3839	Data CRC check error.
ERR_PED_KEY_VALUE_INVALID	-3840	The work key value already exists or does not match the requirements.

6.2 Data Definition

6.2.1 Key type

Table 7 Key Types

Macro	Value	Description
PED_TLK	0x01	Loading Key
PED_TMK	0x02	Master Key
PED_TPK	0x03	PIN Key
PED_TAK	0x04	MAC Key
PED_TDK	0x05	Data Key
PED_TIK	0x10	DUKPT Initial Key
PED_TRK	0x07	MSR Key
PED_TAESK	0x20	AES Key

6.2.2 Display attribute of Asterisk

Table 8 Layout attributes of asterisk

Macro	Value	Description
PED_ASTERISK_ALIGN_LEFT	0	left-aligned
PED_ASTERISK_ALIGN_CENTER	1	center-aligned
PED_ASTERISK_ALIGN_RIGHT	2	right-aligned

Table 9 The color values of asterisk

Macro	Value	Description
RGB(_r, _g, _b)	...	According to the inputted three-primary colors to generate color value with 16-bit.

6.3 Data Structure

6.3.1 Structure of RSA key

ST_RSA_KEY	
typedef struct{	
int ModulusLen;	*/*Modulus length(bits) */*
unsigned char Modulus[512];	*/*Modulus, if the length of modulus <512 bytes, store from right, add 0x00 in left. */*

```
    int ExponentLen;                    /* Exponent Length (bits) */
    unsigned char Exponent [512];       /* When exponent <512 bytes, add 0x00 in left.*/
    unsigned char KeyInfo[128];         /* RSA key information */
}ST_RSA_KEY;
```

6.3.2 RSA key structure for verifying the cipher text IC card PIN

ST_RSA_PINKEY

```
typedef struct{
    int ModulusLen;                     /*Modulus length(bits) */
    unsigned char Modulus[256];         /*Modulus of PIN public key*/
    unsigned char Exponent [4];         /* Exponent of PIN public key*/
    int IccRandomLen;                   /* Length of random data gets from IC card*/
    unsigned char IccRandom[8];         /* Random data gets from IC card*/
}ST_RSA_PINKEY;
```

6.4 Basic PED

6.4.1 OsPedOpen

Prototype	**int OsPedOpen(void);**	
Function	Open PED device.	
Parameters	None	
Return	RET_OK	Success
	ERR_DEV_BUSY	Device is busy.
Instruction	Other PED series functions can be operated only after open device successfully.	

6.4.2 OsPedGetVer

Prototype	**int OsPedGetVer (unsigned char * PedVer);**	
Function	Return to PED version.	
Parameters	PedVer【Output】	PED version, buffer size is 6 bytes.
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction		

6.4.3 OsPedSetInterval

Prototype	**int OsPedSetInterval(unsigned long TpkIntervalMs);**		
Function	Set the minimum interval time between consecutive operations of getting PINBlock.		
Parameters	TpkIntervalMs	= 0	Use the default value (30s)
		<1000	Automatically set to 1000 (1s)
		>600000	Automatically set to 600000 (10 min)
		=0xffffffff	No change in current settings.
Return	RET_OK	Success	
	ERR_DEV_NOT_OPEN	PED device is not open.	
	ERR_INVALID_PARAM	Invalid parameter.	
Instruction	Calculate the interval time of PINBLOCK: It can only be called 4 times, if the default time is 120-second, it means default value of TPKIntervalTimeMs is 30-second, after reset by calling this function, it is limited to call 4 times during the 4*TPKIntervalTimeMs time, for example, if the TPKIntervalTimeMs is 20000 (ms), then within 80 seconds it can only be called 4 times. This function is valid when open/close the machine. For example, set it only can be called 4 times within 80 seconds, if reboot the machine after the third time, then immediately to call 2 times in succession, these five times were all within 80 seconds, but the fifth time could not be successful, and it will prompt to wait.		

6.4.4 OsPedVerifyPlainPin

Prototype	**int OsPedVerifyPlainPin(int IccSlot,** **const char * ExpPinLen,** **int Mode,** **unsigned long TimeoutMs,** **unsigned char * IccRsp);**	
Function	Verify the offline plaintext PIN Get plaintext PIN. According to card command and card slot number, then sending plaintext PIN BLOCK to card.	
Parameters	IccSlot	ICC slot number, and IccSlot=0.
	ExpPinLen【Input】	Enumeration of 0-12 Application enumerates all the possible lengths of PIN, used the ‘,’ to separate each number of length. If it is allowed to input 4 or 6 digits, or enter without pressing any passwords, the string should be ‘0, 4, 6’. 0 means that user can press ‘Enter’ to return but without inputting any digits.
	Mode	0x00, IC card command mode. Currently supports EMV2000 only.
	TimeoutMs	The timeout of PIN input [ms] Maximum is 300000. 0 means no timeout, and the PED without timeout control.
	IccRsp【Output】	2 bytes Card response code (2 bytes: SWA++SWB)
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .

Instruction	1. Prompt cardholder to input PIN;
	2. Prompt cardholder that it is processing;
	3. Convert plaintext PIN to PIN BLOCK.
	Use OsIccexchange () to do the verification interaction with the card, as follows:
	ST_APDU_REQ apdu_s;
	ST_APDU_RSP apdu_r;
	
	memcpy (apdu_s.cmd, icc_command, 4);
	apdu_s.lc = icc_command[4];
	memcpy (apdu_s.data_in, PINBLOCK,apdu_s.lc);
	apdu_s.le = 0;
	if (icc_exchange(icc_slot, 0, &apdu_s, &apdu_r))
	return CMDERR;
	icc_resp[0] = apdu_r.swa;
	icc_resp[1] = apdu_r.swb;

6.4.5 OsPedVerifyCipherPin

Prototype	**int OsPedVerifyCipherPin(int IccSlot, const ST_RSA_PINKEY * RsaPinKey, const char * ExpPinLen, int Mode, unsigned long TimeoutMs, unsigned char * IccRspOut);**	
Function	Verify offline enciphered PIN. Get plaintext PIN, use RsaPinKey provided by the application to encrypt plaintext PIN according to EMV standards. According to card command and card slot number provided by application, and then sending plaintext PIN BLOCK to card.	
Parameters	IccSlot	ICC slot number, Iccslot=0.
	RsaPinKey【Input】	Encrypt the data structure.
	ExpPinLen【Input】	Application enumerates all the possible lengths

<table>
<tr><td rowspan="4"></td><td></td><td>of PIN, used the ‘,’ to separate each number of length. If it is allowed to input 4 or 6 digits, or enter without pressing any passwords, the string should be set as ‘0, 4, 6’.
0 means that user can press ‘Enter’ to return but without inputting any digits.</td></tr>
<tr><td>Mode</td><td>0x00, currently supports EMV2000 only.</td></tr>
<tr><td>TimeoutMs</td><td>The timeout of PIN input [ms]
Maximum is 300000.
0 means no timeout, and the PED without timeout control.</td></tr>
<tr><td></td><td>IccRsp【Output】</td><td>2 bytes
Card response code (2 bytes: SWA+SWB)</td></tr>
<tr><td rowspan="4">Return</td><td>RET_OK</td><td>Success</td></tr>
<tr><td>ERR_DEV_NOT_OPEN</td><td>PED device is not open.</td></tr>
<tr><td>ERR_INVALID_PARAM</td><td>Invalid parameter.</td></tr>
<tr><td>Others</td><td>Refer to the PED Return code list .</td></tr>
<tr><td>Instruction</td><td colspan="2">Encryption algorithm:
Use the public key, and apply RSA functions to the following data listed in the table to get enciphered PIN.
<table>
<tr><th>Name</th><th>Length</th><th>Description</th><th>Format</th></tr>
<tr><td>Head of Data</td><td>1</td><td>Hex. The value is ‘7F’</td><td>b</td></tr>
<tr><td>PIN Block</td><td>8</td><td>PINBLOCK</td><td>b</td></tr>
<tr><td>unpredictable IC card random data</td><td>8</td><td>The random number got from card, and it is provided in RSA_PINKEY,</td><td>b</td></tr>
<tr><td>Random padding bytes</td><td>NIC–17</td><td>the padding data got from the terminal application, it is provided in RSA_PINKEY,</td><td>b</td></tr>
</table></td></tr>
<tr><td></td><td colspan="2">1. Prompt message for PIN input;
2. Prompt cardholder that it is processing;
3. Convert plaintext PIN to PIN BLOCK;
4. Generate data, used for encryption (listed in above table)
5. Use public key, and apply RSA functions to encryption data which is generated in step4, then getting Enciphered PIN.
Use OsIccExchange () to send verification command to card, as follows:
ST_APDU_REQ apdu_s;</td></tr>
</table>

	ST_APDU_RSP apdu_r;
	memcpy (apdu_s.cmd, icc_command, 4);
	apdu_s.LC = icc_command[4];
	memcpy (apdu_s.data_in, EncipheredPIN, apdu_s.LC);
	apdu_s.LE = 0;
	if (OsIccExchange (icc_slot, 0, &apdu_s, &apdu_r))
	return ERR_PED_ICC_CMD_ERR;
	icc_resp[0] = apdu_r.SWA;
	icc_resp[1] = apdu_r.SWB;

6.4.6 OsPedEraseKeys

Prototype	**int OsPedEraseKeys(void);**	
Function	Clear all key information in PED.	
Parameters	None	
Return	RET_OK	Success.
	ERR_DEV_NOT_OPEN	PED device is not open.
	Others	Refer to the PED Return code list .
Instruction		

6.4.7 OsPedSetFunctionKey

Prototype	**int OsPedSetFunctionKey (int KeyFlag);**	
Function	Set some functions of function key. When PED is power on, the default function of CLEAR button is to clear input PIN. Other different functions of CLEAR button can also be set by calling this function.	
Parameters	KeyFlag	**0x00:** The PIN has already been cleared or no input PIN. By keep pressing the CLEAR button will make quit the input status and will return ERR_PED_INPUT_CLEAR. **0x01:** While calling this function, during the input process of the key input interfaces (OsPedGetPinBlock, OsPedGetPinDukpt, OsPedVerifyPlainPin, OsPedVerifyCipherPin etc), press CLEAR button is to clear the latest input PIN digit by digit. The CLEAR button is ineffective after clearing the entire PIN, and it will not exit the PIN input function.

		0x02: It is allowed to press ATM4 button to end the PIN input. This rule does not apply to the models without ATM button. **0x03:** It is allowed to press Function button to end the PIN input. This rule does not apply to the models without FN button. **0xff:** It means restore the default function of the function keys. (Press CLEAR button to clear all PIN, press ATM4/FN key does not exit the PIN input function.)
Return	RET_OK	Success.
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction		

During PIN input, if needs to support keypress to exit or clear input PIN one by one; it should call this function once after startup.

6.4.8 OsPedClose

Prototype	**void OsPedClose(void);**	
Function	Close the PED device.	
Parameters	None	
Return	None	
Instruction	This function should be called to close device while program exit.	

6.5 MK/SK

6.5.1 OsPedWriteKey

Prototype	**int OsPedWriteKey (const unsigned char * KeyBlock);**
Function	Write in a key, including write in and divergent of TLK, TMK and TWK, and use KCV to check the key correction.

Parameters	KeyBlock【Intput】	1 byte	Format: 0x03
		1 byte	SrcKeyType: • PED_TLK • PED_TMK • PED_TPK/PED_TAK/PED_TDK
		1 byte	SrcKeyIdx: • When SrcKeyType = PED_TLK, then SrcKeyIdx = 1; • When SrcKeyType = PED_TMK, then SrcKeyIdx = [1~100] • When writing in plaintext, SrcKeyIdx = 0
		1 byte	DstKeyIdx: • When DstKeyType = PED_TLK, then DstKeyIdx = 1; • When DstKeyType = PED_TMK, then DstKeyIdx = [1~100]; • When DstKeyType = PED_TPK or PED_TAK or PED_TDK, Then DstKeyIdx = [1~100].
		7 bytes	Reserved domain. Random number.
		1 byte	DstKeyType: • PED_TLK • PED_TMK • PED_TPK/PED_TAK/PED_TDK
		1 byte	DstKeyLen: 8/16/24
		8/16/24 bytes	DstKeyValue The destination key plaintext / ciphertext
		1 byte	KcvMode: • 0x00: No authentication

			• 0x01:Performs DES/TDES encryption on 8-byte 0x00, and use first 3 bytes in ciphertext as KCV. • 0x02:Firstly, performs parity check, then doing DES/TDES encryption on "\x12\x34\x56\x78\x90\x12\x34\x56", and use first 3 bytes in ciphertext as KCV. • 0x03:Transfers in a string of KcvData, use source key to perform specified mode MAC on [DstKeyValue + KcvData], and use the result as KCV.
		128 bytes	KcvData: • When KcvMode is 0x00/0x01/0x02, padding with random numbers. • When KcvMode is 0x03, the first byte of KcvData is the length of KCV data which participate in the calculation, the rest is KCV data. The first byte after the KCV data represents the MAC operation mode.
		8 bytes	• When KcvMode = 0x00, padding with random numbers. • When KcvMode =0x01/0x02/0x03, KcvValue points to the KCV value.
		10 bytes	Padding with random number.
Return	RET_OK	Success.	
	ERR_DEV_NOT_OPEN	PED device is not open.	
	ERR_INVALID_PARAM	Invalid parameter.	
	Others	Refer to the PED Return code list .	

Instruction	Writing the ciphertext and plaintext to the specific index position of the specific key type area. Using this function have following key points: 1. When SrcKeyIdx=0, system considers that the DstKeyValue is the plaintext of key and does not judge SrcKeyType and SrcKeyIdx. Directly write the DstKeyValue to DstKeyIdx in DstKeyType area. 2. Only when PED_TLK does not exist, to type-in plaintext or download any key is allowed. 3. When PED_TLK exist, it is not allowed to type in plaintext or download key. PED_TLK can be of 16 or 24 bytes. 8-byte key is not allowed. 4. Format PED to clear all downloaded keys and then write in PED_TLK. 5. If SrcKeyIdx is valid, PED considers the DstKeyValue as the key ciphertext, thus decrypt it using SrcKeyIdx key and write the key to DstKeyIdx. DstKeyType >= SrcKeyType. 6. DstKeyLen only can be of 8 or 16 or 24 bytes. If DstKeyLen = 8 bytes, the key could only be used for DES calculation. If DstKeyLen = 16 or 24 bytes, the key could be used for TDES calculation. DstKeyLen <= SrcKeyLen. 7. a) If DstKeyType=PED_TPK, the key only be used to encrypt PIN Block. b) If DstKeyType=PED_TAK, the key can only be used for MAC encryption. c) If DstKeyType=PED_TDK, the key can only be used for *DES/TDES. 8. KCV is the verification for plaintext. If plaintext is typed-in directly, the KcvMode of KeyIn is not 0 and the system will do the KCV verification for plaintext according to the specified KcvMode. 9. The valid KeyBlock must be 184 bytes, and the users must pass in valid parameters, otherwise an error will occur.

6.5.2 OsPedWriteTIK

Prototype	**int OsPedWriteTIK (const unsigned char * KeyBlock);**		
Function	Write in a TIK, and check the key correction by KCV.		
Parameters	KeyBlock【Intput】	1 byte	Format: 0x03
		1 byte	SrcKeyType: • PED_TLK
		1 byte	SrcKeyIdx: • When SrcKeyType = PED_TLK, then SrcKeyIdx = 1; • When plaintext writing,

			SrcKeyIdx = 0.
		1 byte	DstKeyIdx: DstKeyIdx = [1~100];
		7 bytes	Reserved domain. Random number.
		1 byte	DstKeyType: • PED_TIK
		1 byte	DstKeyLen: 8/16
		24 bytes	DstKeyValue The destination key plaintext / ciphertext
		1 byte	KcvMode: • 0x00: No authentication • 0x01:Performs DES/TDES encryption on the 8-bytes 0x00, and use first 3 bytes in ciphertext as KCV. • 0x02: Performs parity check 1st, then performs DES/TDES encryption on 8 bytes "\x12\x34\x56\x78\x90\x12\x34\x56", and use first 3 bytes in ciphertext as KCV. • 0x03: Sends in data KcvData, use source key to perform specified mode of MAC on [aucDesKeyValue + KcvData], and use the result as KCV.
		128 bytes	KcvData: • When the KcvMode is 0x00/0x01/0x02, padding with random numbers. • When the KcvMode is 0x03 the first byte of KcvData is the length of KCV data which participate in the calculation, the rest is KCV data. The first byte after the KCV data represents the MAC operation mode.
		8 bytes	• When KcvMode = 0x00, padding with random numbers. • When KcvMode = 0x01/0x02/0x03, KcvValue point

			to the KCV value.
		10 bytes	Initialize KSN,
Return	RET_OK	Success	
	ERR_DEV_NOT_OPEN	PED device is not open.	
	ERR_INVALID_PARAM	Invalid parameter	
	Others	Refer to the PED Return code list .	
Instruction	Writes the cryptograph and plaintext of a key to the specific index position of the specific key type area. Using this function has following key points: 1. When SrcKeyIdx=0, system considers that the DstKeyValue is the plaintext of key and will not judge the SrcKeyType and SrcKeyIdx. Write the DstKeyValue to DstKeyIdx in DstKeyType area directly. 2. Only when PED_TLK does not exist, it is allowed to type-in plaintext or download any key. 3. When PED_TLK exist, it is not allowed to type in plaintext or download key. 4. If SrcKeyIdx is valid, PED considers the DstKeyValue as key cryptograph thus decrypts it bySrcKeyIdx key and writes the key to DstKeyIdx. DstKeyType >= SrcKeyType. 5. KCV is verification for plaintext. If plaintext is typed-in directly, and the KcvMode of KeyIn is not 0, the system will process KCV verification for plaintext according to the specified KcvMode. 6. The valid KeyBlock must be 184 bytes, and the users must pass in valid parameters, otherwise an error will occur.		

6.5.3 OsPedWriteKeyVar

Prototype	**int OsPedWriteKeyVar(int KeyType, int SrcKeyIdx, int DstKeyIdx, const unsigned char * KeyVar);**	
Function	Uses the key plaintext that specified by source key index and key type, to do operation with a string of data and writes the result to the location, specified by the destination key index with the same key type.	
Parameters	KeyType	PED_TMK PED_TPK

		PED_TAK PED_TDK
	SrcKeyIdx	The source key index
	DstKeyIdx	The destination key index
	KeyVar【Input】	24 bytes.The extensible input data to be used in exclusive-or, length of it should be same as the key.
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter
	Others	Refer to the PED Return code list .
Instruction	Please refer to AS2805.6.	

6.5.4 OsPedSetAsteriskLayout

Prototype	**int OsPedSetAsteriskLayout(int x, int y, int fontSize, int fontColor, uchar align);**	
Function	Sets how to display the layout attributes of asterisk while inputting PIN.	
Parameters	x	X-coordinate
	y	Y-coordinate
	fontSize	Font size of asterisk: fontSize = 16, represents the character has 16 dots; fontSize = 24, represents the character has 24 dots; fontSize = 32, represents the character has 32 dots; fontSize = 48, represents the character has 48 dots; Display the asterisk with PED internal font, and

		it is not relevant to system installed font.
	fontColor	Font color of asterisk. Using the macro definition RGB(_r, _g, _b) and according to the inputted three-primary colors to generate color value with 16-bit.
	align	Alignment: PED_ASTERISK_ALIGN_LEFT; PED_ASTERISK_ALIGN_CENTER PED_ASTERISK_ALIGN_RIGHT
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction	1. The PIN inputting interface is displayed by the application, this function will only display asterisk. 2. It needs to call this function to set the displaying layout of asterisk before using PedVerifyPlainPin, PedVerifyCipherPin, PedGetPinBlock andPedGetPinDukpt.	

6.5.5 OsPedGetPinBlock

Prototype	**int OsPedGetPinBlock (int KeyIdx, const unsigned char * DataIn, const char * ExpPinLen, int Mode, unsigned long TimeoutMs, unsigned char * PinBlock);**	
Function	Scan the keyboard PIN entry and output the PIN BLOCKin a specific time. Input the PIN in the length specified by ExpPinLenIn, output the PIN BLOCK generated by algorithm encryption specified by Mode.	
Parameters	KeyIdx	[1-100] Index of TPK.
	DataIn【Input】	1. If Mode=0x00, **DataIn** is 16 bytes primary account number after shifting. 2. If Mode=0x01, Input parameters for participation in PinBlock formatting, 8 bytes data (refer to ISO9564 standard, this data can be Random number, transaction

		serial number or time stamp, etc.) 3. If Mode=0x02, **DataIn** is 16 bytes primary account number after shifting. 4. If Mode=0x03, datain is ISN [6 Bytes, ASCII code]
	ExpPinLen【Input】	Enumeration of 0-12 Application enumerates all possible lengths of PIN. ‘,’ will be used to separate each number of length. If no PIN, or 4 or 6 digits of PIN are allowed, the string will be set as ‘0, 4, 6’. 0 means that no PIN is required and pressing ‘Enter’ will return.
	Mode	PIN BLOCK format 0x00 0x00 ISO9564 format 0 0x01 0x01 ISO9564 format 1 0x02 0x02 ISO9564 format 3 0x03 0x03 HK EPS format
	TimeoutMs	The timeout of PIN entry [ms] Maximum is 300000ms. 0: Without timeout or related control for PED.
	PinBlock【Output】	8 bytes. Point to the generated PINBlock.
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction	Press ‘**CANCEL**’ to cancel input.	

6.5.6 OsPedUpdatePinBlock

Prototype	**int OsPedUpdatePinBlock (int UpdateFlag, const unsigned char * KeyInfo, const unsigned char * DataIn, unsigned char * PinBlock, int Mode);**	
Function	Getting PINBlock again and choose to replace TPK.	
Parameters	UpdateFlag	0: Do not replace TPK, Non zero: Replace TPK

	KeyInfo【Input】	▪ It has 184 bytes, details refer to the KeyBlock definition in OsPedWriteKey(). ▪ When UpdateFlag is 0, only ucDstKeyIdx is valid, using ucDstKeyIdx, specify TPKand recalculate PINBLOCK. ▪ When UpdateFlag is 1,please refer to the OsWriteKey
	DataIn【Input】	When Mode=0x03, Transaction serial number ISN [6 Bytes, ASCII code]
	PinBlock【Output】	8 bytes Input original PINBlock data, output new PINBLOCK
	Mode	0x03 HK EPS dedication format[Appendix EPS_PINBLOCK Format]
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction	For EPS.	

6.5.7 OsPedGetMac

Prototype	**int OsPedGetMac(int KeyIdx, const unsigned char *DataIn, int DataInLen, unsigned char *Mac, int Mode);**	
Function	Use MAC key, specified by the KeyID to do the MAC operation for the following Mode algorithm, output the 8-byte result to Mac.	
Parameters	KeyIdx	1~100 TAK index
	DataIn【Input】	<=1024 bytes The data package that needs to do the MAC operation.
	DataInLen	The length of data package. If the length is not multiple of 8, 0x00 will be padded automatically.

	Mac【Output】	8 bytes, output of MAC.
	Mode	1. 0x00: Does the DES/TDES encryption for BLOCK1 by using MAC key. Does the DES/TDES encryption again by using TAK when and after bitwise XOR the previous encryption result with BLOCK 2. Processes in turn to get the 8 bytes encryption result. 2. 0x01: Does bitwise XOR for BLOCK1 and BLOCK 2; Does bitwise XOR again by using previous XOR result with BLOCK3. Does it in turn and finally gets the 8 bytes XOR result. Uses TAK to process DES/TDES encryption for the result. 3. 0x02: ANSIX9.19 standard. Does DES encryption for BLOCK1 by using TAK (only take the first 8 bytes of the key). The encryption result will bitwise XOR with BLOCK 2, and then does the DES encryption by using TAK again. Does it in turn and get the 8 bytes encryption result. Uses DES/TDES to encrypt in the last time.
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction	For EPS.	

6.5.8 OsPedDes

Prototype	**int OsPedDes(int KeyIdx,** **unsigned char * InitVector,** **const unsigned char *DataIn,** **int DataInLen,** **unsigned char *DataOut,** **int Mode);**
Function	Uses the TDK to do the DES/TDES decryption for the data and then outputs plaintext or ciphertext. A specified TDK can be used for encryption and decryption algorithms.

Parameters	KeyIdx	TDK index.1~100.
	InitVecto【Input】	Used for CBC/OFB encryption or decryption. If set to NULL, it will set the initialization vector as "\x00\x00\x00\x00\x00\x00\x00\x00" by default. It is not needed for ECB encryption or decryption, and can be set to NULL.
	DataIn【Input】	Points to the data that needs to be calculated.
	DataInLen	Data length. It should be <=1024, and multiple of 8.
	DataOut 【Output】	Points to the data that has been calculated.
	Mode	• 0x00: ECB Decryption • 0x01: ECB Encryption • 0x02: CBC Decryption • 0x03: CBC Encryption • 0x04: OFB Decryption • 0x05: OFB Encryption
Return	RET_OK	Success
	ERR_DEV_NOT _OPEN	PED device is not open.
	ERR_INVALID _PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction		

Using DES or TDES depends on the key length.

6.5.9 OsPedGetKcv

Prototype	**int OsPedGetKcv (int KeyType,** **int KeyIdx,** **int KcvMode,**

<table>
<tr><td></td><td colspan="2">int KcvDataLen,
unsigned char * KcvData,
unsigned char * Kcv);</td></tr>
<tr><td>Function</td><td colspan="2">Gets KCV value for key verification of two sides:
1. While it isn’t TIK: uses specific key and algorithm to encrypt the data, and then return the first 3 bytes of the cryptograph.
2. While it is TIK: returns the 8-byte KCV which was injected while TIK-injection.</td></tr>
<tr><td rowspan="6">Parameters</td><td>KeyType</td><td>PED_TLK
PED_TMK
PED_TAK
PED_TPK
PED_TDK
PED_TIK</td></tr>
<tr><td>KeyIdx</td><td>Index number of the key, for example:
TLK can only be 1.
TMK takes range value from 1 to 100.
TWK takes range value from 1 to 100.
TIK takes range value from 1 to 100.</td></tr>
<tr><td>KcvMode</td><td>0x00: KCV check mode.</td></tr>
<tr><td>KcvDataLen</td><td>The data length used in the KCV calculation. It should be <=128 bytes and be the multiple of 8.
It can be “0” if the type is TIK.</td></tr>
<tr><td>KcvData【Input】</td><td>Points to the data that needs to be calculated. It can be NULL if the type is TIK.</td></tr>
<tr><td>Kcv【Output】</td><td>3 or 8bytes. Points to KCV.
KCV of TIK has 8 bytes, other types have 3 bytes.</td></tr>
<tr><td rowspan="3">Return</td><td>RET_OK</td><td>Success</td></tr>
<tr><td>ERR_DEV_NOT_OPEN</td><td>PED device is not open.</td></tr>
<tr><td>ERR_INVALID</td><td>Invalid parameter.</td></tr>
</table>

	_PARAM	
	Others	Refer to the PED Return code list .
Instruction		

6.5.10 OsPedDeriveKey

Prototype	**int OsPedDeriveKey (int SrcKeyType, int SrcKeyIdx, int DstKeyType, int DstFromKeyIdx, int DstToKeyIdx, int Mode);**	
Function	Divergent key. Uses the key specified by SrcKeyIdx to do the encryption or decryption for the key specified by DstFromKeyIdx, then derives a new key and save it as the specified key of DstToKeyIdx.	
Parameters	SrcKeyType	Types of the source key. PED_TLK PED_TMK PED_TAK PED_TPK PED_TDK
	SrcKeyIdx	Index number of source key, for example: TLK can only be 1. TMK takes range value from 1 to 100. TWK takes range value from 1 to 100.
	DstKeyType	Types of the destination key PED_TLK PED_TMK PED_TAK

		PED_TPK PED_TDK
	DstFromKeyIdx	Source index of the destination key
	DstToKeyIdx	Destination index of the destination key
	Mode	0x00: DES/TDES decryption 0x01: DES/TDES encryption
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction	The source key level should not be lower than the destination key type.	

6.6 DUKPT

6.6.1 OsPedGetPinDukpt

Prototype	**int OsPedGetPinDukpt(int GroupIdx, const unsigned char * DataIn, const char * ExpPinLen, int Mode, unsigned long TimeoutMs, unsigned char * Ksn, unsigned char * PinBlock);**	
Function	Scans the input PIN in a specified time, and outputs the PINBlock which generated by computing the PIN key of DUKPT.	
Parameters	GroupIdx	1~100:DUKPT group ID

	DataIn【Input】	1. If Mode=0x20, **DataIn** is the 16 bytes primary account number after shifting. 2. If Mode=0x21, inputs parameters for participation in PinBlock formatting, 8 bytes data (refer to ISO9564 standard, this data can be Random numbers, the transaction serial number or time stamp, etc.) 3. If Mode=0x22, **DataIn** is the 16 bytes primary account number after shifting. 4. If Mode=0x23, datain is ISN [6 Bytes, ASCII code]
	ExpPinLen【Input】	0~12 enumerate set. Application enumerates all possible lengths of PIN. ‘,’ will be used to separate each number of lengths. If no PIN, or 4 or 6 digits PIN are allowed, the string should be set to ‘0, 4, 6’. 0 means that no PIN is required and pressing ‘Enter’ will return.
	Mode	Choose the format of PIN BLOCK • 0x20 ISO9564 format 0, KSN not plus 1 automatically. • 0x21 ISO9564 format 1, KSN not plus 1 automatically. • 0x22 ISO9564 format 2, KSN not plus 1 automatically. • 0x23 HK EPS format, KSN not plus 1 automatically.
	TimeoutMs	The timeout of PIN entry [ms] Maximum is 300000ms. 0 means there is no timeout time, not doing timeout control for PED.
	Ksn【Output】	Points to the current KSN.(10 bytes)
	PinBlock【Output】	Points to the generated PIN Block result.(8 bytes)
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction	When KSN does not plus 1, a DUKPT PIN key can only calculate the PIN BLOCK for once.	

6.6.2 OsPedGetMacDukpt

Prototype	**int OsPedGetMacDukpt(int GroupIdx, const unsigned char *DataIn, int DataInLen, unsigned char *Mac, unsigned char *Ksn, int Mode);**	
Function	Calculate MAC by using MAC key of DUKPT.	
Parameters	GroupIdx	1~100:DUKPT group ID
	DataIn【Input】	Points to the data that needs to calculate the MAC.
	DataInLen	The data length should be <=1024. If the length is not the multiple of 8, 0x00 will be padded automatically.
	Mac【Output】	Points to the obtained MAC.
	Ksn【Output】	Points to the current KSN.
	Mode	1. 0x20: Does TDES encryption for BLOCK1 by using MAC key. Does TDES encryption again by using TAK when and after bitwise XOR the previous encryption result with BLOCK 2. Processes in turn to get the 8 bytes encryption result. 2. 0x21: Does bitwise XOR for BLOCK1 and BLOCK 2; Does bitwise XOR again by using previous XOR result with BLOCK3. Does it in turn and finally gets the 8 bytes XOR result. Uses TAK to process TDES encryption for the result. 3. 0x22: ANSIX9.19 standard, Does DES encryption for BLOCK1 by using TAK (only take the first 8 bytes of key). The encryption result will bitwise XOR with BLOCK 2 and then does DES encryption by using TAK again. Does it in turn and gets the 8 bytes encryption result. Uses TDES to encrypt in the last time. 4. 0x20/0x21/0x22: KSN not plus 1 automatically. 5. 0x40/0x41/0x42: The MAC calculation method is

		the same with 0x20/0x21/0x22. 6. 0x40/0x41/0x42: Chooses to response the MAC key. 7. 0x20/0x21/0x22: KSN chooses to request and response MAC key 8. 0x40/0x41/0x42: KSN not plus 1 automatically. 9. Other values are reserved for extended MAC algorithm.
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction	If KSN does not increase, both the response MAC key and the response-request MAC key can calculate MAC for unlimited times.	

6.6.3 OsPedDesDukpt

Prototype	**int OsPedDesDukpt (int GroupIdx, int KeyVarType, unsigned char *InitVector, int DataInLen, unsigned char *DataIn, unsigned char *DataOut, unsigned char *Ksn, int Mode);**	
Function	Uses DES/MAC key of DUKPT to do encryption and decryption for the input data.	
Parameters	GroupIdx	1~100:DUKPT group ID
	KeyVarType【Input】	0x00 Uses the requests and responses of MAC key 0x01 Uses DES key of DUKPT 0x02 Uses the PIN variant to to encrypt the data, and it

		is only available for EBC encryption, that means the Mode can only be 1.
	InitVector【Input】	Used for CBC/OFB encryption or decryption. If set to NULL, it will set the initialization vector as "\x00\x00\x00\x00\x00\x00\x00\x00" by default.(8 bytes) It is not needed for ECB encryption, and can be set to NULL.
	DataInLen	The data needed to be calculated should be <= 8192 bytes.
	DataIn 【Input】	Input data.
	DataOut【Output】	Points to the data that has been calculated.
	Ksn【Output】	Current KSN.(10 bytes)
	Mode	• 0x00: ECB decryption • 0x01: ECB encryption • 0x02: CBC decryption • 0x03: CBC encryption • 0x04: OFB decryption • 0x05: OFB encryption
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction	When KSN is unchanged, a group of the DUKPT Data key can do DES operations for 256 times at most.	

6.6.4 OsPedGetKsnDukpt

Prototype	**int OsPedGetKsnDukpt (int GroupIdx, unsigned char * Ksn);**	
Function	Reads the current KSN value.	
Parameters	GroupIdx	1-100: DUKPT group ID
	Ksn【Output】	Points to the current KSN. (10 bytes)

Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction		

6.6.5 OsPedIncreaseKsnDukpt

Prototype	**int OsPedIncreaseKsnDukpt (int GroupIdx);**	
Function	Increases KSN value of the specific DUKPT group.	
Parameters	GroupIdx	1-100: DUKPT group ID
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction		

6.7 RSA

6.7.1 OsPedReadRsaKey

Prototype	**int OsPedReadRsaKey (int RsaKeyIdx, ST_RSA_KEY * RsaKey);**	
Function	Reads the RSA public key.	
Parameters	RsaKeyIdx	1~10: Index of RSA Key.
	RsaKey【Output】	RSA public key.
Return	RET_OK	Success

	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction	It can only read the RSA public key, while reads private key, returns error.	

6.7.2 OsPedWriteRsaKey

Prototype	**int OsPedWriteRsaKey (int RsaKeyIdx, ST_RSA_KEY * RsaKey);**	
Function	Inject RSA key into the PED.	
Parameters	RsaKeyIdx	1~10: Index of RSA Key.
	RsaKey【Input】	RSA public key. RSA key that needs to be injected into PED.
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction	The type of RSA key is depending on the exponent's length, it is the private key if the length equals to modulus.	

NOTE

1. Currently it does not support that RSA key whose length is more than 256 bytes.
2. RSA key can be rewritten at any time.

6.7.3 OsPedRsaRecover

Prototype	**int OsPedRsaRecover (int KeyIdx,**

	int DataInLen, **unsigned char * DataIn,** **unsigned char * DataOut,** **unsigned char * KeyInfo);**	
Function	Uses the RSA key stored in PED to process data operation.	
Parameters	RsaKeyIdx	1~10: Index of RSA Key.
	DataInLen	The length of operation data, and it is the same with the RSA modulus. 64 bytes or 128 bytes or 256 bytes.
	DataIn【Input】	Points to the data that needs to be calculated.
	DataOut【Output】	Points to the data that has been calculated.
	KeyInfo【Output】	Key information
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction		

6.7.4 OsPedReadCipherRsaKey

Prototype	**int OsPedReadCipherRsaKey (int RsaKeyIdx,** **unsigned char * CipherRsaKey);**	
Function	Reads the ciphertext of RSA key.	
Parameters	RsaKeyIdx	1~10: Index of RSA Key.
	CipherRsaKey【Output】	Points to the ciphertext data of RSA key.

Return	>0	The byte length of the RSA ciphertext.
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction		

6.7.5 OsPedWriteCipherRsaKey

Prototype	**int OsPedWriteCipherRsaKey (int RsaKeyIdx, int CipherRsaKeyLen, unsigned char * CipherRsaKey);**	
Function	Writes the ciphertext of RSA key.	
Parameters	RsaKeyIdx	1~10: Index of RSA Key.
	CipherRsaKeyLen	The byte length of the ciphertext data of RSA key.
	CipherRsaKey【Input】	Points to the ciphertext data of RSA key.
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction		

6.8 AES

6.8.1 OsPedWriteAesKey

<table>
<tr><td>Prototype</td><td colspan="3">int OsPedWriteAesKey (const unsigned char * KeyBlock);</td></tr>
<tr><td>Function</td><td colspan="3">Write inan AES key and use KCV to check the key correction.</td></tr>
<tr><td rowspan="10">Parameters</td><td rowspan="10">KeyBlock【Input】</td><td>1 byte</td><td>Format: 0x03</td></tr>
<tr><td>1 byte</td><td>SrcKeyType:
PED_TLK
PED_TMK</td></tr>
<tr><td>1 byte</td><td>SrcKeyIdx:
When SrcKeyType = PED_TLK,
then SrcKeyIdx = 1;
When SrcKeyType = PED_TMK,
then SrcKeyIdx = [1~100];
If ucSrcKeyIdx = 0, key will be written in PED as plain text.</td></tr>
<tr><td>1 byte</td><td>DstKeyIdx: 1-100.</td></tr>
<tr><td>7 bytes</td><td>Reserved domain. Random number.</td></tr>
<tr><td>1 byte</td><td>DstKeyType: PED_TAESK</td></tr>
<tr><td>1 byte</td><td>DstKeyLen: 16/24/32</td></tr>
<tr><td>32 bytes</td><td>DstKeyValue:
The destination key plain-text or cipher-text.</td></tr>
<tr><td>1 byte</td><td>KcvMode:
0x00: No KCV check.
0x01: Performs AES ECB encryption on 16-byte 0x00, and use first 3 bytes as KCV.
0x02: Perform parity check at first, then perform AESECB encryption on 16 bytes “\x12\x34\x56\x78\x90\x12\x34\x56\x12\x34\x56\x78\x90\x12\x34\x56”, and use first 3 bytes as KCV.
0x03: Transfers in a string of KcvData, use source key to perform specified mode MAC on [DstKeyValue (cipher) + KcvData],</td></tr>
</table>

<table>
<tr><td rowspan="4"></td><td rowspan="4"></td><td></td><td>and use the result as KCV.</td></tr>
<tr><td>128 bytes</td><td>KcvData:
When KcvMode is 0x00/0x01/0x02, padding with random numbers.
When KcvMode is 0x03, the first byte of KcvData is the length of KCV data which participates in the calculation, the rest is KCV data. The first byte after the KCV data represents the MAC operation mode.</td></tr>
<tr><td>8 bytes</td><td>When KcvMode = 0x00, padding with random numbers.
When KcvMode =0x01/0x02/0x03,KcvValue point to the KCV value.</td></tr>
<tr><td>2 bytes</td><td>Padding with random number.</td></tr>
<tr><td rowspan="4">Return</td><td colspan="2">RET_OK</td><td>Success</td></tr>
<tr><td colspan="2">ERR_DEV_NOT_OPEN</td><td>Device is not open.</td></tr>
<tr><td colspan="2">ERR_INVALID_PARAM</td><td>Invalid parameter.</td></tr>
<tr><td colspan="2">Others</td><td>Refer to the PED Return code list.</td></tr>
<tr><td>Instruction</td><td colspan="3">Writing the cryptograph and plaintext of an AES key to the specific index position of the AES area. Using this function have following key points:
When SrcKeyIdx=0, system consider that the DstKeyValue is the plaintext of key and does not judge SrcKeyType and SrcKeyIdx. Write the DstKeyValue to DstKeyIdx in DstKeyType area directly.
Only when PED_TLK does not exist, to inject plaintext or download any key into PED is allowed.
When PED_TLK exist, it is not allowed to inject in plaintext or download key.
If SrcKeyIdx is valid, PED considers the DstKeyValue as the key cryptography, thus decrypt it using SrcKeyIdx key and write the key to DstKeyIdx.
The valid KeyBlock must be 184 bytes, and the users must pass in valid parameters, otherwise an error will occur.</td></tr>
</table>

6.8.2 OsPedAes

Prototype	**int OsPedAes(intKeyIdx, unsigned char * InitVector, const unsigned char *DataIn, intDataInLen, unsigned char *DataOut, int Mode);**	
Function	Uses the specified AES key stored in PED to do the AES encryption or decryption for the data and then output cipher-text or plain-text.	
Parameters	KeyIdx 【Input】	TAESK index: 1~100.
	InitVector 【Input】	Used for CBC/OFB encryption or decryption. If set to NULL, it will set the initialization vector as "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00" by default. It is not needed for ECB encryption or decryption, and can be set to NULL.
	DataIn 【Input】	Points to the data that needs to be calculated.
	DataInLen 【Input】	Data length. It should be <=1024, and multiple of 16.
	DataOut 【Output】	Points to the data that has been calculated.
	Mode 【Input】	• 0x00: ECB Decryption • 0x01: ECB Encryption • 0x02: CBC Decryption • 0x03: CBC Encryption • 0x04: OFB Decryption • 0x05: OFB Encryption
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	Device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to the PED Return code list .
Instruction		

{ This page intentionally left blank }

7LCD

In PROLIN 2.X, the operation of displaying contents on LCD is managed by the GUI, it supports the graphics systems such as Minigui, QT. In this chapter, it will provide OsScrBrightness(), OsScrContrast(), OsScrGetSize() and some related interfaces for application use.

Applications need to develop their own GUI system, or they can use FrameBuffer approach to test the effectiveness of the LCD driver. OsScrContrast(), OsScrBrightness(), OsScrGetSize() and the rest of the LCD operations are managed by the GUI.

Applications (such as driver testing) can also operate FrameBuffer directly. Details are as follows: Operating the FrameBuffer .

Steps of reading and writing the FrameBuffer:

1. Open the FrameBuffer device,
2. Get the fixed screen information by ioctl,
3. Get the variable screen information by ioctl,
4. Map device memory to the process space by mmap,
5. Write the FrameBuffer

```
int open_screen(void)
{
    char vtname[128];
```

```
    int fd, nr;
    unsigned y, addr;
    struct fb_fix_screeninfo fix;
    sb = (screen_buffer*)malloc(sizeof (screen_buffer));
    if ((sb->dev_fd = open(FB_DEV_PATH, O_RDWR)) == -1) {
    perror("open");
    return -1;
    }

    int ret = ioctl(sb->dev_fd, FBIOGET_VSCREENINFO, &fb_vinfo);

    if (ret) {
     sb->width = FB_WIDTH;
    sb->height = FB_HEIGHT;
    sb->bytes_per_pixel = FB_BYTES_PER_PIXEL;
    fprintf(stderr,"in %s line %d",__FUNCTION__,__LINE__);
    } else {
    sb->width = fb_vinfo.xres;
    sb->height = fb_vinfo.yres;
    sb->bytes_per_pixel = fb_vinfo.bits_per_pixel / 8;
    }
    if(sb->bytes_per_pixel == 3)
    sb->bytes_per_pixel = 4;

    if (ioctl(sb->dev_fd, FBIOGET_FSCREENINFO, &fix) < 0) {
     close(sb->dev_fd);
     return -1;
    }

    fbmemlen = sb->width * sb->height * sb->bytes_per_pixel;

    if ((sb->buffer = (uint8_t *) mmap(NULL, fbmemlen, PROT_READ | PROT_
WRITE,MAP_FILE |MAP_SHARED, sb->dev_fd, 0)) == (uint8_t *) -1)
    {
    fprintf (stderr, "rw_sd_inand.c: Can't mmap frame buffer ++\n");
    exit (1);
    }
    memset(sb->buffer, 0, fbmemlen);
    return 0;
}
```

1. Close the FrameBuffer device

```
void close_screen(screen_buffer *sb)
{
    if(!sb)
    return;
    // Unmap the framebuffer
    munmap(sb->buffer, fbmemlen);
    // Close framebuffer device
    close(sb->dev_fd);
    free(sb);
}
```

According to the above mentioned steps, open the framebuffer, you can draw and write, and the content that you filled in the framebuffer will be timely displayed on the screen. But in the multi-process operating environment, such as in a window management environment, while operating the framebuffer, it will show a full screen picture, and at the same time, there will be a process keep updating the current system time, and cause the current picture shows incompletely. Actually it only needs the current process to display a complete picture, and the display area does not want to be interfered by the other operation processes of the framebuffer.

But the question is; how to avoid the above scenario? The right way is to operate the tty device. Open and activate a new terminal tty device, set it to the image mode and make the framebuffer device be used exclusively by the terminal, then output the picture. In this way, even if there are other processes operating the framebuffer, the framebuffer of the current terminal will not be heavy brush, and will achieve the stable output.

In general there are 7 tty devices in Linux system. tty0 is an alias for the current virtual terminal, and the information generated by the system will be sent to it. tty1-tty6 are called virtual terminal device. But how to know which tty device you should open? First, use the ioctl (ConsoleFD, VT_OPENQRY, &vtnumber) to query the number of currently opened virtual terminal. The general release is open 6, i.e., tty1~tty6, this can be controlled in / etc / inittab. In addition, tty0 is opened automatically, but not used for user login, so there will be open 7 query results, vtnumber = 7, and this figure is the available terminal number for the next time. tty7 Code reference: it references the code in tslib: http://blog.csdn.net/fyzhao/archive/2008/12/11/3501099.aspx

```
struct vt_stat vts;
    char vtname[128];
    int fd, nr;
    unsigned y, addr;

    char *consoledevice = "/dev/tty";

    if (strcmp (consoledevice, "none") != 0) {
    sprintf (vtname,"%s%d", consoledevice, 1);
    printf("line %d : vtname is:%s\n",__LINE__,vtname);
    fd = open (vtname, O_WRONLY);
    if (fd < 0) {
     perror("open consoledevice");
     return -1;
    }

    if (ioctl(fd, VT_OPENQRY, &nr) < 0) {
     perror("ioctl VT_OPENQRY");
     return -1;
    }
    close(fd);

    sprintf(vtname, "%s%d", consoledevice, nr);

    printf("line %d : vtname is:%s\n",__LINE__,vtname);
    con_fd = open(vtname, O_RDWR | O_NDELAY);
    if (con_fd < 0) {
     perror("open tty");
     return -1;
    }

    if (ioctl(con_fd, VT_GETSTATE, &vts) == 0)
      last_vt = vts.v_active;

    if (ioctl(con_fd, VT_ACTIVATE, nr) < 0) {
     perror("VT_ACTIVATE");
     close(con_fd);
     return -1;
    }
```

```
if (ioctl(con_fd, VT_WAITACTIVE, nr) < 0) {
 perror("VT_WAITACTIVE");
 close(con_fd);
 return -1;
}

if (ioctl(con_fd, KDSETMODE, KD_GRAPHICS) < 0) {
 perror("KDSETMODE");
 close(con_fd);
 return -1;
}

}
```

7.1 OsScrContrast

Prototype	**void OsScrContrast(int Contrast);**	
Function	Sets the contrast.	
Parameters	Contrast	Contrast level [0~7, darkest: 0, lightest: 7]. Default value: 4. Other values: no action.
Return	None	
Instruction	Only available on monochrome LCD models.	

7.2 OsScrBrightness

Prototype	**void OsScrBrightness(int Brightness);**	
Function	Sets the screen brightness.	
Parameters	Brightness	Brightness level[0~10, 0 represents turn off the backlight. 10 represents the lightest value. All brightness values are visible] Default value: 8. Other values: no action.
Return	None	
Instruction	This function is equivalent to the original backlight settings function, but the level of brightness has been increased.	

7.3 OsScrGetSize

Prototype	**void OsScrGetSize(int *Width, int *Height);**	
Function	Gets the LCD Physical screen size.	
Parameters	Width【Output】	Width (unit :pixel).
	Height【Output】	Height (unit :pixel).
Return	None	
Instruction	The screen size is just a read-only property.	

8Keyboard

The keyboard input of PROLIN 2.X is managed by GUI.

Key value definition

Macro	Value	Description
KEY1	2	KEY”1”
KEY2	3	KEY “2”
KEY3	4	KEY “3”
KEY4	5	KEY “4”
KEY5	6	KEY “5”
KEY6	7	KEY “6”
KEY7	8	KEY “7”
KEY8	9	KEY “8”
KEY9	10	KEY “9”
KEY0	11	KEY “0”
KEYCANCEL	223	KEY “CANCEL”
KEYCLEAR	14	KEY “CLEAR”

KEYENTER	28	KEY “ENTER”
KEYALPHA	69	KEY “Alpha”
KEYF1	59	At the bottom of S800 LCD, the first key from left to right.
KEYF2	60	At the bottom of S800 LCD, the second key from left to right.
KEYF3	61	At the bottom of S800 LCD, the third key from left to right.
KEYF4	62	At the bottom of S800 LCD, the fourth key from left to right.
KEYFUNC	*102*	
KEYUP	*103*	
KEYDOWN	*108*	
KEYMENU	*139*	

The application developers can directly use the input subsystem when they need to test keyboard drivers or transplant other GUI systems.

Details refer to the following examples of calling the input subsystem:

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <linux/input.h>
static int event0_fd = -1;
struct input_event ev0[64];
//for handling event0, mouse/key/ts
static int handle_event0() {
    int button = 0, realx = 0, realy = 0, i, rd;

    rd = read(event0_fd, ev0, sizeof(struct input_event) * 64);

    if ( rd < sizeof(struct input_event) ) return 0;

    for (i = 0; i < rd / sizeof(struct input_event); i++) {
```

```
            printf("", ev0[i].type, ev0[i].code, ev0[i].value);
            if (ev0[i].type == 3 && ev0[i].code == 0)
                realx = ev0[i].value;
            else if (ev0[i].type == 3 && ev0[i].code == 1)
                realy = ev0[i].value;
            else if (ev0[i].type == 1) {
                if (ev0[i].code == 158) {
                    //if key esc then exit
                    return 0;
                }
            }
            else if (ev0[i].type == 0 && ev0[i].code == 0 && ev0[i].value == 0) {
                realx = 0, realy = 0;
            }
            printf("event(%d): type: %d; code: %3d; value: %3d; realx: %3d;
realy: %3d\n", i,
          ev0[i].type, ev0[i].code, ev0[i].value, realx, realy);
      }
      return 1;
}

int main(void) {
     int done = 1;
      printf("sizeof(struct input_event) = %d\n", sizeof(struct input_event));
      event0_fd = open("/dev/input/event0", O_RDWR);
      if ( event0_fd < 0 )
            return -1;
      while ( done ) {
          printf("begin handel_event0...\n");
            done = handle_event0();
          printf("end handel_event0...\n");
      }
      if ( event0_fd > 0 ) {
            close(event0_fd);
            event0_fd = -1;
      }
      return 0;
}
```

8.1 OsKbBacklight

Prototype	**void OsKbBacklight(int OnOff);**	
Function	Switches the keyboard backlight.	
Parameters	OnOff	0: Turn off the backlight. Non-zero: Turn on the backlight.
Return	None	
Instruction		

9Touch Screen

The touch screen input of PROLIN 2.X is managed by GUI.

The application developers can directly use the input subsystem if they need to test touch screen drivers or transplant other GUI systems.

About calling input subsystem, it can refer to the example of the keyboard.

This chapter is only available on touch screen models.

10Signature Board

The related operations of signature board are based on PED device, and it must open PED first and close it after use.

This chapter is only available on touch screen models.

10.1 OsSignCap

Prototype	**int OsSignCap(SIGN_PARAM *sign);**	
Function	Captures signature handwriting.	
Parameters	sign【Input/Output】	*typedef struct _SIGN_POINT { char x[2]; char y[2]; } SIGN_POINT;* *typedef struct _SIGN_PARAM {* *unsigned int rgba_bg; /*Input background color */* *unsigned int rgba_fg; /*Input foreground color */* *SIGN_POINT point_array[SIGN_MAX_ARRAY]; /* Output Save signature track points */* *unsigned int point_len; /* Output The length of the track point */* *} SIGN_PARAM;*
Return	RET_OK	Success

	ERR_INVALID_PARAM	Invalid parameter.
	ERR_DEV_NOT_EXIST	Device does not exist.
	ERR_DEV_BUSY	Device is busy.
	ERR_USER_CANCEL	Cancelled by user.
Instruction		

11Printer

PROLIN 2.X provides the printer function of physical printer and virtual printer and also provides the unified interface for API. For physical printer, the senior application developers can access the printer driver through a POSIX interface to achieve the specific print function.

11.1 Return code list

Table 10 Printer return code list

Macro	Value	Description
ERR_PRN_BUSY	-3701	Printer busy
ERR_PRN_PAPEROUT	-3702	Out of paper
ERR_PRN_WRONG_PACKAGE	-3703	The format of print data packet error
ERR_PRN_OVERHEAT	-3704	Printer over heating
ERR_PRN_OUTOFMEMORY	-3705	The print data is too large, and exceeds the buffer length.
ERR_PRN_OVERVOLTAGE	-3706	Voltage is too high.

11.2 Open and Close

11.2.1 OsPrnOpen

Prototype	**int OsPrnOpen(unsigned int printertype, const char* targetname);**	
Function	Opens the printer (including physical and virtual).	
Parameters	printertype【Input】	Printer type • PRN_REAL: Physical printer. • PRN_BMP: Bmp virtual printer and it generates bmp format files.
	targetname【Input】	For the physical printer, this parameter should be NULL. The output file name of virtual printer, this parameter should fill in the file name generated by virtual printing, for example, /home/app/test.bmp.
Return	RET_OK	Success.
	ERR_DEV_NOT_EXIST	Device does not exist.
	ERR_INVALID_PARAM	Invalid parameter
	ERR_DEV_BUSY	Device is busy.
Instruction	Other functions can be operated only after open device successfully.	

11.2.2 OsPrnReset

Prototype	**void OsPrnReset(void);**	
Function	Resets and re-initializes the printer.	
Parameters	None	
Return	None	
Instruction	Calling this function will restore the printer default settings and clear the print buffer data. This function is applicable to both physical printers and virtual printers.	

After calling OsPrnReset (), the font choice of library file will be set to default font status (only available in English).

11.2.3 OsPrnClose

Prototype	**void OsPrnClose(void);**	
Function	Closes the printer.	
Parameters	None	
Return	None	
Instruction	This function should be called to close the printer device while program exit.	

11.3 Printer settings

11.3.1 OsPrnSetSize

Prototype	**int OsPrnSetSize (unsigned int Width, unsigned int Height);**	
Function	Sets printer parameters.	
Parameters	Width 【Input】	Width
	Height 【Input】	Height
Return	RET_OK	Success.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction	Only applicable to the virtual printer. The default size is 384*5000, and the maximum size is 600*5000. This function is effective in the first calling, and it will not change the first settings in the following repeat calls.	

11.3.2 OsPrnSetDirection

Prototype	**int OsPrnSetDirection (unsigned char Mode);**	
Function	Sets the print direction.	
Parameters	Mode 【Input】	• 0; print horizontally. • Non-zero; print vertically.
Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid parameter.
Instruction	This function is applicable to both physical printers and virtual printers.	

11.3.3 OsPrnSetGray

Prototype	**void OsPrnSetGray(int Level);**
Function	Sets printing gray level.

Parameters	Level	• Level =0, reserved, • Level =1, default level, normal print slip, • Level =2, reserved, • Level =3, two-layer thermal printing, • Level =4, two-layer thermal printing, higher gray level than 3, • The default level is 1. • The illegal value does not change current settings.
Return	None	
Instruction	Before setting gray level, it prints with the default level, after calling this function it will print with the setting level. Only applicable to the physical printer.	

11.4 Type Setting

11.4.1 OsPrnSetSpace

Prototype	**void OsPrnSetSpace(int CharSpace, int LineSpace);**	
Function	Sets the printing space.	
Parameters	CharSpace	Character space (unit: pixel) (It is invalid to the mandatory non-monospaced fonts, such as Arabic fonts, Thai fonts.)
	LineSpace	Line space(unit: pixel)
Return	None	
Instruction	1. Settings will be valid until they are set again or OsPrnReset () is called; 2. Printing character space defaults to 0; 3. Printing line space defaults to 0 for thermal and 2 for spocket; 4. The maximum line space can be 255; 5. The maximum character space can be 255. 6. Illegal value does not change the current settings.	

11.4.2 OsPrnSetReversal

Prototype	**int void OsPrnSetReversal(int Attr);**	
Function	Sets the reverse attribute of font, normal printing with the default settings.	
Parameters	Attr	0: normal Non zero: reversal
Return	None	
Instruction	This function is applicable to both physical printers and virtual printers.	

11.4.3 OsPrnSetIndent

Prototype	**int OsPrnSetIndent (unsigned int Left, unsigned int Right);**	
Function	Sets the left and right margins.	
Parameters	Left【Input】	The left margin: the valid range is [0, 100] and the default value is 0.
	Right【Input】	The right margin: the valid range is [0, 100] and the default value is 0.
Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid parameter
Instruction	If the physical printer is set to print vertically, then the left margin should correspond to the bottom margin of the page, and right margin corresponds to the top margin.	

11.4.4 OsPrnCheck

Prototype	**int OsPrnCheck(void);**	
Function	Checks the current status of printer.	
Parameters	None	
Return	RET_OK	Success.
	ERR_PRN_NOFONTLIB	Has no font library.
	ERR_PRN_BUSY	Printer is busy.
	ERR_PRN_PAPEROUT	Out of paper.
	ERR_PRN_OVERHEAT	Printer overheating.
Instruction	This function can be used to check whether there is a printing font library, whether there is paper, whether printing buffer is full, and whether the printer is over heating or not. Only applicable to the physical printer.	

11.4.5 OsPrnGetDotLine

Prototype	**int OsPrnGetDotLine(void);**
Function	Gets current printed dot line for slip alignment.

Parameters	None	
Return	>=0	Current dot line.
Instruction	Used for slip alignment. This function is applicable to both physical printers and virtual printers.	

11.4.6 OsPrnSetFont

Prototype	**int OsPrnSetFont(const char * fontname);**	
Function	Selects print fonts.	
Parameters	fontname【Input】	Font(file) name
Return	RET_OK ERR_FONT_NOT_EXIST ERR_INVALID_PARAM	Success Font does not exist. Invalid parameter.
Instruction	It can choose a different font style and font size for printing. The system built-in font (file) name can be obtained by calling OsEnumFont() function.	

11.4.7 OsPrnSelectFontSize

Prototype	**void OsPrnSelectFontSize(int SingleCodeWidth, int SingleCodeHeight, int MultiCodeWidth, int MultiCodeHeight);**	
Function	Sets the font size.	
Parameters	SingleCodeWidth	The width control of single code font. (For non-monospaced font, width of each character may not meet the settings). The value ranges from 8 to 64.
	SingleCodeHeight	The height control of single code font. The value ranges from 8 to 64.
	MultiCodeWidth	The width control of multiple code font. The value ranges from 12 to 64.
	MultiCodeHeight	The height control of multiple code font

		The value ranges from 12 to 64.
Return	None	
Instruction	After the first calling of OsPrnOpen ( ), the font width and height are set to the default values (12x24) (24x24). This function is applicable to both physical printers and virtual printers.	

Suggest the height and width of multiple code font should be the same, otherwise, the font may display abnormally.

11.4.8 OsPrnFeed

Prototype	**void OsPrnFeed(int Pixel);**	
Function	Feeds printing paper "pixel" pixels in print buffer.	
Parameters	Pixel	number of pixels
Return	None	
Instruction	1. If the pixel value is positive, then the paper will feed forwards. If it is negative, then feed backwards. If it is 0, then no action. 2. This function is applicable to both physical and virtual printers.	

This is a one-time action.

11.4.9 OsPrnPrintf

Prototype	**void OsPrnPrintf(const char *Str, ...);**	
Function	Formats output string to print buffer.	
Parameters	Str【Input】	Pointer of string that needs to be printed.
Return	None	
Instruction	1. Support variable parameters; 2. Support '\n' (new line) and '\f' (new page) control characters; 3. If the printing data package is too long, then the printing program will overflow; 4. If the string is longer than the current printing line, automatically changes line and continues printing; 5. The maximum buffer size is 2048 bytes; 6. Store str in printing buffer, and print data in printing buffer in sequence	

	after calling OsPrnStart ().

11.4.10 OsPrnPutImage

Prototype	**void OsPrnPutImage(const unsigned char *Logo);**	
Function	Outputs images to the print buffer.	
Parameters	Logo【Input】	Pointer to the logo information; the length cannot be more than 20000 bytes.
Return	None	
Instruction	1. Bitmap data is generated as follows: ▪ Draw a bitmap (usually a logo): use paintbrush program under Windows to draw a bitmap and save it as a "monochromatic, bmp format" file. ▪ Use "Bitmap Converter" provided by PAX to convert the .bmp file into a header file, for instance, Logo.h header file. (If more than one .bmp files are selected, then multiple character arrays (with the corresponding bmp file name) will be defined in the generated header file.) ▪ Printing bitmap size limit: up to 384 pixels in width, spocket with 180 pixels and the height is unlimited. 2. Use the generated array as the input parameter of this function. 3. If the bitmap width is larger than the limit of the printer, then it will be sliced on the right side. 4. If the data packet is too long, then this function will remove the LOGO message.	

NOTE

Description of the array in the header file:

First byte [1 byte]: lines number of the bitmap;

Size of the first bitmap line in byte [2 Bytes, MSB (most significant byte)ahead];

Bitmap data of the first bitmap line [one line of the bitmap have 8 pixels in height];

Size of the second bitmap line in byte [2 Bytes, MSB ahead];

Bitmap data of the second bitmap line;

So on and so forth.

This function only stores logo into printing buffer, and begins printing data in printing buffer in sequence after calling OsPrnStart ().

11.4.11 OsPrnStart

Prototype	**int OsPrnStart(void);**	
Function	Starts printer and prints the data in the buffer.	
Parameters	None	
Return	RET_OK	Success
	ERR_PRN_BUSY	Printer is busy.
	ERR_PRN_PAPEROUT	Out of paper.
	ERR_PRN_WRONG_PACKAGE	The format of print data packet error.
	ERR_PRN_OVERHEAT	Printer overheating.
	ERR_PRN_OUTOFMEMORY	The print data is too large.
Instruction	1. After calling this API, the printer will perform the printing task and return after completing the whole printing task. 2. After completing the whole printing task, this API will return the printer status in return value. Therefore the check printer status is not required. 3. If the printing process is completed, recalling this function will reprint the slip.	

11.5 POSIX

PROLIN 2.X physical printer driver module makes the POSIX programming interface open to the senior application developers.

11.5.1 open

Opens the physical printer, the device name is “/dev/printer”

int handle = open(“/dev/printer”, O_RDWR);

11.5.2 read

Read the printer status. The data format in Read buffer is defined as follows:

char 0:

- 0x00 normal
- 0x01 printer is busy
- 0x02 out of paper
- 0x03 printer over heating
- 0x04~0xFF reserved

```
unsigned char buf[10];
 int ret = read(handle, buf, 2);
 if (ret > 0) {
       //buf[0]
 }
```

11.5.3write

Send the printer configurationbuffer.

The first two bytes of the buffer are gray settings and reserved bit,

char 0: bit0~bit2: The control value of print grayscale,

- 000(0) Reserved
- 001(1) Normal gray level
- 010(2) Reserved
- 011(3) Two-layer thermal printing A
- 100(4) Two-layer thermal printing B
- 101(5) Reserved
- 110(6) Reserved
- 111(7) Reserved

bit3~bit7: Reserved

char 1: from char[2], multiple batches of 48 characters composed a single line (384 dots), if it is less than 48 then it will be padding with blank by the driver.

```
unsigned char buf[50];

 buf[0] = 0x01;
 memset(buf + 2, 0xff, 48);

 int ret = write(handle, buf, 50);

 if (ret < 0) {
       //Error handling……
 }
```

The limit of the maximum write data length is:

To the thermal printing of 384 dots, the driver can deal with 5000 lines each time at most; the longest length of a write buffer should be 384*5000/8 + 2 = 240002 Bytes. The out of bound content will be trimmed off.

To the thermal printing of 384 dots, when prints horizontally, the driver can deal with 5000 lines each time at most, if out of the range, it will not print. When prints vertically, it can deal with 384 lines each time, and each line can print 5000 dots at most, if there are more than 384 lines, it will not print. The longest length of a write buffer should be 384*5000/8 + 2 = 240002 Bytes.

11.5.4 Close

Closes the printer file handles.

close (handle);

12Font Library

PROLIN 2.X supports Freetype as the system font library. Therefore, the system supports a series of vector font and bitmap font.

12.1 Data structure

FT_FONT	
typedef struct {	
char FileName[64];	*/* Font file name */*
char FontName[64];	*/* Font name */*
}FT_FONT;	

FT_DOT	
typedef struct {	
unsigned char Left;	*/*Font offset left from the baseline*/*
unsigned char Top;	*/* Font top offset from the baseline */*
unsigned char Width;	*/* Font width */*

```
    unsigned char    Height;          /* Font height */
    unsigned char    Dot[3072];       /*Valid font data */
}FT_DOT;
```

12.2 Font operation

12.2.1 OsEnumFont

Prototype	**int OsEnumFont(FT_FONT *FontList);**	
Function	Gets the vector font list provided by system.	
Parameters	FontList【Output】	Vector fonts list
Return	>=0	Read the number of vector fonts
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_FONT_NOT_EXIST	Font library does not exist.
Instruction	*Example:*	

```
int i, num;
FT_FONT *FontList;
num = OsEnumFont(&FontList);
if(num <= 0)
    return -1;
for(i=0; i<num; i++)
    printf("[%d]file name: %s, font name : %s\n",
            i,  FontList[i].FileName, FontList[i].FontName);
```

12.2.2 OsOpenFont

Prototype	**int OsOpenFont (const char *FileName);**	
Function	Loads vector fonts.	
Parameters	FileName【Input】	Font file name.
Return	>=0	Font handle

	ERR_INVALID_PARAM	Invalid parameter.
	ERR_FONT_NOT_EXIST	System does not install font library.
Instruction	It needs to cache the dot-matrix data after open the fonts, and it is recommended to call OsGetFontDot() after 3 seconds.	

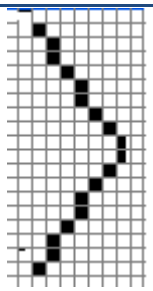
12.2.3 OsCloseFont

Prototype	**void OsCloseFont (int Handle);**	
Function	Closes vector fonts.	
Parameters	Handle 【Input】	Font handle
Return	None	
Instruction	After using vector fonts, please promptly shut down to release the system resources.	

12.2.4 OsGetFontDot

Prototype	**int OsGetFontDot (int Handle,** **const char *Utf8Code,** **const int Width,** **const int Height,** **const int Style,** **FT_DOT *FtDot);**	
Function	Gets the utf-8 encoding standard character font.	
Parameters	Handle 【Input】	Font handle
	Utf8Code 【Input】	Characters of UTF-8 encoding standards
	Width 【Input】	Font width, value range is【8, 128】.
	Height 【Input】	Font height, value range is【8, 128】.
	Style 【Input】	Font style: FONT_STYLE_NONE 0 No style

		FONT_STYLE_BOLD 0x00000001 Bold type FONT_STYLE_ITALIC 0x00000002 Italic type FONT_STYLE_BOLD|FONT_STYLE_ITALIC represents bold italics.
	FtDot 【Output】	The output of font data structure.
Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid parameter
	ERR_FILE_NOT_EXIST	File does not exist.
	ERR_FONT_CODE	Font code error.
	ERR_INVALID_HANDLE	Invalid handle
Instruction	▪ Utf8Code input. UTF-8 code is a variable length, and it needs to end with '\ 0', when the code is composed of letters, Utf8Code requires two bytes, which Utf8Code [0] represents letter, Utf8Code [1] represents ' \ 0'; but for Chinese, Utf8Code requires four bytes, which Utf8Code [0-2] represents the Chinese, and Utf8Code [3] represents ' \ 0'. ▪ The Italic style dot matrix. When using italics effects, the obtained dot matrix width is wider than the set value. It is not recommended that the dot size should be not less than 24, or the dot may cannot show italics effects. For example, when the dot size of Song typeface is less than 19, and the bold font dot size is less than 21, the italic effects cannot be used for Chinese, but it is available for letters. ▪ The format of font data. 1. All of the font dot matrix are in horizontal arrangement mode; 2. The point which corresponds to each byte, the sequence from left to right is 0x80 to 0x01; 3. If the character width is not integer multiple of 8, the bytes of per line dot matrix are (width+7)/8 For example: For the character”>“, with 10(width)×20(height)	



The font data is:

0x00, 0x00,

0x20, 0x00,

0x10, 0x00,

0x10, 0x00,

0x08, 0x00,

0x04, 0x00,

0x04, 0x00,

0x02, 0x00,

0x01, 0x00,

0x00, 0x80,

0x00, 0x80,

0x01, 0x00,

0x02, 0x00,

0x04, 0x00,

0x04, 0x00,

0x08, 0x00,

0x10, 0x00,

0x10, 0x00,

	0x20, 0x00, 0x00, 0x00 After calling this function, the character returns: Width = 10 Height = 20 Dot data: 0x00,0x00,0x20,0x00,0x10,0x00,0x10,0x00, 0x08,0x00,0x04,0x00,0x04,0x00,0x02,0x00, 0x01,0x00,0x00,0x80,0x00,0x80,0x01,0x00, 0x02,0x00,0x04,0x00,0x04,0x00,0x08,0x00, 0x10,0x00,0x10,0x00,0x20,0x00,0x00,0x00

13Code

PROLIN 2.X supports UTF8 as the system default code, and also provides the code-conversion interface.

13.1 Code Convert

13.1.1 OsCodeConvert

Prototype	**int OsCodeConvert (const char *FromCharset, const char *ToCharset, const char *InBuf, char *OutBuf, unsigned int LenOut);**	
Function	Implement conversion of character encoding.	
Parameters	FromCharset 【Input】	The original character encoding
	ToCharset 【Input】	The target character encoding
	InBuf 【Input】	Character string of the original encoding, ending with' \0'. Unicode should end in"\0\0"
	OutBuf	The converted encoding string

	【Output】	
	LenOut 【Input】	Size of array OutBuf, it should be 1.5 times of the array InBuf.
Return	>=0	Success, then returns the length of converted character string
	ERR_INVALID_PARAM	Invalid parameter
Instruction	Supports conversions among the following codes. ISO-8859-(1,2,3,4,5,6,7,8,9,10,11, 13,14,15,16) cp(850,862,866,874,932,936,949,950,1131, 1133,1250,1251,1252,1253,1254,1255,1256,1257,1258) GBK/GB18030(2 bytes part) BIG5 SHIFT_JIS EUC-KR UNICODE UTF-8	

1. The conversion is only recommended between the above codes and UTF-8 codes. Others might fail.
2. UNICODE adopts the Little-Endian mode.

14MSR

PROLIN 2.X provides the function of reading the magnetic stripe data and provides a unified API reading interface for use. In addition, senior application developers can access to the magnetic drive through the POSIX interface and directly get the magnetic stripe bit-stream to achieve different logics of magnetic stripe decoding.

14.1 Return code list

Table 11 MSR return code list

Macro	Value	Description
ERR_MSR_FAILED	-2701	Failed
ERR_MSR_HEADERR	-2702	Did not find the head mark.
ERR_MSR_ENDERR	-2703	Did not find the end mark
ERR_MSR_LRCERR	-2704	LRC check error
ERR_MSR_PARERR	-2705	One bit of MSR check error.
ERR_MSR_NOT_SWIPED	-2706	No swiping
ERR_MSR_PED_DECRYPTERR	-2709	PED decryption failed.

14.2 Data structure

MSR structure: Records information and status of each magnetic track.

ST_MSR_DATA:		
typedef struct {		
	unsigned char TrackData[256];	*/* Track data buffer */*
	int DataLen ;	*/* Track data length */*
	int Status ;	*/* Track data status, status equal 0 indicate read track data succeed, other value indicate failed */*
}ST_MSR_DATA;		

NOTE

When the status of track data is 0, it means reads track successfully, and it has two scenarios:

1. If the data format is correct or there is no data in track, then it needs to be combined with DataLen to be determined;
2. When Status <0, DataLen will be equal to 0, and the TackData will not include the information of magnetic track.

14.3 MSR control interface

14.3.1 OsMsrOpen

Prototype	**int OsMsrOpen(void);**	
Function	Switches on magnetic stripe reader.	
Parameters	None	
Return	RET_OK	Success
	ERR_DEV_NOT_EXIST	Device does not exist.
	ERR_DEV_BUSY	Device is busy.

	ERR_DEV_NOT_OPEN Device is not open.
Instruction	Other functions can be operated only after open device successfully.

Magnetic stripe reader works in interrupt mode. When the magnetic stripe reader is opened, it can read the magnetic track data, even if no card-reading function is called. So it is better to switch off magnetic stripe reader when it is not in use.

14.3.2 OsMsrClose

Prototype	**void OsMsrClose(void);**	
Function	Switches off magnetic stripe reader.	
Parameters	None	
Return	None	
Instruction	This function should be called to close device while program exit.	

14.3.3 OsMsrReset

Prototype	**void OsMsrReset(void);**	
Function	Resets magnetic stripe reader	
Parameters	None	
Return	None	
Instruction	When the magnetic reader is powered on, this function resets the reader and clears the data in the magnetic buffer.	

14.3.4 OsMsrRead

Prototype	**int OsMsrRead(ST_MSR_DATA *Track1, ST_MSR_DATA *Track2, ST_MSR_DATA *Track3);**	
Function	Reads data of magnetic stripe card.	
Parameters	Track1【Output】	Output the data of Track1
	Track2【Output】	Output the data of Track2

	Track3【Output】	Output the data of Track3
Return	RET_OK	Success
	ERR_MSR_NOT_SWIPED	No swiping.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_DEV_NOT_OPEN	Device is not open.
Instruction	If a certain track's data is not needed, set the corresponding pointer to NULL, then the data will not be outputted. After swiped card successfully, and users didn't call this interface to read the track data, the data will be automatically emptied. And all of the returned track data are 0x00.	

Calling OsMsrSwiped () first to detect the swipe actions (or receives the information of WM_MSR_SWIPED), then calls this function to obtain the data of magnetic track. Otherwise, when the function returns, the data included in the buffer is invalid.

For magnetic card conforming to ISO7812:
Track1 needs 79 bytes
Track2 needs 37 bytes
Track3 needs 107 bytes

14.3.5 OsMsrSwiped

Prototype	**int OsMsrSwiped(void);**	
Function	Checks whether a card is swiped or not	
Parameters	None	
Return	TRUE	Card swiped
	FALSE	Not swiped
	ERR_DEV_NOT_OPEN	Device is not open.
Instruction	1. This function returns the corresponding value immediately, doesn't matter card swiped or not. 2. Call this function to check whether a card is swiped or not. 3. If the card is swiped, the system will send the message "MSG_MSR_SWIPED" to the application that used to open the magnetic device, details refer to the "System Information".	

	4. After calling the OsMsrOpen ( ), OsMsrRead ( ) or OsMsrReset ( ), the status of swiping card will be clear.

14.4 POSIX

PROLIN 2.X Magnetic driver module makes the POSIX programming interface open to the senior application developers.

14.4.1 Open

Opens the magnetic stripe reader, and the device name is"/dev/msr"

int handle = open ("/dev/msr", O_RDONLY);

14.4.2 Read

Read the data of the magnetic bit stream.

The data format in Read buffer is defined as follows:

Variable length of data, but it is not more than 3 * 750 bytes. In the case of no card swiping, the returned data length of read is 0, and the read() may return -1.

The data bit stream is represented by using the ASCII code 0/1 and an ASCII code represents a bit, tracks are separated by 0x0A.

0101010101010101010000000000000000000000000000000001111111111111111111111111111\n

0101010101010101010000000000000000000000000000000001111111111111111111111111111\n

0101010101010101010000000000000000000000000000000001111111111111111111111111111\n

```
unsigned char buf[2250];
 int ret = read(handle, buf, 2250);
 if (ret > 0) {
       /*Bit stream decode*/
 }
```

14.4.3 Close

Closes file handles of the magnetic stripe reader.

After closing the handle, the original magnetic data stored in the drive buffer will be cleared.

close (handle);

15ICC Reader

The basic protocol interface is customized according to the ISO7816/EMV.

15.1 Return Code List

Table 12 ICC reader return code list

Macro	Value	Description
ERR_SCI_HW_NOCARD	-2800	No card
ERR_SCI_HW_STEP	-2801	Exchange when no init, warm reset when no active
ERR_SCI_HW_PARITY	-2802	Parity error
ERR_SCI_HW_TIMEOUT	-2803	Time out
ERR_SCI_TCK	-2804	TCK error
ERR_SCI_ATR_TS	-2810	TS error in ATR

ERR_SCI_ATR_TA1	-2811	TA1 error in ATR
ERR_SCI_ATR_TD1	-2812	TD1 error in ATR
ERR_SCI_ATR_TA2	-2813	TA2 error in ATR
ERR_SCI_ATR_TB1	-2814	TB1 error in ATR
ERR_SCI_ATR_TB2	-2815	TB2 error in ATR
ERR_SCI_ATR_TC2	-2816	TC2 error in ATR
ERR_SCI_ATR_TD2	-2817	TD2 error in ATR
ERR_SCI_ATR_TA3	-2818	TA3 error in ATR
ERR_SCI_ATR_TB3	-2819	TB3 error in ATR
ERR_SCI_ATR_TC3	-2820	TC3 error in ATR
ERR_SCI_T_ORDER	-2821	Protocol is not T0 or T1
ERR_SCI_PPS_PPSS	-2830	PPSS error in PPS
ERR_SCI_PPS_PPS0	-2831	PPS0 error in PPS
ERR_SCI_PPS_PCK	-2832	TC3 error in ATRPCK error in PPS
ERR_SCI_T0_PARAM	-2840	Data in transmitting is too long in T0
ERR_SCI_T0_REPEAT	-2841	Too many character repetition in T0
ERR_SCI_T0_PROB	-2842	Procedure byte error in T0
ERR_SCI_T1_PARAM	-2850	Data in transmitting is too long in T1
ERR_SCI_T1_BWT	-2851	BWT exceed in T1
ERR_SCI_T1_CWT	-2852	CWT exceed in T1
ERR_SCI_T1_BREP	-2853	Too many block repetition in T1
ERR_SCI_T1_LRC	-2854	LRC error in T1
ERR_SCI_T1_NAD	-2855	NAD error in T1
ERR_SCI_T1_LEN	-2856	LEN error in T1
ERR_SCI_T1_PCB	-2857	PCB error in T1

ERR_SCI_T1_SRC	-2858	SRC error in T1
ERR_SCI_T1_SRL	-2859	SRL error in T1
ERR_SCI_T1_SRA	-2860	SRA error in T1
ERR_SCI_PARAM	-2880	Parameter not allow

15.2 Data Structure

15.2.1 IC card device control block

sci_dcb_t:

```
/* device control block */
struct sci_dcb_t {
        unsigned int voltage;           /* operation condition: voltage */

        /* frequency adjust integer, default value is 372 */
        unsigned int fi;
        /* speed adjust integer, default value is 1 */
        unsigned int di;

        unsigned int conv;              /* logical converse direction */
        unsigned int protocol;          /* T=0 or T=1 */

        unsigned char option_clock;             /* stop clock options */
        unsigned char option_voltage;           /* voltage options */
        unsigned char option_spu;

        /* these members are appended, you must notice */
        unsigned int spec;
        unsigned int nego;                      /* support PPS protocol */

        /* the guard time between characters, only for T=0 */
        unsigned int cgt;
        /* block guard time, default value is 22, only for T=1 */
        unsigned int bgt;

        /* RESET signal maintain LOW level clock cycles, default is 42500 */
        unsigned int rstt;

        unsigned int wtt; /* TS wait time, default value is ??? */

        /* allowed maximum of ATR duration, only used in EMV mode (spec=0) */
        unsigned int twt;
        unsigned int wwt;       /* T=0, work wait time, default is 0x0A */
```

```
        /* character repetition (in T=0), maximum is 6 */
        unsigned int tpar_retry;    /* send repeat time on parity error */
        unsigned int rpar_retry;    /* recv repeat time on parity error */

        unsigned int bwt;/* T=1, block wait time, default is 0x00 */
        unsigned int cwt;/* T=1, character wait time, default is 0x05 */

        unsigned char repeat;

        /* the maxium frame size of ICC, default value is 32 */
        unsigned int fsc;
        unsigned int fsd;

        unsigned char sci_last_ipcb;
        unsigned char icc_last_ipcb;

        unsigned char sci_last_pcb;
        unsigned char icc_last_pcb;

        /* reset status: cold reset, warm reset, or activation */
        unsigned int status;
};
```

15.2.2 ATR structure

sci_atr_t:

```
/* ATR */
struct sci_atr_t {
        unsigned char ts;
        unsigned char t0;
        unsigned char ta_flag;
        unsigned char tb_flag;
        unsigned char tc_flag;
        unsigned char td_flag;
        unsigned char ta[8];
        unsigned char tb[8];
        unsigned char tc[8];
        unsigned char td[8];
        unsigned char hbytes[15];
        unsigned char tck;
};
```

15.2.3 APDU Request Structure

ST_ APDU_REQ

```
typedef struct
{
    Unsigned char Cmd[4];        /*CLA, INS, P1, P2*/
    int LC;                      /* The valid length of DataIn that sending to
ICC */
    unsigned char DataIn[512];   /* The data that sending to ICC */
    int LE;                      /* The expected returned length */
}ST_ APDU_REQ;
```

ST_APDU_REQ structure:

1. LE is the length of expected returned data. The actual returned data length is related to specific command. Here is an expected length but the actual returned data length will be obtained by LenOut.

2. LE and LC are used in combination as follows:

- LC=0, LE=0. There are neither sending data nor return data.
- LC=0, LE>0. No sending data, but expecting return data. If the length of expected return data is unknown, set Le to 256; otherwise, set it to certain value.
- LC>0, LE=0. The data are sent, but no expected data are returned;
- LC>0, LE>0. The data are sent and the expected data are returned. If the length of expected return data is unknown, set Le to 256; otherwise, set it to certain value.

15.2.4 APDU Response Structure

ST_ APDU_RSP:

```
typedef struct
{
    Int LenOut;                  /* The actual returned data length */
    unsigned char DataOut[512];  /* Returned data pointer from ICC   */
    unsigned char SWA;           /*status word 1 of ICC */
    unsigned char SWB;           /* status word 2 of ICC */
}ST_ APDU_RSP;
```

15.3 API index

15.3.1 sci_open

Prototype	**int sci_open(int id);**	
Function	Opens the corresponding smartcard device.	
Parameters	id【Input】	device id
Return	0	opened successfully
	others	failed to open, return an error code
Instruction	Other functions can be operated only after open device successfully.	

15.3.2 sci_get_fd

Prototype	**int sci_get_dcb(int id);**	
Function	Get the corresponding smartcard device's fd.	
Parameters	id【Input】	Device id, 0-user slot, 1, 2, 3, 4, sam1-sam4.
Return	>=0	device’s id
	others	device not open or return a error code
Instruction	When open a smartcard device, fd is stored in sci_logical_devices[id].fp, get it by this function.	

15.3.3 sci_get_dcb

Prototype	**int sci_get_dcb(int id, struct sci_dcb_t *dcb)**	
Function	Gets the device control block from the device driver layer.	
Parameters	id【Input】	device id
	dcb【Output】	device control block
Return	0	success
	others	error
Instruction		

15.3.4 sci_set_dcb

Prototype	**int sci_set_dcb(int id, struct sci_dcb_t *dcb);**	
Function	Sets the device control block to the device driver layer.	
Parameters	id【Input】	device id
	dcb【Output】	device control block
Return	0	success
	others	error
Instruction		

15.3.5 sci_read

Prototype	**int sci_read(int id, unsigned char *pbuf, int length);**	
Function	Reads bytes from the device driver layer.	
Parameters	id【Input】	device id
	pbuf【Output】	data buffer
	length【Input】	data length

Return	0	success
	others	error
Instruction		

15.3.6 sci_write

Prototype	**int sci_write(int id, unsigned char *pbuf, int length);**	
Function	Writes bytes into the device driver layer.	
Parameters	id【Input】	device id
	pbuf【Iutput】	data buffer
	length【Input】	data length
Return	0	success
	others	error
Instruction		

15.3.7 sci_close

Prototype	**int sci_close(int id);**	
Function	Closes the corresponding smartcard device.	
Parameters	id【Input】	device id
Return	0	success
	others	error
Instruction	This function should be called to close device while program exit.	

15.3.8 sci_lock

Prototype	**int sci_lock(int id);**	
Function	Locks the smartcard lock on the corresponding smartcard device.	
Parameters	id【Input】	device id

Return	0	success
	others	error
Instruction	On smartcard devices, lock the smartcard lock before cold reset, warm reset, reading, writing. It has no effect on usercard device (id = 0).	

15.3.9 sci_unlock

Prototype	**int sci_unlock(int id);**	
Function	Unlocks the smartcard lock on the corresponding smartcard device.	
Parameters	id【Input】	device id
Return	0	success
	others	error
Instruction	On smartcard devices, unlock the smartcard lock while getting all the data (or error info) from transmission by read operation. It has no effect on usercard device (id = 0).	

15.3.10 sci_powerup

Prototype	**int sci_powerup(int id);**	
Function	Card activation on the corresponding smartcard device.	
Parameters	id【Input】	device id
Return	0	success
	others	error
Instruction		

15.3.11 sci_powerdown

Prototype	**int sci_ powerdown (int id);**	
Function	Card deactivation on the corresponding smartcard device.	
Parameters	id【Input】	device id
Return	0	success

	others	error
Instruction		

15.3.12 sci_detect

Prototype	**int sci_detect(int id);**	
Function	Detects whether the user card is in the socket or not.	
Parameters	id【Input】	device id
Return	0	success
	others	error
Instruction	Only id = 0 is accepted.	

15.3.13 sci_cold_reset

Prototype	**int sci_cold_reset(int id, struct sci_atr_t *pstATR);**	
Function	Performs a cold reset sequence and receives the ATR data.	
Parameters	id【Input】	device id
	pstATR【Output】	pointer to ATR data
Return	0	success
	others	error
Instruction		

15.3.14 sci_warm_reset

Prototype	**int sci_warm_reset (int id, struct sci_atr_t *pstATR);**	
Function	Performs a warm reset sequence and receives the ATR data.	
Parameters	id【Input】	device id
	pstATR【Output】	pointer to ATR data

Return	0	success
	others	error
Instruction		

15.4 Protocol processing function

15.4.1 emv_atr_parse

Prototype	**int emv_atr_parse (const struct sci_atr_t *pstATR, struct sci_dcb_t *dcb);**	
Function	Parses the ATR characters according to EMV v4.2 standard.	
Parameters	pstATR【Input】	ATR point
	dcb【Output】	device control block
Return	0	success
	others	error
Instruction		

15.4.2 iso7816_atr_parse

Prototype	**int iso7816_atr_parse (const struct sci_atr_t *pstATR, struct sci_dcb_t *dcb);**	
Function	Parses the ATR characters according to ISO7816 standard.	
Parameters	pstATR【Input】	ATR point
	dcb【Output】	device control block
Return	0	success
	others	error
Instruction		

15.4.3iso7816_pps

Prototype	**int iso7816_pps(int id, struct sci_atr_t *pstATR, struct sci_dcb_t *dcb);**	
Function	PPS protocol process.	
Parameters	id【Input】	device id
	pstATR【Input】	ATR point
	dcb【Output】	device control block
Return	0	success
	others	error
Instruction		

15.4.4iso7816_ocs

Prototype	**int iso7816_ocs(int id, struct sci_atr_t *pstATR);**	
Function	Select operating conditions.	
Parameters	id【Input】	device id
	pstATR【Output】	pointer to ATR data
Return	0	success
	others	card class selection abort
Instruction	Auto class selection, try 1.8V, then 3V, then 5V. It can be invoked as a cold reset with auto vcc selection.	

15.4.5iso7816_t1_ifsd_request

Prototype	**int iso7816_t1_ifsd_request(int id);**	
Function	ifsd request in T=1.	
Parameters	id【Input】	device id
Return	0	success

	others	error
Instruction		

15.4.6 iso7816_t0_exchange

Prototype	**int iso7816_t0_exchange(int id, ST_APDU_REQ *apdu_req, ST_APDU_RSP *apdu_resp);**	
Function	Transmission on T=0 protocol under ISO 7816-3 standard.	
Parameters	id【Input】	device id
	apdu_req【Input】	pointer to APDU request, terminal request information
	apdu_resp【Output】	pointer to APDU response, card response information
Return	0	success
	1	success with a warning
	others	Error
Instruction		

15.4.7 iso7816_t1_exchange

Prototype	**int iso7816_t1_exchange(int id, ST_APDU_REQ *apdu_req, ST_APDU_RSP *apdu_resp);**	
Function	Transmission on T=1 protocol under ISO 7816-3 standard.	
Parameters	id【Input】	Device id
	apdu_req【Input】	Pointer to APDU request, terminal request information.
	apdu_resp【Output】	Pointer to APDU response, card response information.
Return	0	success

	others	Error
Instruction		

15.5 Encapsulated Interfaces

15.5.1 OsIccOpen

Prototype	**int OsIccOpen(int Slot);**	
Function	Open the ICC reader.	
Parameters	Slot	channel number: ICC_USER_SLOT User card ICC_SAM1_SLOT SAM slot 1 ICC_SAM2_SLOT SAM slot 2 ICC_SAM3_SLOT SAM slot 3 ICC_SAM4_SLOT SAM slot 4
Return	RET_OK	Succeed
	ERR_DEV_NOT_EXIST	Device does not exist.
	ERR_DEV_BUSY	Device is busy.
Instruction		

1. Other functions can be operated only after open IC device successfully.
2. For various machines, the numbers and types of slots are different. For specify numbering of slots, please read manual or consult professional staff.

15.5.2 OsIccDetect

Prototype	**int OsIccDetect(int Slot);**	
Function	Check whether there is a card in the specified slot.	
Parameters	Slot	channel number:
Return	RET_OK	Card-inserted
	Others	Please refer to ICC Return code list
Instruction	1. This function will return immediately no matter whether there is a card in slot or not. 2. For USER_SLOT, if card-insert or card-extract happens, system will send MSG_ICCSIG message to the application which used to open the device. 3. It is not applicable to SAM card.	

For SAM card, please make sure call this interface firstly before reset the SAM card. This interface will lead to the SAM card power off.

15.5.3 OsIccInit

Prototype	**int OsIccInit(int Slot, unsigned long Option, unsigned char *Atr);**	
Function	Initialize the IC card device.	
Parameters	Slot	Channel number. Please refer to OslccOpen()
	Option	**(Bit 0~1)card voltage options:** 00 - 5V, 01 - 1.8V, 10 - 3V, 11 - 7V **(Bit 2)Support for PPS protocol:** 0 – not support, 1 – support; **(Bit 3~4)Rate used in ATR** 00 – Standard rate 9600 01 – Twice rate 19200 10 – Four times rate 38400 The rate mentioned here is a reference value which the cards are operated under the typical frequency (3.57MHz). The communication rate between the IC card and the reader component is closely related to the working clock frequency which was provided to card by a specific machine. **(Bit 5)Specification** 0 – EMV 1 - ISO7816 If the specific is marked as EMV mode, the power rate will be marked as invalid. It uses the standard rate by default. **(Bit 6 ~31)Reserved** Option is set as 0 by default(that is 5V, do not support PPS, Standard rate, and follow EMVx)
	*Atr 【Output】	1. Answer To Reset. Up to 34 bytes will be returned from card. 2. Content is composed of length of ATR (1 byte) and ATR[output]
Return	RET_OK	Succeed
	Others	Please refer to ICC Return code list

Instruction	1. ATR output buffer should be allocated at least 34 bytes. 2. Whether PPS protocol is available or not, it depends on the specific cards is support or not. 3. For SAM card, some terminals can only make one card effective at a time. If multi cards need to be operated, it should initialize cards one by one.(OsIccInit ( )) - > operation (OsIccExchange) - > close ( )

Most of the SAM card only supports ISO7816, so the Option should be set as 0x20, but not 0x00.

15.5.4 OsIccExchange

Prototype	**int OsIccExchange(int Slot, int CtrlFlag, cosnt ST_APDU_REQ *ApduReq, ST_APDU_RSP *ApduRsp);**	
Function	Interacts command with IC card.	
Parameters	Slot	Channel number. Please refer to OslccOpen()
	CtrlFlag	1. Bit0 represents whether to send "Get Response" instruction automatically under T=0 protocol. 1 Yes 0 No 2. Bit1~Bit31 Reserved
	ApduReq 【Input】	The data structure which was sent to IC card.
	ApduRsp 【Output】	The data structure which was received from IC card.
Return	RET_OK	Succeed
	Others	Please refer to ICC Return code list
Instruction		

15.5.5 OsIccClose

Prototype	**int OsIccClose(int Slot);**	
Function	Close the IC card device.	
Parameters	Slot	Channel number. Please refer to OslccOpen()
Return	RET_OK	Succeed
	Others	Please refer to ICC Return code list
Instruction		

{ This page intentionally left blank }

16RF Reader

This part conforms to the ISO14443 and the specification of 'EMV Contactless Book D V2.1' and mainly describes the application programming interfaces of RF reader.

16.1 Return Code List

Macro	Value	Description
PCD_ERR_PAR_FLAG	-2901	Parity error
PCD_ERR_CRC_FLAG	-2902	CRC error
PCD_ERR_WTO_FLAG	-2903	Timeout or no card
PCD_ERR_COLL_FLAG	-2904	Multi-card conflict
PCD_ERR_ECD_FLAG	-2905	Frame format is wrong
PCD_ERR_EMD_FLAG	-2906	Interference
PCD_ERR_COM_FLAG	-2907	Chip error, it cannot communicate properly
PCD_ERR_AUT_FLAG	-2908	M1 authentication error
PCD_ERR_TRANSMIT_FLAG	-2909	Transmit error
PCD_ERR_PROTOCOL_FLAG	-2910	Protocol error

PCD_ERR_PARAMFILE_FLAG	-2911	Configuration file does not exist
PCD_ERR_USER_CANCEL	-2912	Users cancel the transaction
PCD_ERR_CARRIER_OBTAIN_FLAG	-2913	Didn't obtain the carrier
PCD_ERR_CONFIG_FLAG	-2914	Configuration register occurs error
PCD_ERR_NOT_ALLOWED_FLAG	-2951	Parameter error or the value is invalid
PCD_CHIP_ABNORMAL	-2952	Chip is abnormal or does not exist
PCD_CHIP_NOT_OPENED	-2953	Module is not open
PCD_CHIP_CARDEXIST	-2954	Card does not remove
PCD_ERR_NOT_IDLE_FLAG	-2955	Card is out of the idle state
PCD_ERR_NOT_POLLING_FLAG	-2956	Card without POLLING
PCD_ERR_NOT_WAKEUP_FLAG	-2957	Card does not wakeup
PCD_ERR_NOT_ACTIVE_FLAG	-2958	Card is not activated

16.2 Data Structure

About request structure and response structure please refer to the section 15.2.3 and 15.2.4.

16.2.1 User Configuration Structure

PCD_USER_ST

```
typedef struct pcd_user_t{
unsigned char wait_retry_limit_w;      /* Written enable for the number of   S(WTX) response*/
unsigned int wait_retry_limit_val;     /* The most repeat times of S(WTX) response */
unsigned char check_cancel_key_w;      /*Written enable for checking the cancel key*/
unsigned char check_cancel_key_val;    /* 0 represents has no response to the cancel key, 1 represents response to
```

```
                                    the cancel key */
int (*check_cancel_key_function)(void);/* Detect whether has pressed the cancel
                                    key or not, if set the
                                    check_cancel_key_w=1 and
                                    check_cancel_key_val=1, the
                                    check_cancel_key_function will be called
                                    during the RF card transaction, when it
                                    returns 0, it represents that does not
                                    pressed the cancel key, otherwise, it
                                    means it does, and then it will be forced to
                                    exit the transaction */
unsigned char reserved[60];    /* Reserved byte, for future expansion */
} PCD_USER_ST;
```

16.2.2 Configuration Parameter Definition

struct PCD_PARAM_ST

```
    /* Card protocol check enable switch,1- check;0 – do not check */
    unsigned int uiProtocolCheckEn;
    /* the maximum block length that the card received .(unit: byte) */
    unsigned int uiFSC;
    /*The longest time that waiting for card response, the unit is according to
    the current ETU time */
    unsigned int uiFWT;
    /*the protection time of sending, the unit is according to the current ETU
  time */
    unsigned int uiSFGT;

    /* Electrical conductivity of the A card */
    unsigned int uiTypeAConduct;
    /*Reserved*/
    unsigned int uiReserved;
    /* Receiver sensitivity of the A card */
    unsigned int uiTypeARxThreshold;
  /* Antenna gain of A card */
  unsigned int uiTypeAGain;

  /* Electrical conductivity of the B card */
    unsigned int uiTypeBConduct;
  /* Modulation depth of B card*/
```

```
unsigned int uiTypeBModulDepth;
/* Receiver sensitivity of the B card */
unsigned int uiTypeBRxThreshold;
/* Antenna gain of B card */
unsigned int uiTypeBGain;

/* Electrical conductivity of the Felica card */
unsigned int uiFelicaConduct;
/* Modulation depth of Felica card */
unsigned int uiFelicaModulDepth;
/* Receiver sensitivity of the Felica card */
unsigned int uiFelicaRxThreshold;
/* Antenna gain of Felica card */
unsigned int uiFelicaGain;

/*Reserved for future use. */
unsigned int uiRFU[60];
};
```

16.3 ISO14443 --- Type A

16.3.1 iso14443_3a_req

Prototype	**int iso14443_3a_req(unsigned char *atqa);**	
Function	Sends REQA command to PICC, and receives the ATQA from PICC.	
Parameters	atqa【Output】	The buffer for ATQA, 2 bytes
Return	0	Success, the ATQA is valid, consists of two bytes.
	others	Error
Instruction	<EMV Contactless Book D - Contactless Comm Protocol 2.1,section 5.3.2>	

16.3.2 iso14443_3a_wup

Prototype	**int iso14443_3a_wup(unsigned char* atqa) ;**	
Function	Sends WUPA command to PICC, and receives the ATQA from PICC.	
Parameters	atqa【Output】	the buffer for ATQA, 2 bytes
Return	0	Success, the ATQA is valid, consists of two bytes.

	others	Error
Instruction	<EMV Contactless Book D - Contactless Comm Protocol 2.1,section 5.3.2>	

16.3.3iso14443_3a_antisel

Prototype	**int iso14443_3a_antisel(unsigned char *uid, int uid_ln, unsigned char *sak);**	
Function	Sends ANTICOLLISION command to PICC, and receives the UID from PICC.	
Parameters	uid	the unique number of PICC
	uid_ln	the length of the unique number of PICC
	sak	the last selected command response
Return	0	Success.
	others	Error
Instruction	• < EMV Contactless Book D - Contactless Comm Protocol 2.1, section 5.4.2> • < ISO/IEC 14443-3:2001(E) Section 6.4.3.1 and 6.4.4 >	

16.3.4iso14443_3a_halt

Prototype	**int iso14443_3a_halt( );**	
Function	Sends HALT command to PICC.	
Parameters	None	
Return	0	Success.
	Others	Error.
Instruction		

16.3.5iso14443_3a_rats

Prototype	**int iso14443_3a_rats(unsigned char* ats);**	
Function	Requests answer to selection.(defined by iso14443-4)	
Parameters	ats【Output】	the response from PICC (must be greater than 256 bytes)
Return	0	Success.

	others	Error
Instruction		

16.4 ISO14443 --- Type B

16.4.1 iso14443_3b_req

Prototype	**int iso14443_3b_req(unsigned char *atqb);**	
Function	Sends REQB command to PICC, and receives the ATQB from PICC.	
Parameters	atqb【Output】	The buffer for ATQB, 12 or 13 bytes
Return	0	Success, the ATQB is valid; consists of 12 or 13 bytes.
	others	Error
Instruction	<EMV Contactless Book D - Contactless Comm Protocol 2.1, section 5.3.2>	

16.4.2 iso14443_3b_wup

Prototype	**int iso14443_3b_wup(unsigned char* atqb);**	
Function	Sends WUPB command to PICC, and receives the ATQB from PICC.	
Parameters	atqb【Output】	the buffer for ATQB, 12 or 13 bytes
Return	0	Success, the ATQB is valid; consists of 12 or 13 bytes.
	others	Error
Instruction	<EMV Contactless Book D - Contactless Comm Protocol 2.1,section 5.3.2>	

16.4.3 iso14443_3b_attri

Prototype	**int iso14443_3b_attri(const unsigned char *pupi, unsigned char* data, int *txr_ln);**	
Function	PCD sends ATTRIB command to PICC, and receives the SAK from PICC.	
Parameters	pupi	the picc's uid, 4 bytes
	dat【Input&Output】	higher layer INF.(command and response)
	txr_ln	the number of higher layer INF.

Return	0	Success
	others	Error
Instruction		

16.4.4 iso14443_3b_halt

Prototype	**int iso14443_3b_halt( );**	
Function	Sends HALTB command to PICC.	
Parameters	None	
Return	0	Success
	others	Error
Instruction		

16.5 Half-duplex transmission protocol

16.5.1 iso14443_4_transfer

Prototype	**int iso14443_4_transfer(unsigned char *src, int tx_ln, unsigned char *des, int *rx_ln);**	
Function	Implements the half duplex communication protocol with ISO14443-4.	
Parameters	src	The data will be transmitted by PCD.
	tx_ln	The number of transmitted data by PCD.
	des	The data will be transmitted by PICC
	rx_ln	The number of transmitted data by PICC.
Return	0	Success
	others	Error
Instruction		

16.6 Encapsulated Interfaces

16.6.1 OsPiccOpen

Prototype	**int OsPiccOpen(void);**	
Function	Power on the PCD module, make the module into the preparatory work state.	
Parameters	None	
Return	0	Succeed.
	Others	Failed to open device. (Details refer to the return code list.)
Instruction		

16.6.2 OsPiccClose

Prototype	**int OsPiccClose (void);**	
Function	Power off the PCD module.	
Parameters	None	
Return	0	Succeed.
	Others	Failed. (Details refer to the return code list.)
Instruction		

16.6.3 OsPiccResetCarrier

Prototype	**int OsPiccResetCarrier (void);**	
Function	Reset the carrier wave.	
Parameters	None	
Return	0	Succeed.
	Others	Failed. (Details refer to the return code list.)
Instruction	When do the carrier reset operation for RF reader, it will change the card state in the RF field, no matter what the state is, the card will enter the idle state after calling this interface.	

16.6.4 OsPiccPoll

Prototype	**int OsPiccPoll(char* pcPiccType, unsigned char* pucATQx);**	
Function	Detect cards, it only can do the roll polling for card A and card B.	
Parameters	pcPiccType 【Output】	Card types: 'A' - card A 'B' - card B
	pucATQx 【Output】	In response to the WUPA commands, a Picc of

		card A, will return an ATQA with a length of 2 bytes. In response to the WUPB commands, a Picc of card B, will return an ATQB with a length of 12 bytes.
Return	0	Succeed.
	Others	Failed. (Details refer to the return code list.)
Instruction	Mifare card is a special A card, after calling this interface, M card returns as an A card.	

16.6.5 OsPiccAntiSel

Prototype	**int OsPiccAntiSel(const char cPiccType, unsigned char *pucUID, const unsigned char ucATQ0, unsigned char* pucSAK);**	
Function	Do anti-collision and selection operations for cards.	
Parameters	pcPiccType 【Input】	Card types: 'A' - card A 'B' - card B
	pucUID 【Output】	Unique identifier of the card: Card A-- 4, 7 or 10 bytes, the value of the UID shall be a fixed number or a random number which is dynamically generated by the Picc. Card B—4 bytes, the value of the pucUID of Type B card shall be fixed number or a random number which is dynamically generated by the Picc.
	ucATQ0 【Input】	It is unused.
	pucSAK 【Output】	The response data of card while selecting card, it has 1 byte. SAK represents the data that response to the last SELECT command of card A. This parameter is ignored by card B.
Return	0	Succeed.
	Others	Failed. (Details refer to the return code list.)
Instruction	If users want to differentiate between the picc of card A and Mifare card, they can carry out the decision according to the output parameters value of the pucSAK.	

16.6.6 OsPiccActive

Prototype	**int OsPiccActive(const char cPiccType, unsigned char *pucATS);**	
Function	Activate the card.	
Parameters	pcPiccType 【Input】	Card types: 'A' - card A 'B' - card B
	pucRATS 【Output】	The response data: PucRATS represents the data that responsed to ATS command of card A. PucRATS represents the data that response to ATTRIB command of card B.
Return	0	Succeed.
	Others	Failed (Details refer to the return code list.)
Instruction	The output data of PucRATS mainly includes the card frame waiting time, buffer size, maximum frame sizes, start-up frame guard time, etc. For details, see the 'EMV Contactless Book D V2.1' in section 5.7 and 6.4.	

16.6.7 OsPiccTransfer

Prototype	**int OsPiccTransfer(const unsigned char*pucTxBuff, int iTxLen, unsigned char* pucRxBuff, int *piRxLen);**	
Function	Realize the transparent transmission/reception in accordance with the half-duplex communication protocol in the ISO14443-4	
Parameters	pucTxBuff 【Input】	The data buffer to be transmitted.
	iTxLen 【Input】	The byte count of data to be transmitted.
	pucRxBuff 【Output】	Response data buffer of received card.
	piRxLen 【Output】	The length of received card's response data.
Return	0	Succeed.
	Others	Failed (Details refer to the return code list.)
Instruction		

16.6.8 OsPiccRemove

Prototype	**int OsPiccRemove (void);**
Function	In accordance with the EMV mode to remove card.

Parameters	None	
Return	0	Succeed.
	Others	Failed. (Details refer to the return code list.)
Instruction		

16.6.9 OsMifareAuthority

Prototype	**int OsMifareAuthority(unsigned char *uid, unsigned char blk_no, unsigned char group, unsigned char *psw);**		
Function	Verify the Mifare card.		
Parameters	uid	【Input】	Card ID, 4 bytes.
	blk_no	【Input】	Block number
	group	【Input】	Password types, can be value as ‘A’ or ‘B’
	psw	【Input】	Authentication password, 6 bytes.
Return	0		Succeed.
	Others		Failed. (Details refer to the return code list.)
Instruction			

16.6.10 OsMifareOperate

Prototype	**int OsMifareOperate (unsigned char ucOpCode, unsigned char ucSrcBlkNo, unsigned char* pucVal, unsigned char ucDesBlkNo);**		
Function	Do operations of reading and writing block for the specified blocks of Mifare card, and increasing, decreasing or backup the specified data block of Mifare card, and then update it into other specified value block.		
Parameters	ucOpCode	【Input】	Operation types include reading block, writing block, increment, decrement and backup.
	ucSrcBlkNo	【Input】	Specify the visiting block number
	pucVal	【Input/Output】	For read block, it is output. Output the block value. For write block, it is input. The contents which points by pucVal should be written in the specified block. For increase/decrease value, it represents the first address of amount buffer is waiting for

		increasing or decreasing value.
	ucUpdateBlkNo 【Input】	Specify the block number which used to written in the operation result.(while reading and writing block, it is NULL)
Return	0	Succeed.
	Others	Failed. (Details refer to the return code list.)
Instruction		

16.6.11 OsPiccInitFelica

Prototype	**int OsPiccInitFelica(unsigned char ucSpeed, unsigned char ucModInvert);**	
Function	Initialize the configuration for the Felica card.	
Parameters	ucSpeed 【Input】	Set the transmission rate which used to interact with card. 1-424Kbp Others-212Kbps
	ucModInvert 【Input】	Set the FeliCa modulate mode. 1 : forward modulate output; Others: reverse modulate output.
Return	0	Succeed.
Instruction		

16.6.12 OsPiccIsoCommand

Prototype	**int OsPiccIsoCommand(int cid, ST_APDU_REQ *ApduReq, ST_APDU_RSP *ApduRsp);**	
Function	Send the APDU data and receive response in the specified channel.	
Parameters	cid 【Input】	Used for specifying the logical channel number of the card, it values from 0 to 14, currently the value is 0.
	ApduReq 【Input】	The structure sends to PICC card.
	ApduRsp 【Output】	The response structure returns from PICC card.
Return	0	Succeed
	Others	Failed (Details refer to the return code list.)
Instruction		

16.6.13 OsPiccSetUserConfig

Prototype	**int OsPiccSetUserConfig(PCD_USER_ST *pcd_user_config) ;**

Function	Set the user configuration.	
Parameters	pcd_user_config【Input】	User configuration structure
Return	0	Succeed
	Others	Failed (Details refer to the return code list.)
Instruction		

16.6.14 OsPiccGetUserConfig

Prototype	**int OsPiccGetUserConfig(PCD_USER_ST *pcd_user_config);**	
Function	Get the user configuration.	
Parameters	pcd_user_config【Output】	User configuration structure
Return	0	Succeed
	Others	Failed (Details refer to the return code list.)
Instruction		

16.7 Note of touch screen and RF reader programming

It has configured touch screen and RF reader on S300 and S800. When the RF card doing the A/B transaction, application developers should note that touch screen cannot be used during the period. The remove card function should be called after finishing the transaction. When operating Mifare card, it must call OsPiccRemove () or OsPiccClose () at last. When operating Felica card, the RF module should be closed at last.

{ This page intentionally left blank }

17Communication Port

17.1 Data Definition

Table 13 Macro definition list of communication ports

Macro	Value	Description
PORT_COM1	0	UART 1
PORT_COM2	1	UART 2
PORT_COM3	2	UART3
PORT_PINPAD	3	Built-out PinPad
PORT_BT	9	Bluetooth
PORT_USBDEV	11	USB device mode port
PORT_USBHOST	12	USB host mode port

Table 14 Return code list of USB port functions

Macro	Value	Description
USB_ERR_NOT_OPEN	-3403	Channel is not open.
USB_ERR_BUF	-3404	Send buffer error.

USB_ERR_NOT_FREE	-3405	Has no free port.
USB_ERR_NO_CONF	-3411	The device has not completed enumeration and configuration process.
USB_ERR_DISCONN	-3412	The device has been disconnected with the host.
USB_ERR_MEM_SYSTEM	-3413	System memory is abnormal.
USB_ERR_BUSY	-3414	USB system is busy.
USB_ERR_RC_SYSTEM	-3415	The application for system resources is failed.
USB_ERR_DEV_ABSENT	-3416	The device on USB host is absent.
USB_ERR_INVALID	-3417	USB communication state is invalid.

17.2 Communication control

17.2.1 OsPortOpen

Prototype	**int OsPortOpen(int Channel, const char *Attr);**	
Function	Opens communication port and sets communication parameters.	
Parameters	Channel	Please refer to the Macro definition list of communication ports, PORT_COM2 and PORT_PINPAD can be reused but only one port at a time.
	Attr 【Input】	Only when the channel is PORT_COM1/PORT_COM2/PORT_COM3USB/PORT_WL, attr does not work, neither does other type and Attr can be NULL. When the channel is UART port: 1. attr = "9600, 8, n, 1", it represents that the baud rate is 9600bps; 8 data bits; no parity; 1 stop bit. ',' will be used to separate characters. 2. Baud rate: One of 1200, 2400, 4800, 9600, 19200, 38400, 57600,115200

		3. Data bit: 7 or 8; 4. Parity method: o-odd parity; e-even parity; n-no parity 5. Stop bit: 1 or 2
Return	RET_OK	Success.
	ERR_DEV_BUSY	Device is busy.
	ERR_DEV_NOT_EXIST	The port does not exist.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction	Other functions can be operated only after open device successfully.	

1. In Prolin2.4, it defaults to use the USB to start the XCB service. In order to avoid the resource conflict, when application needs to use the USB or serial port, it should call OsRegSetValue("persist.sys.xcb.enable", "0") in mian() to close the XCB service firstly.
2. We can start the XCB service in these ways.
   a) Call the OsRegSetValue("persist.sys.xcb.enable", "1") in the main application;
   b) Select a connection way among COM, USB and Network in TM.

17.2.2 OsPortClose

Prototype	**void OsPortClose(int Channel);**	
Function	Closes the specified port.	
Parameters	Channel	Please refer to the Macro definition list of communication ports .
Return	None	
Instruction	This function should be called to close the device while program exit.	

17.2.3 OsPortSend

Prototype	**int OsPortSend(int Channel,**

	const void *SendBuf, **int SendLen);**	
Function	Sends data to the specified communication port.	
Parameters	Channel	Please Refer to the Macro definition list of communication ports .
	SendBuf【Input】	Buffer of sending data.
	SendLen	Length of sending data.(<=8*1024)
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	Port is not open.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction	1. The buffer size is 8K, when the send data is less than the free space of the buffer, this function will not block and the data will be stored in the send buffer. 2. When calling OsPortClose (), the system will block until the send buffer data has been sent out.	

17.2.4 OsPortRecv

Prototype	**int OsPortRecv(int Channel,** **void *RecvBuf,** **int RecvLen,** **int TimeoutMs);**	
Function	Receives data from specified communication port.	
Parameters	Channel	Please Refer to the Macro definition list of communication ports .
	RecvBuf【Output】	Buffer of receiving data.
	RecvLen	The data length that want to receive. When the length is 0, it means clear the receive buffer.
	TimeoutMs	Receive timeouts 【unit:ms】 (The minimum precision is

		100ms)
Return	>=0	The actual length of receive data.
	ERR_DEV_NOT_OPEN	Port does not open.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction	1. The received data will return immediately when it is equal to the RecvLen. 2. If did not reach the RecvLen, it will wait for timeouts.	

17.2.5 OsPortReset

Prototype	**int OsPortReset (int Channel);**	
Function	Reset the port. This function will clear any buffered data in send and receive buffers of COM port.	
Parameters	Channel【Input】	Please Refer to the Macro definition list of communication ports .
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	Port is not open
	ERR_INVALID_PARAM	Invalid parameter
Instruction		

17.2.6 OsPortCheckTx

Prototype	**int OsPortReset(int Channel);**	
Function	Check the remaining bytes in sending buffer of the specified COM port.	
Parameters	Channel【Input】	Please Refer to the Macro definition list of communication ports .
Return	>=0	The data size remained in the send buffer
	ERR_DEV_NOT_OPEN	Port is not open
	ERR_INVALID_PARAM	Invalid parameter
Instruction		

17.3 POSIX

PROLIN 2.X opens serial POSIX interfaces for advanced applications developers to use.

17.3.1 Open

Opens the uart, and the device name are ttyAMA0, ttyAMA1, ttyAMA2, ttyAMA3.

```
int fd;
/* Open the uart with read-write access mode */
fd = open("/dev/ttyAMA1", O_RDWR);
if(-1 == fd){
    perror("Open uart error!");
}
```

17.3.2 Read

Read data from communication port.

char buff[1024];

int Len = 1024;

int readByte = read(fd, buff, Len);

17.3.3 Write

Write data into the communication port. (send)

char buffer[1024];

int Length = 1024;

int nByte;

nByte = write (fd, buffer, Length);

17.3.4 Close

Close communication port.

close (fd);

17.3.5 Query the buffer data of communication port

```
int remain;
int count;
/* Inquiry the number of bytes remained in send buffer */
ioctl(fd, TIOCOUTQ, &remain);
/* Inquiry the number of bytes which remained in receive buffer */
ioctl(fd, TIOCINQ, &count);
```

17.3.6 Clear the buffer data of communication port

```
/* clear the buffer data */
tcflush(fd, TCIOFLUSH);
```

17.3.7 Set the configuration parameters of communication port

```
/* Set the baud rate, data bits, parity bits and stop bits of uart*/
int SetTermios (int fd, int nSpeed, int nBits, char cEvent, int nStop)
{
   struct termios newtio, oldtio;

   /* Get configurations of the original uart */
   if (tcgetattr (fd, &oldtio) != 0)
      {
         printf("Get serial error\n");
         return -1;
      }

   /* Initialize the variable of new configuration */
   bzero (&newtio, sizeof (newtio));
   newtio.c_cflag |= CLOCAL | CREAD;
   newtio.c_cflag &= ~CSIZE;

   /* set the data bits */
   switch (nBits)
      {
      case 7:
         newtio.c_cflag |= CS7;
         break;
      case 8:
         newtio.c_cflag |= CS8;
```

```
        break;
    }

/* Configurate the parity bit*/
switch (cEvent)
    {
    case 'o':
        newtio.c_cflag |= PARENB;
        newtio.c_cflag |= PARODD;
        newtio.c_iflag |= (INPCK | ISTRIP);
        break;
    case 'e':
        newtio.c_iflag |= (INPCK | ISTRIP);
        newtio.c_cflag |= PARENB;
        newtio.c_cflag &= ~PARODD;
        break;
    case 'n':
        newtio.c_cflag &= ~PARENB;
        break;
    }

/* Set the baud rate*/
switch (nSpeed)
    {
    case 1200:
        cfsetispeed (&newtio, B1200);
        cfsetospeed (&newtio, B1200);
    case 2400:
        cfsetispeed (&newtio, B2400);
        cfsetospeed (&newtio, B2400);
        break;
    case 4800:
        cfsetispeed (&newtio, B4800);
        cfsetospeed (&newtio, B4800);
        break;
    case 9600:
        cfsetispeed (&newtio, B9600);
        cfsetospeed (&newtio, B9600);
        break;
    case 19200:
        cfsetispeed (&newtio, B19200);
```

```
            cfsetospeed (&newtio, B19200);
            break;
        case 38400:
            cfsetispeed (&newtio, B38400);
            cfsetospeed (&newtio, B38400);
            break;
        case 57600:
            cfsetispeed (&newtio, B57600);
            cfsetospeed (&newtio, B57600);
            break;
        case 115200:
            cfsetispeed (&newtio, B115200);
            cfsetospeed (&newtio, B115200);
            break;
        default:
            printf ("Not support the speed %d\n", nSpeed);
            cfsetispeed (&newtio, B9600);
            cfsetospeed (&newtio, B9600);
            return -1;
        }

    /* set the stop bits */
    if (nStop == 1)
        newtio.c_cflag &= ~CSTOPB;
    else if (nStop == 2)
        newtio.c_cflag |= CSTOPB;

    /* Set the waiting time and the minimum number of characters , there is no
specific request for waiting time and receive characters ,and it can be set to 0
*/
    newtio.c_cc[VTIME] = 0;
    newtio.c_cc[VMIN] = 0;

    /* Clear the send buffer*/
    tcflush (fd, TCIFLUSH);

    /* Set the new configuration message */
    if ((tcsetattr (fd, TCSANOW, &newtio)) != 0)
        {
            printf("Set serial error\n");
            return -1;
```

```
        }
    return 0;
}
```

18MODEM

In PROLIN 2.X, it can use the built-in UART to send AT commands to Modem and implement the Modem communication functions; at the same time, it can encapsulate some Modem communication interfaces for the developers to use.

18.1 Return code list

Table 15 Modem return code list

Macro	Value	Description
MODEM_CONNECTING	10	Dialing
MODEM_CONNECTED	0	Connected
MODEM_HAVE_DIALED	6	Start sending numbers (only from automatically sending mode to manually answering mode)
MODEM_RECV_POOL_HAVE_DATA	8	Receive buffer is not empty (received remote data)
MODEM_RECVDATA_SEND_IS_FULL	9	Receive buffer is not empty, the send buffer is full.
MODEM_SEND_POOL_FULL	1	Send buffer is full. (In OsModemCheck (), the full status of send buffer

		represents that the modem is using the send buffer, at this time, the OsModemSend () cannot be used.)
MODEM_IDLE	11	Idle
ERR_MDM_TXOVER	-3100	Sending buffer full.
ERR_MDM_BYPASS_BUSY	-3101	The paralleled line is busy. The hardware of NGFP S800 has no side telephone port, and also no such return value.
ERR_MDM_LINE_BUSY	-3102	Telephone line is not properly connected, or parallel line is occupied.
ERR_MDM_NO_CARRIER	-3103	Carrier wave of telephone lost. (Built synchronization chain failure)
ERR_MDM_NO_ANSWER	-3104	No response for dialing.
ERR_MDM_CALLEE_BUSY	-3105	Line busy.
ERR_MDM_NO_LINE	-3106	Telephone line is not connected (Line voltage is 0).
ERR_MDM_CMD_BUF_FULL	-3108	The excommand () buffer is full.
ERR_MDM_CMD_TOO_LONG	-3109	Command of excommand () is too long, exceeded 100.
ERR_MDM_CMD_NOT_SUPPORT	-3110	Excommand () does not support the command.
OTHERS	-3XXX (-3111 ~ -3199)	Abnormal error code will not appear frequently. Setting abnormal error code is for the purpose of maturity and maintainability. Details about what error code means are not important.
	-3115	Calling synchronization handshake receiving process 1 error
	-3116	Calling synchronization handshake

	receiving data package error.
-3117	Calling synchronization handshake receiving package type error.
-3118	Calling synchronization handshake receiving process 2 error
-3119	Calling synchronous communication receiving process 1 error
-3120	Calling synchronous communication chip hang up
-3121	Calling synchronous communication receiving the packet series number error
-3122	Calling synchronous communication receiving process 2 error
-3123	Calling synchronous communication sent overload
-3124	Calling synchronous communication sent under run
-3130	Calling synchronous communication line rate is illegal.
-3131	Calling synchronous communication send stateful packet 1 errors
-3132	Calling synchronous communication sent data packets retry more than the specified time.
-3133	Calling synchronous communication sent data packets timeout
-3134	Calling synchronous communication receiving the acknowledgement packet retry more than the specified time
-3135	Calling synchronous communication sent stateful packet 2 error
-3136	Calling synchronous communication sent stateful packet 3 error
-3137	Calling synchronous communication

	sent stateful packet 4 error
-3138	Calling synchronous communication receiving data packets retry more than the specified time
-3139	Calling synchronous communication sent stateful packet 5 error
-3140	Calling synchronous communication sent stateful packet 6 error
-3144	Sent number automatically and not to pick up the phone timely.
-3145	Called synchronization handshake sent handshake packets failed
-3146	Called synchronization handshake receiving handshake packets failed
-3147	Called synchronization handshake more than the specified time.
-3148	Called synchronous communication sent stateful packet 1 error
-3149	Called synchronous communication receiving process 1 error
-3150	Called synchronous communication chip hang up
-3151	Called synchronous communication receiving process 2 error
-3152	Called synchronous communication receiving retry more than the specified time
-3153	Called synchronous communication sent stateful packet 2 error
-3154	Called synchronous communication sent data packet error
-3155	Called synchronous communication receiving process 3 error
-3156	Called synchronous communication receiving the packet retry more than the specified time
-3157	Called synchronous communication sent

	stateful packet 3 error
-3160	Called connection receiving ring information error
-3161	Called connection detecting the line voltage failed
-3162	Called connection detecting the line voltage data format error
-3163	Called connection voltage is less than the threshold
-3164	Called connection timeout
-3165	Called asynchronous line rate format is incorrect
-3166	Called asynchronous line rate is illegal.
-3167	Called connection information format is incorrectly.
-3170	Called connection set the instruction string 1 failed
-3171	Called connection set the instruction string 2 failed
-3172	Called connection set the extended instruction string failed
-3175	Calling connection set instruction string 1 failed.
-3176	Calling connection set instruction string 2 failed.
-3177	Calling connection set instruction string 3 failed.
-3178	Calling connection set instruction string 4 failed.
-3180	Calling connection set instruction string 5 failed.
-3181	Calling connection asynchronous line rate is illegal
-3182	Calling connection set instruction string 6 failed.

-3183	Calling connection set extended instruction string failed
-3185	Calling connection has no dial tone.
-3186	Calling connection chip indicate an error.
-3187	Calling connection detect the digital lines.
-3188	Calling connection has no dial tone and the voltage is too low.
-3189	Calling connection has other exception errors.
-3192	Non-pre-dial-up dial up timeout (300s)
-3193	When FSK sends data, the DCD signal timeout
-3194	When FSK sends data, the CTS signal timeout
-3195	FSK sends data timeout.
-3196	Called synchronous communication sent data packet format error
-3197	Asynchronous communication does not support the ConnectFormat parameters
-3198	Daemon to create thread failed
-3199	The process with Daemon communication failure or error
-3200	Modem is using the bound uart.
-3201	Socket creation failed
-3202	Socket link failed
-3203	Socket send failed
-3204	Create semaphore failed
-3205	Set the semaphore value failed
-3206	Semaphore has been pre-empted.
-3207	Semaphore cannot be released.
-3208	Semaphor initialization failed
-3209	Gettimeofday failed

	-3210	More than 2 links are using the modem daemon
	-3211	Received the cancel button in the dial-up process.
	-3212	The request of receiving data is rejected. (Receive buffer is empty.)
	-3213	The command string 7 of calling connection Setting is failed.
	-3214	The command string 8 of calling connection Setting is failed.
	-3215	FSK sending is overtime, but still has data in send buffer.
	-3216	Invalid data length (len=0 or len>2048), will not send data.
ERR_MDM_INIT	-3217	Modem initialization failed.
	-3218	If does not implement OsModemConnect (), or implemented OsModemConnect () wrongly, then implement OsPppomLogin () or OsPppomCheck ().
	-3219	The Modem or ModemPPP is being used, Modem cannot be power off.
	-3220	Modem does not power on.

18.2 Data structure

ST_MODEM_SETUP:

```
typedef struct {

    int CallMode;
    int CommMode;
    int CodeType;
```

```
    int CodeDuration;
    int CodeSpacing;
    int DetectLineVoltage;
    int DetectDialTone;
    int DialToneTimeout;
    int CommaPauseTime;
    char ConnectRate[20];
    char ConnectFormat[20];
    int ConnectTimeout;
    int DialTimes;
    int IdleTimeout;
    int Pppom;

    int Reserved[9];
}ST_MODEM_SETUP;
```

Table 16 Variable definition of ST_MODEM_SETUP

CallMode	MODEM_PRE_DIAL	Caller pre-dial
	MODEM_DIAL	Caller dial
	MODEM_WAIT_CALL	Called/Answered the call
CommMode	MODEM_COMM_SYNC	synchronous
	MODEM_COMM_ASYNC	asynchronous
CodeType	MODEM_CODE_DTMF	DTMF (Dual Tone Multi Frequency) dialing
	MODEM_CODE_PULSE1	Pulse dialing 1 (Pulse rate 10/s; Intermittent proportion 1.6:1;Signal interval >=500ms)
	MODEM_CODE_PUL	Pulse dialing 2 (Pulse rate 10/s; Intermittent

	SE2	proportion 2:1; Signal interval >=600ms)
	Other values	Reserved
CodeDuration	The duration of two-tone dialing a single number (Unit:10ms,valid range 5~25)	
CodeSpacing	The interval time between two numbers of two-tone dial-up. It cannot be set to 93011 chips, and it is not applicable to S800. (Unit:10ms, valid range 5~25)	
DetectLineVoltage	TRUE	Detect the parallel telephone occupation (Caller dialing, No assigned number switch to manual answer mode)
	FALSE	Not detected the parallel telephone occupation (Caller dialing, No assigned number switch to manual answer mode)
DetectDialTone	TRUE	Dial tone detection. Refer to the instruction of DailTone Timeout.
	FALSE	Does not detect dial tone. If the 8th bit of DetectDialTone is 1(0x80), while set is called, it will not postback in 8s and the drive will send 15 to client, or the drive will postback 15 to client.
DialToneTimeout	Dial tone detection: The longest time to wait for the dial tone. Exit waiting when the dial tone has been detected during this time. Dial tone not detected: The waiting time for dial tone when off /hook. Unit: 100ms, Minimum and default value is 20, valid range is 20~50. In both cases, it starts the timer since offhook about 450 to 500 ms.	
CommaPauseTime	“,”wait time when dial outside line (Unit: 100ms). This value will be set up according to the actual application environment. It is better to keep interface of manually setting in the application. (Range is 0~255. The range is not applicable to S800) The valid range of S800 is 1~26s (Because of the modem patch, it is inconsistent with the Datasheet)	
ConnectRate[20]	The rate of connection and communication(Expressed as a string): “1200”//1200 bps fast connect	

	“1200,V22”//1200 bps normal connect “1200,V23C”//1200 bps for V.23C(FSK) “1200,B202”//1200 bps for Bell 202(FSK) “2400,FC”//2400 bps fast connect “2400”//2400 bps normal connect “4800”//4800 bps “7200”//7200 bps “9600”//9600 bps “12000”//12000 bps “14400”//14400 bps “19200”//19200 bps “24000”//24000 bps “26400”//26400 bps “28800”//28800 bps “31200”//31200 bps “33600”//33600 bps “48000”//48000 bps “56000”//56000 bps For null string "\ 0"and synchronous communication, the system will select “1200” by default. For asynchronous communication, the system will by default select the maximum rate that the chip can support. S800 supports the baud rate. Asynchronous: “1200”//1200 bps

	“1200,V23C”//1200 bps for V.23C(FSK) “1200,B202”//1200 bps for Bell 202(FSK) “2400”//2400 bps “4800”//4800 bps “7200”//7200 bps “9600”//9600 bps “12000”//12000 bps “14400”//14400 bps “19200”//19200 bps “24000”//24000 bps “26400”//26400 bps “28800”//28800 bps “31200”//31200 bps “33600”/33600 bps “48000”//48000 bps “56000”//56000 bps Synchronous: “1200”//1200 bps “1200,V22”//1200 bps normal connect “2400,FC”//2400 bps fast connect “2400”//2400 bps “9600”//9600 bps
ConnectFormat[20	Format of connection and communication(Expressed as a string):

	“8, n, 1” “8, e, 1” “8, o, 1” “7, e, 1” “7, o, 1” For null string "\ 0", the system will select “8, n, 1” by default. For Synchronous communication, the system will select “8, n, 1” automatically.
ConnectTimeout	Timeouts of waiting for connection,【unit: s】, (valid range 0~300)
DialTimes	The total number of dial-up cycle (convert 0 to 1 if it is 0). Dialing all the numbers in a dial number string is one cycle (valid range is 1~255).
IdleTimeout	There is no application-layer data exchange in the specified time. MODEM then will hang up. Unit 10s, no timeout if it is 0. The maximum timeout is 900s.This value is invalid to ModemPPP.
Pppom	TRUE Modem PPP communication FALSE Common communication
Reserved[9]	Reserved.

18.3 OsModemOpen

Prototype	**int OsModemOpen(void);**	
Function	Switches on the Modem device.	
Parameters	None	
Return	RET_OK	Success
	ERR_DEV_NOT_EXIST	Device does not exist.
	ERR_DEV_BUSY	Device is busy.
	ERR_NO_PORT	No communication port.
Instruction	Other functions can be operated only after open device successfully. It should call the OsModemSwitchPower () firstly, before using the Modem.	

18.4 OsModemClose

Prototype	**void OsModemClose(void);**	
Function	Switches off the Modem device.	
Parameters	None	
Return	None	
Instruction	This function should be called to close device while program exit.	

18.5 OsModemSwitchPower

Prototype	**int OsModemSwitchPower(int OnOff);**	
Function	Manages the Modem Power.	
Parameters	int OnOff	OnOff=1, power on, OnOff=0, power off.
Return	RET_OK	Success
	-3219	The Modem or ModemPPP is being used, Modem cannot be power off.
	-3220	Modem does not power on.
Instruction	1. It should power on the Modem before using Modem or ModemPPP. 2. This function is independent of the interface functions in Modem module. 3. It will not automatically perform OsModemClose () when Modem module is power off.	

18.6 OsModemConnect

Prototype	**int OsModemConnect(const ST_MODEM_SETUP *Setup, const unsigned char *TelNo);**	
Function	Sets the communication link function, for both calling and being called.	
Parameters	Setup【Input】	Modem parameter, while mdm_setup==NULL, default dialing parameter will be used. Default dialing mode includes: Synchronous, 1200, DTMF, connection timeout for 10 seconds, idle hang up for 60 seconds.

	TelNo【Input】	Telephone number.
Return	RET_OK	Success
	ERR_MDM_BYPASS_BUSY	The paralleled line is busy.
	ERR_MDM_LINE_BUSY	Telephone line is not properly connected, or parallel line is occupied.
	ERR_MDM_NO_ANSWER	No response for dialing.
	ERR_MDM_CALLEE_BUSY	Line is busy.
	ERR_MDM_NO_LINE	Telephone line is not connected (Line voltage is 0).
	ERR_MDM_NO_CARRIER	Carrier wave of telephone lost.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_MDM_CANCEL_KEY_DOWN	Press CANCEL key while dialing.
	ERR_DEV_NOT_OPEN	Device is not open.
Instruction	1. The function can also be used to set Modem mode as being called. 2. Telephone icon will be controlled by a program. It shows hang-up icon while connecting; it shows pickup icon when communication is established, or during switching time of pre-dial after hangup. 3. It needs to call the OsModemCheck () to query the result of pre-dial. 4. It must call OsModemConnect () before using the ModemPPP, sets ST_MODEM_SETUP.Pppom=1, and it doesn't need to call OsModemOpen (), then calls OsPppomLogin ().	

Meanings of telephone symbols:
0-9,*, #,A~D — Telephone numbers
, — Dialing delay
; — Transmitting next telephone number
. — End of numbers, which is used to keep connected with

application after sending numbers

.. — End of extension numbers, which is used to switch to manual receiving after sending numbers.

18.7 OsModemCheck

Prototype	**int OsModemCheck(void);**	
Function	Checks the result of the last Modem dialing.	
Parameters	None	
Return	MODEM_CONNECTING	Dialing
	MODEM_CONNECTED	Connected
	MODEM_IDLE	Idle
	ERR_MDM_BYPASS_BUSY	The parallel line is busy.
	ERR_MDM_LINE_BUSY	Telephone line is not properly connected, or parallel line is occupied.
	ERR_MDM_NO_ANSWER	No response for dialing.
	ERR_MDM_CALLEE_BUSY	Line busy.
	ERR_MDM_NO_LINE	Telephone line is not connected (Line voltage is 0).
	ERR_MDM_NO_CARRIER	Carrier wave of telephone lost.
	ERR_MDM_RECVPOOL_NOT_EMPTY	Receive buffer is not empty (received remote data)
	ERR_MDM_RECVPOOL_SENDPOOL_BOTH_NOT_EMPTY	Receive buffer is not empty (received remote data), and the send buffer is sending data.

	ERR_DEV_NOT_OPEN	Device is not open.
Instruction	1. This function can be used to check whether communication has been established or not by the redial. 2. After calling OsModemOpen (), OsModemHangup () or OsModemClose (), the status of the last Modem dial will become: MODEM_IDLE.	

18.8 OsModemExCmd

Prototype	**int OsModemExCmd(const char *Cmd, char *Rsp, int *RespLen, int TimeoutMs);**	
Function	Sets additional AT control command for OsModemConnect (), to control Modem dialing.	
Parameters	Cmd【Input】	Input AT control command.
	Rsp【Output】	Contents of response data.
	RespLen【Output】	Length of response data.
	TimeoutMs	Waiting time for response.(unit:ms)
Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_MDM_CMD_BUF_FULL	Command buffer overflow.
	ERR_MDM_CMD_TOO_LONG	Command is too long.
	ERR_MDM_CMD_NOT_SUPPORT	Does not support the command. (command that begin with AT,S3,S7,WT=)

	ERR_DEV_NOT_OPEN	Device is not open.
Instruction	1. The function is needed to be called before OsModemConnect (), and it is only valid for the entire process of OsModemConnect (). 2. While the function is executing, it will automatically hang up current dialing or communication process. 3. The function can be called 100 times continuously. If it is more than 100, the exceeding callings will be discarded and will return, failed.	

1. Maximum 100 bytes of string can be inputted for each calling. If it is more than 100 bytes, the entire control command will be discarded, and will return, failed.
2. Every input of control command has to be AT control command, which should be supported by this Modem chip. Otherwise, it will lead to OsModemConnect ( ) failure.

18.9 OsModemHangup

Prototype	**void OsModemHangup(void);**	
Function	Hangs up Modem or terminates Modem dialing.	
Parameters	None	
Return	None	
Instruction	If dialing the number again, right after hanging up, the Modem will wait and start redialing after 3 seconds, in order to allow PABX finish hanging up and transmitting dialing tone.	

18.10OsModemSend

Prototype	**int OsModemSend(const void *SendBuf, int SendLen);**	
Function	Sends packets out by Modem.	
Parameters	SendBuf【Input】	Pointer of packets, which will be sent
	SendLen	Length of packets, which will be sent (bytes)
Return	RET_OK	Success
	ERR_NOT_CO	Not connected.

	NNECT	
	ERR_MDM_TX OVER	Send buffer is full.
	ERR_MDM_NO _CARRIER	No carrier waves.(Disconnected)
	ERR_INVALID _PARAM	Invalid parameter
	ERR_DEV_NOT _OPEN	Device is not open.
Instruction	It can send 2048 bytes each time at most. Receiving and sending data of synchronous called are up to 2053 bytes, because there are more than 5 control characters.	

18.11OsModemRecv

Prototype	**int OsModemRecv(void *RecvBuf, int BufSize, int Timeout);**	
Function	Receives packets by MODEM.	
Parameters	RecvBuf【Output】	Pointer of the packets that have been received. [Buffer size can be defined according to the requirements of different cases.]
	BufSize	Size of RecvBuf (<=2048bytes)
	Timeout	Timeouts【ms】
Return	>= 0	The actual number of receiving data.
	ERR_NOT_CONNECT	Not connected.
	ERR_MDM_NO_CARRIER	No carrier waves. (Disconnected)
	ERR_INVALID_PARAM	Invalid parameter
	ERR_DEV_NOT_OPEN	Device is not open.
Instruction	1. It can send 2048 bytes each time at most. Receiving and sending data of	

	synchronous called are up to 2053 bytes, because there are more than 5 control characters. 2. If the size of actual data is not larger than the specified size of receive buffer, it will return immediately. 3. While receiving data, if there's a line error, it will immediately return the corresponding error code. 4. For SDLC synchronous communication, it will immediately return after receiving a packet. (even if the received packet length is less than the BufSize) 5. For asynchronous communication, it will immediately return after receiving byte data of BufSize, or wait until the timeout. 6. For synchronous receiving, it will receive a complete frame each time, and it does not affected by restriction of BufSize. 7. For FSK, the timeout does not work.

18.12OsPppomLogin

Prototype	**int OsPppomLogin(const char *Name const char *Password, long Auth, int TimeOutMs);**	
Function	Establishes the Modem PPP network link.	
Parameters	Name【Input】	User name; Length<=50 bytes; It cannot be NULL. For the 96169 background of China Telecom, it can enter any user name, and it must enter a character at least.
	Password【Input】	Password; Length<=50 bytes;; It cannot be NULL. For the 96169 background of China Telecom, it can enter any password, and it must enter a character at least.
	Auth	Authentication Algorithms;

		Currently the following Authentication algorithms are supported PPP_ALG_PAP 0x1 PAP Authentication Algorithm PPP_ALG_CHAP 0x2 CHAP Authentication Algorithm PPP_ALG_MSCHAPV1 0x4 MSCHAPV1 Authentication Algorithm PPP_ALG_MSCHAPV2 0x8 MSCHAPV2 Authentication Algorithm PPP_ALG_ALL 0xff (all Authentication Algorithms are supported) At least one type of authentication algorithm has to be selected, more than one authentication algorithm will also be allowed by using (+) or (|), for example, PPP_ALG_PAP | PPP_ALG_CHAP. If the algorithm is unknown, fill with PPP_ALG_ALL.
	TimeOutMs	Timeouts【ms】; The valid range is 0~3600000; if it is <0, it will automatically set it to 0, if more than 360000. It will automatically set it to 3600000.
Return	PPP_LOGINING	Being processed.
	RET_OK	The link established successfully.
	ERR_INVALID_PARAM	Invalid parameter
	ERR_NET_PASSWD	wrong password
	ERR_NET_SERVER_BUSY	Server is busy, communication failed.
Instruction	1. Before using this function, it should call OsModemConnect () first and set the ST_MODEM_SETUP.Pppom as 1. also it doesn't need to call OsModemOpen(); 2. While TimeOutMs=0 means return immediately, 3. Call OsPppomCheck ( ) to check the link status 4. The login time may vary from settings to settings of ST_MODEM_SETUP parameters. The modem chip of S800 supports up	

	to 33600 asynchronous baud rate, dial-up while the setting is less than or equal to 33600, there will be a low re-training rate and high success rate. 5. For the 96169 background (Guidway A8010), if a re-training is occurred and the time period after sending number is more than 20 seconds, then the background communication will no longer be according to ppp protocol, which will end up in failure. 6. After the link sets up successfully, it can communicate through the IP network communication function. 7. In the process of dialing, when users want to hang up by pressing the cancel button, the methods of operation are as follows: Application porting a thread and take the key, if it is the cancel key, perform OsPppomLogin ("a", "a", 1, -1), the first 3 parameters should be filled in accordance with the requirements, and the fourth parameter must be set to -1, then ModemPPP will hang up and automatically logout.

18.13OsPppomCheck

Prototype	**int OsPppomCheck(void);**	
Function	Checks the link status of Modem network.	
Parameters	None	
Return	PPP_LOGOUTING	The link is in the status of disconnection.
	PPP_LOGINING	Being processed.
	RET_OK	The link established successfully
	ERR_NET_PASSWD	wrong password
	ERR_NET_LOGOUT	Has been calling the OsPppomLogout ( ) to disconnect the link.
	ERR_NET_IF	Link has been disconnected.
Instruction		

18.14OsPppomLogout

Prototype	**int OsPppomLogout(void);**	
Function	Exits network, disconnect the Modem PPP link.	
Parameters	None	
Return	RET_OK	Success
Instruction	1.	

{ This page intentionally left blank }

19Network Communication

PROLIN 2.X uses the unified TCP/IP to manage different physical connections. For the network communications programming, it provides a standard socket programming (socket).

19.1TCP programming

```
/* Server code in C */

   #include <sys/types.h>
   #include <sys/socket.h>
   #include <netinet/in.h>
   #include <arpa/inet.h>
   #include <stdio.h>
   #include <stdlib.h>
   #include <string.h>
   #include <unistd.h>

   int main(void)
   {
      struct sockaddr_in stSockAddr;
      int SocketFD = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
```

```
    if(-1 == SocketFD)
    {
       perror("cannot create socket");
       exit(EXIT_FAILURE);
    }

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

    stSockAddr.sin_family = AF_INET;
    stSockAddr.sin_port = htons(1100);
    stSockAddr.sin_addr.s_addr = INADDR_ANY;

    if(-1 == bind(SocketFD,(struct sockaddr *)&stSockAddr,
sizeof(stSockAddr)))
    {
       perror("error bind failed");
       close(SocketFD);
       exit(EXIT_FAILURE);
    }

    if(-1 == listen(SocketFD, 10))
    {
       perror("error listen failed");
       close(SocketFD);
       exit(EXIT_FAILURE);
    }

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

       if(0 > ConnectFD)
       {
          perror("error accept failed");
          close(SocketFD);
          exit(EXIT_FAILURE);
       }

       /* perform read write operations ...
       read(ConnectFD,buff,size)*/
```

```
        if (-1 == shutdown(ConnectFD, SHUT_RDWR))
        {
           perror("cannot shutdown socket");
           close(ConnectFD);
           exit(EXIT_FAILURE);
        }
        close(ConnectFD);
     }
return EXIT_SUCCESS;
}

   /* Client code in C */

   #include <sys/types.h>
   #include <sys/socket.h>
   #include <netinet/in.h>
   #include <arpa/inet.h>
   #include <stdio.h>
   #include <stdlib.h>
   #include <string.h>
   #include <unistd.h>

   int main(void)
   {
      struct sockaddr_in stSockAddr;
      int Res;
      int SocketFD = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);

      if (-1 == SocketFD)
      {
         perror("cannot create socket");
         exit(EXIT_FAILURE);
      }

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

      stSockAddr.sin_family = AF_INET;
      stSockAddr.sin_port = htons(1100);
      Res = inet_pton(AF_INET, "192.168.1.3", &stSockAddr.sin_addr);
```

```
    if (0 > Res)
    {
      perror("error: first parameter is not a valid address family");
      close(SocketFD);
      exit(EXIT_FAILURE);
    }
    else if (0 == Res)
    {
      perror("char string (second parameter does not contain valid
ipaddress)");
      close(SocketFD);
      exit(EXIT_FAILURE);
    }

    if (-1 == connect(SocketFD, (struct sockaddr *)&stSockAddr,
sizeof(stSockAddr)))
    {
      perror("connect failed");
      close(SocketFD);
      exit(EXIT_FAILURE);

    }

    /* perform read write operations ... */

    shutdown(SocketFD, SHUT_RDWR);

    close(SocketFD);
    return EXIT_SUCCESS;
  }
```

19.2 UDP programming

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
```

```
#include <unistd.h> /* for close() for socket */
#include <stdlib.h>

int main(void)
{
   int sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);
   struct sockaddr_in sa;
   char buffer[1024];
   ssize_t recsize;
   socklen_t fromlen;

   memset(&sa, 0, sizeof sa);
   sa.sin_family = AF_INET;
   sa.sin_addr.s_addr = INADDR_ANY;
   sa.sin_port = htons(7654);
   fromlen = sizeof(sa);

   if (-1 == bind(sock,(struct sockaddr *)&sa, sizeof(sa)))
   {
      perror("error bind failed");
      close(sock);
      exit(EXIT_FAILURE);
   }

   for (;;)
   {
      printf ("recv test....\n");
      recsize = recvfrom(sock, (void *)buffer, sizeof(buffer), 0, (struct sockaddr
*)&sa, &fromlen);
      if (recsize < 0) {
         fprintf(stderr, "%s\n", strerror(errno));
         exit(EXIT_FAILURE);
      }
      printf("recsize: %z\n ", recsize);
      sleep(1);
      printf("datagram: %.*s\n", (int)recsize, buffer);
   }
}

#include <stdlib.h>
#include <stdio.h>
```

```
#include <errno.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <unistd.h>
#include <arpa/inet.h>

int main(int argc, char *argv[])
{
   int sock;
   struct sockaddr_in sa;
   int bytes_sent;
   char buffer[200];

   strcpy(buffer, "hello world!");

   sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);
   if (-1 == sock) /* if socket failed to initialize, exit */
      {
         printf("Error Creating Socket");
         exit(EXIT_FAILURE);
      }

   memset(&sa, 0, sizeof sa);
   sa.sin_family = AF_INET;
   sa.sin_addr.s_addr = inet_addr("127.0.0.1");
   sa.sin_port = htons(7654);

   bytes_sent = sendto(sock, buffer, strlen(buffer), 0,(struct sockaddr*)&sa,
sizeof sa);
   if (bytes_sent < 0) {
      printf("Error sending packet: %s\n", strerror(errno));
      exit(EXIT_FAILURE);
   }

   close(sock); /* close the socket */
   return 0;
}
```

20Network Configuration

PROLIN 2.X provides the interface for network configuration, including ARP settings, Ping services, DNS configuration, network card configuration, DHCP service, routing settings and PPPoE service.

20.1 Return code list

Table 17 Network Configuration return code list

Macro	Value	Description
ERR_NET_SERVER_BAD	-3301	IP server error.
ERR_NET_SERVER_BUSY	-3302	IP server is busy; it can only provide service for 5 applications at a time.
ERR_NET_NO_ROUTE	-3305	Set no route.
ERR_NET_FULL	-3306	Connection is full; the application can set up to 20 connections at a time.
ERR_NET_IF	-3307	Network interface link is unavailable (The link has not set up or there is no corresponding device.)
ERR_NET_SESS_BROKEN	-3308	TCP / UDP session connection has been broken and is unavailable

ERR_NET_PASSWD	-3309	Password is incorrect.
ERR_NET_LOGOUT	-3310	Application logout initiatively.
ERR_NET_INIT	-3311	Initialization failed.
ERR_NET_DHCP_DISCOVER	-3312	Has not found DHCP Server.
ERR_NET_DHCP_OFFER	-3313	DHCP cannot assign the IP address.
ERR_NET_DHCP_START	-3314	DHCP not started.
ERR_NET_DNS	-3315	DNS cannot analyse the corresponding domain.
ERR_NET_SERV_USING	-3316	The port specified by the server is in use.
ERR_NET_NODNServer	-3317	Has not configured the domain name server
ERR_NET_LINKDOWN	-3318	Link is disconnected by the server.
ERR_NET_CONN	-3319	Cannot connect to the specified server.
ERR_NET_TIMEOUT	-3320	Connection timeout
ERR_NET_PPP	-3321	PPP connection error
ERR_NET_SERV	-3322	Has not found the PPPoE server
ERR_NET_AGAIN	-3323	The request resource wasn't found, please try again.
ERR_NET_AUTH	-3324	Has no authority to access to the RADIUS server.
NET_DOING	1	Some related operations are being done.(such as PPP login, DHCP)

20.2 Data Definition

20.2.1 Physical channel list

Macro	Description
NET_LINK_ETH	Ethernet

NET_LINK_WL	Wireless, including GPRS, CDMA, TDSCDMA
NET_LINK_WIFI	WiFi(Wireless Local Area Network)
NET_LINK_PPPOE	ADSL link
NET_LINK_MODEM	Modem PPP link

All of the macro values, list in the above table are positive integers, and are greater than 0.

Details refer to osal.h.

20.3 Network Configuration interface

20.3.1 OsNetAddArp

Prototype	**int OsNetAddArp(int NetLink,** **const char *Addr,** **const char MAC[6]);**	
Function	Adds the IP address to MAC address mapping table, which is static.	
Parameters	NetLink	Physical channel can only configuration with Ethernet and WiFi; • NET_LINK_ETH Ethernet; • NET_LINK_WIFI WiFi(Wireless Local Area Network)
	Addr【Input】	IP address. The format is "XXX.XXX.XXX.XXX", and XXX ranges from 0 to 255. For example: "192.168.0.1"; Cannot be NULL.
	MAC【Input】	the MAC address corresponding to the IP address; Cannot be NULL, the space has 6 bytes.
Return	RET_OK	Success
	<0	Failed, the value is error code.

Instruction	

1. Static ARP table is used to resist the attack from ARP Cheat,
2. If there is no static ARP table, the system will be dynamically obtained, and it doesn't need to be configured by the application.

20.3.2 OsNetPing

Prototype	**int OsNetPing(const char *Addr, int TimeOutMs);**	
Function	Pings a specific machine to check whether the network is running smoothly or not.	
Parameters	Addr【Input】	The IP address of destination machine; Format is "XXX.XXX.XXX.XXX", XXX represents 0~255, for example:"192.168.0.3";
	TimeOutMs	Timeout【unit:ms】, the valid range is 3000~3600000.
Return	RET_OK	Success
	ERR_NET_IF	Link is unavailable, means the link has not been set correctly (or has been disconnected).
	ERR_INVALID_PARAM	Invalid parameter
	ERR_TIME_OUT	Timeout
Instruction	Before calling OsNetPing( ), it should call OsNetSetRoute( ) to set the physical channel first.	

20.3.3 OsNetDns

Prototype	**int OsNetDns(const char *Name, char *Addr, int TimeOutMs);**
Function	Analyzes the IP address corresponding to the domain name and stores the

	results in Addr parameter.	
Parameters	Name【Input】	Information of domain name, e.g.: [www.sina.com.cn]. The maximum length cannot be NULL and cannot exceed 255 characters.
	Addr【Output】	1. It is an output parameter; 2. Use to store the IP address, mapped by the domain name, the format is "XXX.XXX.XXX.XXX", XXX represents 0~255, e.g:"192.168.0.3"; 3. It cannot be NULL; there are at least 16 bytes in the space.
	TimeOutMs	Timeout【unit:ms】, the valid range is 1000~3600000
Return	RET_OK	Success
	ERR_NET_IF	The Link is unavailable, means the link has not been set correctly (or has been disconnected).
	ERR_INVALID_PARAM	Invalid parameter
	ERR_TIME_OUT	Timeout.
	ERR_NET_DNS	The domain server cannot analyze the corresponding domain, or it does not exist.
Instruction	Before calling OsNetDns(), it should call OsNetSetRoute() to set the physical channel first.	

20.3.4 OsNetSetConfig

Prototype	**int OsNetSetConfig(int NetLink, const char *Addr, const char *Mask, const char *Gateway, const char *DNSServer);**	
Function	Configures the network information.	
Parameters	NetLink	Physical channel can only configure with Ethernet and WiFi; • NET_LINK_ETH Ethernet;

		• NET_LINK_WIFI WiFi(Wireless Local Area Network)
	Addr【Input】	the address used by POS Format is "XXX.XXX.XXX.XXX", XXX represents 0~255; e.g: "192.168.0.3"; if it is " " or NULL, means do not change the original configuration;
	Mask【Input】	Mask code used by POS; Format is "XXX.XXX.XXX.XXX"; e.g.: "255.255.255.0"; if it is " " or NULL, means do not change the original configuration;
	Gateway【Input】	Address of the default gateway Format is "XXX.XXX.XXX.XXX", XXX represents 0~255; e.g.: "192.168.0.1"; if it is " " or NULL, means do not change the original configuration;
	DNSServer【Input】	DNS server address Format is "XXX.XXX.XXX.XXX", XXX represents 0~255; e.g.: "192.168.0.1"; if it is " " or NULL, means do not change the original configuration;
Return	RET_OK	Success
	ERR_NET_IF	The link is unavailable, means the link has not been set correctly (or has been disconnected) or the WIFI module does not exist.
	ERR_INVALID	Invalid parameter.

	_PARAM
Instruction	1. After calling successfully, the system will stop the DHCP function. 2. The function does not check the matching relationship between Addr, Mask and Gateway; it only checks the legality of their formats. 3. Wireless link, PPPoE, and ModemPPP can only be dynamically allocated and cannot be configured by this interface.

20.3.5 OsNetGetConfig

Prototype	**int OsNetGetConfig(int NetLink, char *Addr, char *Mask, char *Gateway, char *DNSServer);**	
Function	Gets the network configuration information, such as IP, subnet mask, gateway and DNS.	
Parameters	NetLink	Physical channel can only configure with Ethernet and WiFi; • NET_LINK_ETH Ethernet; • NET_LINK_WIFI WiFi(Wireless Local Area Network)
	Addr【Output】	The address use by POS is an Output parameter. There are at least 16 bytes in the space. The format is "XXX.XXX.XXX.XXX", XXX represents 0~255; e.g.: "192.168.0.3"; It can be NULL.
	Mask【Output】	Mask code used by POS is an Output parameter. There are at least 16 bytes in the space. Format is "XXX.XXX.XXX.XXX"; e.g.: "255.255.255.0"; It can be NULL.
	Gateway【Output】	Address of the default gateway is an Output parameter. There are at least 16 bytes in the space. Format is "XXX.XXX.XXX.XXX", XXX represents

		0~255; e.g.: “192.168.0.1”; It can be NULL.
	DNSServer 【Input】	DNS server address is an Output parameter. There are at least 16 bytes in the space. Format is “XXX.XXX.XXX.XXX”, XXX represents 0~255; e.g.: “192.168.0.1”; It can be NULL.
Return	RET_OK	Success
	ERR_NET_IF	The link is unavailable, means the link has not been set correctly (or has been disconnected) or the WIFI module does not exist.
Instruction		

Addr, Mask, Gateway and DNSServer return the string as ‘’ that means it has not been set.

20.3.6 OsNetStartDhcp

Prototype	**int OsNetStartDhcp(int NetLink);**	
Function	Starts the DHCP function to obtain a dynamically assigned address.	
Parameters	NetLink	Physical channel can only configure with Ethernet and WiFi; • NET_LINK_ETH Ethernet; • NET_LINK_WIFI WiFi(Wireless Local Area Network)
Return	RET_OK	Success
	ERR_NET_IF	Has not configured Ethernet or WiFi.
Instruction	This interface is only used to start the DHCP function, does not wait to assign addresses, but it can check whether the address distribution is completed by calling the OsNetCheckDhcp () or not.	

Before starting the DHCP, it should close all the connections because these connections may not be able to communicate properly in the subsequent activity.

20.3.7 OsNetCheckDhcp

Prototype	**int OsNetCheckDhcp(int NetLink);**	
Function	Check DHCP status.	
Parameters	NetLink	Physical channel can only configure with Ethernet and WiFi; • NET_LINK_ETH Ethernet; • NET_LINK_WIFI WiFi(Wireless Local Area Network)
Return	NET_DOING	DHCP is doing the dynamic allocation.
	RET_OK	Dynamic allocation has been done.
	ERR_NET_DHCP_DISCOVER	DHCPServer has not been found.
	ERR_NET_DHCP_OFFER	DHCP cannot assign the IP address.
	ERR_NET_DHCP_START	DHCP does not start.
Instruction		

20.3.8 OsNetStopDhcp

Prototype	**void OsNetStopDhcp(int NetLink);**	
Function	Stops DHCP function.	
Parameters	NetLink	Physical channel can only configure with Ethernet and WiFi; • NET_LINK_ETH Ethernet; • NET_LINK_WIFI WiFi(Wireless Local Area Network)
Return	None	
Instruction	1. After the DHCP function stops, the application needs to re-configure the network using OsNetSetConfig (); 2. After the success of DHCP, the system will regularly update the configuration information, when the update fails, it will cause the network	

	become unavailable.

20.3.9 OsNetSetRoute

Prototype	**int OsNetSetRoute(int NetLink);**	
Function	Sets the physical channel which used for connection.	
Parameters	NetLink	Physical channel, please refer to Physical channel list.
Return	RET_OK	Success, the new link came into effect.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction	1. When the application starts, it must call this function to establish the TCP / UDP connection; 2. While setting the physical channel, the system does not check the availability of link. It will check only when the TCP connection is established. 3. After setting the physical channel, it is only effective to the new connection. The old ones continue to use the old channel.	

20.3.10 OsNetGetRoute

Prototype	**int OsNetGetRoute(void);**	
Function	Reads the physical channel which is currently used by the system.	
Parameters	None	
Return	>0	Physical channel, please refer to Physical channel list .
	<0	Has not initialized the settings.
Instruction	1. After system initialization, the user needs to call OsNetSetRoute (), otherwise OsNetGetRoute() will return <0; 2. After calling OsNetSetRoute () successfully, the system return values must be NetLink that set by OsNetSetRoute ().	

20.3.11 OsPppoeLogin

Prototype	**int OsPppoeLogin(const char *Name,**

	const char *Password, **int TimeOutMs);**	
Function	PPPoE link Login.	
Parameters	Name【Input】	User name; Cannot exceed 50 bytes Cannot be NULL, if there is no password, use an null string “ “;
	Password【Input】	Password; Cannot exceed 50 bytes Cannot be NULL, if there is no password, use ““;
	TimeOutMs	Timeout,【unit:ms】 The valid range is 0~3600000;
Return	NET_DOING	Logging
	RET_OK	Success
	<0	Failed
Instruction		

20.3.12 OsPppoeCheck

Prototype	**int OsPppoeCheck(void);**	
Function	Checks the PPPoE link.	
Parameters	None	
Return	NET_DOING	Logging
	RET_OK	The link has been successfully established.
	<0	The link has been disconnected.
Instruction		

20.3.13 OsPppoeLogout

Prototype	**voidO sPppoeLogout(void);**

Function	Disconnects the PPPoE link.	
Parameters	None	
Return	None	
Instruction		

21GPRS/CDMA

PROLIN 2.X provides supports for GPRS and CDMA, for the utility and configuration of these wireless modules. It also provides a series of APIs for application developers to use.

21.1 Return code list

Table 18 GPRS/CDMA return code list

Macro	Value	Description
PPP_LOGINING	1	PPP is logining.
PPP_LOGOUTING	2	PPP is logouting.
ERR_WL_POWER_ONING	-3501	Module is power on, please wait.
ERR_WL_POWER_OFF	-3502	Module is power off.
ERR_WL_NOT_INIT	-3503	Has not called OsWlInit () to initialize successfully and cannot work normally.
ERR_WL_NEEDPIN	-3504	Sim card needs PIN.
ERR_WL_RSPERR	-3505	Module response error.
ERR_WL_NORSP	-3506	Module no response.

ERR_WL_NEEDPUK	-3507	Sim card needs PUK.
ERR_WL_WRONG_PIN	-3508	PIN of Sim card error.
ERR_WL_NOSIM	-3509	SIM card not inserted
ERR_WL_NOREG	-3510	Cannot register on the GPRS network.
ERR_WL_AUTO_RST	-3511	Module reset automatically.
ERR_WL_BUF	-3512	Module memory error.
ERR_WL_GET_SIGNAL	-3513	Getting the signal, please wait for 3s.
ERR_WL_NOTYPE	-3514	Module cannot be recognized.
ERR_WL_PPP_ONLINE	-3515	PPP is on line, and it cannot be sleeping.
ERR_WL_ERR_BUSY	-3516	Module is busy.
ERR_WL_SLEEP_ONING	-3517	Module is in sleeping.
ERR_WL_SLEEP_FAIL	-3518	Sleeping failed.
ERR_WL_SIM_FAILURE	-3519	SIM card Operation failed.
ERR_WL_NO_SIMSOCKET	-3520	The machine does not have a SIM card slot.
ERR_WL_ONLY_ONE_SIMSOCKET	-3521	The machine only has one SIM card slot.
ERR_WL_SIMSOCKET_CONFIGFILE	-3522	The ro.fac.simsocket in config file is incorrectly set.

21.2 Wirless module interface

21.2.1 OsWlLock

Prototype	**int OsWlLock(void);**	
Function	Opens the wireless module.	
Parameters	None	
Return	RET_OK	Open successfully.

	ERR_DEV_BUSY	Device is being used by another application.
	ERR_DEV_NOT_EXIST	Device does not exist.
Instruction	1. Before calling OsWlInit (), OsWlPowerSwitch (), OsWlLogin () or OsWlLogout (), it must call this function to open the wireless module first. 2. It must call OsWlUnlock () to close wireless module if does not carry out any operation.	

21.2.2 OsWlUnlock

Prototype	**void OsWlUnlock(void);**	
Function	Close the wireless module.	
Parameters	None	
Return	None	
Instruction		

21.2.3 OsWlInit

Prototype	**int OsWlInit(const char *SimPin);**	
Function	Initializes wireless module.	
Parameters	SimPin【Input】	SIM card PIN, The PIN length can't exceed 50 characters. NULL means it does not need the PIN.
Return	RET_OK	Success
	ERR_DEV_NOT_OPEN	Fail to call WlOpen ().
	ERR_DEV_NOT_EXIST	Wireless module does not exist.
	ERR_NO_PORT	Not enough physical uart.
	ERR_WL_NEEDPIN	SIM card needs PIN.
	ERR_WL_RSPERR	Response error.
	ERR_WL_NORSP	Module has no response.
	ERR_WL_NEEDPUK	SIM card needs PUK.

	ERR_WL_WRONG_PIN	PIN error.
	ERR_WL_NOSIM	No SIM card.
	ERR_WL_NOTYPE	Module cannot be recognized
Instruction	1. Before using this function, be sure to call OsWlLock () successfully. 2. It needs to call this function successfully at system startup time and then call OsWlLogin () to establish the link. 3. SIM card will automatically check the password if it requires. 4. If the module does not power on, the system will automatically supply power for it. 5. When it returns ERR_WL_NOSIM (No SIM card), the application can use some functions without SIM card. 6. After calling OsWlSwitchPower (), it should wait for 15 seconds until the the module is stable, otherwise, it will be failed to perform OsWlInit ().	

21.2.4 OsWlSwitchPower

Prototype	**int OsWlSwitchPower(int OnOff);**		
Function	Powers on /off the wireless module.		
Parameters	OnOff	0	Power off
		1	Power on
Return	RET_OK	Success	
	ERR_DEV_NOT_EXIST	Module does not exist.	
	ERR_DEV_NOT_OPEN	Fail to call OsWlLock ().	
Instruction			

1. The time cost to power on the modules is different.
2. If the power is off, the O/S will shut off power to module.
3. While the power is on, if the wireless module has been detected by the monitor, the system will power on automatically.
4. Please wait for 8 seconds between call off and on.
5. Before power off, it will quit ppp automatically.
6. In the state of power off, it should wait for 15 seconds before initializing module and getting signal. The login connection will be performed after 15 seconds, and it results a long time landing.

21.2.5 OsWlSwitchSleep

Prototype	**int OsWlSwitchSleep(int OnOff);**	
Function	Makes the wireless module sleep or wake up.	
Parameters	OnOff	**0:** Wake up **1 :** Sleep **Others :** Unknown errors.
Return	RET_OK	Success
	ERR_DEV_NOT_EXIST	Module does not exist.
	ERR_DEV_NOT_OPEN	Fail to call OsWlLock( ).
	ERR_WL_PPP_ONLINE	PPP is on line.
Instruction		

If it detects the PPP is on line, it will not be in sleep.

For MG323, when there is no inserted SIM card, or has not activated the PIN, or has not registered to the network, it will not be in sleep.

21.2.6 OsWlGetSignal

Prototype	**int OsWlGetSignal(void);**	
Function	Gets the wireless signal strength.	
Parameters	None	
Return	0~5	Represents the signal strength, the higher the number the stronger the signal, 0 means no signal and 5 represents the strongest signal.
	ERR_DEV_NOT_EXIST	Module does not exist.
	ERR_NO_PORT	Not enough physical uart
	ERR_WL_POWER_ONING	Module is power on.
Instruction	1. It doesn't need to call OsWLock() when use this function; 2. When the wireless link is not established, this API will get the signal values from the module through AT commands; 3. After calling OsWlLogin () and establishing wireless link, if obtained	

	signal value is not the latest, then enter the data mode and if cannot obtain real-time signal, it will return to ERR_WL_RSPERR.

21.2.7 OsWlCheck

Prototype	**int OsWlCheck(void);**	
Function	Checks the status of wireless link.	
Parameters	None	
Return	PPP_LOGOUTING	Link disconnecting.
	PPP_LOGINING	Link establishing.
	RET_OK	Link established successful.
	ERR_DEV_NOT_EXIST	Module does not exist.
	ERR_WL_POWER_ONING	Module is power on.
	ERR_WL_POWER_OFF	Module is power off.
	ERR_WL_NOT_INIT	Initialization failed.
	ERR_NET_PASSWD	Password is incorrect.
	ERR_NET_LOGOUT	OsWlLogout () disconnect the link.
	ERR_NET_IF	Link has been disconnected.
Instruction		

21.2.8 OsWlLogin

Prototype	**int OsWlLogin(const char *APN,** **const char *Name,** **const char *Password,** **long Auth,** **int TimeOutMs,** **int KeepAlive,**

	int ReserParam);	
Function	Logins on the wireless network and set up a wireless link.	
Parameters	APN【Input】	APN – AccessPointName for GPRS communication, dialing number for CDMA. Length <= 50 characters; When pointer points to NULL, the application should dial-up first and the protocol stacks direct login on PPP.
	Name【Input】	User name; Length <= 50 bytes; It cannot be NULL, if there is no user name, replace it with an empty string " ";
	Password【Input】	Password; Length <= 50 characters; It cannot be NULL, if there is no password, replace it with an empty string " ";
	Auth	The supported authentication algorithms that can support: PPP_ALG_PAP 0x00000001 PAP authentication algorithm PPP_ALG_CHAP 0x0000000 CHAP authentication algorithm PPP_ALG_MSCHAPV1 0x00000004 MSCHAPV1 authentication algorithm PPP_ALG_MSCHAPV2 0x00000008 MSCHAPV2 authentication algorithm PPP_ALG_ALL 0xff All algorithms are supported. At least one type of authentication algorithm has to be chosen; more than one authentication algorithm will also be allowed to choose by using (+) or (|), for example, PPP_ALG_PAP | PPP_ALG_CHAP. If the algorithm is unknown, fill it with PPP_ALG_ALL.
	TimeOutMs	Timeout,【unit:ms】; The valid range is 0~3600000; if it is <0, automatically set it to 0, if more than 360000, automatically set it to 3600000.

	KeepAlive	Interval for link check, unit is ms; The valid range is 10000~3600000; If the link is longer than KeepAlive time but without any messages; the system automatically starts the link check interval;
	ReserParam	Reserved parameter, used for extension.
Return	PPP_LOGINING	Processing
	RET_OK	Link set up successfully.
	ERR_DEV_NOT_EXIST	Wireless module does not exist.
	ERR_DEV_NOT_OPEN	Perform OsWlLock() failed
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_WL_POWER_ONING	Module is power on.
	ERR_WL_POWER_OFF	Module is power off.
	ERR_WL_NOT_INIT	Initialized failed.
	ERR_NET_PASSWD	Password is incorrect.
	ERR_NET_SERVER_BUSY	Server is busy, communication failure.
	ERR_NET_AUTH	Has no authority to access to the RADIUS server.
Instruction	1. Before using this function, be sure to call OsWlLock () successfully. 2. When TimeOutMs=0, it means return immediately; 3. Calling OsWlCheck( ) to check the link status; 4. For different modules and different network environments, the login time will be different; if with-in 60 seconds the login did not complete, it means login failure or the module has an exception error. 5. Unsuccessful login for three consecutive times, it indicates the module has no response, the application must call OsWlSwitchPower () to reset the module. 6. After the link set up successfully, it can communicate through the IP network communication function.	

21.2.9 OsWlLoginEx

Prototype	**int OsWlLoginEx(const char *DialNum,** **const char *APN,**

	const char *Name, **const char *Password,** **long Auth,** **int TimeOutMs,** **int KeepAlive,** **int ReserParam);**	
Function	Logins on the wireless network and set up a wireless link.(it supports modifying the dialing instructions)	
Parameters	DialNum【Input】	The PPP dialing instruction. Length <= 50 characters; When it is NULL, it adopts the default instruction of system.
	ANP【Input】	APN – AccessPointName for GPRS communication, dialing number for CDMA. Length <= 50 characters; It cannot be NULL.
	Name【Input】	User name; Length <= 50 bytes; It cannot be NULL, if there is no user name, replace it with an empty string “ ”;
	Password【Input】	Password; Length <= 50 characters; It cannot be NULL, if there is no password, replace it with an empty string “ “;
	Auth	The supported authentication algorithms: PPP_ALG_PAP 0x00000001 PAP authentication algorithm PPP_ALG_CHAP 0x0000002 CHAP authentication algorithm PPP_ALG_MSCHAPV1 0x00000004 MSCHAPV1 authentication algorithm PPP_ALG_MSCHAPV2 0x00000008 MSCHAPV2 authentication algorithm PPP_ALG_ALL 0xff All algorithms are supported. At least one type of authentication algorithm

		has to be chosen; more than one authentication algorithms can be allowed to choose by using (+) or (|), for example, PPP_ALG_PAP | PPP_ALG_CHAP. If the algorithm is unknown, fill it with PPP_ALG_ALL.
	TimeOutMs	Timeout,【unit:ms】; The valid range is 0~3600000; if it is <0, automatically set it to 0, if more than 360000, automatically set it to 3600000.
	KeepAlive	Interval for link check, unit is ms; The valid range is 10000~3600000; If the link is longer than KeepAlive time but without any messages; the system automatically starts the link check interval; (This functionality does not work now.)
	ReserParam	Reserved parameter, used for extension.
Return	PPP_LOGINING	Processing
	RET_OK	Link set up successfully.
	ERR_DEV_NOT_EXIST	Wireless module does not exist.
	ERR_DEV_NOT_OPEN	Perform OsWlLock() failed。
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_WL_POWER_ONING	Module is power on.
	ERR_WL_POWER_OFF	Module is power off.
	ERR_WL_NOT_INIT	Initialized failed.
	ERR_NET_PASSWD	Password is incorrect.
	ERR_NET_SERVER_BUSY	Server is busy, communication failure.
Instruction	1. This function is similar to OsWlLogin (), when DialNum is NULL, the two functions are the same. 2. Before using this function, be sure to call OsWlLock () successfully. 3. When TimeOutMs=0, it means return immediately; 4. Calling OsWlCheck( ) to check the link status; 5. For different modules and different network environments, the login time will be different; if the login did not complete in 60 seconds, it means login failure or the module has an exception error.	

	6. Unsuccessful login for three consecutive times, it indicates the module has no response, the application must call OsWlSwitchPower () to reset the module. 7. After the link set up successfully, it can communicate through the IP network communication function.

21.2.10 OsWlLogout

Prototype	**int OsWlLogout(void);**	
Function	Logouts the wireless network and disconnects the wireless link.	
Parameters	None	
	RET_OK	Disconnect the wireless link successfully.
	ERR_DEV_NOT_OPEN	Fail to call OsWlLock ().
Instruction	Before using this function, be sure to call OsWlLock () successfully.	

21.3 Wireless module information settings

21.3.1 OsWlSelSim

Prototype	**int OsWlSelSim(int simno);**	
Function	Selects Sim card.	
Parameters	simno 【Input】	• 0: Selecting card slot 1 • 1: Selecting card slot 2 • Others:Parameter error
Return	RET_OK	Success
	ERR_DEV_NOT_EXIST	Module does not exist.
	ERR_DEV_NOT_OPEN	Fail to call OsWlLock ().
	ERR_WL_ERR_BUSY	Module is busy.
	Other non-zero value	Refer to the Return code list.
Instruction	1. Before using this function, be sure to call OsWlLock () successfully. 2. After select sim card, the module will be power on/off, and this function blocks about 15 seconds. Application needs to re-calll OsWilInit() to initialize the module. 3. If users select a card slot with bad cards or without any cards, the function will also return successfully. And it can detect the problems while logining.	

22WIFI

Wifi module is added in the NGFP products for the first time, so NGFP Prolin increases some APIs about the WiFi network management.

22.1 Return code list

22.2 Data Definition

Table 19 Definition list of WiFi data

Macro	Value	Description
SZ_SSID_LEN	120	Length of SSID.
SZ_BSSID_LEN	6	Length of AP MAC address
SZ_KEY_LEN	100	Length of WiFi password
WEP_KEY_NUM	4	Numbers of WEP password

Table 20 Macros of WiFi protocol

Macro	Value	Description
AUTH_MODE_OPEN	0x00000001	
AUTH_MODE_SHARED	0x00000002	
GROUP_CIPHERS_CCMP	0x00000001	
GROUP_CIPHERS_TKIP	0x00000002	
GROUP_CIPHERS_WEP40	0x00000004	

GROUP_CIPHERS_WEP104	0x00000008
KEY_MGMT_IEEE8021X	0x00000001
KEY_MGMT_NONE	0x00000002
KEY_MGMT_WPA_EAP	0x00000004
KEY_MGMT_WPA_PSK	0x00000008
PARE_CIPHERS_CCMP	0x00000001
PARE_CIPHERS_NONE	0x00000002
PARE_CIPHERS_TKIP	0x00000004
PROTOCOL_RSN	0x00000001
PROTOCOL_WPA	0x00000002
WIFI_KEY_TYPE_HEX	0x01
WIFI_KEY_TYPE_ASCII	0x02

22.3 Data Structure

22.3.1 WifiScanInfo

ST_WifiScanInfo

```
typedef struct _WifiScanInfo
{
    Char szESSID[SZ_SSID-LEN] ;                    /* AP name */
    Int iNetId;
}ST_ WifiScanInfo;
```

ST_WifiScanInfoList

```
typedef struct _WifiScanInfoList
{
    ST_WifiScanInfo info ;
```

```
    struct _WifiScanInfoList
*pNext;

}ST_ WifiScanInfoList;
```

22.3.2 WifiConfiguration

ST_WifiConfiguration:

```
typedef struct _WifiConfiguration
{
    char intNetID;
    char szBSSID[SZ_BSSID_LEN];
    char szESSID[SZ_SSID_LEN];
    unsigned long AuthMod;
    unsigned long GroupCiphers;
    unsigned long KeyMgmt;
    unsigned long PareCipher;
    unsigned long Protocols;
    char szPreSharedKey[SZ_KEY_LEN];
    charszWepKey[WEP_KEY_NUM][ SZ_KEY_LEN];
    char intWepKeyID;
    char intKeyType;
}ST_ WifiConfiguration;
```

22.3.3 OsWifiOpen

Prototype	**int OsWifiOpen(void);**	
Function	Gets the right to use the WIFI module.	
Parameters	None	
Return	RET_OK	Success

	ERR_DEV_USED	Module has been occupied.
Instruction		

22.3.4 OsWifiClose

Prototype	**int OsWifiClose (void);**	
Function	Releases the right of using the WIFI module.	
Parameters	None	
Return	RET_OK	Success
	ERR_DEV_USED	Module has been occupied.
Instruction		

22.3.5 OsWifiSwitchPower

Prototype	**int OsWifiSwitchPower (int type);**	
Function	Sets WIFI module to the state of enabled power off or sleep.	
Parameters	Type	• 0: Module hardware entered into the active state. • 1: Module hardware entered into the power off state. • 2: Module hardware entered into the sleeping state.
Return	RET_OK	Success
	ERR_DEV_NOT_EXIST	Module does not exist.
	ERR_DEV_USED	WIFI has been occupied.
Instruction		

22.3.6 OsWifiScanAP

Prototype	**int OsWifiScanAP (ST_WifiScanInfoList **pls);**	
Function	Searches the AP.	
Parameters	pls【Output】	Output ESSID of the searched AP with the form of linked list, and logic a network number; If failed, * PLS will be NULL, if successes, it

		should be the pointer value of the linked list head.
Return	RET_OK	Success
	ERR_MEM_FAULT	Memory error.
	ERR_INVALID_PARAM	Parameter error.
	ERR_DEV_NOT_EXIST	WIFI module driver loading abnormal or module error.
	ERR_DEV_USED	WIFI has been occupied.
Instruction		

22.3.7 OsWifiConnectById

Prototype	**int OsWifiConnectById(int netid,** **const char *key,** **int keylen,** **int keyType);**	
Function	Connects to AP with the searched information and password.	
Parameters	netid【Input】	The logic network number of AP
	Key【Input】	Keys that used to connect with the AP.
	Keylen【Input】	Key length.
	KeyType【Input】	Key type: • WIFI_KEY_TYPE_HEX • WIFI_KEY_TYPE_ASCII
Return	RET_OK	Success
	ERR_INVALID_PARAM	Parameter error.
	ERR_DEV_NOT_EXIST	WIFI module driver loading abnormally or module error.
	ERR_DEV_USED	WIFI has been occupied.

Instruction	The latest configuration information is effectively.

22.3.8 OsWifiConnectByPara

Prototype	**int OsWifiConnectByPara (ST_WifiConfiguration config);**	
Function	Uses the configuration parameter to connect with AP.	
Parameters	config【Input】	Configuration parameter.
Return	RET_OK	Success
	ERR_INVALID_PARAM	Parameter error.
	ERR_MEM_FAULT	Memory error.
	ERR_DEV_NOT_EXIST	WIFI module driver loading abnormally or module error.
	ERR_DEV_USED	WIFI has been occupied.
Instruction		

22.3.9 OsWifiCheckStatus

Prototype	**int OsWifiCheckStatus(char *essid, int *sig);**	
Function	Gets the current status of WIFI, it will should return AP essid and signal strength when connected successfully.	
Parameters	essid【Output】	ESSID name
	sig【Output】	Signal strength level: 0-5
Return	RET_OK	Success
	ERR_NOT_CONNECT	Has not connected to AP.
	ERR_INVALID_PARAM	Parameter error.
Instruction	It doesn't need to call OsWlLock () before using this function. But it does need to ensure that the buffer size of essid should not be less than SZ_SSID_LEN.	

23File System

PROLIN 2.X uses the YAFFS2 as Nandflash file management system and supports the FAT file system on the SD card and U disk. It also supports the NFS network file system at the application development stage.

The file system uses the standard ANSI.C API to get the access. For details, refer to <stdio.h>. For real-time operation, it can use the standard POSIX interface to get the access.

24System Information

In Prolin, the system messages generated by the hardware device will be realized by asynchronous notification. The system provides two kinds of hardware system messages, magnetic cards and IC cards.

- SIGMAG
- SIGICC

The SIGMAG is only valid when the magnetic stripe reader device is open.

The code of registered asynchronous notification function is shown as follows:

EXAMPLE

```
void input_handler(int signum)
{
    printf("receive a signal from globalfifo,signalnum:%d\n",signum);
}

main()
{
    int fd, oflags;

    fd = open("/dev/globalfifo", O_RDWR, S_IRUSR | S_IWUSR);
    if (fd !=  - 1)
    {
        // Start the signal driven mechanism
        signal(SIGICC, input_handler);// Let the input_handler() deal with
        SIGICC signal
        fcntl(fd, F_SETOWN, getpid());
        oflags = fcntl(fd, F_GETFL);
        fcntl(fd, F_SETFL, oflags | FASYNC);
```

```
            while (1)
            {
                sleep(100);
            }
        }
}
```

25Audio

The speaker volume of PROLIN 2.X is divided into five levels, ranging from 0 to 4, 0 means mute. In general, the volume setting is unified, and it can set in the TM interface.

25.1 OsPlayWave

Prototype	**int OsPlayWave(const char *Buf, int Len, int Volume, int DurationMs);**	
Function	Play WAV audio files.	
Parameters	Buf 【Input】	The audio data buffer
	Len【Input】	Length of the data buffer
	Volume【Input】	Volume, the values range from 0 to 4, 0 represents using the default volume.
	DurationMs 【Output】	Play duration【Unit:ms】
Return	RET_OK	Success
	ERR_FILE_FORMAT	File format error
	ERR_ACCESS_DENY	Access denied
	ERR_INVALID_PARAM	Invalid parameter
Instruction	When Volume <0, set it as 0; when Volume >4, set is as 4. If the DurationMs is more than the Len play time, it will play on a continuous loop of the file. On the contrary, DurationMs shall prevail. When DurationMs=0, the actual length of the audio data shall prevail.	

26Barcode

26.1 General Definiton

26.2 APIs

26.2.1 OsScanOpen

Property	**int OsScanOpen (void);**	
Function	Open the barcode scanning module.	
Parameter	None	
Return	RET_OK	Success
	ERR_DEV_BUSY	Device has been occupied.
	ERR_DEV_NOT_OPEN	Device is not open.
	ERR_DEV_NOT_EXIST	Device does not exist.
Instruction		

26.2.2 OsScanRead

Property	**int OsScanRead(char *Buf, int Len, int TimeoutMs);**		
Function	Read the barcode.		
Parameter	Buf	【Output】	The buffer used to store the barcode data. One-dimensional code recommends being more than 512 bytes, and two-dimensional code suggests being 3072 bytes.
	Len	【Input】	Buffer length

	TimeoutMs 【Input】	Timeout of reading barcode,【unit:ms】 The valid range is 1500~36000, if it is less than 1500, set is as 1500, and if it is greater than 36000, set it as 36000. The error between the actual timeout value and set value may be no more than 1s, it is recommended to set the timeout value as 3000ms.
Return	>=0	The actual length of the read barcode data.
	ERR_DEV_NOT_OPEN	Device is not open.
	ERR_TIME_OUT	Timeout.
	ERR_INVALID_PARAM	Invalid parameter
Instruction		

26.2.3 OsScanClose

Property	**void OsScanClose (void);**
Function	Close the scanning device.
Parameter	None
Return	None
Instruction	

27Power Management

27.1 OsCheckBattery

Prototype	**int OsCheckBattery(void);**	
Function	Checks the battery.	
Parameters	None	
Return	BATTERY_LEVEL_0	0~20% It needs immediately charge to the battery. At this time it should not do the transaction, wireless communications and printing. When the battery capacity is low, the system will automatically turn off.
	BATTERY_LEVEL_1	20%~40%
	BATTERY_LEVEL_2	40%~60%
	BATTERY_LEVEL_3	60%~80%
	BATTERY_LEVEL_4	80%~100%
	BATTERY_LEVEL_CHARGE	Battery is being charged.
	BATTERY_LEVEL_ABSENT	Battery does not exist. It needs an electric power supply.
	BATTERY_LEVEL_COMPLETE	Battery is fully charged, use the electric power supply.
	ERR_SYS_NOT_SUPPORT	System does not support checking the battery. S800/S300 returns this value.
Instruction	1. When using an electric power supply, it can detect whether the battery is full	

	charged or not, but the battery level only can be detected in the state of battery powered. 2. It is not recommended to call this function during printing, because th e printer requires high current in the printing procedure, otherwise, it will make the interface obtain an inaccurate charge.

27.2 OsSysSleep

Prototype	**int OsSysSleep(void);**	
Function	System hibernates.	
Parameters	None	
Return	RET_OK	Success
	ERR_SYS_NOT_SUPPORT	System does not support this function.
Instruction	It needs to refresh screen after wake up. It is not recommended to hibernate during using RF card, if so, after wake up, it must call OsPiccClose() firstly, and then call OsPiccOpen() and other cards operations.	

Appendix 1 PIN Block Format

Format 0 PIN block

This PIN block is constructed by modulo-2 addition of two 64-bit fields: the plain text PIN field and the account number field. The formats of these fields are described in 1.1.1 and 1.1.2 respectively.

The format 0 PIN block shall be reversibly enciphered when transmitted.

Plain text PIN field

The plain text PIN field shall be formatted as follows.

Bit

1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 64

C	N	P	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

where

C = Control field: shall be binary 0011;

N = PIN length: 4-bit binary number with permissible values of 0100(4) to 1100(12);

P = Pin digit: 4-bit field with permissible values of 0000(zero) to 1001(9);

P/F = PIN/Fill digit: designation of these fields is determined by the PIN length field;

F = Fill digit: 4-bit field value 1111(15).

Account number field

The account number field shall be formatted as follows.

Bit

1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 64

0	0	0	0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12

where

0 = Pad digit: a 4-big field with the only permissible value of 0000(zero);

A1…A12 = Account number: content is the 12 rightmost digits of the primary account number (PAN) excluding the check digit. A12 is the digit immediately preceding the PAN's check digit. If the PAN excluding the check digit is less than 12 digits, the digits are right justified and padded to the left with zeros. Permissible values are 0000 (zero) to 1001 (9).

Format 1 PIN block

This PIN block is constructed by concatenation of two fields: the plain text PIN field and the transaction field.

The format 1 PIN block should be used in situations where the PAN is not available.

The format 1 PIN block shall be reversibly enciphered when transmitted.

The format 1 PIN block shall be formatted as follows.

Bit

1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 64

C	N	P	P	P	P	P/T	P/T	P/T	P/T	P/T	P/T	P/T	P/T	T	T

where

C = Control field: shall be binary 0001;

N = PIN length: 4-bit binary number with permissible values 0100(4) to 1100 (12);

P = PIN digit: 4-bitfield with permissible values 0000 (zero) to 1001 (9);

P/T = PIN/Transaction digit: designation of these fields is determined by the PIN length field;

T = Transaction digit: 4-bit binary number with permissible values of 0000 (zero) to 1111 (15).

The transaction field is a binary number formed by [56-(N*4)] bits. This binary shall be unique (except by chance) for every occurrence of the PIN block and can, for example, be derived from a transaction sequence number, time stamp, random number or similar.

The transaction field should not be transmitted and is not required in order to translate the PIN block to another format since the PIN length is known.

Format 2 PIN block

The format 2 PIN block has been specified for local use with IC cards. The format 2 PIN block shall only be used in an offline environment and shall not be used for online PIN verification.

Format 3 PIN block

Format 3 PIN block construction

The format 3 PIN block is the same as format 0 PIN block except for the fill digits.

This PIN block is constructed by modulo-2 addition of two 64-bit fields: the plain text PIN field and the account number field. The formats of these fields are described in 1.4.2 and 1.4.3 respectively.

Plain text PIN field

The plain text PIN field shall be formatted as follows.

Bit

1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 64

C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

where

C = Control field: shall be binary 0011;

N = PIN number: 4-bit binary number with permissible values of 0100 (4) to 1100 (12);

P = PIN digit: 4-bit field with permissible values of 0000 (zero) to 1001 (9);

P/F = PIN/Fill digit: designation of these fields is determined by the PIN length field;

F = Fill digit: 4-bit field, with values from 1010(10) to 1111(15), where the

Fill-digit values are randomly or sequentially selected from this set of six possible values, such that it is highly unlikely that the identical configuration of fill digits will be used more than once with the same account number field by the same PIN encipherment device.

Account number field

The account number field shall be formatted as follows.

Bit

1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 64

0	0	0	0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12

For more details related to PIN Block format please refer to ISO 9564-1:2002(E).

Appendix 2 EPS PINBLOCK Format

String format: “1234”+ISN [6 byte] +PIN [Byte bit];

If the PIN less than 6 bytes, add ‘0’ before PIN;

The data string will be converted to BCD code, Using TPK to process DES/TDES encryption for BCD code.

Appendix 3 Register Table

The names begin with "ro.fac." can be read-only, for "persist.sys.", it can be both write and read.

System configuration name	Note
ro.fac.hwver	Hardware version number. Main board- Interface board.
ro.fac.mach	Product type name.
ro.fac.conf.ver	Version information of configuration files.
ro.fac.pn	PN number
ro.fac.sn	SN number
ro.fac.eth	Whether there is network cable.(0- does not exist, 1- exist)
ro.fac.usb.host	Whether there is the main device interface.(0- does not exist, 1- exist)
ro.fac.usb.device	Whether there is an USB device interface.(0- does not exist, 1- exist)
ro.fac.usb.otg	Whether there is an USB OTG interface.(0- does not exist, 1- exist)
ro.fac.leddt	Whether there is LED digital tube.(0- does not exist, 1- exist)
ro.fac.keybroad	Key types.(0- have no keys, 1- indicates the presence of physical buttons, 2- indicates the presence of a touch-screen buttons)
ro.fac.buzzer	Whether there is a Buzzer module.(0- does not exist, 1- exist)
ro.fac.simsocket	The number of sim card slot.
ro.fac.battery	Whether there is a battery.(0- does not exist, 1- exist)
ro.fac.wifi	Name of WIFI module.(If none, it means does not exist)
ro.fac.bt	Name of Bluetooth module. (If none, it means does not exist)
ro.fac.radio	Wireless module information, and the parameter information is optional. It needs to isolate when there are multiple wireless modules. (If none, it means does not exist)
ro.fac.modem	Name of Modem module. (If none, it means does not exist)
ro.fac.printer	Name of Printer module. (If none, it means does not exist)
ro.fac.pcd	Name of PCD module. (If none, it means does not exist)
ro.fac.sci	Name of ICC Reader. (If none, it means does not exist)
ro.fac.msr	Name of MSR Reader. (If none, it means does not exist)
ro.fac.videocard	Name of videocard module. (If none, it means does not exist)
ro.fac.audiocard	Name of audiocard module. (If none, it means does not exist)
ro.fac.sensor.gravity	Name of Gravity sensor module. (If none, it means does not exist)
ro.fac.touchscreen	Name of Touch-screen module. (If none, it means does not exist)
ro.fac.sdhc	The specification, capacity range and speed level that support by

	SD card. (If none, it means does not exist)
ro.fac.scanner	Name of Scanner module. (If none, it means does not exist)
ro.fac.pcd.param1	PCD antenna parameter 1. (If none, it needn't to fill in.)
ro.fac.pcd.param2	PCD antenna parameter 2. (If none, it needn't to fill in.)
ro.fac.pcd.param3	PCD antenna parameter 3. (If none, it needn't to fill in.)
ro.fac.lcd.rotate	The LCD clockwise rotation degrees. ("0","90","180","270")
persist.sys.eth0.enable	Supported Ethernet or not. ("true" or " "- support, "false"- does not support)
persist.sys.eth0.dhcp	DHCP is open or not. ("true" - opened, "false" or " "- closed)
persist.sys.eth0.ip	Ethernet ip address
persist.sys.eth0.mask	Ethernet subnet mask
persist.sys.eth0.gateway	Ethernet gateway
persist.sys.dns1	System Preferred DNS
persist.sys.dns2	System alternative DNS
persist.sys.eth0.speed	network port speed.("eth_auto" represents automatic configuration, "eth_10mhd" represents 10M half-duplex, "eth_10mfd" represents 10M full-duplex, "eth_100mhd" represents 100M half-duplex, "eth_100mfd" represents 100M full-duplex)

Appendix 4 Validity of models and contents

According to the differences of the hardware design, some OSAL interfaces cannot take an effect on a certain model. Details refer to the table below.

whether there is Wireless module, Modem module or Ethernet module, it depends on the model configuration.(refer to the POS PN number)

Chapters	**S300**	**S800**	**S900**	**D200**
Thread	√	√	√	√
System Function	√	√	√	√
Encryption and Decryption	√	√	√	√
PED	√	√	√	√
LCD	√	√	√	√
Keyboard	√	√	√	√
Touch Screen	√	NA	√	NA
Signature Board	√	NA	√	NA
Printer	NA	√	√	NA
Font Library	√	√	√	√
Code	√	√	√	√
MSR	√	√	√	√
ICC Reader	√	√	√	√
RF Reader	√	√	√	√
Communicatio	PORT_COM1	PORT_COM1	PORT_COM1	PORT_COM

n Port	PORT_USBDEV PORT_USBHOST	PORT_COM2 PORT_PINPAD PORT_USBDEV PORT_USBHOST	PORT_USBDEV PORT_USBHOST	1 PORT_USBDEV
MODEM	NA	√	NA	NA
Network Communication	√	√	√	NA
Network Configuration	√	√	√	NA
GPRS/CDMA	NA	√	√	NA
WIFI	NA	NA	√	AT Command
File System	√	√	√	√
System Information	√	√	√	√
Audio	√	√	√	NA
Power Management	√	√	√	√

Above table is based on the fully configured models.

Prolin API Programming Guide

PAX Technology Limited
www.pax.com.hk

Hong Kong
Room 2416, 24/F, Sun Hung Kai Centre, 30 Harbour Road,
Wanchai, Hong Kong
Tel: +852-25888800
Fax: +852-28023300

Shenzhen
4/F, No.3 Building, Software Park, Second Central Science-Tech Road,
High-Tech Industrial Park, Shenzhen, Guangdong 518057, P.R. China
Tel: +86-755-86169630
Fax: +86-755-86169634